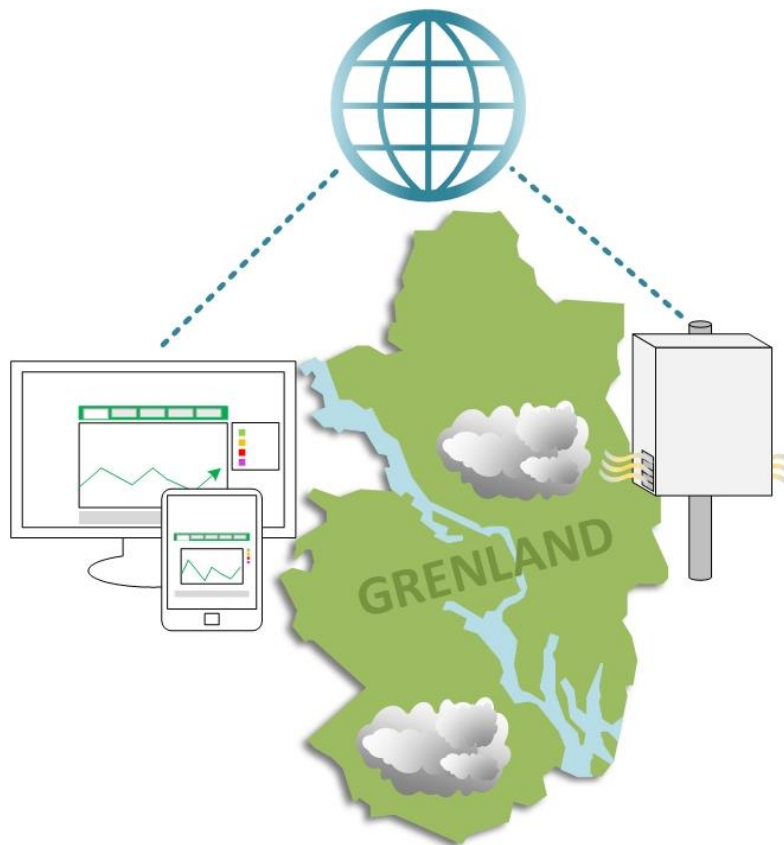


PRH612 Project 2017

Information Management System for Environmental and Public Health Information



IA6-3-17

Course: PRH612 Project, 2017

Title: Information Management System for Environmental and Public Health Information.

This report forms part of the basis for assessing the student's performance in the course.

Project group: IA6-3-17

Availability: Open

Group participants:

Hans-Martin L. Kristensen
Henrik Mølmen
Joakim Johansson
Kjetil Berg Skjelbred
Thomas Prestvik

Supervisor:

Hans-Petter Halvorsen

Project partner:

Porsgrunn municipality, Tel-Tek, Telemark Hospital-
Department of Occupational Medicine

Approved for archiving: _____

Summary:

This bachelor thesis has been written to answer the assignment "Information management system for environmental and public health information" given within "PRH612 Bachelor's Thesis", a sixth semester course at the University College of Southeast Norway.

The assignment description lists several different subjects, so a primary goal was established to define a suitable scope. In short, the goal was to develop a complete system for automatic retrieval of air pollutant data and presentation of the data in an informative manner. A mobile measuring station was also developed.

The web application was developed in Visual Studio using ASP.Net Core 1.1 framework. MS SQL Server was used for database management. The mobile measuring station contains several different components. The Arduino programming was done in Arduino IDE using C++. Visual Studio with C# was used for programming the Raspberry Pi.

The bachelor thesis, developed web application, and built Mobile Measuring Station Prototype adds up to a solution that meets the primary goal defined at the start of the bachelor project. A low-cost MMSP has been designed and built. The MMSP transfers data automatically to the web application. The web application presents data in an informative and user-friendly manner. Cost estimates of local web application hosting compared to a cloud solution have been performed. The difference in cost is low, but the chance of unforeseen expenses is greater for the local solution.

The University College of Southeast Norway takes no responsibility for the results and conclusions in this student report.

Preface

This bachelor thesis is written by students from the computer science and industrial automation study program at the faculty of Technology, Natural Sciences and Maritime Sciences. The thesis is written during the sixth semester. The content of this report along with the developed web application and mobile measuring station prototype, hereby referred to as MMSP, will be handed over to our faculty and the project partners at the end of this project. The bachelor group hopes that the work that is done can be used, and further developed in future projects.

The thesis documents the planning, design and work done on a web application and MMSP. The web application will deliver information to the public about local pollution data, and can be found at <http://luftforurensing.azurewebsites.net/>. The MMSP is built to show the concept of how a mobile measuring station may be built using microsensors. User manual, electrical drawing and datasheets are all attached to the thesis. The figure on the frontpage was designed by the group.

Tools and software used during the project includes the following: MS Office Suite, Visual Studio, MS SQL Server, MS Project, MS Visio, Fritzing, Google SketchUp, Arduino IDE and Windows 10 IoT. The bachelor's thesis assignment, project goals, WBS and Gantt chart are all attached to the thesis as respectively appendix A, appendix B, appendix C and appendix D.

All program code is stored on a memory stick, this is attached to the thesis.

Log in information for the administrator on the website is as follows:

Username: admin

Password: passord1

To get a full understanding of the thesis, basic knowledge of electrical circuits and programming is an advantage.

The bachelor group would like to express their gratitude towards the project supervisor Hans Petter Halvorsen for all guidance and help. The group would also like to thank the project partners for their guidance and feedback during work on the thesis, especially Tonje W. Thomassen (Tel-Tek) and Børge Iversen (Porsgrunn municipality).

Porsgrunn, 11.05.2017

Nomenclature

ADC - Analog to Digital Converter

API - Application Programming Interface

APN - Access Point Name. Gateway between a GSM, GPRS, 3G or 4G and another computer network

Arduino Uno - Programmable microcontroller with I/O module

ASP.NET Core 1.1 - Programming framework

CSS - Cascading Style Sheets

GPIO - General-Purpose Input/Output

GPRS - General Package Radio Service

GSM - Global system for Mobile communication

HTML - HyperText Markup Language

I/O - Input/Output

IoT - Internet of Things

JS - JavaScript

JSON - JavaScript Object Notation

MMSP - Mobile Measuring Station Prototype

MVC - Model View Controller

NEA - Norwegian Environment Agency

NILU - Norwegian Institute for Air Research

NO - Nitrogen monoxide

NO₂ - Nitrogen dioxide

NPRA - Norwegian Public Road Administration

OS - Operative system

PHP - Hypertext Preprocessor. Server scripting language

PM₁₀ (Particulate Matter 10) - Dust particles smaller than 10µm (micrometre)

PM_{2.5} (Particulate Matter 2.5) - Dust particles smaller than 2.5µm (micrometre)

Raspberry Pi 3 - Small single board computer with I/O module.

REST - Representational State Transfer

SO₂ - Sulphur dioxide

SSB - Statistics Norway

VSTS - Visual Studio Team Services

Contents

Preface	3
Nomenclature	4
Contents.....	5
1 ..Introduction	8
1.1 Objectives.....	8
1.2 Methods	8
1.3 Report structure.....	9
2..Problem description and planned solution	10
2.1 Problem description	10
2.1.1 <i>Local situation</i>	10
2.1.2 <i>Existing systems</i>	10
2.2 Planned solution	12
2.2.1 <i>Web application</i>	12
2.2.2 <i>Mobile measuring station for air quality</i>	12
3..Web application - Requirements and design.....	13
3.1 Web application requirements and planned solution	13
3.1.1 <i>Admin log in and data entry</i>	14
3.1.2 <i>Current measurement status and -information</i>	15
3.1.3 <i>Historic data and graphs</i>	16
3.1.4 <i>Reports and extraction of data from database</i>	16
3.1.5 <i>Security measures</i>	16
3.2 Web application design.....	17
3.2.1 <i>Design template</i>	17
3.2.2 <i>Home page</i>	18
3.2.3 <i>Measurement station page</i>	19
3.2.4 <i>Login page</i>	20
4..Mobile Measuring Station Prototype - Requirements and design	21
4.1 Governmental requirements.....	21
4.2 Sensor technology available today.....	22
4.3 Requirements	22
4.4 Prototype Design	23
4.4.1 <i>Prototype cabinet</i>	23
4.5 Controller	25
4.6 Communication.....	26
4.7 Sensors.....	26
4.8 Power source	27
4.8.1 <i>Battery</i>	27
4.8.2 <i>230 V supply</i>	27
4.9 Equipment quote.....	28
4.9.1 <i>Battery option</i>	28
4.9.2 <i>230 V option</i>	29
5..Database and data gathering	30
5.1 Database	30
5.1.1 <i>Original database</i>	30

5.1.2 <i>Current database</i>	31
5.2 Data gathering.....	32
5.2.1 <i>Air pollution</i>	32
5.2.2 <i>Air pollution data gathering program</i>	33
5.2.3 <i>Data on traffic</i>	33
5.2.4 <i>Water pollution</i>	34
5.2.5 <i>Soil pollution</i>	34
5.2.6 <i>Industrial pollution</i>	34
5.3 Local- vs. cloud hosting.....	35
5.3.1 <i>Local hosting</i>	35
5.3.2 <i>Cloud hosting</i>	35
5.3.3 <i>Cost comparison</i>	35
6..Web application - Solution	38
6.1 Web application - Architecture and design.....	38
6.1.1 <i>Web application - Final design</i>	38
6.1.2 <i>Dynamic web application</i>	39
6.1.3 <i>Responsive design</i>	40
6.2 Choosing and implementing a graph for the web application	42
6.2.1 <i>Choosing a graph solution</i>	42
6.2.2 <i>Implementing the graph</i>	43
6.3 Choosing and implementing a map for the web application	44
6.3.1 <i>Choosing a map solution</i>	44
6.3.2 <i>Implementing the map</i>	45
6.4 Information about miscellaneous code on the web application	47
6.4.1 <i>Data from the MMSP</i>	47
6.4.2 <i>Login for the web application</i>	48
6.4.3 <i>Upload data</i>	50
7..Mobile Measuring Station Prototype - Solution.....	51
7.1 Prototype concept	52
7.2 Technology choice	52
7.2.1 <i>Microcontrollers</i>	54
7.2.2 <i>Data Transmission</i>	54
7.2.3 <i>Sensors</i>	55
7.2.4 <i>Power source</i>	59
7.3 Wiring and built solution.....	60
7.4 Code and Software Solution	61
7.4.1 <i>Raspberry Pi</i>	61
7.4.2 <i>Arduino</i>	62
7.5 Field testing of performance and durability.....	63
7.6 Maintenance requirement	66
7.7 End-user functionality.....	67
8..Discussion	69
8.1 Future Work.....	70
8.1.1 <i>Improvements on the MMSP's air channel</i>	70
8.1.2 <i>MMSP mounting solution</i>	70
8.1.3 <i>Sensor calibration</i>	70
8.1.4 <i>MMSP power source</i>	71
8.1.5 <i>MMSP memory</i>	71
8.1.6 <i>GPS module</i>	71
8.1.7 <i>Advised solutions for MMSP</i>	71
8.1.8 <i>Indication for amount of times the limit values have been exceeded</i>	73
8.1.9 <i>Downloading automatically generated reports from the web application</i>	73
8.1.10 <i>Rerouting from the home page map</i>	73

8.1.11	<i>Data from NRPA</i>	74
8.1.12	<i>Alert function - power failure in the MMSP</i>	74
9	Summary	75
	References	76
	Appendices	78

1 Introduction

This bachelor thesis will be written as an answer to the assignment “Information Management System for Environmental and Public Health Information” given within “PRH612 Bachelor’s Thesis”, a sixth semester course at The University College of Southeast Norway. The assignment is given by HSN in cooperation with Tel-Tek, Porsgrunn municipality and Telemark hospital-Department of Occupational Medicine as external partners.

Today's increased focus on the environment, forces the government and the local municipalities to increase their focus on local pollution. There is a lot of environmental data available, from both local and national sources. Although most of the data is currently available online, it is spread across different websites and a good overview of all data is hard to achieve. To remedy this, Porsgrunn municipality have asked Tel-Tek to develop an information management system for presenting local pollution data. This system should be useful for members of the public as well as representatives from the municipality and local industry. To develop the technical aspects of the system, such as a database, web application and measurement systems, Tel-Tek have acquired the help of students and faculty members from the department of electrical engineering, IT and cybernetics, at the faculty of Technology, Natural Sciences and Maritime Sciences at HSN Campus Porsgrunn. In the fall of 2016, a group of master students were tasked with developing a database and a web application for use in the system. This database will be used in this project, but some necessary changes will be made.

1.1 Objectives

The assigned bachelor’s thesis is attached to the report as Appendix A. It lists several different subjects that must be addressed, researched, and developed in order to deliver a complete information management system. It is not feasible to cover all the subjects in the time frame given for this thesis, so one of the first tasks for the bachelor group is to decide on a suitable scope for the project. To help define the scope, the group has set the following primary goal:

Develop a complete system for automatic data retrieval of environmental air pollutants and presentation of the data in an informative and easy to grasp manner. The system should contain a web page for displaying information, a database for storing data, and measuring stations for data gathering. A mobile measuring station will be developed. Automatic data gathering from the measuring stations should be the preferred solution.

This project will consider cost aspects of data and web application hosting and maintenance, as well as the cost of building the mobile measuring station prototype.

1.2 Methods

Different methods will be used while working on the thesis. Development of the web application will be done in Visual Studio using the ASP.NET Core 1.1 framework. MS SQL Server is used for modifying and maintaining the database. Arduino programming will be done in Arduino IDE using the C++ programming language. Windows IoT is used as OS on

1 Introduction

the Raspberry Pi. GUI and other necessary programming for the Raspberry Pi will be done in Visual Studio using the C# programming language. Google SketchUp is used for 3D modelling of the prototype. Fritzing will be used to make electrical drawings. Microsoft Azure is used for hosting of the web application and database. For project management purposes, a WBS document will be developed in MS Visio and a Gantt chart will be developed in MS Project. Visual Studio Team Services will be used to help with project management, source code control and team collaboration (scrum).

1.3 Report structure

Chapter 2 gives a brief problem description and a short summary of the planned solution.

Chapter 3 and Chapter 4 gives a description of the planned web application and MMSP. Requirements, proposed solutions, and design is described in different subchapters.

Chapter 5 gives a description of the database and information about how data is gathered.

Chapter 6 contains detailed information about the web application.

Chapter 7 has details about the built MMSP.

Chapter 8 is a discussion chapter where different aspects of the system will be discussed, as well as suggestions for work to be done by other groups in the future.

Chapter 9 gives a summary of the project.

2 Problem description and planned solution

This chapter gives an introduction regarding the need for the information management system, an overview of the planned solution will also be given in this chapter. The planned solution is described in detail in chapter 3 and 4.

2.1 Problem description

Global warming is a threat to everyday life. All countries and levels of government must do their part to slow down, or stop emissions of greenhouse gasses.

Environmental data from both local and national sources are available, most of it online, but it is spread across different websites and databases. This means that obtaining a complete overview of the local situation is both difficult and time consuming.

In the regulations relating to pollution control (Pollution regulation) part 3 [1], issued by The Royal Norwegian Ministry of Climate and Environment, it is stated that all municipalities in Norway are required to measure local air quality and make this information available to the public.

2.1.1 Local situation

Grenland is a district in the south-eastern part of Telemark county, composed of the municipalities Bamble, Drangedal, Kragerø, Porsgrunn, Siljan, and Skien. The area has been an industrial centre since Norsk Hydro AS built their first factory at Herøya in the late 1920s. Several large companies such as Yara, Eramet, Norcem, Ineos and Noretyl AS have factories in Grenland. Emissions from the local industry made Grenland a highly-polluted area during the 1970s [2]. Rules, regulations, and efforts have since then reduced the environmental impact of emissions from the local industries. However, the local fjords still have challenges with polluted sediments on the soft bottom [3] [4]. The fjords in Grenland are closely monitored by several institutions, and several reports from the recent years are available. The air quality in Grenland has reached critical levels several times in the recent years [5] [6]. This shows the need for continuous measuring of the areas air quality.

2.1.2 Existing systems

There are several websites that gives the user information about pollution data. These are national websites, and contains information for all of Norway.

NILU, the Norwegian Institute for Air Research has developed <http://luftkvalitet.info/> on behalf of the Norwegian Public Roads Administration and the Norwegian Environment Agency. NILU is also responsible for hosting and maintaining the website. This website offers the users a real-time overview of the air quality in Norway. It is also possible for users to get more detailed information about a single measurement station and what kind of pollutants the station measures. Figure 2.1 shows the national status window with colour symbols that displays the current status of an area.

