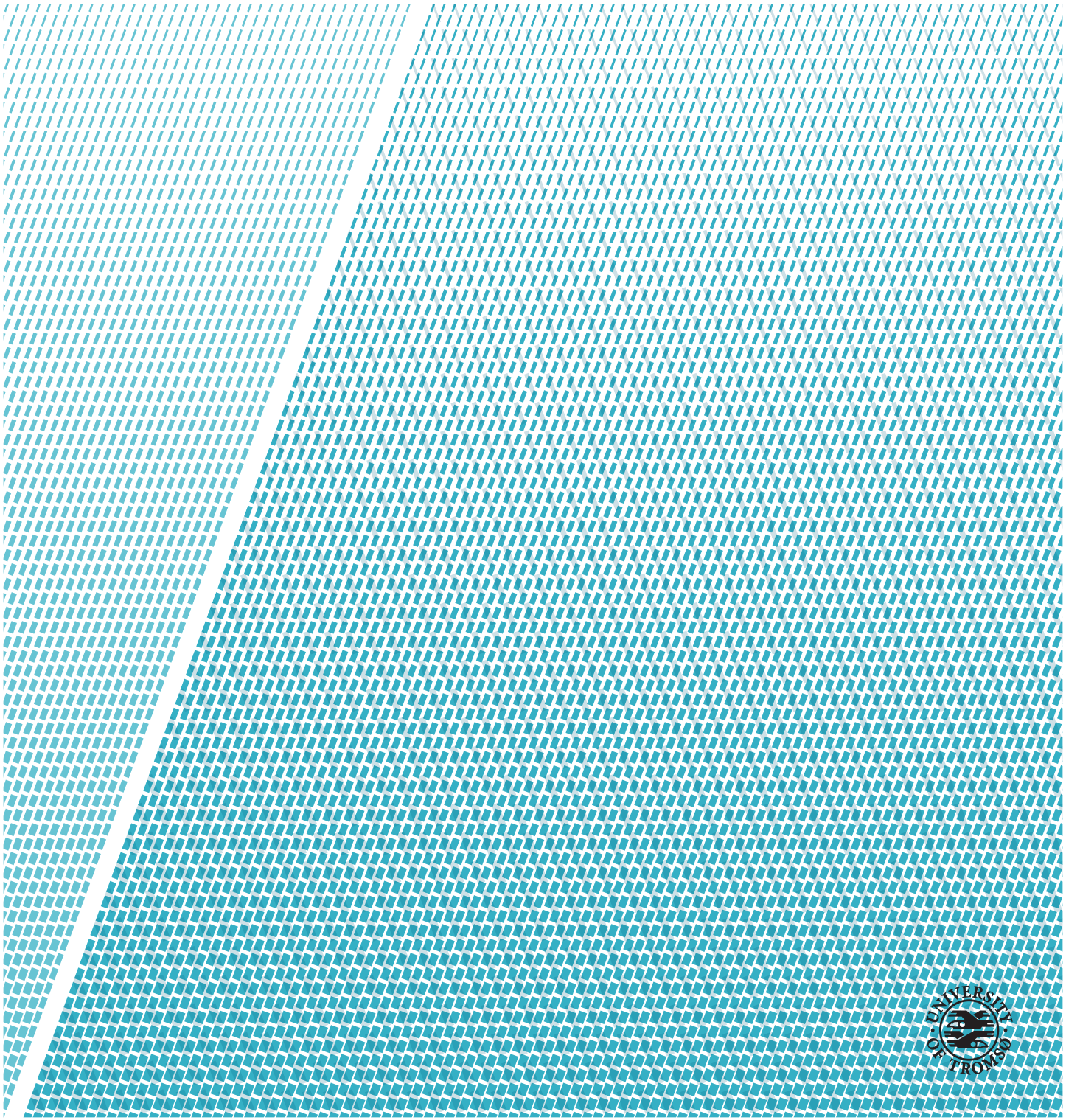
UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Computer Science

Low-cost portable air quality sensor kit and cloud service

Nina Angelvik

Individual Special Curriculum, December 2016



This thesis document was typeset using the *UiT Thesis LaTeX Template*.
© 2016 – http://github.com/egraff/uit-thesis

Abstract

Every year the air quality in Tromsø worsen during months with rapidly changing weather. There are currently only two official air quality monitoring stations in Tromsø, both stationary and located in heavily trafficked areas. Due to their permanent location, only a limited area of Tromsø is monitored and it does not include locations where people spend time outdoors. To address this issue, we have developed an idea for a low-cost portable air quality sensor kit and a data analysis service, that will allow us to crowdsource the air monitoring to high school students. The goal of the project is to test and evaluate a prototype for the air quality sensor kit, and to implement the web service, in order to see how well they will work in educational projects in high schools.

The portable air quality sensor kit is a battery-powered Arduino Uno with sensor components that can register time and GPS coordinates, measure dust density, temperature and humidity, and store the recordings on an on-board memory card. We have also implemented a cloud service consisting of a back-end storage system and a front-end providing an interface for uploads, to upload and store the collected data.

This report provides a requirement analysis and a description of the design and implementation of both the portable air quality sensor kit and the web service. Our evaluation of the sensor kit, shows that the components deliver a variable level of accuracy and that its battery time depends on data collection frequencing. The web service is able to store 500MB of data, and can easily be scaled if need be.The cost per kit is estimated to be 451,45 NOK and the monthly cost to run the web service is 207 NOK for database management and for running the application online.

Based on these results, we conclude that the portable air quality sensor kit produces high enough quality data to be used as an educational tool in high schools. It is simple enough for high school students to build and program, and together with the collected data it can be used to teach about air pollution, climate, electronics and computer science. The sensor kit does not produce as accurate results as the state of the art stations, but we are optimistic that later versions will improve the data quality. We also believe this is a promising first

step towards generating air pollution data for areas where people spend their time outside.

Acknowledgements

I would like to thank Lars Ailo Bongo, Bjørn Fjukstad and Morten Grønnesby for their help and guidance during this project. I could not have done it without you!

I would also like to thank Bjørn Fjukstad for building the prototype for the portable air quality sensor kit and writing code for the components.

I would like to thank Hedinn Gunhildrud at the Science Center of Northern Norway for designing the shield and the case for the air quality sensor kit, and creating the circuit sketch.

Also a big thank you to Maria Wulff Hauglann for always being a great help when needed and to the School Laboratory for letting me be a part of this exciting project.

Contents

List of Figures

List of Tables

1

Introduction

Air pollution is a serious health problem in cities due to increased traffic. Air conditions in Tromsø worsen during months with rapidly changing weather, due to cars using studded tires on bare roads and the usage of salt to melt iced roads. The project has been driven by the interest for getting more relevant information about the air condition in Tromsø, seeing there are only two official stationary stations gathering data on air pollution in the city. These are located in areas with heavy traffic and traffic congestion, illustrated in red on figure 1.1, and even though they are able to tell when the level of pollution exceeds what is allowed by the Norwegian government, they cannot describe the general air condition in the city. Portable air quality sensor kits could provide more information about the air condition in places where non-car commuters are spending their time outdoors.

This project is a collaboration between the School Laboratory at the faculty of science and technology and a high school class at Kongsbakken VGS. The School Laboratory hosts science activities on a regular basis for students and teachers in primary, secondary and high school. Our main aim is to build air quality sensor kits that can be used to measure air pollution, and then use crowdsourcing to collect data from all over Tromsø. Our crowd will be the students, who will build the portable air quality sensors as a part of a project in a science course. Kongsbakken VGS is located near the city center, marked in green on the map in figure 1.1. But due to the students being able to attend any high school in the county, we can assume that they live anywhere in Tromsø. We also assume that they, during their spare time, cover most parts of

town.

The requirements for the portable air quality sensors kit are the following:

- It must measure dust particles in the air, using the standard metrics particulate matter (PM).
- It must map the collected data to geographical coordinates, using for instance a GPS.
- It must store data intermediately on a storage unit which can be connected to a computer.
- It should be able to collect data continuously for at least 20 hours between each battery charge.
- It must be possible for a class of high school students to build it in a classroom.
- The total cost should not exceed that of a regular textbook, for instance in mathematics, which is approximately 700-800 NOK [1]

In order to use crowdsourcing to gather data, we need a web service which can collect and store data from multiple air quality sensor kits. The web server should:

- be simple and intuitive to use, such that the students should not need special training to understand how to upload the data.
- provide fast uploads. Once we receive the file, it should not take more than 1 second to store it in our system.
- be able to store at least 500MB of data per project.
- enble future analysis and visualization of the collected data.

Regarding existing portable air quality sensors, The Norwegian Institute for Air Research, which is in charge of the two stationary air quality stations, have also done a project where they have developed and used portable air quality sensor packs to measure air pollution in cities. Their portable sensor packs are however more expensive than ours and they are integrated into a storage

1. http://www.adlibris.com/no/bok/sinus-matematikk-r2-9788202457105

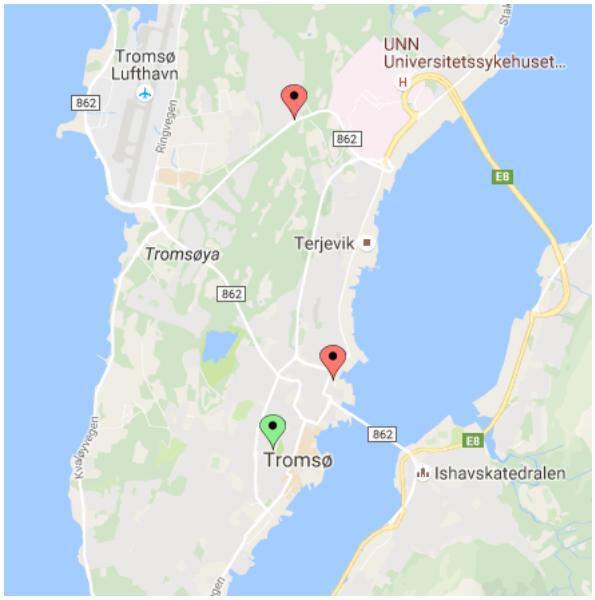


Figure 1.1: The two official air quality stations (marked in red) are located in areas that experience heavy traffic in Tromsø. The green mark represent the location of the school where the participating student attend.

system that more closed than the one we want to create.

In this project, the portable air quality sensor kits are built using an Arduino Uno microcontroller and sensors that register GPS coordinates, dust particles, temperature and humidity. The Arduino is also connected to an microSD card reader/writer, and the data is stored as a CSV file on an on-board microSD card. Files are uploaded to a Heroku server with an integrated PostgreSQL and the web service interface has a very simple layout, with only an upload button on the main page as shown by figure 1.2.

We have evaluated the air quality sensor kit by collecting data for a couple of minutes outside and 7/16 hours inside. The GPS coordinates of a location has been evaluated by comparing them to those of the same location on Google maps, and temperature and humidity has been compared with data from weather forecasts. Because the prototype still lacked a shield and a case during the evaluation, making it sensitive to weather and transportation, it was not possible do measurements that could have been compared with the stationary air quality stations. The GPS and temperature sensors both provided accurate data. The humidity sensor however needs additional calibration to provide accurate measurements, so does the dust sensor but we believe we can improve both of these for the pilot project. We evaluated battery life by modifying the frequency of writes to the on-board SD card. Writing data every second

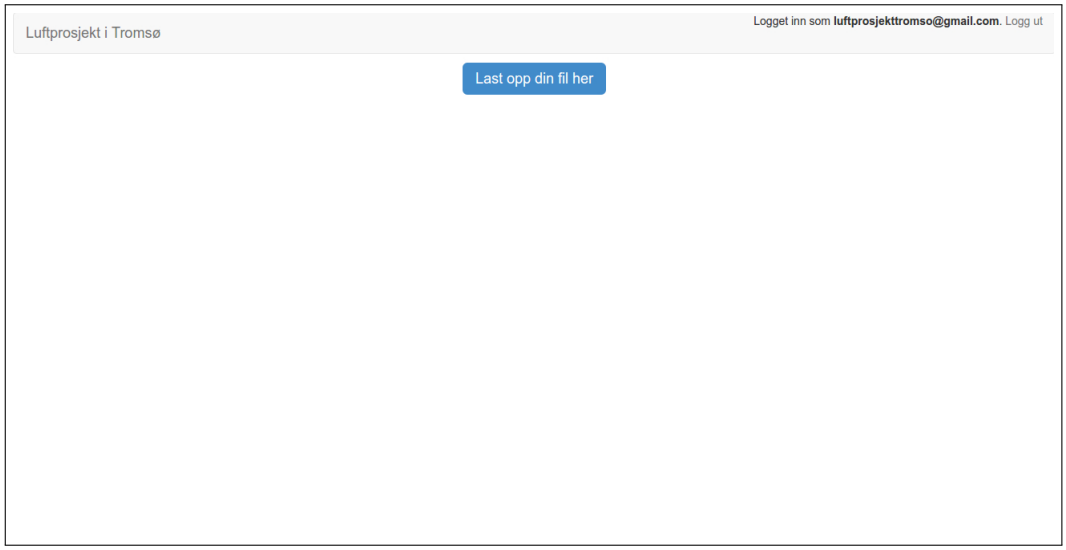


Figure 1.2: The web application has a simple and intuitive interface, where the upload button is placed on the main page

resulted in 7 hours battery life, while writing every three seconds we improved battery life to 16 hours.

Doing calculations using an exchange rate of 1 USD = 9 NOK, a portable air quality sensor kit will cost 451,45 NOK, while the web service will cost 207 NOK/month. If the web server is up and running for two months, the total cost of the project is estimated to be 8306 NOK, while the cost per student for a class with 28 students is approximately 297 NOK.

The upload time has been measured by uploading 10 1.2MB files using the load data time measured in the Chrome Developer Tools to calculate an average. Our results show that the web server is able to upload a 1.2MB file with an average time of 0.973 seconds.

In this project we have managed to create a portable air quality sensor kit that is both simple and affordable to build, and easy to use. Through our evaluation we have identified points to improve the data quality from the commodity components used in the sensor kit. By using the Heroku platform to deploy the web service, the backend storage system becomes easy to manage. The front-end is also designed to be intuitive to use. The high school class that will use the Arduino and web service in their project, will have to pay 297 NOK per student, which is much more affordable than a regular textbook.

We provide the following two contributions:

1. Implementation of a Arduino based portable air quality sensor kit and evaluation of the components in terms of performance, robustness and cost.

2. Implementation, testing and evaluation of a web service for uploads and storage of collected air quality data.

2

Background

2.1 The origin of the project

This project came to after the School Laboratory at the Faculty of Science and Technology at UiT had a meeting with science high-school teachers in Tromsø. The School Laboratory presented themselves and what kind of projects they were interested in doing, and the teacher from Kongsbakken VGS liked the idea of combining programming and air pollution research and wanted his class to participate.

2.2 Air Pollution

Air pollution is a general term describing air contaminated by anything physical, biological or chemical, such as dust, smoke or gas. It can cause respiratory and heart conditions, and the World Health Organization has deemed air pollution as the largest single environmental health risk in the world [1]. Common causes of outdoor air pollution are burning of fossil fuels, agricultural activities, exhaust from factories and industries and studded tires [2]. These emissions fill the air with different types of air pollutants, such as particulate matter (PM_10 and $PM_2.5$), nitrogen dioxide (NO_2), ozone (O_3), sulfur dioxide (SO_2) and carbon monoxide (CO).

Particulate matter is often referred to as “PM” followed by a specific number

indicating the size of the particles in micrometers. The particles come in a wide range of sizes, but those of 10 micrometers or less in diameter are considered a health risk as they can get into our lungs and cause severe health problems. Particulate matter is therefore divided into two categories: $PM_2.5$ and PM_10. $PM_2.5$ include all fine particles smaller than 2.5 micrometers in diameter, and are typically produced from all types of combustion (e.g. motor vehicles and wood burning). PM_10 include all coarse particles smaller than 10 micrometers in diameter, thus also including the $PM_2.5$ particles. These particles are, among other things, caused by exhaust from factories and industries, and studded tires. Nitrogen dioxide (NO_2) is mainly caused by combustion from traffic and industry. It is mainly a problem in the larger cities in Norway, where the level of NO_2 exceeds what is allowed by the government. NO_2 irritates the lungs and can cause respiratory infections, especially with those who suffer from asthma and bronchitis. Ozone (O_3) is created when NOx (the combination of nitrogen oxide (NO) and nitrogen dioxide (NO_2)) and volatile organic compound (VOC) mix with sunlight. When in the stratosphere, ozone will protect the earth against UV radiation, but at ground level the substance can cause harm to people and the environment. Sulfur dioxide (SO_2) is caused by burning substances containing sulfur, such as the fossil fuels oil and coal. SO_2 as an air pollutant in only a problem in cities where industries such as power plants and oil refineries cause local emissions. Carbon monoxide (CO) is created as a result of incomplete combustion of fuels and wood. The level of CO in Norwegian cities is normally too low to cause any health-related problems [3].

The stationary air quality stations in Tromsø are monitored by The Norwegian Institute for Air Research (NILU), an independent Norwegian non-profit institution which monitors climate change, global air quality and air pollution transport pathways [4].They also monitor air quality stations in 21 other cities in Norway. These stations measure the level of PM_10, $PM_2.5$ and NO_2, the three air pollutants which contribute the most to air pollution in Norway. According to NILU's API, they stationary air quality stations also able to measure CO, O_3, PM_1 and SO_2[1].