2 Problem description and planned solution

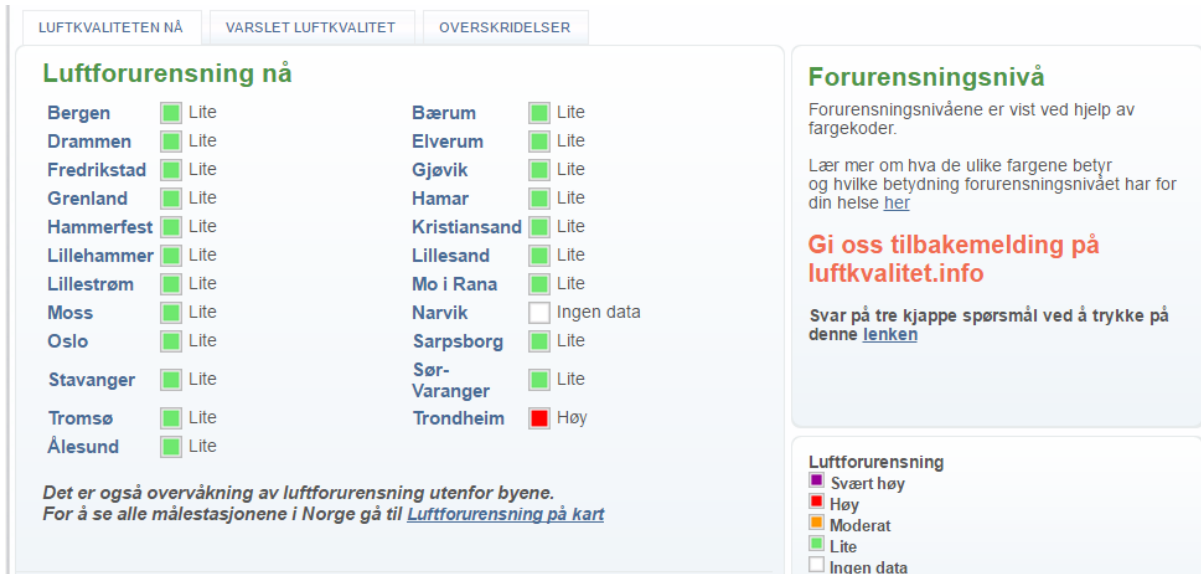


Figure 2.1 luftkvalitet.info national status window [7]

The site <http://www.norskutslipp.no> gathers data from industries, counties, Statistics Norway (SSB), and the Norwegian Environment Agency. The site offers information on several different kinds of pollution data. Under the section for land-based industries, the information shown can be limited down to local industries by choosing county and municipality. This is shown in Figure 2.2 below.

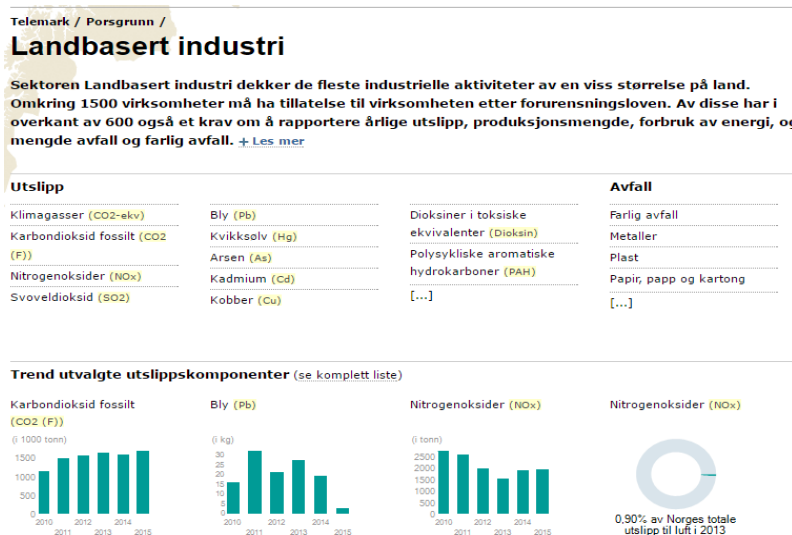


Figure 2.2 Information on local land-based industries [8]

These two websites are just some of the many sites that offers information on local and national pollutant data.

2.2 Planned solution

To make it easier for politicians, scientists and the public to get an overview, Porsgrunn municipality wants to develop an information management system where all the local pollutant data is gathered and presented in a manner that is easy to grasp. Different pollutant data such as air quality measurements, sea pollution, road traffic and emission reports from industries will be collected in one database. A web application for presentation of the collected data must be created. The web application must be designed to enable users to get an overview of the current pollution status and enable users to print reports (graphs) on the current situation. A prototype of a mobile measuring station will also be designed and built.

2.2.1 Web application

A web application will be developed to present the data from the database to whoever wishes to use the available information. The focus will be to develop a web application which is user-friendly. Further details on the requirements and planned design of the web application will be given in chapter 3.

2.2.2 Mobile measuring station for air quality

Measuring stations currently in use for air quality are not mobile, because of their size. To show that it is possible to make the task of measuring air quality easier and cheaper, a prototype of a mobile measuring station will be developed. The focus of the prototype is to show the concept of how a mobile measuring station will work, and not necessarily the quality of the measurements, as high-quality microsensors are too expensive for use in a prototype. Details about the design and requirements of the build is given in chapter 4.

3 Web application - Requirements and design

This chapter gives the reader an overview of the project partners' expectations for the web application, as well as the design concepts suggested by the group for the web application.

3.1 Web application requirements and planned solution

Based on the project partners' expectations and requirements, as well as the groups suggestions, the following features described in this subchapter is needed to develop a functional and informative web application. The web application will display current and historic data on pollution in Grenland. Current data will be displayed visually with specific colours depending on the level of pollutants in the air. This lets the users get a quick overview of the current air quality. Historic data will be displayed both visually in a graph and a table. It will be possible to log in as an administrator, this lets the administrator edit and add information about measurement stations and pollutants. Figure 3.1 shows a use case diagram of the planned web application.

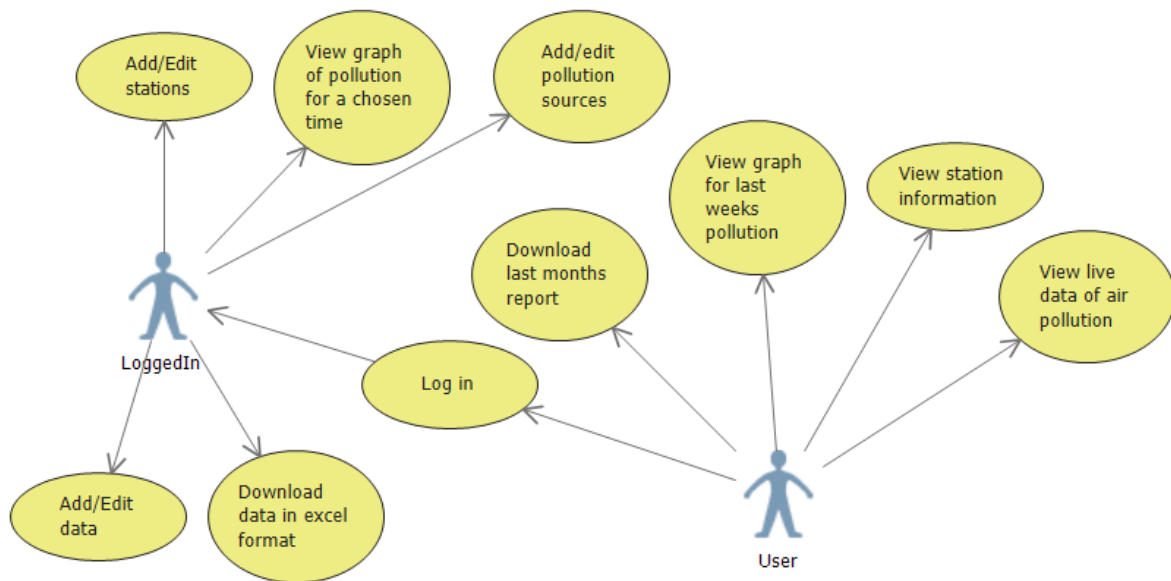


Figure 3.1 Use case diagram for the planned web application

3 Web application - Requirements and design

3.1.1 Admin log in and data entry

To edit information and add new measurement stations, a login function for admins will be created. This will be done using ASP.NET Core Identity. Identity is a tool that gives the users two options: Create a new account specific to this web application, or log in using Facebook, Google, or other third-party options. To restrict admin access to only people working in/with Porsgrunn municipality, new users will need to apply for admin rights and an existing admin needs to approve them. Figure 3.2 shows the planned sequence diagram for user login.

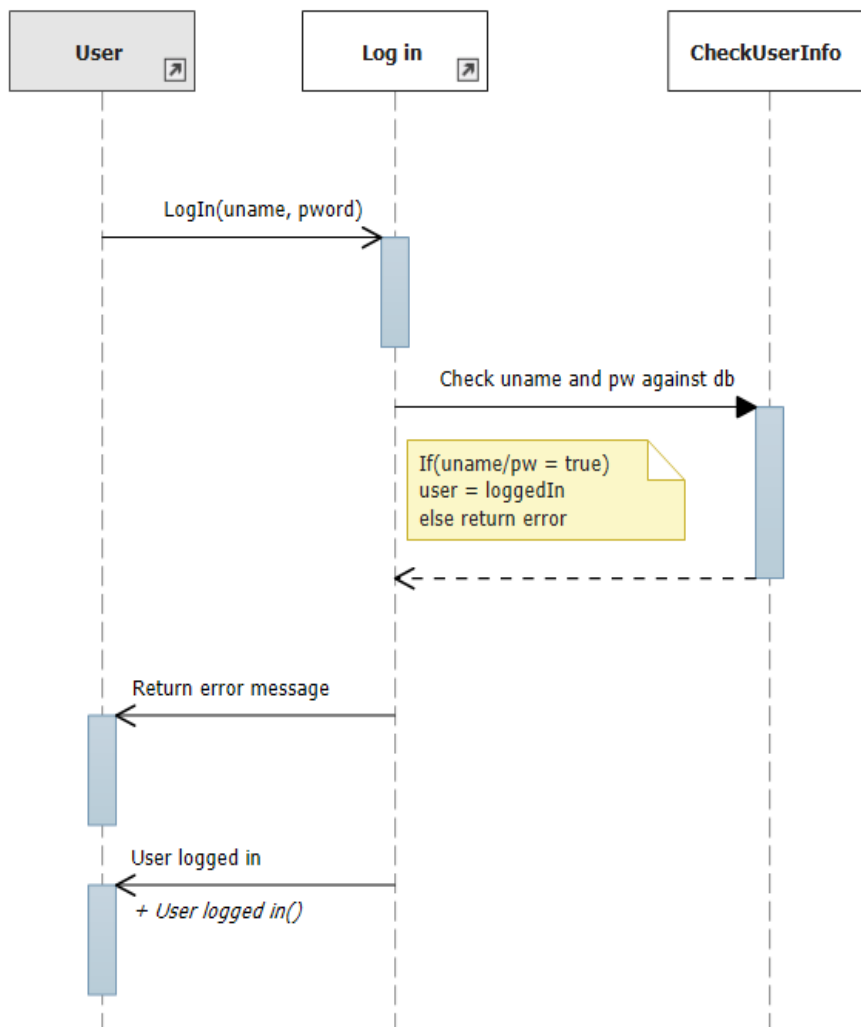


Figure 3.2 Sequence diagram user login

3 Web application - Requirements and design

Admins will also be able to add new data from various pollution sources. This data needs to be in an Excel file and will require the sheet to have a predefined format. Figure 3.3 is a sequence diagram showing the planned upload function.

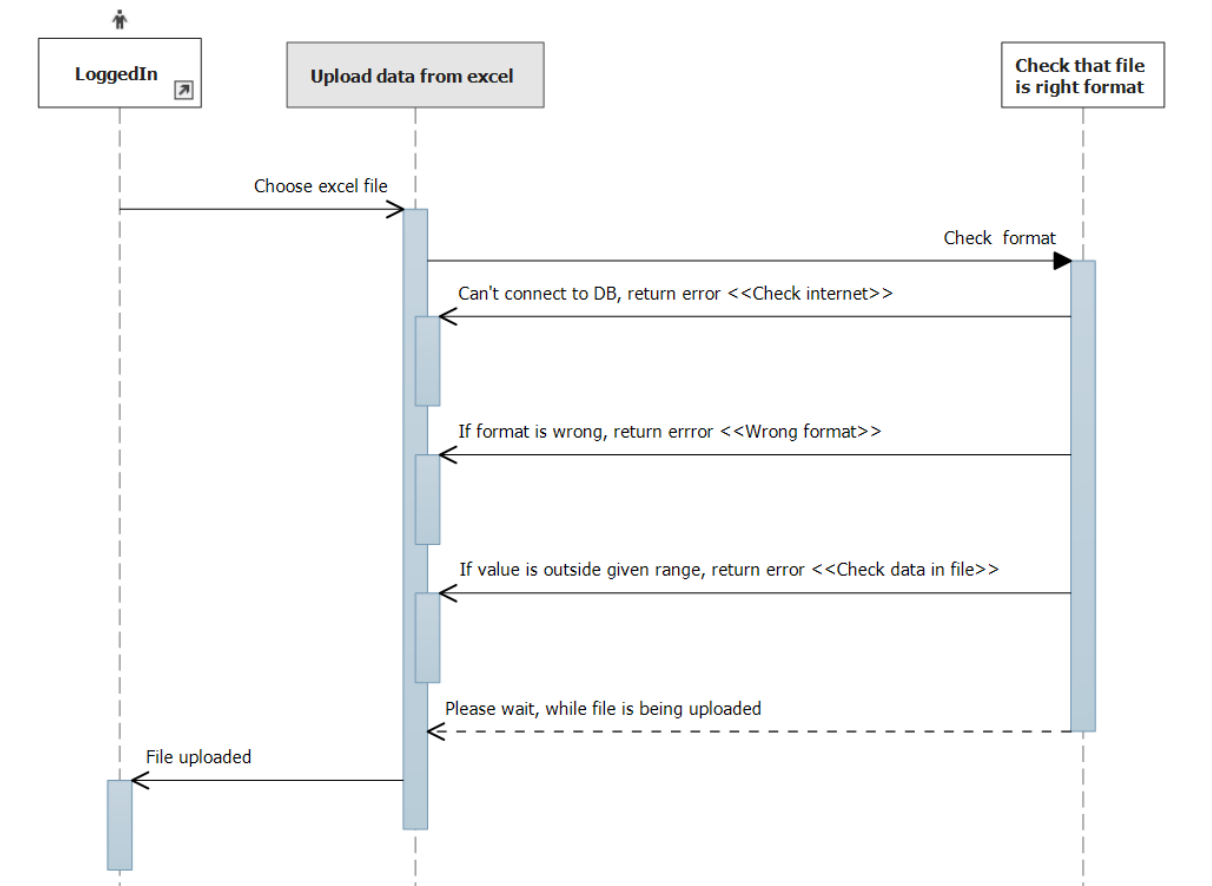


Figure 3.3 Sequence diagram file upload

3.1.2 Current measurement status and -information

The web application will have live updates of the current air pollution around the different measurement stations. This will allow users to get a quick overview of the area surrounding the measurement station. These live updates will be updated using information from NILU's REST API, which updates once per hour.

Each station will get a dedicated page (view) where more detailed information can be found. Examples of more detailed information would be a map showing the exact location of the station, a list of pollutants the station measures and their current value, together with a graph showing pollution data from the last 7 days. The user will also be able to hover over each pollutant and get more information about it.

3 Web application - Requirements and design

3.1.3 Historic data and graphs

To be able to provide users with information, gathering of historic data is necessary. This data along with live data will be displayed in graphs.

The historic data must be gathered manually from the relevant databases. To ease the work of adding this data to the projects database, a method to upload data will be created. This method will also be used to enable admin users to enter data into the database via the web application.

A well-functioning graph solution is an effective way of displaying information to users of the web application. Different graph solutions must be evaluated based on functionality and cost. The selected graph should show historic data, it should also be possible to manually select the timespan of displayed data. If there are more than one set of data shown on the graph at the same time, the user should be able to select or deselect the different data.

Graphs for comparison and other features for displaying information can be added if there is time and resources to implement these. One example of this is a table showing the amount of times a limit value were exceeded, on a monthly or yearly basis.

3.1.4 Reports and extraction of data from database

Admin users should have the possibility to extract data in the form of reports. The timespan of these reports must either be predefined or selected by the user. If time and resources are available, a function where users can generate a pdf document with charts and tables for a chosen timespan should be implemented.

3.1.5 Security measures

The connection string to the database is saved server side. This ensures that users on the client side never will get access to it.

Entity framework will be used to get information from the database. This prevents the user from seeing the SQL queries, which makes the database even more secure.

3.2 Web application design

Concept drawings for the webpages are presented in this subchapter.

3.2.1 Design template

Figure 3.4 illustrates the template design for the web application.