NILU has also done research on how portable air quality sensor packs and crowdsourcing can be used to get better view of the air pollution in a city. From 2012 to 2016 they have been in the lead of the CITI-SENSE project where portable sensors have been used to measure the air quality in nine cities [5]. The goal was to empower the citizens of the nine participating cities to contribute and participate in environmental governance. During the project, the CITI-SENSE team developed seven tools, to involve the citizens in assessments of air and environmental quality: (1) The CITI-SENSE Citizens' Observatories Web Portal

1. `https://api.nilu.no//lookup/components`

Figure 2.1: The LEO mobile sensor pack has been used to measure the air quality in nine different cities during the CITI-SENSE project[2]

functions is an access point to all of the sensor tools, apps and widgets used in the project. It provides information about how to acquire and use them, as well as information about the project itself. (2) The Personal Air Monitoring Toolkit consists of three parts: the LEO (Little Environmental Observatory) mobile sensor pack, illustrated by figure 2.1, the ExpoApp which connects to the sensor units and uploads data to a server, and a computer application to manage the sensors.

(3) The CityAir smartphone application allows the public to express their perception of the outdoor air quality at their location. Provided access to a network and GPS signals, it will mark the user's current location onto a displayed map and the user is free to add a pollution marker, an assumed source of pollution or a comment. Any data provided by the user will then be uploaded by the app to the CITI-SENSE platform, making it available for other CITI-SENSE users to explore or download. (4) The On-line air quality perception questionnaire is a tool for collecting and analysing air quality issues as perceived by users. The results of the questionnaires have been illustrated in graphs such as the one in figure 2.2.

(5) Environmental monitoring toolkit for public places is a toolkit used to perform subjective and objective monitoring of environmental quality and satisfaction. The tools consist of sensors that register information about the weather and noise in a wanted area, applications that connect to and manage the sensors, an application where users can enter information about their personal perceptions and view real-time measurements from the sensors and

2. http://social.citi-sense.eu/UseExamples/PersonalAirMonitoringToolkit.aspx
3. http://oslo.citi-sense.eu/Browsedata.aspx

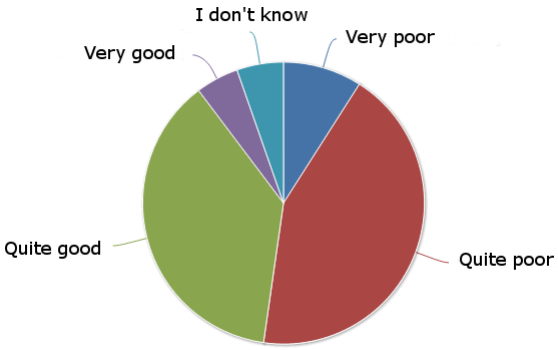


Figure 2.2: The graph shows the results from a survey where the citizens of Oslo were asked to describe the air quality in Oslo.[3]

an application to detect noise in a user's surroundings. The toolkit is meant for dedicated campaigns and the responsible campaign team is responsible for initializing the kit and train the users. (6) A data visualization web page, illustrated in figure 2.3, is a tool for viewing data collected from stationary and portable sensors, perceptions from the AirCity app and answers from the online questionnaires.

In terms of how data about air pollution is collected and visualized, CITI-SENSE is very similar to what we want to achieve. Our project does, however, also present an educational aspect that CITI-SENSE does not. We want to create a portable air quality sensor kit that can teach people not only about air pollution, but also about programming and electronics. We also want the kit to be used in schools, so that children and youth can learn how they can use programming to create useful things.

2.3 Crowdsourcing

In this project we are interested in gathering information about the air situation in areas that are important to people who spend much time outdoors. A way of accomplishing this is to use crowdsourcing, which is when work is being outsourced to a crowd of people [6]. There are several types of crowdsourcing, one of them being microtasking. Microtasking involves splitting large tasks into microtasks, which then are handed out to a large group of individuals. This type of crowdsourcing works well with problems that can be solved by applying the same simple process to each part of the larger data set, such as

4. http://srv.dunavnet.eu/new/citisense/OutdoorDataPortal/

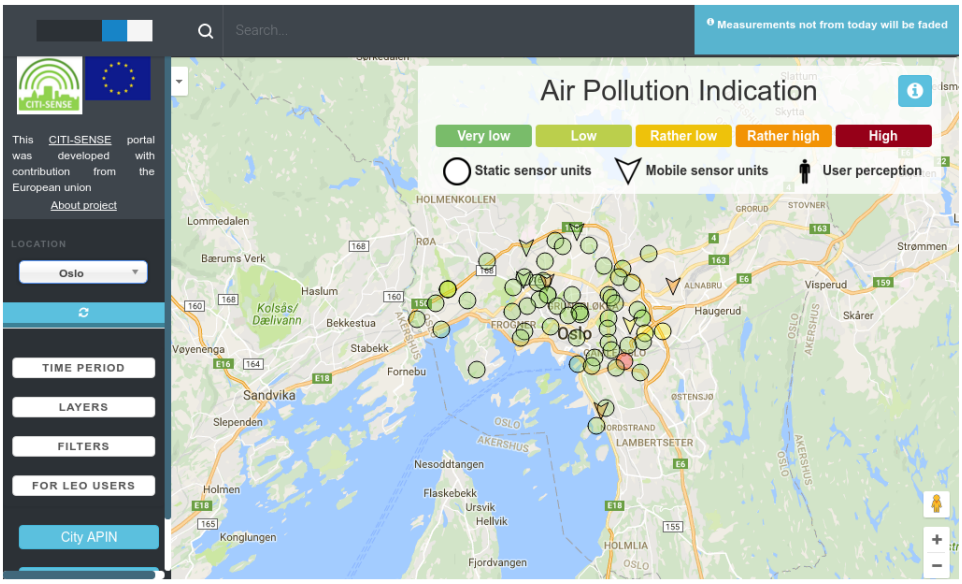


Figure 2.3: Screenshot of the The CITI-SENSE data visualization web page showing the last air pollution measurements that was collected in Oslo before the project ended. This web page is now no longer in use.[4]

when collecting data using for example smartphones or portable devices.

The Ocean Sampling Day (OSD) is an example of a campaign where crowdsourcing has been used to collect samples from the ocean all over the world. OSD is a scientific and global sampling campaign, where the cumulative samples have helped analysing marine microbial biodiversity and function. The first OSD took place on the summer solstice of 2014 and was repeated in 2015 and 2016. Each year between 500 and 600 people from all over the world participated by taking water samples and identifying microbes in it [7]. Participating members were not only scientific teams, but also citizen scientists. To engage the public in joining the OSD, Micro S3 created the OSD Citizen Science project (MyOSD) [8]. To participate in MyOSD, regular citizens could obtain a sampling kit containing necessary equipment such as a handbook, a log sheet, thermometer, syringes and gloves [9, pp.5]. They would use this to take water samples, which they would analyse the samples as described in the MyOSD Handbook and enter the results in the OSD Citizen App. The data would then be uploaded to and visualized on the OSD mapserver [10]. Figure 2.4 shows areas sampled during the Ocean Sampling Day of 2016 visualized on the mapserver. The green spots indicate single samples, while the red indicate a group of single samples within the same area. As the figure shows, many of the samples were collected in Europe, especially in the area of Germany.

5. https://mb3is.megx.net/osd-app/samples

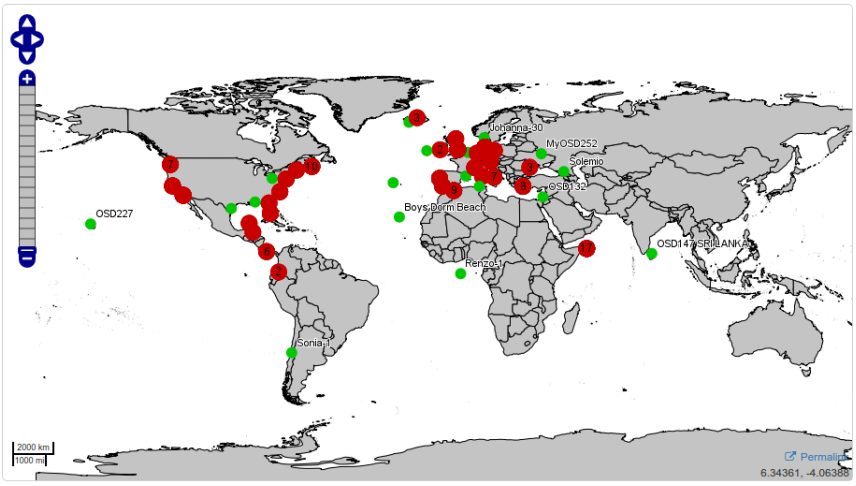


Figure 2.4: The Ocean Sampling Day 2016 was hosted in Germany and many of the samples that day was taken in areas in and around Germany.[5]

Ocean Sampling Day is a great example of how we can generate a very large amount of data in a short amount of time, which can provide us with a great amount of information about for example microbial diversity. What we want to achieve in this project is of a much smaller scale, but the idea is the same. By providing people with the tools they need, and give them instructions on how to deliver the results, we can exploit people's interest in the field and use it to collect more data.

An example of how crowdsourcing can be used to generate a large database which is useful to many, is CrowdSignals.io, a campaign starting in 2016. It is hosted by AlgoSnap, a company that specializes in creating intelligent algorithms and collecting datasets for devices such as smartphones, smartwatches and activity trackers [11]. The goal of the campaign is to create a large dataset containing mobile and sensor data collected from smartphones and smartwatches [12]. The dataset will include system and network logs, social connections, communications, geographical location, user interaction and user-provided survey feedback. One of the largest drivers of innovation within areas such as public health, social science and computing is personal device data, but the data collection campaigns that are necessary to obtain such data, are also expensive, time consuming and legally challenging [13]. Researchers who lack data for their projects often do so because they lack the funding and expertise to collect data from personal devices. Thus the dataset created by CrowdSignal.io is meant to provide researchers, students, scientists and product groups with the data they need to solve important societal problems. Figure 2.5 illustrates how CrowdSignal.io works. The campaign itself is crowdfunded, meaning that it is funded by small contributions from hundreds of

researchers. Volunteers who are willing to share their anonymized data will be informed about the risks and benefits of sharing data, before they may consent to participate. During the campaign they are free to opt-out at any time and they will receive financial compensation after the data has been collected. They will also be given access to insights and analytics about their own data, they will know who is using their data and for what purposes, and they will receive updates on how their data has enabled advances in various scientific areas. Before the collected data is shared with the contributors, it is encrypted and securely transmitted. All sensitive information is anonymized and datasets are watermarked so that unauthorized transfers to third parties can be traced. The sponsors contribute with a given amount of money and in return they get access to the core, anonymized data only a few weeks after it has been collected. Higher-level sponsors can also specify demographic surveys to be answered by the volunteers. About 12-18 months after the data is collected, it will be made available to non-sponsors for research purposes, free of charge. Everyone who intends to use the data (both sponsors and non-sponsors) must also sign a legal agreement to commit to ethical use of the data.

At this point, CrowdSignals.io is not available to anyone outside the United States, making it unavailable to be used in this project. Still, if it had been, we would probably still have chosen to create our own web service, as it would give us complete control of the data.

2.4 Arduino

"Arduino is an open-source electronics platform based on easy-to-use hardware and software."[7] It was originally created as an easy prototyping tool for students without a background in electronics and programming. Today, the simplicity of the platform, as well as the very affordable hardware, has made Arduino useful for thousands of projects, both for hobbyists and professionals [14].

The heart of Arduino is the microcontroller, a simple computer which main purpose is to interface with sensors and devices. Using the Arduino programming language and the Arduino IDE, the microcontroller can be programmed to read inputs from these components and turn it into an output, such as reading coordinates from a GPS sensor and turning on a light if the coordinates are valid. Arduino is great for hardware projects where the tasks are simple and repetitive and because it only requires a battery pack to keep the voltage above a certain level, it is also very portable [15]. As soon as a power source is

6. http://crowdsignals.io
7. https://www.arduino.cc/en/Guide/Introduction

Figure 2.5: CrowdSignal.io is heavily based on crowdsourcing, both when it comes to funding and collecting data.[6]

provided it will start executing code and when the power is unplugged, it will stop.

In terms of easy-to-use platforms used by makers worldwide, Arduino is not alone. Raspberry Pi is a fully functional computer created by the Raspberry Pi Foundation, meant as a tool for helping people learn about computing, problem solving and digital making. It is in the same price range as Arduino and it also offers the possibility of connecting components to it, but the hardware is more complex. Unlike an Arduino, Raspberry Pi can run an operating system from an SD card, has a dedicated processor and uses the SD card as flash memory for the entire system. Raspberry Pi is great for software applications, as it can multitask and the operating system makes it possible to install additional software needed for the application. However, due to the operating system, the Raspberry Pi requires a constant 5V power to stay on. To do this, one needs to set up a power supply and some additional hardware, and by that, portability becomes an issue. Should the power drop, one might end up with a corrupt operating system as the Pi, like a regular computer, needs to be shut down via a software process.

Regarding networking, the built-in Ethernet port makes it easy for the Raspberry Pi to access any network. Neither is it hard to achieve wireless internet on the Pi, as it only requires a USB Wi-Fi dongle and the installation of a driver. To get the Arduino online, the process is a bit more complicated. The microcontroller does not come with a built-in support for either Ethernet or Wi-Fi connection, but shields that can be plugged onto the microcontroller to add those functionalities do exist (though it will require some extra code to get them to work).

As mentioned earlier, both the Raspberry Pi and the Arduino is able to connect to sensors. It is however easier to connect analog sensors to the Arduino, as the microcontroller can use the code written by the maker to interpret and respond to a wide range of sensors. In order to interface effectively with the same type of sensors, the Pi requires software, and if you only want to perform a simple task such as measuring dust particles, this might be considered to be a bit of an overkill.

The Arduino is usually the better choice when it comes to hardware projects, the same way the Raspberry Pi is better at software applications. But for projects consisting of tasks that are both simple and complex, a mix between the two might be the better choice. In our project, we could for example have used an arduino as our portable air quality sensor kit, and a Raspberry Pi for uploading the data to a web server. We have, however, chosen to use a web application to upload data from the air quality sensor kits, instead of a Raspberry Pi, for two reasons: 1) Portability: Uploading files using only a Raspberry Pi would limit the areas where participants could upload their data and they would

have to wait in turn to do it (unless there were multiple Raspberry Pis). A web application will let them upload data wherever they can access a computer and they could do it simultaneously. 2) Cost: By not making the project dependent on a Raspberry Pi, the overall cost would be less expensive.

3

Implementation and design

3.1 Architecture

Figure 3.1 shows the architecture of the project. A crowd of people, divided into small groups or individuals, will assemble and program their own portable air quality sensor kit. They will carry it around with them outdoors to collect data, which they will transfer to a computer and upload to a web server via a web application. In case of future visualization, the same web application that is used for uploads may also be used to visualize the data.

3.2 Use case

In our project, the air quality sensor kits will be built, programmed and used by a high school class that has previously worked with Arduino. It will be used as an educational tool to help them learn about air pollution in general, whilst improving their skills in electronics and programming. They will also use the data gathered by the air quality sensor kits, together with weather data from the Norwegian Meteorological Institute, to explore how the weather affects the air quality in Tromsø. The class consist of 28 students, who will work in groups of 3-4. The students will participate in a workshop at the School Laboratory, where each group will receive an Arduino kit with all of the needed components. They will also be given the help they need to put the air quality sensor kit together and program it. Members of the group will then carry the air quality

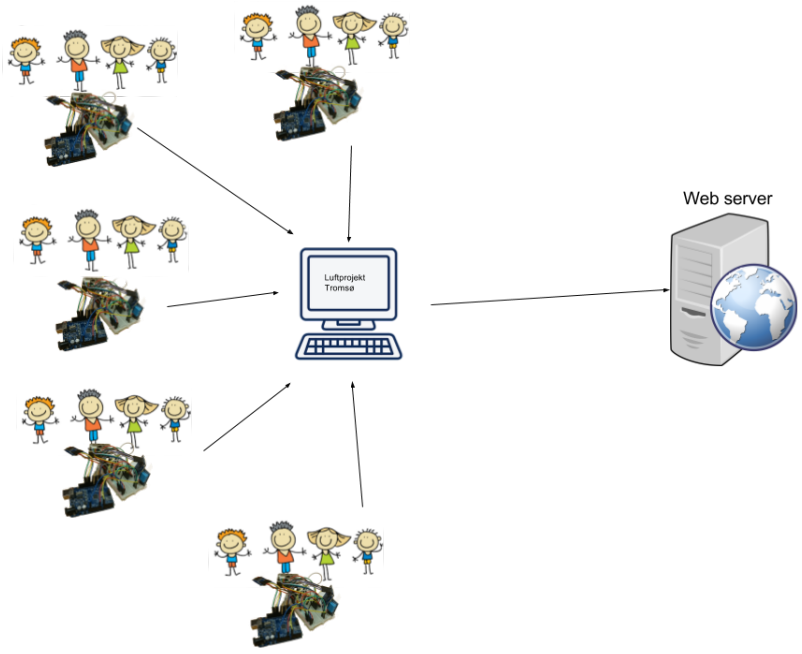


Figure 3.1: The architecture of the project consists of a crowd of students, multiple air quality sensor kits, a web application used to upload data and a web server that stores the data.