Figure 3.4 Web application template

3.2.2 Home page

Figure 3.5 illustrates the concept for the web application layout and how the home page will be designed.



Figure 3.5 Home page design

3 Web application - Requirements and design

3.2.3 Measurement station page

Each measurement station will have its own page that displays measurement data and general information about the station, see Figure 3.6 for the planned layout and features for the station pages.

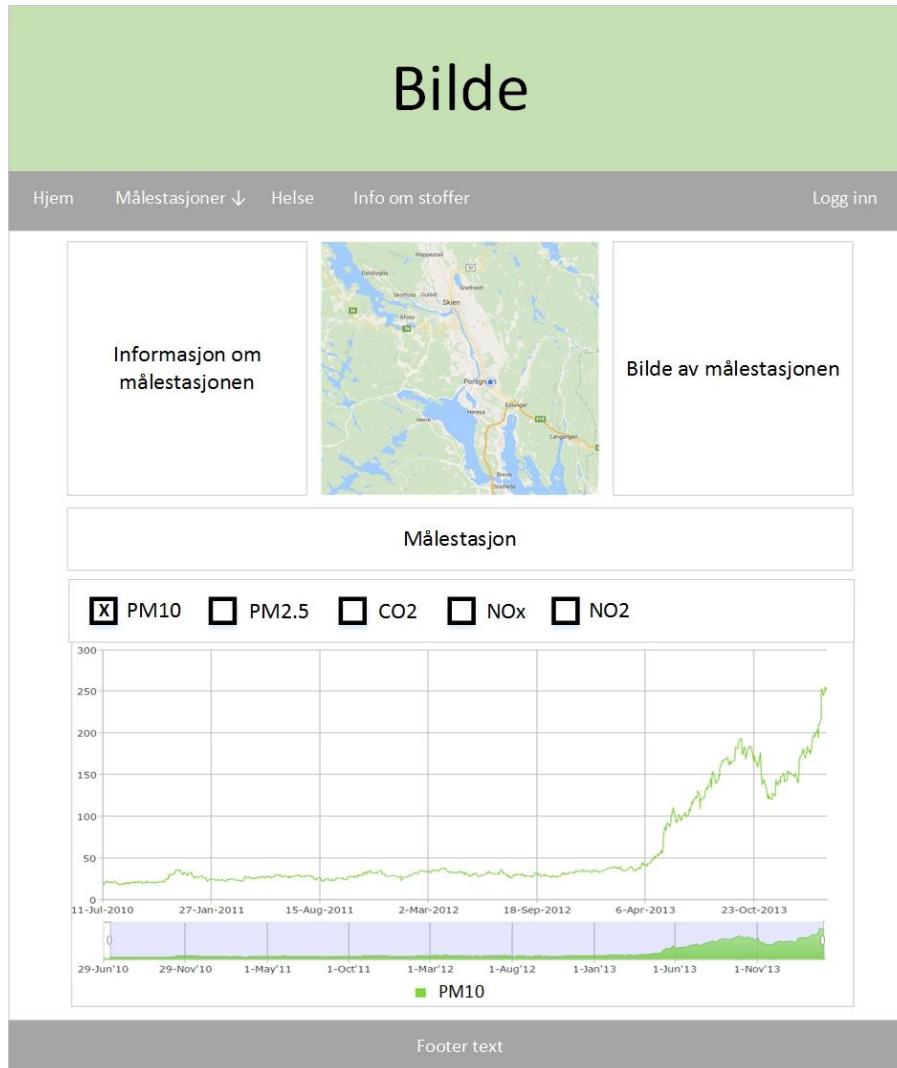


Figure 3.6 Measurement station page

3 Web application - Requirements and design

3.2.4 Login page

The planned design for the login function is shown in Figure 3.7.

Bilde

Hjem Målestasjoner ↓ Helse Info om stoffer Logg inn

Epost:

Passord:

Logg inn

Footer text

Figure 3.7 Login page for authorized users

4 Mobile Measuring Station Prototype - Requirements and design

This chapter focuses on the development of the MMSP. The idea behind this prototype is to create a platform for transferring measurement data automatically from a remote measuring station to the web application. The technology for this is already on the market, but these pre-fabricated units often have a relatively high cost, and many of the older systems are often installed in immobile sheds or containers. This project will focus on a cost-effective measuring station that is easy to implement into the information management system. It should be easy to expand and implement more stations, and use microsensors to reduce cost and size of the measuring station.

The main goal is not to retrieve accurate data, but rather to show that the concept of automatic data gathering is working and that the system is easy to implement. A crucial feature is to make it easy to replace sensors and hardware without too much technical difficulties. The use of environmental information for public and scientific purposes has a strict set of standards and rules in terms of data accuracy and reliability. Adhering to these standards will not be prioritized [9].

4.1 Governmental requirements

As mentioned in chapter 2.1, all cities and municipalities in Norway are required to gather data from pollution in their respective region. NEA, under the Norwegian government, controls the national regulations for data gathering of pollution [1].

The data is gathered and sent to NILU for quality assurance and data-processing, before it is released to the public and governmental institutions. In addition, the existing stations are required to have periodic calibration and maintenance in order to acquire accurate data.

Figure 4.1 shows all the instances involved in the process.

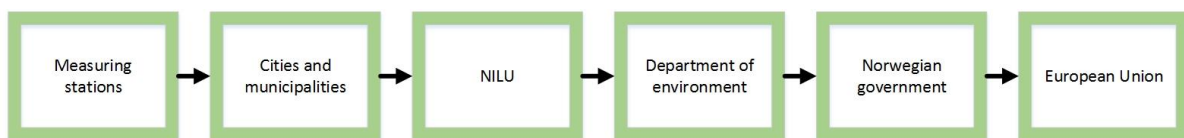


Figure 4.1 Hierarchy of the involved instances

Norway is also committed to report information about pollution and other environmental problems to the European Union. This is to get a better picture of how pollution can affect people's everyday life, and decide if measures must be taken [10].

4.2 Sensor technology available today

From NILU's seminar regarding microsensors, a lot of information on the topic of sensor technology was given. They are working on several projects, where testing of microsensor technology play an important role. It was also stated that the sensors they have been testing are getting better, and that the technology has a rapid development, but they have not been able to find a microsensor that meets all the requirements. More information from the seminar is available on NILU's web site [9].

4.3 Requirements

As mentioned in earlier chapters, the MMSP will be smaller and cheaper than existing stations. This will make it easier for the municipalities to deploy several of these measuring stations, which leads to a better overview of the air quality in the region.

The prototype should consist of several functionalities. The main functionality will be to gather data on air pollution, and several sensors will be implemented to measure the different particles and pollutants. In addition, it needs a power source and a GSM-module for transmitting measurement data to the web application. A GPS unit should also be installed to keep track of the position of the MMSP. To ease the job of placing the MMSP, a display should be installed to show the signal strength of the GSM-network at the deployment position. Figure 4.2 illustrates the MMSP's functions and data transmission principle to the web application.

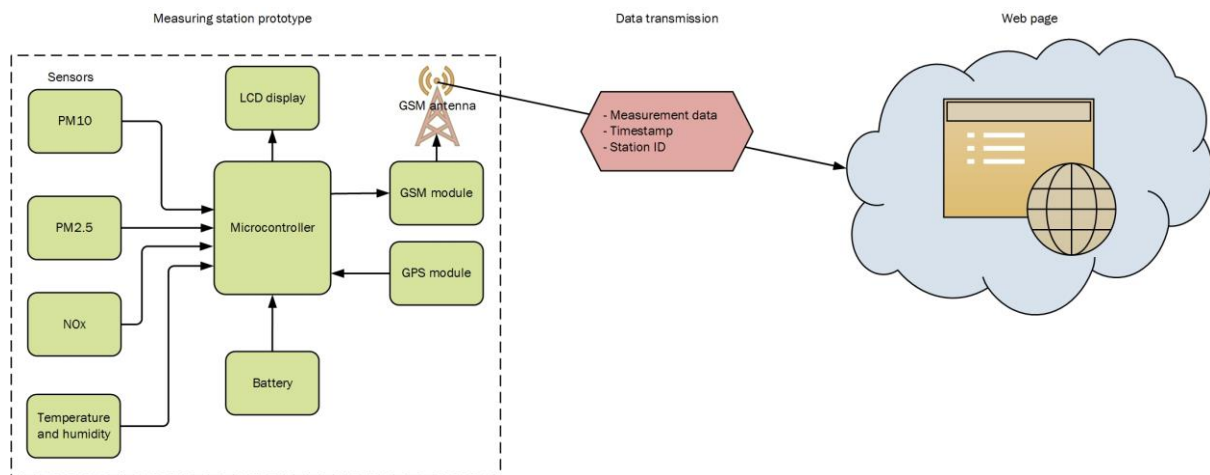


Figure 4.2 Planned system layout

4.4 Prototype Design

In terms of design and development of the prototype, both technical and practical aspects must be considered. Ideally, it should be small enough to be carried around by a person, and placed at any location that is covered by the GSM-network. It should also be possible to mount it to different objects.

There are several factors to consider in terms of reliability. In addition to the data transmission, the MMSP should also withstand environmental conditions. A decision must be made, whether the system should be powered by battery or connected to existing electrical infrastructure. Figure 4.3 shows the planned design and dimensions for the prototype.

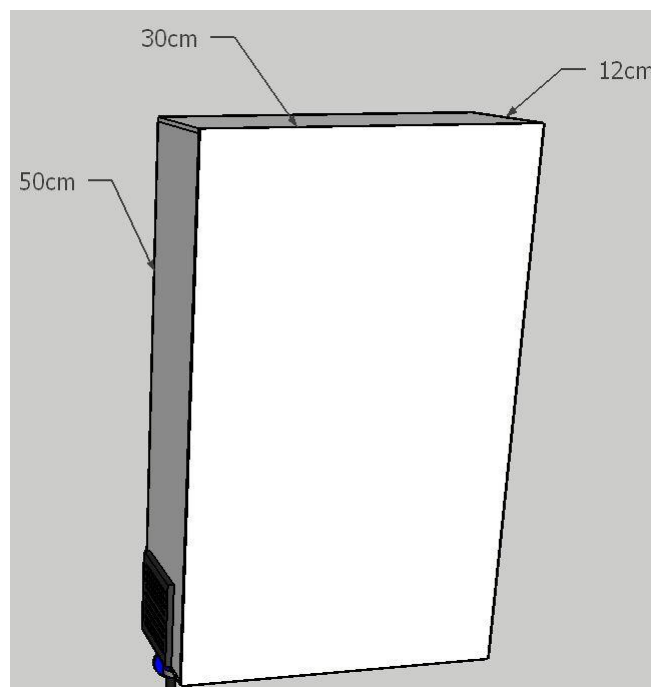


Figure 4.3 Planned design for the prototype

4.4.1 Prototype cabinet

The idea is to fit a standard plastic cabinet with the required equipment. Figure 4.4 and Figure 4.5 shows the concept drawings of the MMSP.

4 Mobile Measuring Station Prototype - Requirements and design

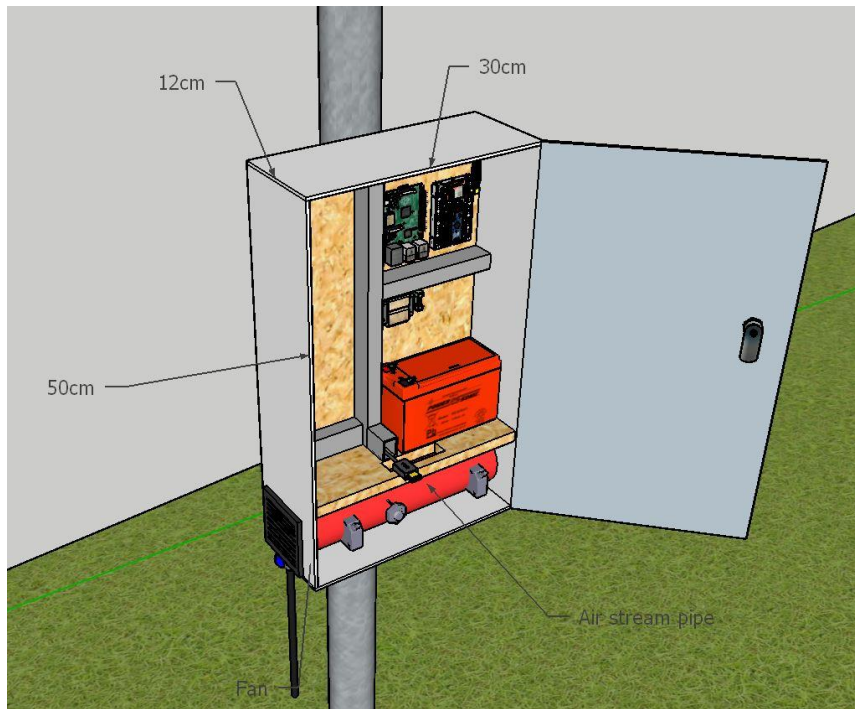


Figure 4.4 Concept sketch with Dimensions

Figure 4.4 shows the planned dimensions for the prototype, and placement of the components. The actual size of the prototype can divert from the planned dimensions.

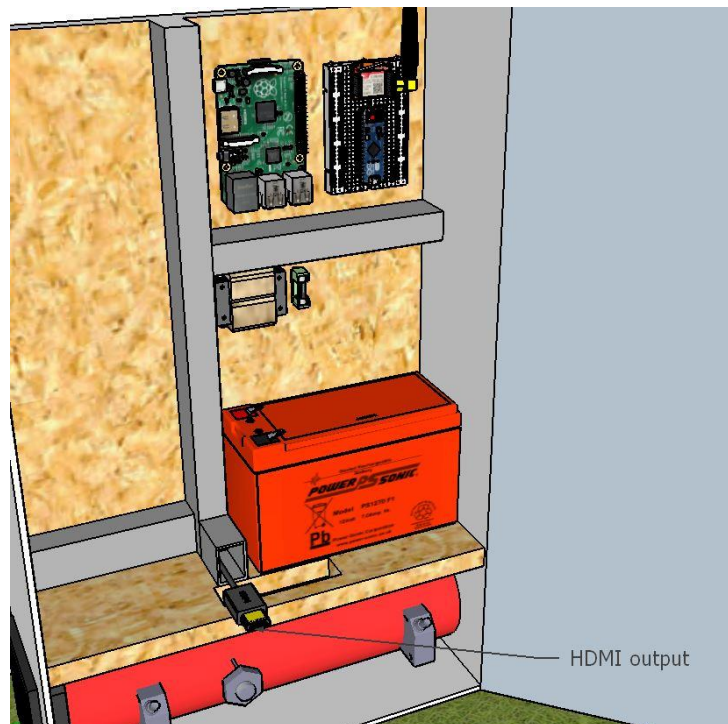


Figure 4.5 Prototype equipment with a HDMI-output to support a GUI

4.5 Controller

There are many types of microcontrollers to choose from. The choice of controller will be based on the requirements from chapter 4.3, and from the choice of sensors described in chapter 4.7. To maintain the overall goal for this project, the controller must be low-cost and compatible with different electronic devices such as a GSM-module. It must also be able to use serial communication between other controllers. All these different factors are important when choosing controllers, as well as their ability to operate in different temperatures.

Another planned feature is the ability to have a GUI that can be used for configuring the prototype. This GUI should run on a microcontroller and be presented on a device with a screen. The GUI should also be easy to use and have a simple design. The planned design for the two GUI-layouts are illustrated in Figure 4.6.

The figure shows two side-by-side screens for the 'Lilleåsen målestasjon' GUI. The left screen is a login page with fields for 'Bruker-ID' (containing 'admin') and 'Passord' (containing '*****'), and a 'Logg inn' button. The right screen is a configuration page with a header showing the time '11.15' and date '24.02.2017', a signal strength indicator, and the station name 'Lilleåsen målestasjon'. The configuration section includes:

- 'Stasjon navn' field with 'Lilleåsen'.
- 'Sendingsinterval' dropdown menu set to 'Hver time'.
- A table for sensor configurations:

	Nåverdi	Målesignal	Kalibreringsfaktor
PM ₁₀	xx.xx	4-20mA	xx.xx
PM _{2.5}	xx.xx	4-20mA	xx.xx
Temperatur	xx.xx	1-10V	xx.xx
- 'SIM-informasjon' box containing:
 - SIM-nummer: xxx xx xxx
 - Resterende saldo: 150 kr
 - SIM-leverandør: One call
- 'Endre passord og bruker-ID' and 'Logg ut' buttons.

Figure 4.6 Planned design for the prototype GUI

4.6 Communication

The technology used for transmitting data at the existing measuring stations in Grenland is GSM. Data is transferred from the sensor equipment to a data logger, then passed to the GSM-module and then to NILU's data system. Measurements are continuously gathered over the course of 1 hour, a median value is generated from these hourly measurements, which is used as the represented value.