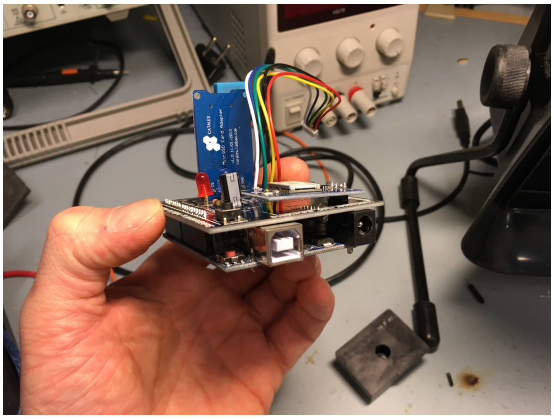


Figure 3.2: The prototype for n air quality sensor kit.

sensor kits with them when walking outside, collecting data from the outdoor environments. The collection period during the spring semester and will last for about four weeks. They will use an SD card reader to transfer the data file from the SD card to a computer and then use the web application to upload the file to the web server.

3.3 Portable air quality sensor kit

The air quality sensor kit consists of the following components: An Arduino Uno, a NEO6MV2 GPS module, A Sharp GP2Y1010AU0F optical dust sensor, a DHT11 humidity sensor for measuring temperature and humidity, a microSD card reader/writer, a microSD card and a portable power pack. It has two leds that require one 220 ohm resistor each, and the dust sensor also require a 150 ohm resistor and a 220uf capacitor. Picture 3.2 shows the newest prototype of the portable air quality sensor kit. We will also build a case that can fit the air quality sensor kit, in order to make it easier to carry around.

Figure 3.3 shows the circuit sketch of how all of components are connected together. All components except for the battery, are connected to ground and power, and they also have additional connections to the microcontroller. The SD card reader is connected to digital pins 10 to 13. The dust sensor has three internal connection, as well as a one to digital pin 8 and analog 0. The GPS sensor uses digital pins 3 and 4, and it also has an antenna connected onto itself. The dust and humidity sensor is connected to digital pin 2 and the two led lights that will be used to indicate if the GPS signal is valid or not, use digital pins 6 and 7. The difference between digital and analog pins is that the analog pins can receive analog signals (a continuous voltage range) and

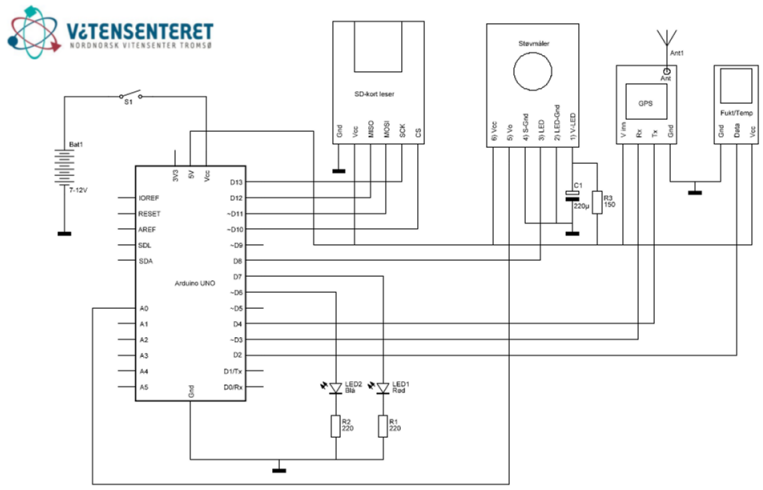


Figure 3.3: A circuit sketch illustrating how the the components are connected together.

convert them to digital numbers within a given numerical range. By digital numbers, we mean numbers converted into 0s and 1s, which are the only values that det digital pins operate with [16].

Figure 3.4 shows the shield and the placement of the sensors. The code used to program the Arduino is written in C++ like programming language in the Arduino IDE. It consists of a set-up part and a loop, where the set-up part contains everything that needs to be done once before the program runs. In our case this includes initializing the sensors and the SD card reader, and creating/opening a file to which data can be written. In order to translate signals from the sensors to readable data, we make use of both internal and external libraries (DHT11[1], TinyGPSPlus[2], SoftwareSerial [3], SD [4] and the standard library). The dust sensor detects particles by using an infrared emitting diode and a phototransistor that catches the reflected light of the dust [17]. Its output is an analog voltage proportional to the dust density, and to collect data from the dust sensor, we have to perform some mathematical calculations. We use a led to measure an analog signal for a given amount of time. The measured voltage is then mapped to a digital number between 0 and 1023, before the dust density is calculated using a linear equation desribed in Chris Nafis' paper on Air quality Monitoring [18].

1. http://playground.arduino.cc/Main/DHTLib
2. http://arduiniana.org/libraries/tinygpsplus/
3. https://www.arduino.cc/en/Reference/SoftwareSerial
4. https://www.arduino.cc/en/Reference/SD

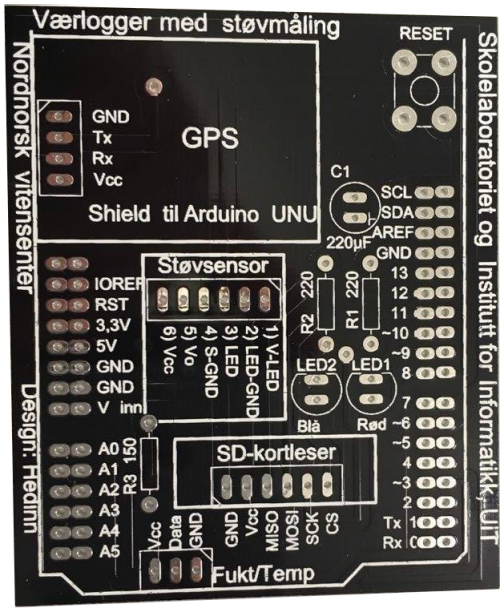


Figure 3.4: This shield has been specially designed for the air quality sensor kit by The Science Center of Northern Norway .

3.4 Web service

3.4.1 Web application

The web application is implemented in Ruby on Rails 4, and provides a simple web interface with a login page and a main page containing an “upload“ button. In the implementation we have used multiple gems, which are Ruby libraries. The authentication mechanism is implemented using a gem called Devise [5]. In addition to the log-in mechanism, Devise also include a solution for registration, resetting passwords and updating user accounts, but these options have been removed as we want to restrict access to the web application. This means that all users must be added manually. The files are uploaded using a simple form, only allowing csv files to be uploaded, and the design on the page is made with Bootstrap [6].

3.4.2 Web server

TThe web application is deployed on Heroku, which is a cloud application platform that lets the developers focus on developing their applications, while they

5. https://github.com/plataformatec/devise
6. http://getbootstrap.com/

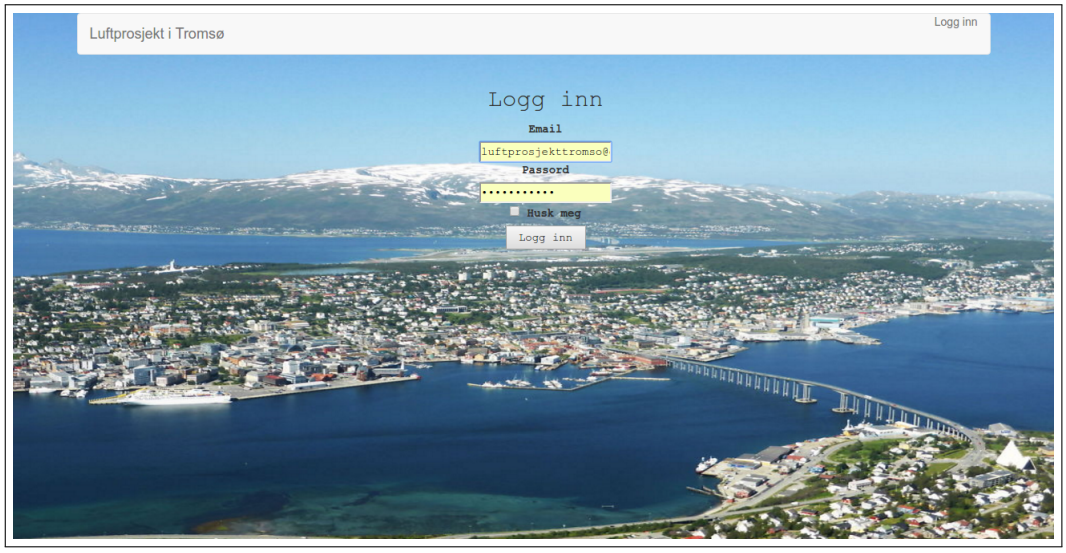


Figure 3.5: The current the log in page on the web application at https://luftprosjekttromso.herokuapp.com/

provide services that handle deployment, server management and scaling [19]. Heroku provides language support, making it possible to develop applications in a variety of programming language, such as Ruby, Python and Go. They also provide "add-ons", which are already implemented services managed by Heroku, including database services and monitoring. Applications are run in dynos, containers that either handle HTTP requests (web dynos) or do work asynchronously (worker dynos).