When using the GSM- and GPRS system, a SIM card is needed to establish a communication ID for the device. The amount of data transferred must be evaluated to find a suitable network subscription that can handle this amount.

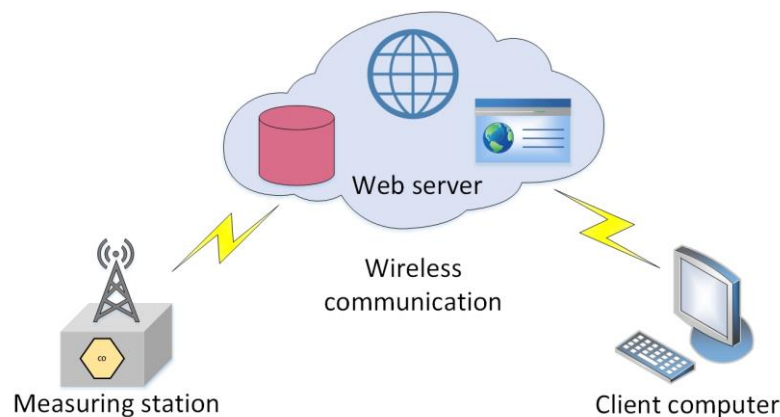


Figure 4.7 Communication principle

As illustrated in Figure 4.7, it is most convenient to use wireless communication and not be dependent on local infrastructure such as an ethernet connection. Another topic to consider is the ability to establish contact with the GSM network in terms of signal strength on site as well as the electronics' ability to send and receive data.

4.7 Sensors

The MMSP should be able to measure the same particles as the existing measuring stations, but in this project the particles to focus on would be PM_{10} , $PM_{2.5}$, NO_x . A temperature sensor will also be implemented to see if the temperature has an impact on measurements. It is important to find sensors that supports the signal input of the controller, usually 0-5 V analog or a digital signal. In addition, the sensors also needs a power input that corresponds with the controller it is connected to.

The price of the sensor may have a correlation with its accuracy. Due to funding, a low-price sensor will be used in this project. Sensors used in the existing measurement stations are expensive and too big, and are therefore not an option [11]. The focus will be to find a sensor that can be implemented with the chosen controller, and show the concept of the MMSP.

4.8 Power source

The MMSP needs an energy source to operate. The main factor when choosing a power source is how long the prototype is meant to be deployed. If the operating time is short, a battery may be enough. However, if deployed over longer periods a connection to existing electrical infrastructure will be needed.

4.8.1 Battery

A critical factor when choosing a battery, is how the changing outdoor temperature affects the discharge rate. Ideally the prototype should have low energy consuming components to ensure long operating time if powered by a battery. This limits the equipment options such as air heaters and fans, since these often have a high power consumption.

4.8.2 230 V supply

Since this option would require a connection to infrastructure, mounting the prototype to a bus shelter could be a possibility. A lot of bus shelters nowadays have lights or electric commercial signs, this means that new infrastructure is not required. Bus shelters are usually also positioned next to roads where measuring might be relevant.

4.9 Equipment quote

This subchapter gives a brief overview of the cost of parts needed for construction of the prototype. There are two build options currently being considered, the cost of the different solutions is presented in separate tables.

4.9.1 Battery option

By using a battery as the power source, the prototype is not depended on any infrastructure to work. An important issue to consider is the battery size compared to price and endurance, as well as the maintenance and lifespan. Table 4.1 shows the estimated cost of the components needed for the battery solution.

Table 4.1 Cost of components for battery option

Equipment	Cost estimate [NOK]
Raspberry Pi 3	420
Arduino MKR1000	720
Breadboard	20
Sensor	340
Battery	1400
GSM module	270
Transformer (12-5V)	40
Cabinet	670
Wiring	100
Fuse	60
Cable gland	30
Plug	30
Fan	250
Total cost	4350

4 Mobile Measuring Station Prototype - Requirements and design

4.9.2 230 V option

This solution would require electrical infrastructure where the prototype is mounted. This would be best as a long-term solution. A 12 VDC power supply would be needed, and a cable for connecting it to an electric socket. Another advantage is that it takes up less space, because no battery is needed. Table 4.2 shows the cost of the components needed for the 230 V option.

Table 4.2 Cost of components for the 230 V option

Equipment	Cost estimate [NOK]
Raspberry Pi 3	420
Arduino MKR1000	720
Breadboard	20
Sensor	340
Power supply	830
GSM module	270
Transformer (12-5V)	40
Cabinet	670
Wiring	100
Fuse	60
Cable gland	30
Plug	30
Fan	250
Total cost	3780

5 Database and data gathering

This chapter contains information about the database designed in this project, as well as the different sources used for data gathering.

5.1 Database

As mentioned in chapter 1, the base of this project was to make it easier for the public and the people working with environmental issues in Grenland, to easily find the information they need on one web application. This can be achieved by receiving information from different external databases through APIs. Doing this would achieve the same result as having a local database. However, if these external databases were to shut down or stop working, the web application would be directly affected and historic data could potentially be lost. To avoid this problem, a local database has been designed and created to store information about different pollutants.

5.1.1 Original database

The database was originally designed and created by the group of master students mentioned in chapter 1. Figure 5.1 Shows an overview of the original database.

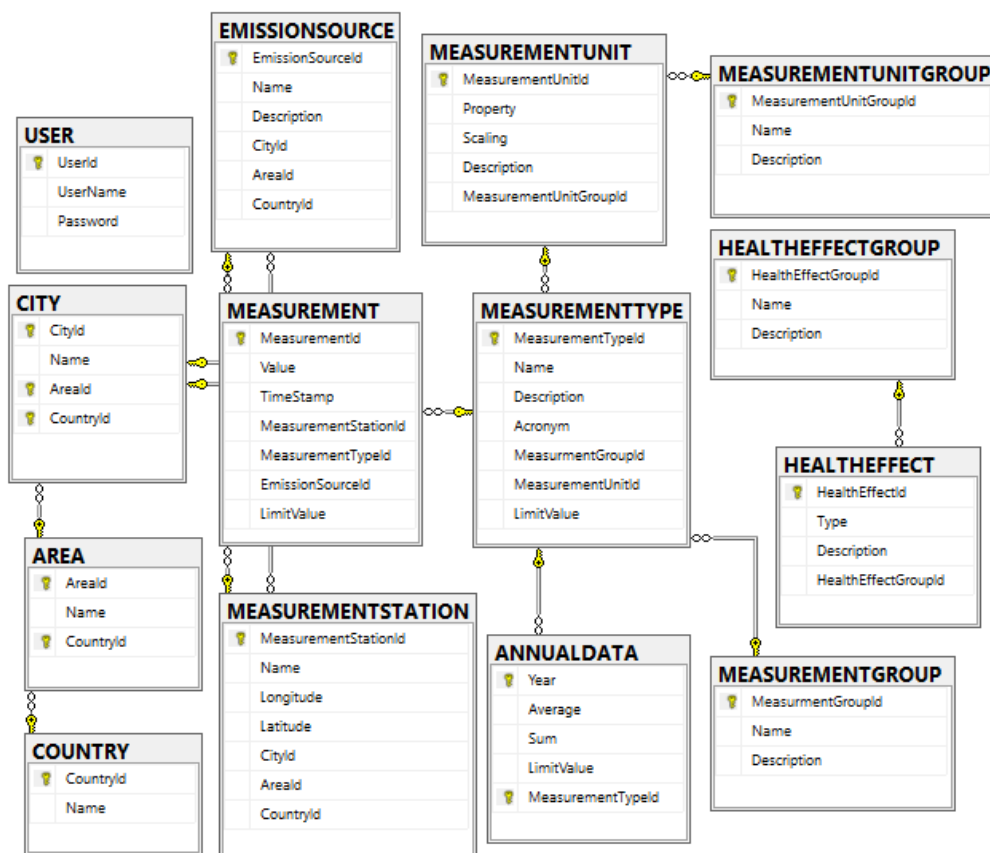


Figure 5.1 ER diagram of the original database

5 Database and data gathering

More details on the original database can be found in the report written by the master group responsible for developing the original database [12].

5.1.2 Current database

To get the necessary functionality from the database, the solution was to redesign certain elements of the database. 9 new tables had to be added to ensure that the database works as intended. Figure 5.2 shows the current database after the redesign.

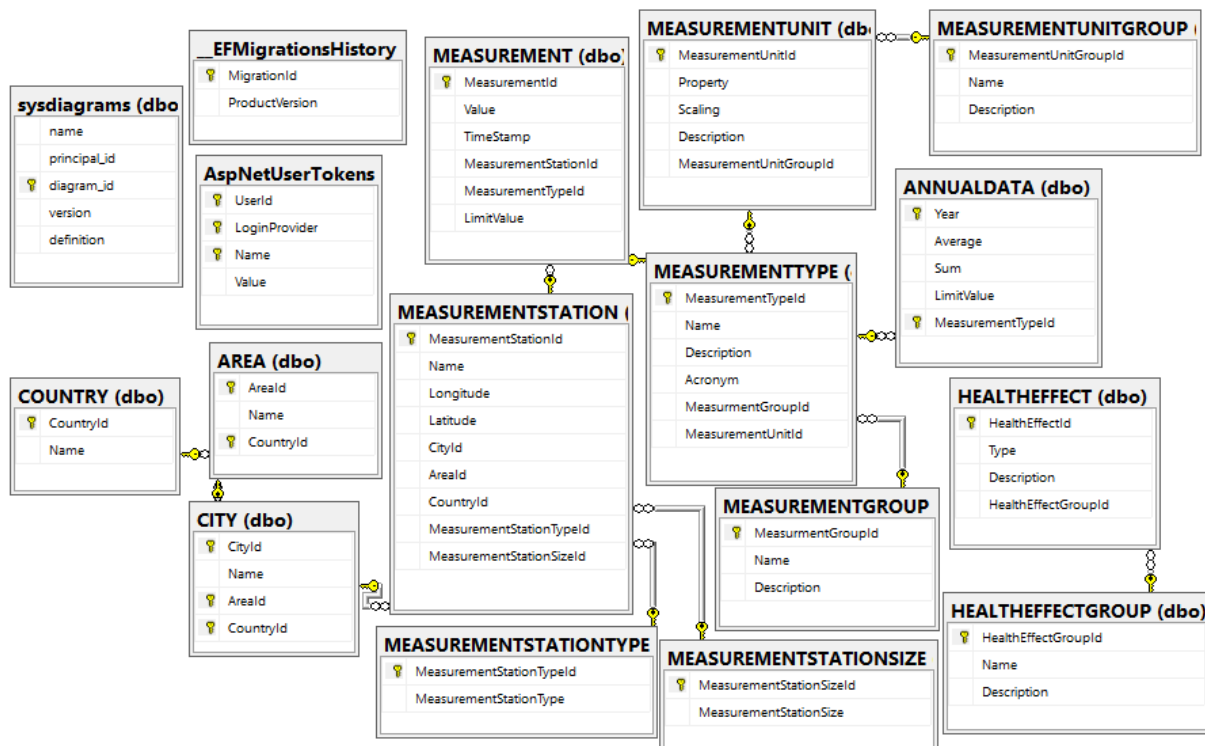


Figure 5.2 ER diagram showing the main tables of the current database

“MEASUREMENTSTATIONTYPE” was added to be able to differentiate between different measurement station types such as air, soil, and water. “MEASUREMENTSTATIONSIZE” adds the possibility to differentiate between measurement stations of different physical sizes.

All the tables shown in Figure 5.3 were added to make ASP.NET Core Identity work. This is where all the login information used on the web application is stored. Information about ASP.NET Core Identity can be found in chapter 6.4.2.

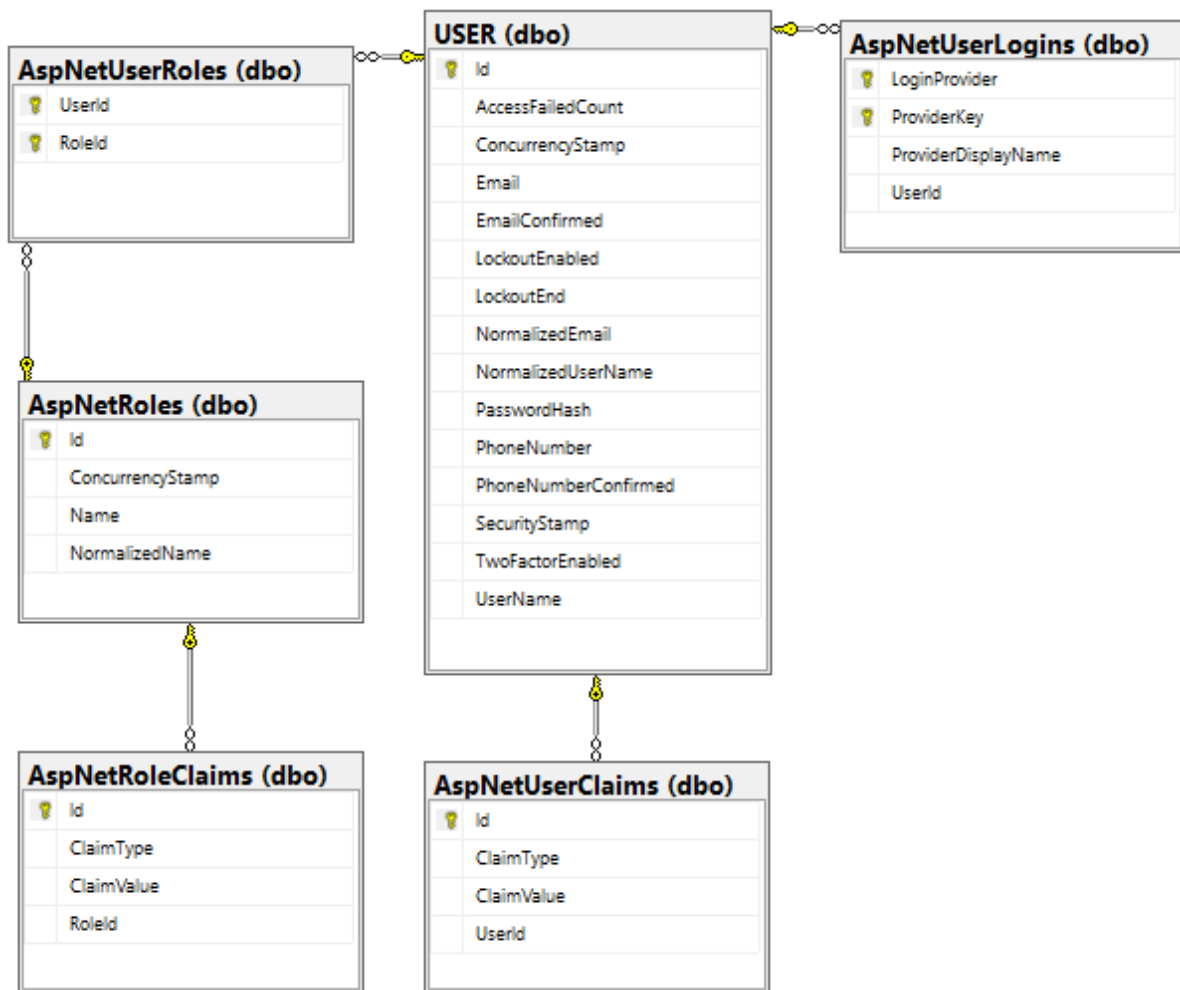


Figure 5.3 Database tables needed for ASP.NET Core identity functionality

5.2 Data gathering

This subchapter contains information about the different data sources used and considered for future use.

5.2.1 Air pollution

The source used for data on air pollution in Norway is NILU's database. The measurements stored in this database are the measurements used by NILU for their research. This together with the fact that NILU has 40 years of experience with research on air pollution, makes this database a credible source.

NILU has an API connected to this database, this allows people to retrieve data from the database either manually or through various commands and parameters in their code. In this project, all historic data from Grenland has been manually downloaded to fill the local database.

5 Database and data gathering

To solve the problem of having to do this manually in the future, a program has been developed to automatically copy the new data from NILU's database over to the local database.

5.2.2 Air pollution data gathering program

A program that gathers data from NILU's database was developed. This was done to avoid having to download all the data manually, and then upload them to the database. This program downloads JSON formatted data from NILU's REST API, and then saves it in the local database. The program loops every minute, and checks what time it is. When the time reaches 14 minutes past the hour, the program will download data, and save it to the database. The cycle for saving the data to the database is seen in Figure 5.4.

```
if (dataSaved == false)
{
    var niluData = new NiluData();
    var data = niluData.GetData();

    if (data != null)
    {
        foreach (var item in data)
        {
            Console.WriteLine();
            var saveData = new MEASUREMENT
            {
                TimeStamp = Convert.ToDateTime(item.toTime),
                Value = item.value,
                MeasurementStationId = db.MEASUREMENTSTATION.Where(c => c.Name == item.station).Select(c => c.MeasurementStationId).FirstOrDefault(),
                MeasurementTypeId = db.MEASUREMENTTYPE.Where(c => c.Acronym == item.component).Select(c => c.MeasurementTypeId).FirstOrDefault()
            };
            db.MEASUREMENT.Add(saveData);
            db.SaveChanges();

            Console.WriteLine("Date and time:" + item.toTime + "----- " + "Station:" + item.station + "----- " + "Value:" + item.value);
        }
        dataSaved = true;
        Thread.Sleep(60000);
    }
}
```

Figure 5.4 Data gathering saving cyclus

5.2.3 Data on traffic

Data on traffic around the areas where the measurement stations are deployed was also something the project partners were interested in. This would be used to see how much of an impact traffic has on the amount of pollution.