The web application is deployed through a git integration on Heroku and runs on two dynos: a web dyno and a worker dyno. It also uses three add-ons: Heroku Postgres (the hobby basic plan)[7], MemCachier[8] and Redis To Go[9].

The Postgres add-ons provides the application with a PostgreSQL database with 10 million available records. This add'on is required due to Heroku's ephemeral filesystem [20], where each dyno has its own temporary file space which is invisible to other processes and the files are deleted when the dyno is stopped or restarted. The files from the air quality sensor kits are uploaded by the web dyno and stored as a record in the database. Each file consists of many individual recordings from the sensors on the air quality sensor kit, so the worker dyno must parse the file and store each recording as an individual record

7. https://elements.heroku.com/addons/heroku-postgresql
8. https://elements.heroku.com/addons/memcachier
9. https://elements.heroku.com/addons/redistogo

in the database. An additional index in the table ensures that all records have a unique combination of timestamp, latitude and longitude. Invalid records (e.g records that have one or more empty fields) are also rejected. Memcachier is a memcache server that used the Dalli gem [10] to create cache store. The cache store is used to make the uploaded files available to the worker, without it having to query the database. After the worker is done parsing and putting the data into the database, it will remove the file from the cache. To perform work asynchronously we use Resque, a gem that uses Redis To Go to create background jobs and put them on queues [11].

To run the application on Heroku, we must create a Procfile [12] that will define process types and the commands that they should run at start-up. Our procfile contains $web : bundle exec rails server -p $PORT$ which defines a web process that will start up the web server. To set up the worker dyno, a Resque server must be created and mounted to a route in the application. Then, a Redis instance must be created and mapped to the the Resque server. The environment for the worker is described in a rake task, which is executed by the Rakefile (Ruby's alternative to a Makefile in C).

10. https://github.com/petergoldstein/dalli
11. https://github.com/resque/resque
12. https://devcenter.heroku.com/articles/procfile

4

Evaluation

To evaluate if the portable air quality sensor kit and the web service implemented in this project can be used to measure air pollution, we will answer the following questions:

1. Does an Arduino Uno and compatible sensor-based air quality sensor kit provide accurate enough data to measure air pollution?

2. How easy is it to build the portable air quality sensor kit?

3. How long can it last collecting data outdoors, in terms of battery time and robustness of the components?

4. How much will it cost to build the portable air quality sensor kit?

5. How much will cost to run the web server?

4.1 Methodology and results

1. Does an Arduino Uno and compatible sensor-based air quality sensor kit provide accurate enough data to measure air pollution?

To answer this question, we have compared the measurements from the air quality sensor kit with coordinates from Google Maps, temperature and humidity from weather forecasts in Tromsø. The prototype that we used when we did our testing, did not yet have a shield or a case, making it both sensitive to weather and difficult to carry around without pulling out chords. Thus we have not been able to compare the measurements from the dust sensor with data from the stationary air quality stations, As the air quality sensor kits used by the student will have both a shield and a case, we expect it to handle both weather and transportation when used in the pilot project. The dust particle data which are collected by the stationary air quality stations and presented by luftkvalitet.info [1] is the hourly average of an unknown amount of data recordings, so ideally we would have placed our air quality sensor kit next to one of them for an hour. However, this and amore extensive testing of the other sensors outdoors, has been postponed until the case is ready. Results described in this section are generated by collecting data outdoors every 4 seconds for 3 minutes.

We observed an accuracy of 10 meters for the GPS module, which we think is accurate enough for this project. We also noted that if the Arduino is without power for a long while, the GPS module spends some time trying to get a valid signal before it succeeds. We recommend to use the air quality sensor kit often, as it seems like the GPS is able to connect faster when it is used frequently. The sensor also operates with coordinated universal time (UTC+0), which means that the recorded time must be altered to our time zone before it is saved to the database.

According to the weather forecast at the time the data was collected, the temperature outside was 0-2 degrees celsius and the humidity at about 90%. Chart 4.1 shows that the temperature sensor provided us with pretty accurate data, at least in the end of the measuring session. The higher temperature in the beginning is most likely due to the air quality sensor kit having lain inside prior to collecting data outside. Chart 4.2, on the other hand, that the humidity sensor is way off. This could be due to a calibration error, but If not, the project does not depend humidity measurements and will do fine without.

Graph 4.3 illustrates the amount of dust particles measured by the dust sensor during the 3 minute measuring session. Investigating the output from the dust sensor, it reveals an interesting pattern. It only collects new data at an interval of 30 seconds, rather than every four seconds. The consistency in time between the recordings, gives us a reason to think that we might be able to achieve more frequent recordings by making changes to the code that controls the sensor. If not, we will extend the write interval to every 30 seconds in the pilot

1. http://www.luftkvalitet.info/home.aspx

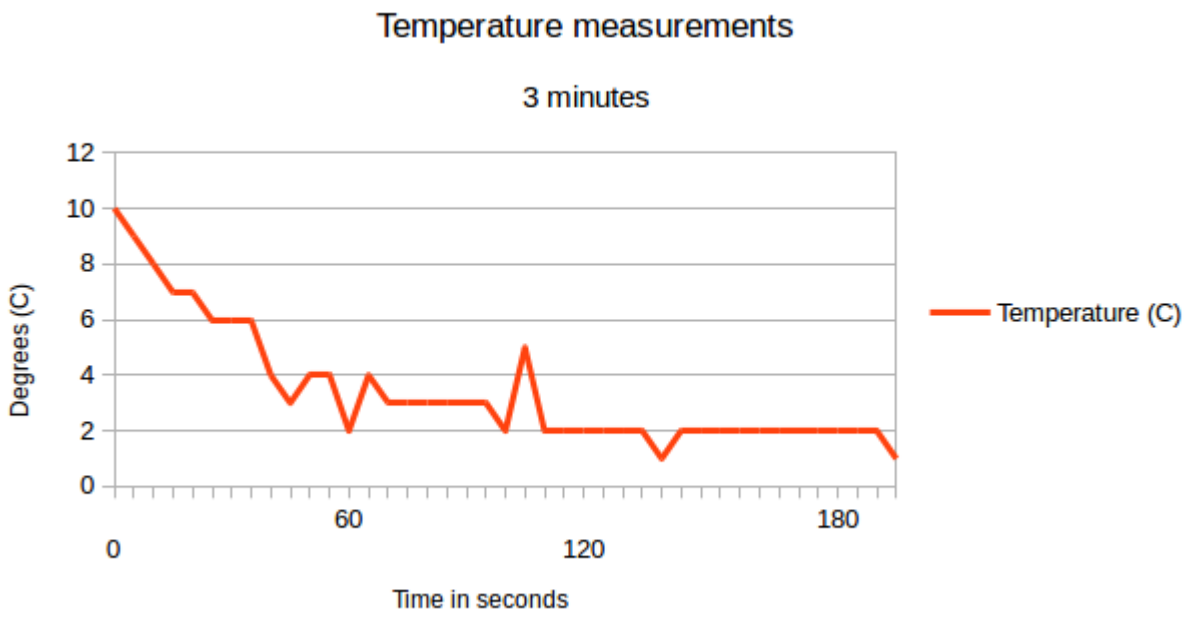


Figure 4.1: With an expected temperature of 0-2 degrees celsius outside, the temperature sensor was able to provide us with pretty accurate data.

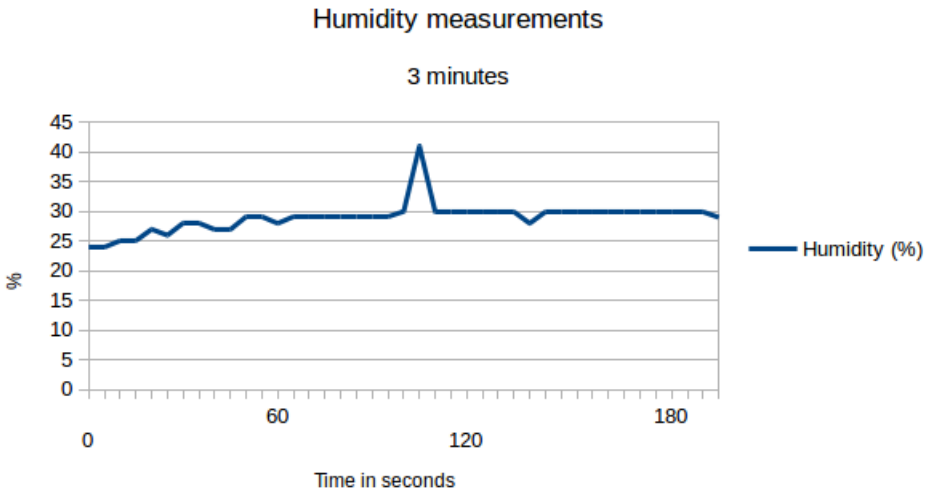


Figure 4.2: The humidity was not able to register humidity properly, most likely due to a calibration error.