The NPRA (Norwegian Public Roads Administration) has a database called the National Road Databank (NVDB). NPRA was contacted to see if it was possible to get “live” data from this database through an API. The reply was that no API currently existed and none of the data in NVDB was “live”, the database was updated once a year with a summary of last year's traffic. They did however say that they were working on an API, but it would not be available within the time frame of this project. For full email chain see Appendix E.

NRPA offered to manually make an Excel file and send it via email. This file contains yearly and monthly values from 2012 till 2016, it is available for download on the web application [13]. See 8.1.11 for more information.

5.2.4 Water pollution

Data sources for water pollution have not been prioritized in this project. NEA (Norwegian Environment Agency) hosts a website which is a source for data and information on water pollution in Grenland and Norway in general [14]. Whether the data from this database should be copied to the local database will be up to a future project group to decide. For now, a link to this website can be found on this projects web application.

5.2.5 Soil pollution

The same premises as water pollution in chapter 5.2.4, also applies for soil pollution. A source for data on this subject can be found on the NEA's website on soil pollution [15], a link to this site can also be found on this projects web application.

5.2.6 Industrial pollution

The NEA's website on land based industry, as mentioned in chapter 2.1.2, is a source for data on industrial pollution in Norway. This website contains data on 660 different industries in Norway and specifically 61 different in Telemark. The database is updated once a year and all the data on the website is checked by a qualified person. For the time being, this data is not locally hosted. Whether this will be done is up to a future project group, but a link to NEA's website can be found on the projects web application.

5.3 Local- vs. cloud hosting

This subchapter will go through the different advantages and disadvantages of cloud- and local hosting relevant to this project, as well as a cost estimate on the difference between cloud and local hosting.

5.3.1 Local hosting

Advantages:

1. Control. With no third-party interference, control of hardware and software is completely up to the person maintaining the server.
2. Internet connection. An internet connection is not needed to access the local data, this means that the data is always accessible on site.

Disadvantages:

1. Data safety. If the data storage crashes, or there is something worse, like a fire, the data can be lost or damaged. This would not be an issue if the data was hosted in the cloud.
2. Cost. Hardware and software upgrades must be handled in house, this can potentially be expensive depending on the need for upgrades.

5.3.2 Cloud hosting

Advantages:

1. Cost effective. The cloud provider takes care of all maintenance and hardware/software upgrades.
2. Scalability. Storage size in the cloud can be scaled up and down depending on the current need, this is done automatically.
3. Back up. If something was to happen on site, the data will be safe in the cloud.
4. Accessibility. Data can be accessed from anywhere, as long as there is an internet connection.

Disadvantages:

1. Internet connection. An internet connection is needed to access the data.

5.3.3 Cost comparison

For the local database, a server matching the approximate needs for the applications in this project was found on Lenovo's website [16]. The data sent from the existing measurement stations takes up approximately 5 MB a year per station. If a 1 TB HDD was chosen it will be able to run the current system for many years before storage is a problem, this HDD size also ensures that there is plenty of storage for future additions to data logging.

Table 5.2 and Table 5.4 shows a rough estimate of what hosting the server locally compared to cloud would cost over the course of 5 years. It shows that the price difference between local and cloud is small, but there is a bigger chance of unforeseen expenses in the local host solution. The decision on where to host the applications is yet to be made.

5 Database and data gathering

Table 5.1 Investment cost local hosting

Description	Price pr./hour [NOK]	People	Hours	Price [NOK]
Equipment				17 000
Work	450	1	22.5	10 125
Total [NOK]				27 125

Table 5.1 shows the investment costs involved with local hosting.

Table 5.2 Total cost local hosting

Description	Year 0	Year 1	Year 2	Year 3	Year 4	Price
Investment cost [NOK]	27 125					27 125
Maintenance [NOK]	13 500	13 500	13 500	13 500	13 500	67 500
Hw/Sw Upgrade [NOK]					23 300	23 300
Total [NOK]						117 925

Table 5.2 shows the cost of hosting the database and web application locally over 5 years. The maintenance cost is based on 450 NOK an hour and 30 hours of work a year. On top of this, a solution for backing up data must be considered. Equipment is based on the prices found on Kompletts website [17] [18].

Table 5.3 Investment cost cloud hosting

Description	Price pr./hour [NOK]	People	Hours	Price [NOK]
Equipment				
Work	450	1	14	6 300
Total [NOK]				6 300

5 Database and data gathering

Table 5.3 shows the investment costs involved with cloud hosting.

Table 5.4 Total cost cloud hosting

Description	Year 0	Year 1	Year 2	Year 3	Year 4	Price
Investment [NOK]	6 300					6 300
Rental cost [NOK]	21 600	21 600	21 600	21 600	21 600	108 000
Total [NOK]						114 300

The rental cost in Table 5.4 is based on the price of an Azure application server, the start size of the storage is 10 GB but this can be scaled up through the user interface in Azure.

6 Web application - Solution

This chapter gives an overview of how the web application was programmed, and different choices that had to be taken during the programming period.

6.1 Web application - Architecture and design

This subchapter describes the programming and design of the web application.

6.1.1 Web application - Final design

This subchapter shows how the planned design in chapter 3 compares with the final design that is currently hosted in Azure.

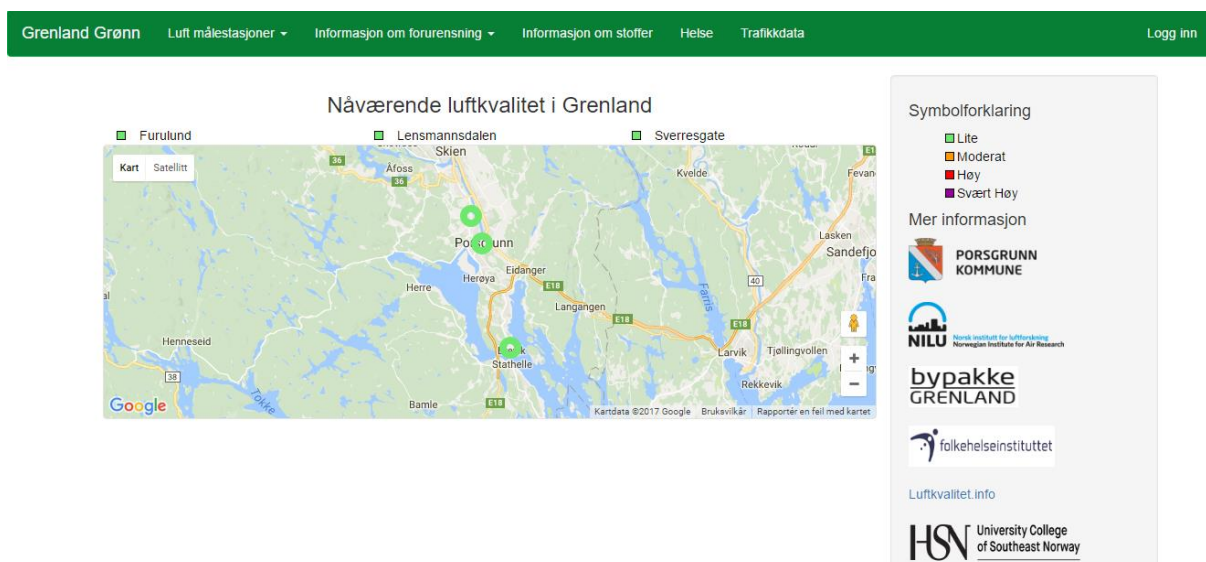


Figure 6.1 Picture of the home page

Figure 6.1 shows how the front page is designed. If compared to the design that was originally planned, the biggest difference is the addition of a map, and links to websites that contain more information.

Figure 6.2 shows how the layout of the information sites used for measuring stations is designed. Since the web application is designed to be dynamic, the layout page looks exactly the same for all stations. The biggest difference when compared to the original design, is the info box on the right side of the map that contains the values of the last registered measurements. This information site also has a graph like the one showed in Figure 6.10.

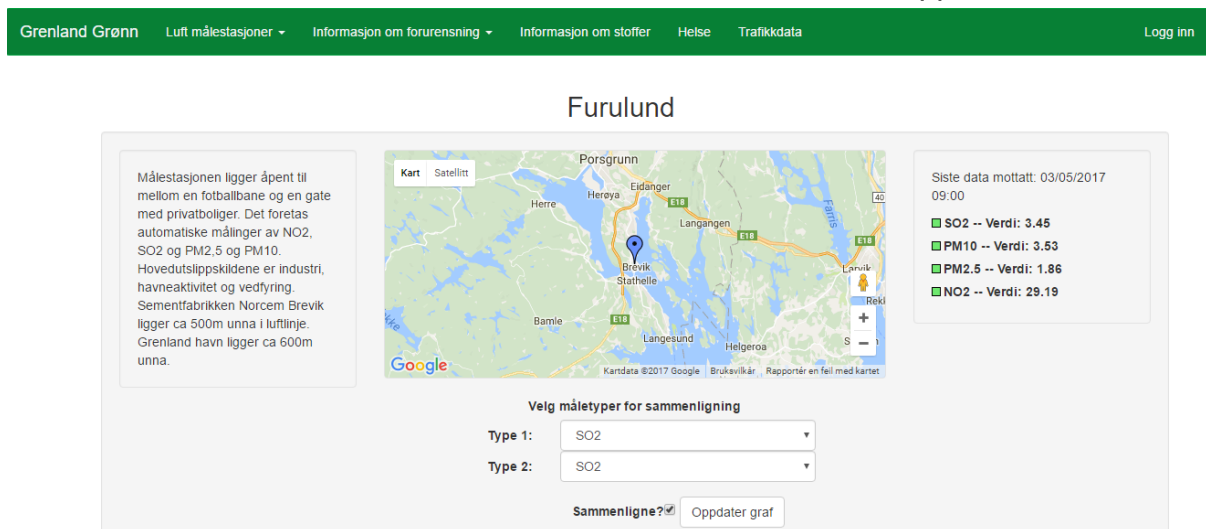


Figure 6.2 Information site for stations

6.1.2 Dynamic web application

Since the web application is supposed to be maintained without having a dedicated IT-person updating it. The web application had to be dynamic, this means that the information on the website changes depending on what data was in the database.

Most elements in the web application needed different data from the database. When programming using ASP.Net Core, the HTML pages are written in a combination of HTML and C#, this is possible by using Razor. Razor is a markup syntax that lets you embed server-based code (Visual Basic and C#) into web pages [19]. By using Razor, it is possible to use C# code in HTML pages, like foreach-loops and if-statements.

Figure 6.3 shows the code of the navigation bar and how the model is looped through. Doing it this way makes the navbars appearance change depending on the information in the database.

```

@foreach (var type in Model.MeasurementStations.Select(x => x.MeasurementStationType).Distinct())
{
    <li class="dropdown">
        <a class="dropdown-toggle" data-toggle="dropdown" href="">
            @type.MeasurementStationType
            <span class="caret"></span>
        </a>
        <ul class="dropdown-menu">
            <li class="dropdown-submenu">
                @foreach (var size in type.Measurementstation.Select(c => c.MeasurementStationSize).Distinct())
                {
                    <a class="test" tabindex="-1" href="#">@size.MeasurementStationSize <span class="caret"></span></a>
                    <ul class="dropdown-menu">
                        @foreach (var station in Model.MeasurementStations)
                        {
                            if (station.MeasurementStationTypeId == type.MeasurementStationTypeId && station.MeasurementStationSizeId == size.MeasurementStationSizeId)
                            {
                                <li><a asp-controller="MeasuringStations" asp-action="Index" asp-route-id="@station.MeasurementStationId">@station.Name
                                    @if (station.Measurement.OrderBy(c => c.Timestamp).Select(x => x.Timestamp)
                                        .LastOrDefault() >= DateTime.Now.AddDays(-1)) { <text>(Aktiv)</text> } </a>
                                </li>
                            }
                        }
                    </ul>
                }
            </li>
        </ul>
    }
}
</li>
}

```

Figure 6.3 Method for dropdown items on navigation bar

6 Web application - Solution

Figure 6.4 shows how the controller class gets the data from the database and how it passes the model to the view.

```
var model = new NavBarViewModel()
{
    MeasurementStations = await _context.Measurementstation.OrderBy(c => c.Name).ToListAsync(),
    MeasurementStationSizes = await _context.Measurementstation.Select(x => x.MeasurementStationSize).ToListAsync(),
    MeasurementStationTypes = await _context.Measurementstation.Select(x => x.MeasurementStationType).ToListAsync()
};

foreach (var station in model.MeasurementStations)
{
    station.Measurement = _context.Measurement.Where(c => c.MeasurementStationId == station.MeasurementStationId).ToList();
}

return View(model);
```

Figure 6.4 Code for getting data from database to navbar view

6.1.3 Responsive design

The web application uses Bootstrap's CSS framework. This framework is developed to make web applications responsive on all devices. Bootstrap is the most popular CSS framework for web applications [20]. Figure 6.5 shows the web application without Bootstrap.



Figure 6.5 Web application without bootstrap viewed on a Samsung Galaxy S7 Edge

6 Web application - Solution

Figure 6.6 shows the web application with Bootstrap, when comparing the different figures it is clear to see the difference Bootstrap makes on the appearance of the web application.

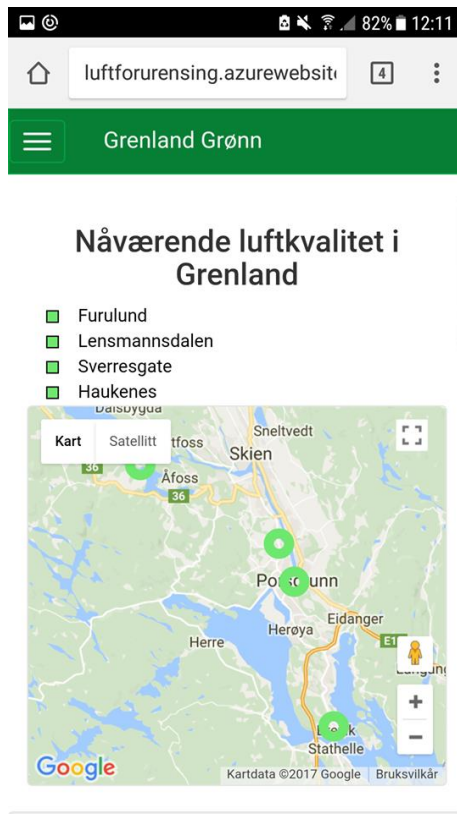


Figure 6.6 Web application with Bootstrap viewed on a Samsung Galaxy S7 Edge

Figure 6.7 shows how the application appears in the web browser of a computer.

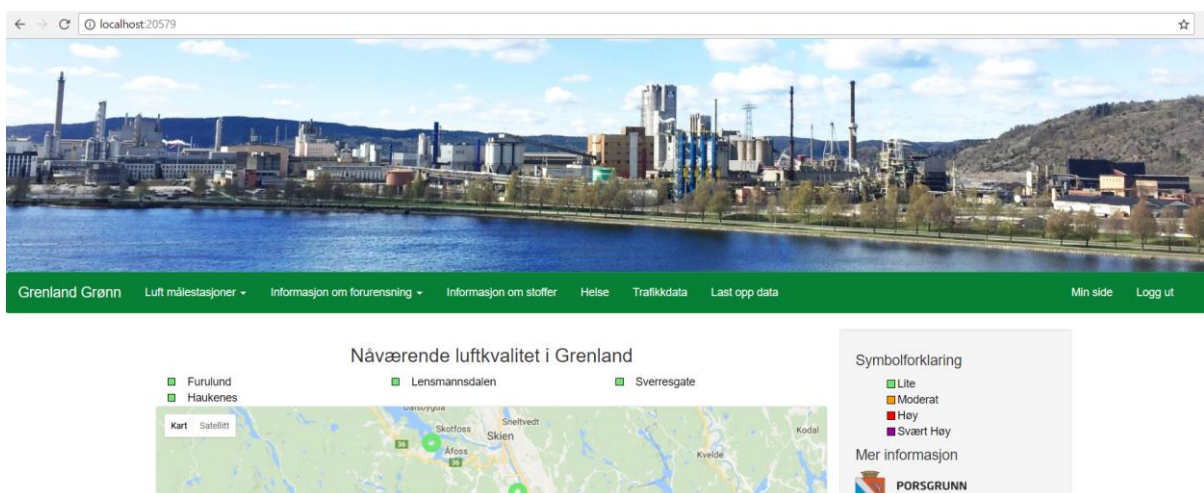


Figure 6.7 Web application viewed on a computer

6.2 Choosing and implementing a graph for the web application

This subchapter explains the process of choosing a graph and how it was implemented.