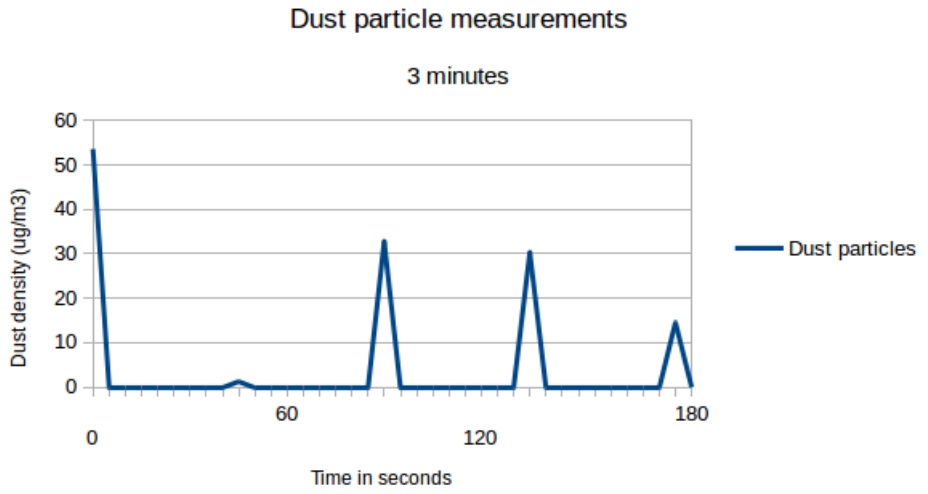


Figure 4.3: The dust sensor did not collect data at the expected 4 second intervals. Instead, it showed an interesting pattern by collecting data every 30 seconds.

project, so to only get recordings containing dust measurements. Results from Chris Nafis' Air Quality Monitoring [18] also show that when compared to a laser particle counter and after some calibration, the the dust sensor is able to generate almost the same results as the laser particle counter. This gives us further reason to believe that the Sharp sensor can be used to measure dust particles in the air in this project. In terms of what particle sizes the sensors detects, Sharp's own description of the component claims that it is especially good at detecting fine particles [21]. This will become more clear when we compare our results with data from the stationary air quality stations.

Even though the data generated by the components is variable, the results do show that it is possible to use our portable air quality sensor kit to measure air pollution in the city. As it is meant to be an educational tool more than a state of the art air quality station, it is more important that the data can detect air pollution, rather than being 100% accurate.

2. How easy is it to build the portable air quality sensor kit?

To evaluate how easy it is to build and program the portable air quality sensor kit, I tried to understand the construction and programming myself. As a master student in computer science with no previous experience with Arduino or the IDE, I found it a bit challenging to wrap my head around the electronics. However, given a basic understanding of the platform, and some guidance to how the portable air quality sensor kit should be assembled, I consider this

Table 4.1: Cost of a air quality sensor kit

Component	**USD**	**NOK**
Arduino Uno	3.14	28,26
NEO6MV2 GPS module	8.19	73,71
Sharp GP2Y1010AU0F dust sensor	5.99	53,91
DHT11 humidity sensor	1.00	9
MicroSD card reader/writer	1.04	9.36
MicroSD card	3.69	33.21
Shield		54
Case		20
Power pack		169
Capacitor	0.09	0.81
Resistor x 3	0.02	0.19
Total :		**451,45**

project to be achievable for a class of high school students, but we will not no for sure until the students have completed their project.

3. How long can it last collecting data outdoors in terms of battery time and robustness of the components?

The battery time has been measured by starting the air quality sensor kit with a fully charged battery and letting it collect data continuously until the power runs out. This has been done while writing data to the SD card once every second and once every three seconds. Using one second intervals, the battery lasted 7 hours and 40 minutes, and by extending the intervals to three seconds, the battery was able to last 16 hours and 51 minutes. This is a good indication that the number of writes to the SD card strongly affects how long the battery will last. Thus there is a trade-off between battery time and quantity of data. By increasing the write interval from every second to every three seconds, we reduced the gathered data from 5MB to 1.2MB. Even though we have not fulfilled the requirements regarding battery time, we have chosen to keep the write interval at every three seconds. However, should be not be able to make the dust sensor collect new data more frequently, we will extend the write interval to every 30 seconds, which prolong the battery life time even more. In terms of robustness, the air quality sensor kit should handle being carried around very well, once all of the components are soldered to the shield and the entity is placed in a box.

4. How much will it cost to build the portable air quality sensor kit? The cost of one portable air quality sensor kit has been measured as a worst case cost, using an exchange rate of 1 USD = 9 NOK.

All of these components can be bought from web sites offering free shipping or in a local store, which will eliminate that cost.

The estimated cost of one air quality sensor kit is 451.45 NOK. Regarding how many air quality sensor kits a class of 28 students will need, we estimate about 9 functioning air quality sensor kit or one per group. However, because of the risk that something might break in the middle of the building process or be malfunctioning upon arrival, we advise the class to buy equipment enough for 15 air quality sensor kits. The Science Center of Northern Norway will also charge 40 NOK per student to assist the students with soldering the air quality sensor kit. For a class of 28 students this bring the total cost of the 15 air quality sensor kits to 7891.75 NOK, with a cost per student at 281.85 NOK.

5. How much will cost to run the web server?

The cost of the web server will very much depend on how much data the students collect. Storing data every three seconds will generate 1.2MB of data in 16 hours and 50 minutes. If we assume that each group of students spend maximum 4 hours outside every day gathering data, the battery will last 4.45 days before it needs to be recharged. This means that each group will generate 1.2MB/4.45 days = 0.27MB of data every day, and 1.9MB of data every week. The total amount of data collected by all groups during the whole collecting period is therefore estimated to be approximately 1.9MB x 4 weeks x 9 groups = 69MB data.

The most basic PostgreSQL database plans on Heroku operate with a number of available records rather than MB and GB, so we need to transfer the estimated amount of MB into an estimated number of records. A CSV file containing 1MB of data from the portable air quality sensor kit, consists of roughly 17 600 records. As the Arduino collects the same amount of data every time it writes to the SD card, the number of records/MB per file will remain pretty consistent. If 1MB is 17 600 records, we will need at least 69 x 17 600 = 1 214 400 available records to store all of the weather data. To fulfill the requirement of being able to fit at least 500MB of data in the database, we need to have 8 800 000 available records.The cheapest PostgreSQL database plan that can fulfill these requirements, is the Hobby Basic plan, which provides 10 million available records at $9/month.

The application also requires a web dyno to run and a worker dyno to do the asynchronous work, and these cost $7/month each. The total cost of the web service is $23/month or 207 NOK. For a class of 28 students, the cost per student will be approximately 7 NOK/month.

We have evaluated the response time by uploading 10 1.2MB files and taking the average of the response time of each of the requests, which we got from the load event info in the Chrome Developer Tools. We found that the average upload time for one file was 972.4ms. However, due to bandwidth limitations other people might experience slower uploads. Seeing as the Hobby Basic supports 20 simultaneous connections, this could potentially increase the response time. We do not consider this to be much of a problem, at least not in the pilot project, as there will only be 9 portable air quality sensor kits to upload data from and it is not likely that multiple groups will upload their data at the exact same time.

4.2 Total cost of the project

The total cost of running the web server and building the air quality sensor kit will be 8098,75 NOK if the server runs for 1 month and 8305,75 NOK if the server runs for two months, given an exchange rate of 1 USD = 9 NOK. For a class of 28 students the cost per student will be 290 NOK or 297 NOK, depending on how long the server runs. And that is much cheaper than a regular textbook.

5 Discussion

5.1 Sensitive information

As we, in this project, use students to collect air pollution data from all over the city, we will collect data that can be used to track individual persons. This is the main reason why the web application requires user authentication and that users can only be created manually by an administrator. We do not want data to be retrieved from the database or visualized in any way that will make it possible for people outside the project to know the home location of the students. To solve this, we plan on removing the first five minutes from each datalog and we believe this to be sufficient for anonymizing the data. It will also not be possible to retrieve usernames/full names with the data. Once this is implemented, we will make the data open and available to other people.

5.2 Choice of storage

During the implementation of the web service, we had to decide on where to store the collected data. We considered three options: Amazon Web Service, Heroku or setting up our own server on campus, and decided to use Heroku. This decision was mainly based on the simplicity of the platform and and since it requires least amount of time managing the server. If we want to use another data store, this can be done without too many difficulties and should we need more space, we can always upgrade our add-on. The easy deploys of

web applications and the version control, were also good reasons to why we chose Heroku. Future work will be to look more into how the Amazon Web Service and the Amazon Simple Storage Service (S3) will work in our project, as it is less expensive than Heroku.