6.2.1 Choosing a graph solution

After deciding that the data should be displayed in a graph, a choice had to be made on what type of graph to use and how it should be displayed. There are many types of graphs that does pretty much the same thing, and choosing the right one for the job can be challenging.

The common way of integrating a graph into a web application is by using a JS library (JavaScript library) to manipulate the webpage based on the data given. After researching different suppliers of charts, there were four that was up for evaluation:

- Chart.js, available from <http://www.chartjs.org/>, free to use under the terms of The MIT Open Source Initiative license [21]
- Telerik, available from <http://www.telerik.com/purchase/aspnet-core-ui>, use of this chart would mean a cost of 1149\$ per developer after a free trial period.
- D3.js, available from <https://d3js.org/>, free to use under the 3-Clause BSD license [22]
- Google Charts, available from <https://developers.google.com/chart/>, free to use under the Attribution 3.0 Unported license [23]

The project partners wanted something that looked like the graph shown in Figure 3.7. The reason for this was that this type of graph had a datepicker that is easy to use. After thoroughly checking the four graphs, it was decided that Google Charts was the best option. This was because it was open source, and free. It was also easy to implement this into the web application.

It was decided that the graph should show 1 week of measurements, one of the reasons for this was because of a survey that NILU performed in Oslo [9]. This survey showed that info about past air quality measurements only was essential for approximately 8 % of the participants. The graph will also get slow and unresponsive if it contains too much information. This was a major factor when deciding how much information the graph should contain.

6.2.2 Implementing the graph

The graph was implemented using the JS library for Google Chart. Using a JS library means that code is already created, and the graph can be altered using various methods and options.

```
data.addColumn('datetime', 'Date');
data.addColumn('number', '@Model.ChartData.Select(x=>x.Type).FirstOrDefault()');
data.addColumn('number', '@Model.ChartData2.Select(x=>x.Type).FirstOrDefault()');
```

Figure 6.8 Adding columns to the chart

Figure 6.8 shows how to implement the columns needed for the chart. After these were added, data needed to be passed to the chart. This is done by using the code shown in Figure 6.9.

```
@foreach (var item in Model.ChartData.Zip(Model.ChartData2, Tuple.Create))
{
    System.Globalization.CultureInfo ci = System.Globalization.CultureInfo.InvariantCulture;
    var d = item.Item1.Timestamp.Value.ToString("yyyy-MM-dd HH:mm:ss", ci);

    <text>
    data.addRow([new Date("@d"), @item.Item1.Value.ToString().Replace(",","."),@item.Item2.Value.ToString().Replace(",",".")]);
    </text>
}
}
```

Figure 6.9 Adding rows to the chart

As seen in Figure 6.9, a foreach-loop is used to loop through all the objects in the model. For each item of data that is supposed to be in the chart, one row is added.



Figure 6.10 Graph on the measurement station page

Figure 6.10 shows how the chart looks on the web application. It is also possible to compare two different pollutants in the same graph, this is done using the dropdown boxes shown in Figure 6.11.

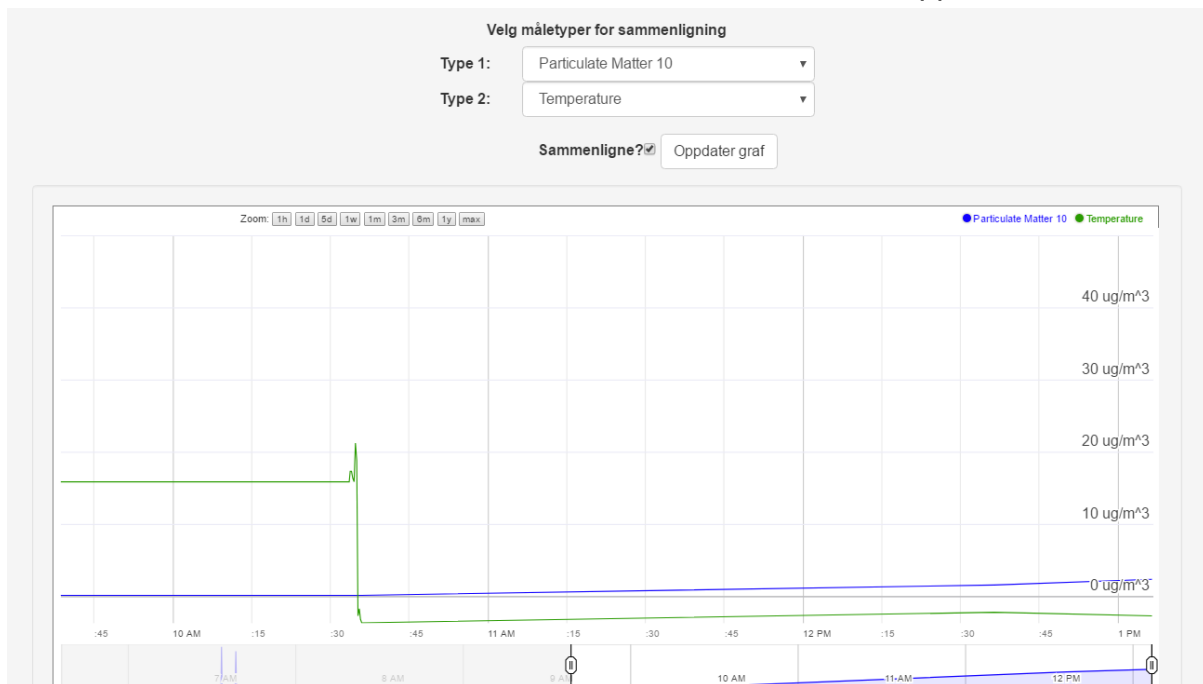


Figure 6.11 Comparing 2 different pollutants in the graph

6.3 Choosing and implementing a map for the web application

This subchapter explains the process of choosing a map solution and how it was implemented.

6.3.1 Choosing a map solution

In chapter 3.2, it was decided that the measurement station subpages were to have a map showing where the station was positioned. Later it was suggested to have a map on the home page showing the position of every station in Grenland, together with a colour indicator showing the level of pollution on each specific position. To achieve these two solutions, a map API with easy implementation and good support had to be found.

Just like the graph in chapter 6.2, the normal way to integrate maps into a web application is to use a JS library. The choice stood between two different providers:

- Bing Maps, available from <http://www.bing.com/api/maps/sdkrelease/mapcontrol/isdk>, free to use if there are less than 50 000 map loads per day [24].
- Google Maps, available from <https://developers.google.com/maps/>, free to use if there are less than 25 000 map loads per day [25].

After considering both options, it was decided that Google was going to be used. As Google Maps is well known and widely used, finding help on the internet would be easier than Bing Maps. Another reason is that most people are familiar with the Google Maps user interface, this would make the page easier to use for first time visitors.

6.3.2 Implementing the map

The Google Maps API offers a wide range of options when it comes to how items on the map are presented. As mentioned in chapter 6.3.1, one map solution shows one single marker, while the other shows multiple markers combined with a colour indication. To easily implement this, two different JS files were created. one for the home page, and one for the measuring station pages.

When first using the Google Maps API, an API key needs to be activated and used in the JavaScript source URL, as shown in Figure 6.12. More information on how to register an API key can be found at <https://developers.google.com/maps/>.

The map for the individual measuring stations is simple. It only shows a blue marker on the position of the measuring station. The reason for the marker being blue, is to avoid confusion for the user since the map on the home page uses various colour codes, and blue is not one of them.

```
<script src="https://maps.googleapis.com/maps/api/js?
key=AIzaSyAGQXY7v05YAvGJ4MeFmAhGmwUujw9ojRI&callback=initMap"></script>
<script src="~/js/map.js"></script>
<script>initMap(@Model.Latitude.ToString().Replace(",","."),
               @Model.Longitude.ToString().Replace(",","."))
</script>
```

Figure 6.12 Code used to show map in measuring station view

The code in Figure 6.12 shows how the data needed to position markers on the map is passed to the JS file. “initMap” refers to the function in Figure 6.13, this function has latitude and longitude as parameters. The data for these parameters is retrieved from the database using models, in this case “@Model.Latitude” and “@Model.Longitude”. Using models like this contributes to keeping the web application dynamic as mentioned in chapter 6.1.2.

```
function initMap(lat, lon) {
    var position = { lat: lat, lng: lon };
    var map = new google.maps.Map(document.getElementById('map'), {
        zoom: 10,
        center: position
    });
    var marker = new google.maps.Marker({
        position: position,
        map: map,
        icon: 'http://maps.google.com/mapfiles/ms/icons/blue-dot.png'
    });
}
```

Figure 6.13 JS for the measuring stations map

6 Web application - Solution

Figure 6.13 shows the code within the JS file. The map on the front page is more complex than the one for the measuring stations. This map shows multiple markers, one for each of the stations in Grenland, and a colour indication that changes depending on the level of pollution on the last reading.

```
<script src="https://maps.googleapis.com/maps/api/js?
  key=AIzaSyAGQXY7v05YAvGJ4MeFmAhGmwUujw9ojRI&callback=initMap">
</script>
<script src="~/js/MapHome.js"></script>
<script>
  var lat = [];
  var lon = [];
  var id = [];
  var station = [];
  var color = [];
  @foreach (var item in Model)
  {
    @:lat.push("@item.Latitude.ToString().Replace(",", ".")");
    @:lon.push("@item.Longitude.ToString().Replace(",", ".")");
    @:id.push("@item.Id");
    @:station.push("@item.MeasurementStationName");
    @:color.push("@item.Color");
  }
  initMap(lat,lon, id, station, color);
</script>
```

Figure 6.14 Code used to show map on the home page

Figure 6.14 shows the code, while Figure 6.15 shows how the map looks on the home page. Since this map has multiple markers, it needs more than one piece of data. Because of this, the data is passed to the JS file by looping through all the data in the model, and for every item in the model, the “.push” method fills an item into the array. When the loop is done, the filled arrays are sent as parameters to the JS file, using the initMap function, just like the previous map function.

Another difference from the code shown in Figure 6.12, is that this function also takes the parameters id, station, and colour. The “id” parameter holds information about what ID the different stations have, this was added to allow the user to click a desired station and automatically redirect the user to the information page belonging to the selected station. This feature is currently not implemented on the web application, but it can be added in the future, see more in chapter 8.1.10.

The “station” parameter has information on the names of the different locations, this information is collected from the local database. This is currently used to let the user click any marker on the map, when clicked the stations name will pop up above the marker as shown in Figure 6.15.

6 Web application - Solution

To allow for the marker on each position to change colour depending on the current level of pollution, the “color” parameter had to be added. The information in the “color” parameter is collected from NILU’s API mentioned in chapter 5.2.1.

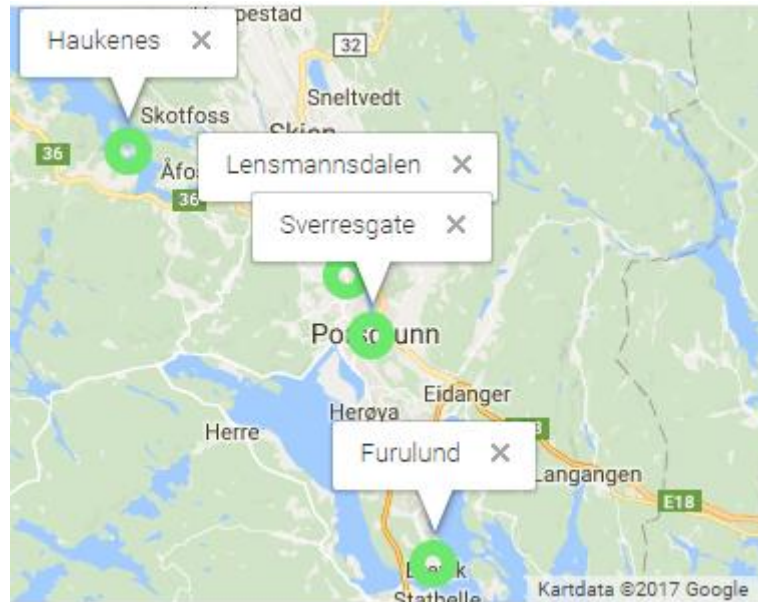


Figure 6.15 Map on the home page

6.4 Information about miscellaneous code on the web application

This subchapter gives information about miscellaneous code on the web application.

6.4.1 Data from the MMSP

To get data from the prototype, a separate URL was developed in the web application, that retrieves data via post requests. The web application gets JSON formatted data from the MMSP, and saves this the same way as the data gathering program. Figure 6.16 shows how the web application takes the JSON formatted data and saves it in a model, and then adds it to the database.

6 Web application - Solution

```
public IActionResult Post([FromBody]AddDataViewModel measurement)
{
    var tsi = TimeZoneInfo.FindSystemTimeZoneById("Romance Standard Time");
    DateTime date = DateTime.UtcNow;

    if (tsi.IsDaylightSavingTime(date))
    {
        date = date.AddHours(2);
    }
    else
    {
        date = date.AddHours(1);
    }

    var saveData = new Measurement();
    saveData.MeasurementId = null;
    saveData.Value = measurement.Value;
    saveData.Timestamp = date;
    saveData.MeasurementStationId = measurement.MeasurementStationId;
    saveData.MeasurementTypeId = measurement.MeasurementTypeId;

    _context.Add(saveData);
    _context.SaveChanges();

    // _context.SaveChanges();
    return Json(measurement);
}
```

Figure 6.16 Code for retrieving JSON data from post request

6.4.2 Login for the web application

It was decided that a login was needed. This was to ensure that only authorized people could upload data, edit measurement stations, and do other “admin” related tasks. To implement the login function in ASP.Net Core, 2 Nuget Packages were needed:

- Microsoft.AspNetCore.Authorization
- Microsoft.AspNetCore.Identity.EntityFrameworkCore

Some updates to the database were needed, ASP.Net Core Identity needs the tables shown in Figure 5.3.

```
[AllowAnonymous]
[HttpPost]
0 references | Hans Martin Kristensen, 18 hours ago | 1 author, 3 changes
public async Task<IActionResult> Login(LoginVM vm) //Sender med data fra Account/Login viewet, for å prøve og logge inn brukeren.
{
    if (ModelState.IsValid)
    {
        var result = await _signInManager.PasswordSignInAsync(vm.Email, vm.Password, vm.RememberMe, false);

        if (result.Succeeded)
        {
            //Hvis brukeren har angitt riktig epost og passord, så blir den personen logget inn, og sendt til Account/Index viewet.
            return RedirectToAction("Index", "Account");
        }
        //Hvis brukeren ikke blir logget på, blir en ModelState lagt til, for visning i viewet.
        ModelState.AddModelError("", "Feil ved innlogging");
        return View(vm);
    }
    return View(vm);
}
```

Figure 6.17 Code for logging into the web application

6 Web application - Solution

Figure 6.17 shows the code needed for the login function, The Nuget Packages installed handles all the password hashing and cookie data. After logging into to the application, the administrator is taken to the main admin page, as seen in Figure 6.18.

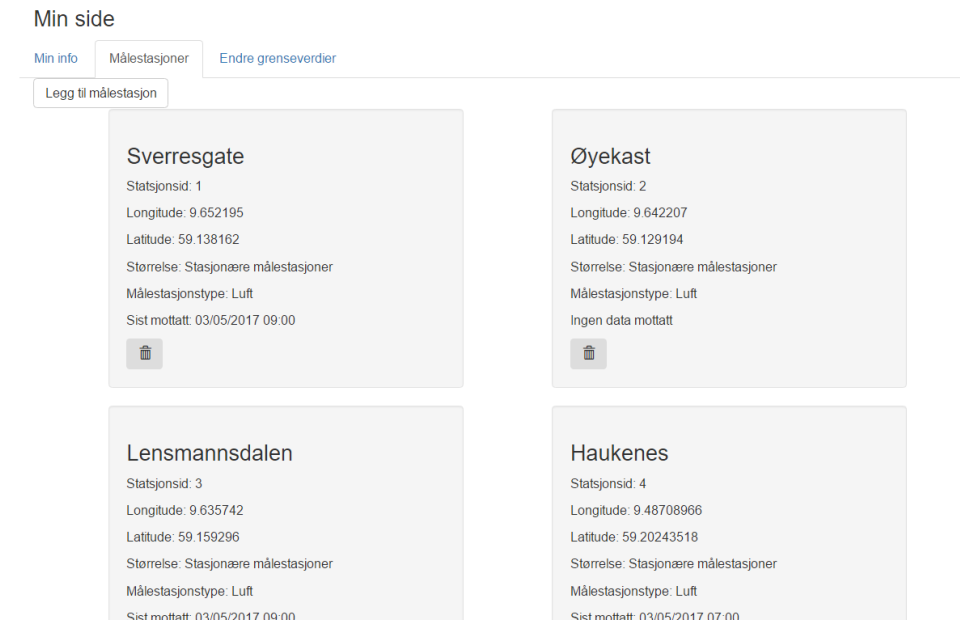


Figure 6.18 The main user page

In the main admin page, the admin can see information about the different measurement stations, and change the limit value of the different measurement types. The admin also needs to be logged in, to be able to upload data to the database. Figure 6.19 shows the upload data website.

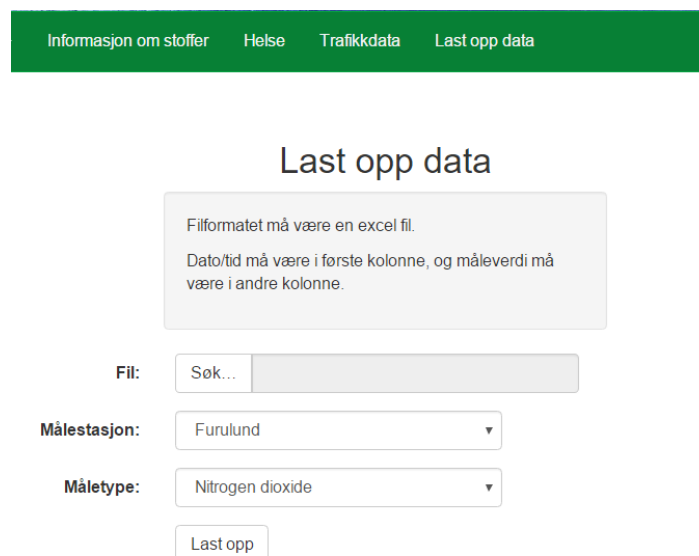


Figure 6.19 Upload data website

6.4.3 Upload data

All data not available through an API can be added manually from an Excel file. This is done using the Nuget Package EPPlus.Core.

```

for (var row = 2; row <= lastRow; row++)
{
    var excelDate = Convert.ToDecimal(workSheet.Cells[row, 1].Value);
    DateTime date = new DateTime(1899, 12, 31).AddDays(Convert.ToDouble(excelDate));
    var value = Convert.ToDouble(workSheet.Cells[row, 2].Value);
    //if (value > 1000 || value < -1)
    //{
    //    var error = new Error();
    //    error.ErrorName = "Value too high or low, in row " + row + ". The value is " + value;
    //    error.ErrorDescription = "Value in given row is outside of the set limit for the given value.";
    //    return View("Error", error);
    //}
    var newRecord = new Measurement
    {
        Value = value,
        TimeStamp = date,
        MeasurementStationId = model.MeasuringStationId,
        MeasurementTypeId = model.MeasuringTypeId
    };

    _context.Measurement.Add(newRecord);
}

```

Figure 6.20 Adding new measurements from an Excel file

The way the uploading code works, is that it loops through all the rows of the document, and adds each row to the database, as seen in Figure 6.20. The user chooses what station and measurement type that is in the file, and selects the file that contains the data. How the Excel file should be formatted is described on the upload data website, as seen in Figure 6.19.

7 Mobile Measuring Station Prototype - Solution

This chapter gives an insight into the built solution of the MMSP. It also focuses on the equipment used, as well as the construction and programming. It is worth noticing that this build has been done in accordance with the funding of this project, and the focus has been to build a low-cost solution that proves the concept of a small and cheap measuring station. Due to the time frame of the project, some features that were initially planned in chapter 4, had to be discarded to complete the build. For example, it was planned to measure NO_x and install a display on the prototype, but these features were not prioritized in this project. The focus has therefore been to implement PM_{10} -, $\text{PM}_{2.5}$ -, and temperature sensors. Figure 7.1 shows the inside of the prototype.

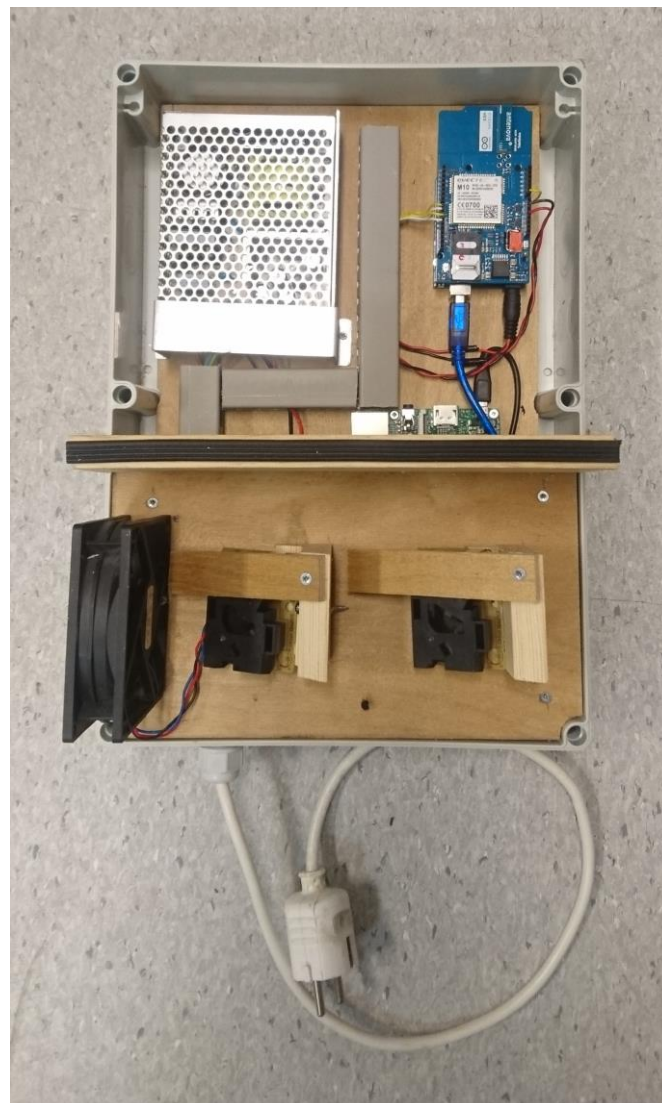


Figure 7.1 Inside of the MMSP

7.1 Prototype concept

The main concept behind this prototype is to build a low-cost solution for data transmission to a web server. The data transmission uses GSM-technology by sending a HTTP-post to the web server, containing the measurement data and station ID. The GSM shield uses a SIM-card that must be pre-configured with correct PIN-number and APN-connection. The APN-connection defines which type of communication format to use, whether it is SMS, MMS, or as in this case internet connection. Measurement data is sent as JSON format to the web server.

The sensors currently used does not have any documentation that proves their accuracy to be accurate enough for public information. When more accurate low-cost sensors arrive on the marked they should be easy to implement in the MMSP. Figure 7.2 illustrates the system and how data is passed from sensors and presented on the web application.

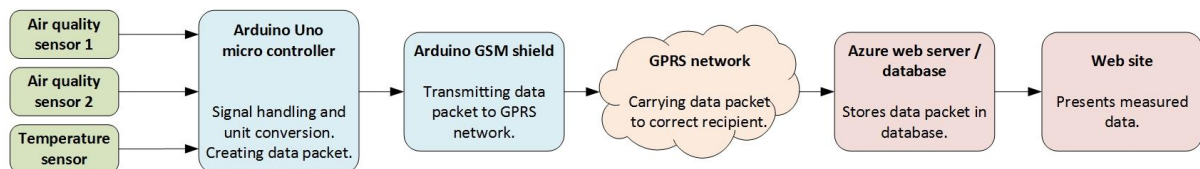


Figure 7.2 Data flow of the system

7.2 Technology choice

The price range of the used equipment is relatively low compared to the equipment on the existing measurement stations, and a budget quote was made, see Table 7.1. The concept is built around microcontrollers. A microcontroller contains sets of physical inputs and outputs which can be connected to electrical devices and equipment to monitor and control a process. The following subchapters will explain the individual components used for the prototype, and a summary of their characteristics.

After developing and assembling the measurement station, there were some unforeseen costs compared to the estimate in chapter 4.9. The reason for the extra expenses was due to long delivery times on certain items, and the need to buy alternative items with a shorter delivery time. The total cost is listed in Table 7.1.

7 Mobile Measuring Station Prototype - Solution

Table 7.1 Total build cost

Equipment	Price equipment [NOK]
Air vents	170
Rubber seal list	70
Cabinet	880
PG-Gland	50
RJ-45 plugs	50
Air quality sensors	600
Transport Straps x2	100
USB micro power supply	50
Arduino Uno	280
Power supply	240
Raspberry Pi 3	450
Arduino GMS shield	650
PC-Fan	100
Micro sensor	250
Arduino GSM shield 2	800
Total cost	4740

7 Mobile Measuring Station Prototype - Solution

7.2.1 Microcontrollers

Due to the need for both a controller that can handle analog inputs and one that can support a user interface program, the prototype consists of 2 types of microcontrollers. The main controller is an Arduino Uno, while the controller that supports the user interface is a Raspberry Pi. This is not an ideal solution, as it requires knowledge about both platforms, and in addition this makes the prototype more complicated, the reason behind this choice is described in chapter 4.5.

Specifications on the Raspberry Pi can be found in Appendix F, and the Arduino Uno specs in Appendix G.

7.2.2 Data Transmission

Transfer of data is done over GPRS with the GSM shield. This solution was chosen because the Arduino GSM shield supports this system. The GPRS coverage in Grenland is shown in Figure 7.3.

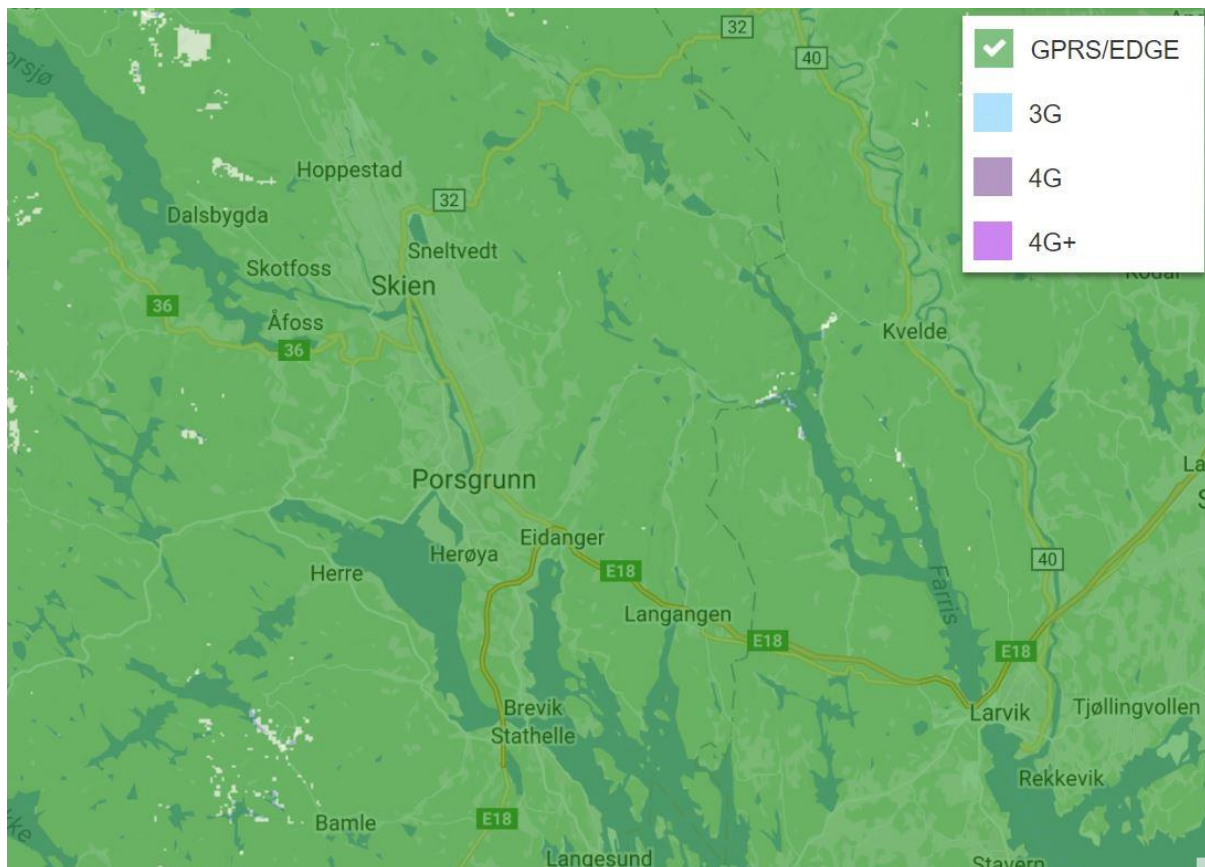


Figure 7.3 Coverage of Telia network in Grenland [26]

Theoretic transmission speed is limited to 171 kbit/s, but practically it is more realistic to achieve 30 to 70 kbit/s [27]. Since the data packet sent from the prototype is roughly around 6.6 kbit per transmission (Calculated from 165 characters), this is within the transmission speed. In this project, the network operator Telia was selected because a project member used

7 Mobile Measuring Station Prototype - Solution

this, and therefore it was easy to use a twin SIM-card for testing of the prototype. The subscription must be able to connect to the internet, for the MMSP to send HTTP-post packets to the web application. Equation (7.1) shows how the bits per transmission is calculated, 165 is the estimated amount of characters, 8 is the number of bits in 1 byte and 5 is the amount of JSON objects per transmission. It is worth to notice that this is an approximation and the real amount can divert some from this result, depending on the amount of characters and possible control bits.

$$data\ bits = 165 * 8 * 5 = 6.6\ kbit \quad (7.1)$$

An original Arduino GSM shield is used as the interface between the microcontroller and the GSM-network. The GSM-module first purchased did not support the European frequencies [28], and it was decided to use an Arduino shield instead to avoid having to troubleshoot the GSM module any further. The Arduino shield selected has an integrated antenna, but there are options where an external antenna could be installed, should the signal be too weak. The GSM shield is mounted directly to the top of the Arduino microcontroller. This leaves the USB serial port free and it is therefore used as an interface to the Raspberry Pi.

Data packet sent as JSON format				
URL: luftforurensing.azurewebsites.net				
Connection string: http://luftforurensing.azurewebsites.net/adddata/post				
Station-ID: 8		Station-ID: 9		
Measurement type-ID: 4	Measurement type-ID: 5	Measurement type-ID: 4	Measurement type-ID: 5	Measurement type-ID: 8
PM ₁₀ Sensor 1	PM _{2.5} Sensor 1	PM ₁₀ Sensor 2	PM _{2.5} Sensor 2	Temperature

Figure 7.4 Data packet sent as JSON format

This project relies on GPRS technology to communicate with the web application. GPRS is a 2G technology and newer generations like 3- and 4G are not supported. The GPRS supports TCP/UDP and HTTP protocols and has a maximum data up- and downlink transfer speed of 85.6 kbit/s [29]. The data transfer uses HTTP over TCP port 80, and the content is in JSON format illustrated in Figure 7.4. In terms of power supply, the GSM shield is connected directly to the Arduino Uno. It is recommended to have an external power supply that can provide at least 700-1000 mA. The GSM shield draws up to 2 amperes when in heavy use. More info on the Arduino GSM shield can be found in Appendix H.

7.2.3 Sensors

It was decided to implement two air quality sensors which would work in parallel, and then see how their data outputs compared. The Arduino microcontroller handles all the sensors,

7 Mobile Measuring Station Prototype - Solution

including a temperature sensor which has an analog 0-5 V output [30]. The air quality sensor selected was a Grove dust sensor [31]. This sensor was used on a similar project [32] and supports both Arduino and Raspberry Pi. In addition, it was low-cost and easy to implement. Figure 7.5 shows one of the air quality sensors used.

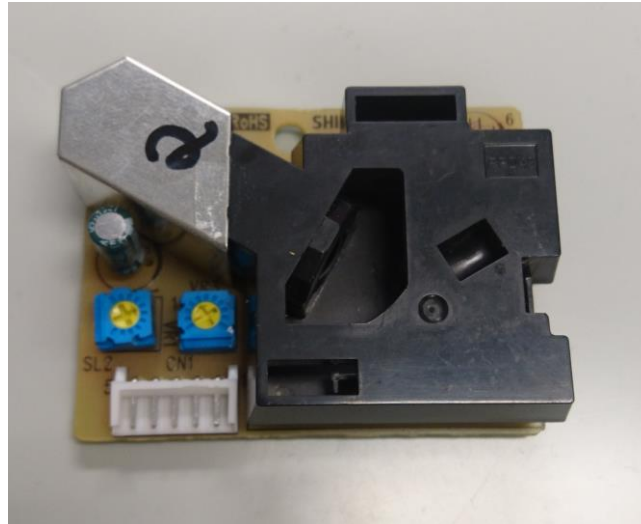


Figure 7.5 Air quality sensor

The sensor is powered from the Arduino, and has an output signal for both PM_{10} and $PM_{2.5}$. It utilizes a negative logic by dropping the voltage output from 4 V to under 0.7 V when a particle with a diameter greater than $1 \mu m$ passes through the sensor. Figure 7.6 shows the inside of the air quality sensor.