6

Conclusion

We have created an easy and affordable Arduino-based prototype of a portable air quality sensor kit that can measure temperature, humidity and dust density, and map the findings to geographical locations. We have also created a web service, consisting of a frontend that provides an interface to upload data and a backend storage system. The air quality sensor kit has been tested by collecting air samples outside for a couple of minutes and validating the results with valid sources such as Google Maps and weather forecasts. Though some of the sensors proved to be a bit inaccurate, we have suggested some possible solutions regarding how this can be fixed. Some sensors will also need to be tested further once the prototype gets a case, making it easier to carry around outside. The battery time was evaluated by letting the air quality sensor kit collect data continuously until the battery ran out, testing different write intervals each time. Writing to the SD card is what affects the battery time the most, thus there is a trade-off between battery time and data quantity. By recording data every three seconds, we are able to collect a good amount of data while still making the battery last for an acceptable amount of time. However, due to inconsistencies in the how the dust sensor collects data, we might have to do our recordings at larger intervals. Regarding the web service, we have chosen a storage system which can meet the requirements in terms of storage capacity and expected response time. It is able to support querying of data, which means that it will be able to support visualization in the future. For a class of 28 students, as the one participating in the project, the cost of the portable air quality sensor kit and web service is estimated to be between 290 NOK and 297 NOK per student, depending on the use of the server.

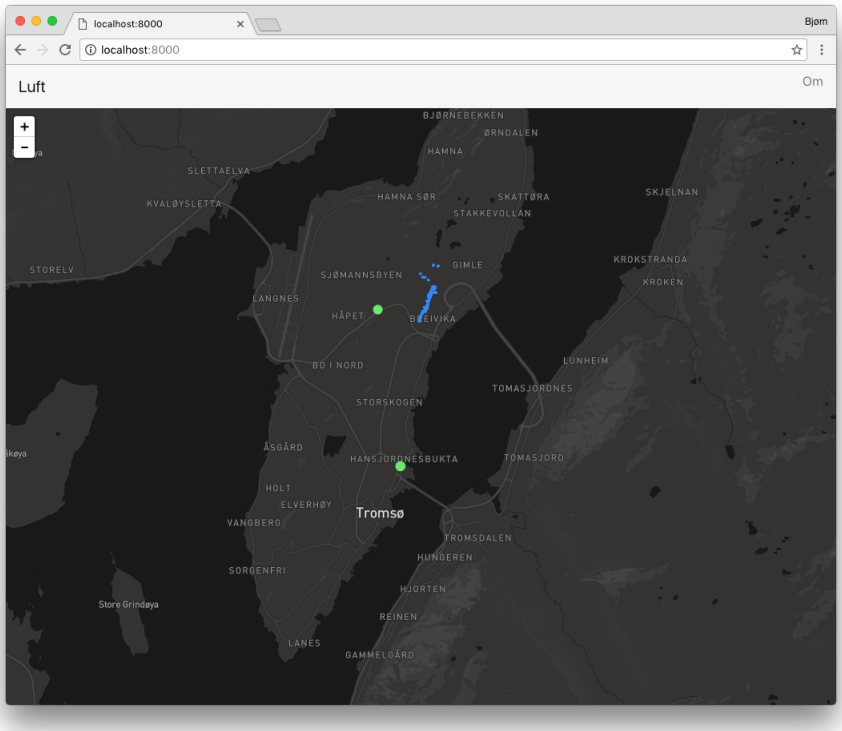


Figure 6.1: The prototype for the visualization tool that will be used in our project. The green circles indicate the two stationary air quality stations, while the blue dots are measurements from the prototype air quality sensor kit.

We therefore conclude that we have built an air quality sensor kit and a web service that is easy and affordable enough to be a part of an educational project in a high school. Even though tests show that the collected data is not as accurate as the stationary air quality stations, they will still be able to give us information about the air quality in Tromsø, at the same time as it will help the students learn about air pollution, electronics and programming.

6.1 Future work

During the spring of 2017, the participating high school class will do their project on air pollution and use the air quality sensor kits and the web service to collect and upload data. By then we will have created a visualization tool that can visualize the collected data on the web application. Our plan is to let the user specify a geographical location and a time frame, retrieve the results and visualize them as shown by the prototype in figure 6.1.

We will also look more into using Amazon Web Service and S3 instead of Heroku, as this is a cheaper storage alternative.

We want to create a development kit containing equipment for the air quality sensor, instructions on how to build and program the Arduino and access key to the web server. We want this kit to be available to all high schools in Tromsø, so that more students can learn more about programming and air pollution. We would also like it to be available to high schools in other parts of Norway, so that they can collect data from their surroundings.

As more people join the project, we have a vision to create a database similar to CrowdSignal.io, where participants of the project can buy in on the web service and access data from all over the country.

Bibliography

[1] World Health Organization, "World Health Assembly closes, passing resolutions on air pollution and epilepsy." http://www.who.int/mediacentre/news/releases/2015/wha-26-may-2015/en, 2015. [Online; accessed 2016-12-11].

[2] National Geographic, "Air Pollution." http://environment.nationalgeographic.com/environment/global-warming/pollution-overview. [Online; accessed 2016-12-11].

[3] Luftkvalitet.info, "Luftforurensing." http://www.luftkvalitet.info/Theme.aspx?ThemeID=6fc2e3cd-424f-4c03-ad0c-2b9c15369cd9. [Online; accessed 2016-12-04].

[4] NILU - Norwegian Institute for Air Research, "About NILU." http://www.nilu.no/OmNILU/tabid/67/language/en-GB/Default.aspx. [Online; accessed 2016-12-04].

[5] CITI-SENSE, "CITI-SENSE CITIZENS' OBSERVATORIES AND WHAT THEY CAN DO FOR YOU." http://www.citi-sense.eu/Portals/106/Documents/Dissemination%20material/CITI-SENSE%20selected%20products%20information%20material_15032016.pdf. [Online; accessed 2016-12-14].

[6] D. Bratvold, "What is Crowdsourcing?." https://dailycrowdsource.com/training/crowdsourcing/what-is-crowdsourcing. [Online; accessed 2016-12-04].

[7] Micro B3, "Ocean Sampling Day." https://www.microb3.eu/osd, 2015. [Online; accessed 2016-12-07].

[8] Micro B3, "MyOSD." https://www.microb3.eu/myosd, 2015. [Online; accessed 2016-12-07].

[9] Micro B3, "Deliverable No 1.10: OSD and MyOSD 2015." https://www.

microb3.eu/sites/default/files/deliverables/MB3_D1_10_PU.pdf. [Online; accessed 2016-12-07].

[10] Micro B3, "How to join MyOSD?." https://www.microb3.eu/myosd/how-join-myosd. [Online; accessed 2016-12-07].

[11] LinkedIn, "AlgoSnap Inc." https://www.linkedin.com/company/algosnap-inc. [Online; accessed 2016-12-09].

[12] CrowdSignals.io, "About Crowdsignals." http://crowdsignals.io/. [Online; accessed 2016-12-07].

[13] CrowdSignals.io, "FAQ." http://crowdsignals.io/. [Online; accessed 2016-12-07].

[14] Arduino, "What is Arduino?." https://www.arduino.cc/en/Guide/Introduction. [Online; accessed 2016-12-14].

[15] B. Bourque, "Arduino vs. Raspberry Pi: Mortal enemies, or best friends?." http://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/, 2015. [Online; accessed 2016-12-09].

[16] Society of Robots, "Introduction to Microcontrollers." http://www.societyofrobots.com/microcontroller_tutorial.shtml. [Online; accessed 2016-12-09].

[17] Arduino, "Sharp Dust Sensor GP2Y1010AU." http://www.arduino.org/learning/tutorials/boards-tutorials/sharp-dust-sensor-gp2y1010au. [Online; accessed 2016-12-10].

[18] C. Nafis, "Air Quality Monitoring." http://www.howmuchsnow.com/arduino/airquality/, 2012. [Online; accessed 2016-12-10].

[19] Heroku, "About Heroku." https://www.heroku.com/about. [Online; accessed 2016-12-10].

[20] Heroku Dev Center, "Dynos and the Dyno Manager - Isolation and Security." https://devcenter.heroku.com/articles/dynos#isolation-and-security. [Online; accessed 2016-12-10].

[21] Sharp, "GP2Y1010AU0F Compact Optical Dust Sensor." https://www.sparkfun.com/datasheets/Sensors/gp2y1010au_e.pdf. [Online; accessed 2016-12-13].

Appendices

A

The web application described in this report can be found at https://luftprosjekttromso.herokuapp.com/

The source code for both the air quality sensor kit and the web application can be found at https://github.com/ninaangelvik/luftprosjekttromso