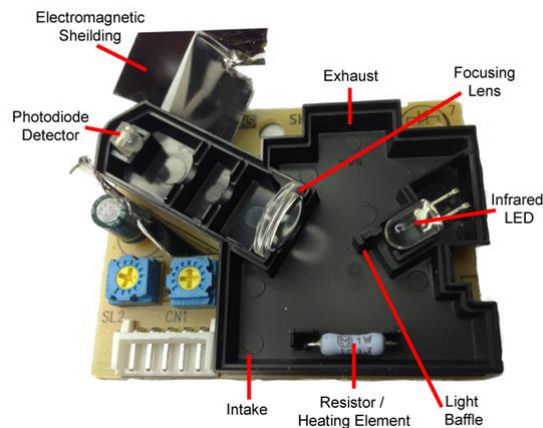


Figure 7.6 Inside of the sensor [33]

7 Mobile Measuring Station Prototype - Solution

Figure 7.7 contains the specifications for the Grove dust sensor, and more information is found in Appendix I.

Items	Min	Norm	Max	Unit
VCC	4.75	-	5.25	V
Standby Current Supply	-	90	-	mA
Detectable range of concentration	-	0~28,000 / 0 ~ 8000	-	pcs/liter / pcs/0.01cf
Operating Temperature Range	0	-	45	° C
Output Method	Negative Logic, Digital output,Hi over 4.0V(Rev.2) Lo: under 0.7V			
Detecting the particle diameter	>1 um			
Dimensions	59(W) × 45(H) × 22(D) [mm]			
Humidity Range	95%rh or less			

Figure 7.7 Dust sensor specifications

Initially the sensor measures dust concentration particles per volume (pcs/l), but since the particle matter uses $\mu\text{g}/\text{m}^3$ as the measuring unit, it must be converted. The sensor samples the number of negative logic pulses over a defined period, and outputs this as a concentration of particles, see Figure 7.8.

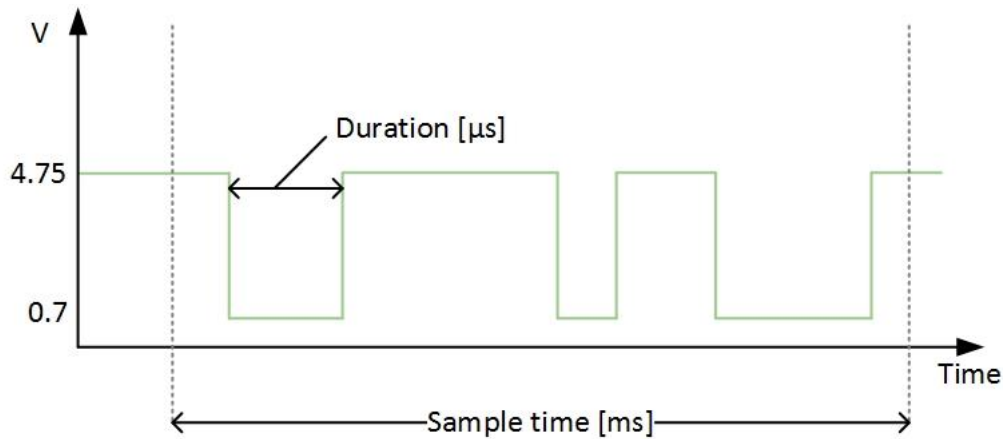


Figure 7.8 Measuring principle with sensor output signal

Each particle with a diameter greater than $1\ \mu\text{m}$ will result in a low-pulse output, where the duration time indicates the time the particle uses to pass. The concentration measured as percentage of particles is then calculated by using equation (7.2):

$$R = \frac{\sum_0^{\text{Sample time}} \text{Duration}}{\text{Sample time} * 1000} * 100 [\%] \quad (7.2)$$

7 Mobile Measuring Station Prototype - Solution

When it comes to the conversion from particle count to $\mu\text{g}/\text{m}^3$, it must be noted that these conversions are made from approximations since it is hard to exactly determine their properties [34]. NILU was contacted regarding advice on these calculations, but no reply was received. Therefore, assumptions used in the code for this conversion is based on previous and existing projects due to time limitations [32] [34]:

- The shape of each particle is considered as spherical
- Particle density is $1.65 * 10^{12} \mu\text{g}/\text{m}^3$
- The radius of $\text{PM}_{2.5}$ particle is $0.44 \mu\text{m}$
- The radius of PM_{10} particle is $2.60 \mu\text{m}$

The number of particles are calculated by using the spec sheet curve equation found in Appendix I, see equation (7.3):

$$n = 1.1 * R^3 - 3.8 * R^2 + 520 * R + 0.62 \quad (7.3)$$

Equation (7.4) is used to show that the relation between 1 ft^3 to 283 ml is:

$$K = \frac{1 \text{ ft}^3}{283 * 10^{-6} \text{ m}^3} = 3531.5 \quad (7.4)$$

The mass of the particle is calculated using equation (7.5):

$$m = \rho * V = \rho * \frac{4}{3} * \pi * r^3 [\mu\text{g}] \quad (7.5)$$

Then finally the PM-concentration is calculated using equation (7.6):

$$PM_{concentration} = m * K * n [\mu\text{g}/\text{m}^3] \quad (7.6)$$

These mathematical expressions are implemented into the Arduino code to convert between pcs/l and $\mu\text{g}/\text{m}^3$.

7 Mobile Measuring Station Prototype - Solution

7.2.4 Power source

During the project planning, two power solutions were suggested: Battery powered or connecting the prototype to the existing electrical infrastructure as described in chapter 4.8. The battery solution would not need any kind of infrastructure to work, but would not be able to power the system for longer periods. Therefore, connecting the prototype to existing electrical infrastructure was selected. This option requires electric infrastructure on the deployment site, but most areas where it would be interesting to deploy the prototype has some sort infrastructure. The power consumption of the prototype at its largest is 15.1 W. A 12 VDC power supply is mounted inside the prototype, to provide the correct voltage level for the electronic components.

A calculation of the total power consumption was done to estimate the operation time for the prototype should it be powered by a battery. A list of the power consuming components and their characteristics are listed in Table 7.2.

Table 7.2 Electrical characteristics

Equipment	Nominal voltage [V]	Max current consumption [mA]
Arduino Uno	5	800
Arduino GSM Shield	5	1000
Raspberry Pi 3	5	1200
Fan	5	20
Total	5	3020

The total operating time is calculated in equation (7.10) by using equation (7.7), (7.8), and (7.9):

$$P = U * I \quad (7.7)$$

$$P = 5 V * 3,02 A = 15,1 W \quad (7.8)$$

$$I = \frac{15,1 W}{12 V} = 1,267 A \quad (7.9)$$

$$\text{Operating time} = \frac{30 Ah}{1,267 A} = 23,67 h \quad (7.10)$$

7 Mobile Measuring Station Prototype - Solution

The result shows that by using a 30 Ah battery, it will be necessary to recharge it after 23 hours. A 30 Ah battery was chosen as an example because of its physical size, which is limited by size and weight in terms of making the prototype as small as possible. In addition to this, the battery capacity would also be affected by the temperature [35].

7.3 Wiring and built solution

When building the MMSP, some wiring and equipment fitting had to be done. The focus was to build it as small as possible, to make it easier to deploy more of these MMSP's in the future. The prototype is built inside a 36 x 25.4 x 16.5 cm plastic box, and has an air channel at the bottom of the box where the air flows through with the help of a fan, see Figure 7.9.

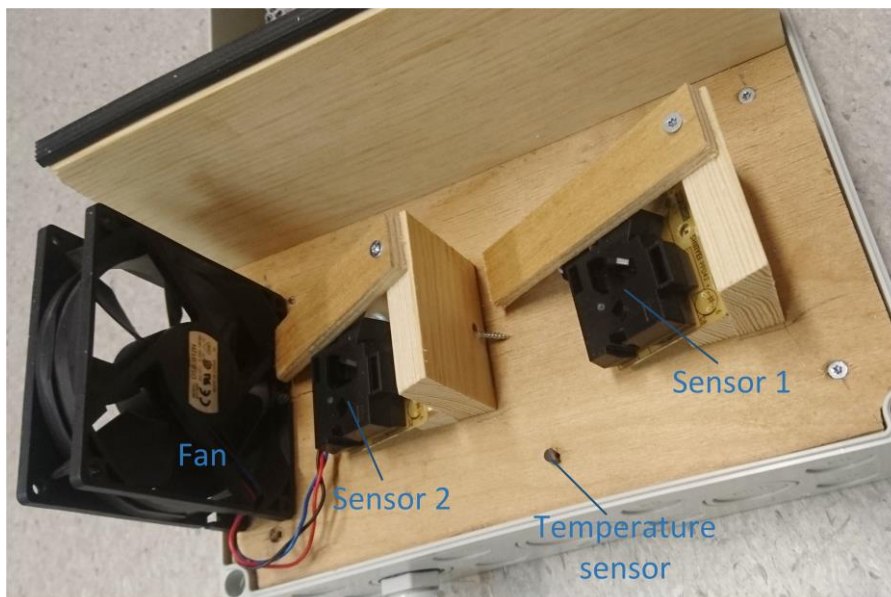


Figure 7.9 Fan is mounted to create air flow

The air quality and the temperature sensors are mounted inside this air channel. All the electronic parts are mounted above the air channel to avoid damage from dust and moisture. A power cable goes through the bottom of the box, and can be directly connected to electric infrastructure.

The unit itself is a capsuled plastic cabinet with drain holes in the bottom and two air vents on each side to enable air to flow through the air channel. The cabinet must be mounted according to the user manual in order to achieve proper protection from rain and snow. The two air vents are covered with a perforated metal plate to minimize environmental exposure to the internal components, but still enables air to pass through. For simplicity and ease of making changes to the components inside, plywood is used as a mounting plate. To protect the electrical components from short-circuiting, a 5 A quick fuse is installed on the 12 V.

The prototype operates with different voltage levels. The main power supply uses 230 VAC, and runs 12 VDC as output, with a maximum load of 6 A. Figure 7.10 show the wiring of the prototype. The electrical wiring diagram is attached to the report as Appendix J.

7 Mobile Measuring Station Prototype - Solution

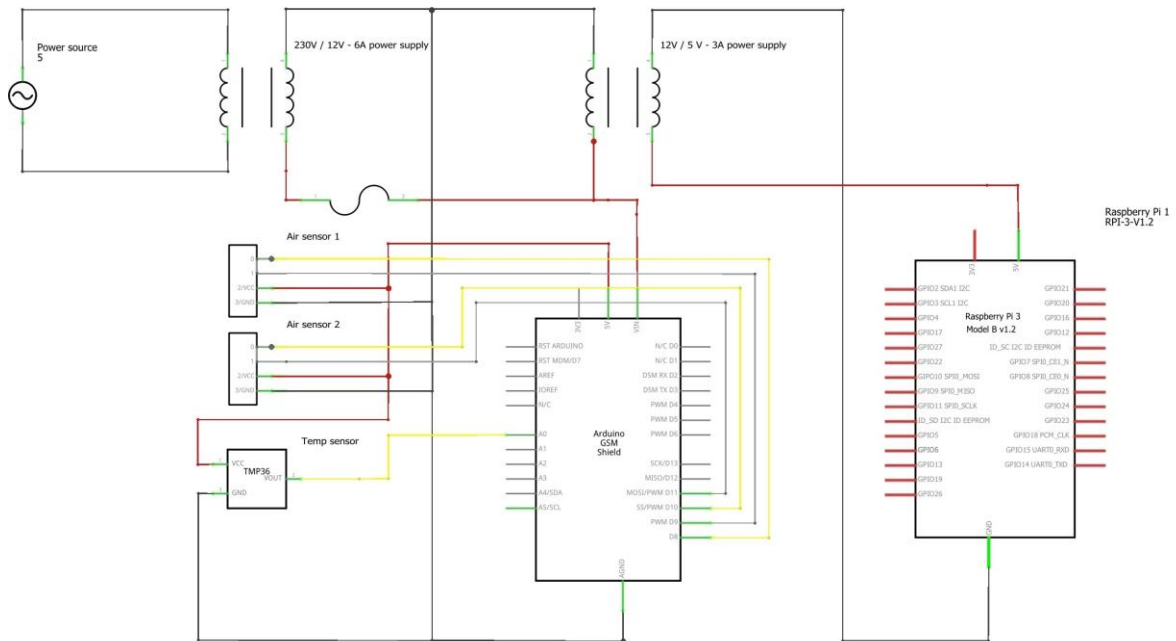


Figure 7.10 Wiring diagram of the MMSP

The Arduino is directly connected to the 12 V, while a 12 V to 5 V transformer is used to deliver power to the Raspberry Pi. The GSM shield and sensors are powered from the Arduino. The GSM shield can draw an ampere peak of around 2 A when in work mode [29], and therefore it is vital to use a power supply that can handle this peak.

7.4 Code and Software Solution

Coding of the prototype is essential to make it function as intended. The code takes care of communication and correct handling of signals from the sensors. In addition, the code is written to ensure correct start-up after a power failure and correct initiation of stored values.

7.4.1 Raspberry Pi

One of the main reasons for choosing Raspberry Pi as a platform, was because it supports the Microsoft operative system called IoT. This OS supports programs written in C# and therefore, the GUI application could be developed in Visual Studio. Visual Studio offers help when writing the program, and contains helpful tools and functions to ease the coding process.

In the planning part of the project the idea was to create the entire prototype around the Raspberry Pi platform, but due to the I/O-configuration and the hardware available the solution became a combination of Raspberry Pi and Arduino. The Raspberry Pi takes care of the user interface and the options for setting parameters as well as initiating the MMSP. The Arduino functions more like a connection module that takes care of all sensor- and component connections. Another issue with the Raspberry Pi is the lack of analog inputs which is used for the temperature sensor. The particle matter sensors can be connected to the Raspberry Pi since it uses digital signals, but is currently connected to the Arduino Uno.

7 Mobile Measuring Station Prototype - Solution

Although the prototype would work without the Raspberry Pi, it was implemented to enable users to set parameters and edit these as necessary. If this solution was not implemented, the parameters would have to be hard coded into the Arduino, and would therefore be more complicated to change on a later occasion. It is also possible to get GSM shields that connects directly to the Raspberry Pi, but due to time limitation and price of this shield, the Arduino shield was selected instead.

7.4.2 Arduino

The Arduino platform is an open source platform that does not require a lot of programming experience. A negative side when coding in Arduino IDE, is the lack of debugging and help when writing code compared to Visual Studio.

The code works by running a setup method that initiates serial communication and other functions. After the setup is complete, the code then runs in a continuous loop which contains a method for sampling measurement data. This method also opens the connection to the web application, and creates a JSON object with all the data stored inside it. This JSON object is then passed to the GSM-connection as a HTTP-post package. A set of code lines tries to open a connection to the web application, and if succeeded the JSON object is sent to the server, in a pre-defined format that matches the database table the data is stored in.

The code contains parameters that are necessary to get the data transferred correctly to the right recipient. The program must have the correct URL to where the data should be sent, a path to designated web application, and which type of port to open for the communication. The HTTP-post connection uses port 80 [36].

Data readings from the air quality sensors are done in the Arduino by registering negative pulses. The code is written to sample a number of these pulses over a defined period, it then calculates the number of pulses per period to measure the concentration percentage of particles in the air. The air quality sensor initially measures particles per volume (pcs/283ml), and must therefore be converted into $\mu\text{g}/\text{m}^3$ to match the standard unit for particle matter. The method for converting between these units is explained more in chapter 7.2.3.

7.5 Field testing of performance and durability

An important topic in the project was to test the MMSP against an existing measuring station. The field test was originally planned for a 24-hour period, but ended up being 21 hours due to issues with the GSM shield. The test was done to monitor the prototype's behaviour and measurement accuracy, compared to NILU-data.

The prototype successfully transferred data throughout the entire period, and the values were logged into the database and displayed on the website. The field test was performed at Lensmannsdalen measuring station in Skien municipality, see Figure 7.11.



Figure 7.11 Prototype mounted on the existing measuring station

A comparison between the prototype readings and the actual data readings from NILU are presented in Figure 7.12 and Figure 7.13. Accurate readings from the sensors were not expected, so this test was done to determine the accuracy of the sensors, and if they followed the trend of the existing measuring station.

7 Mobile Measuring Station Prototype - Solution

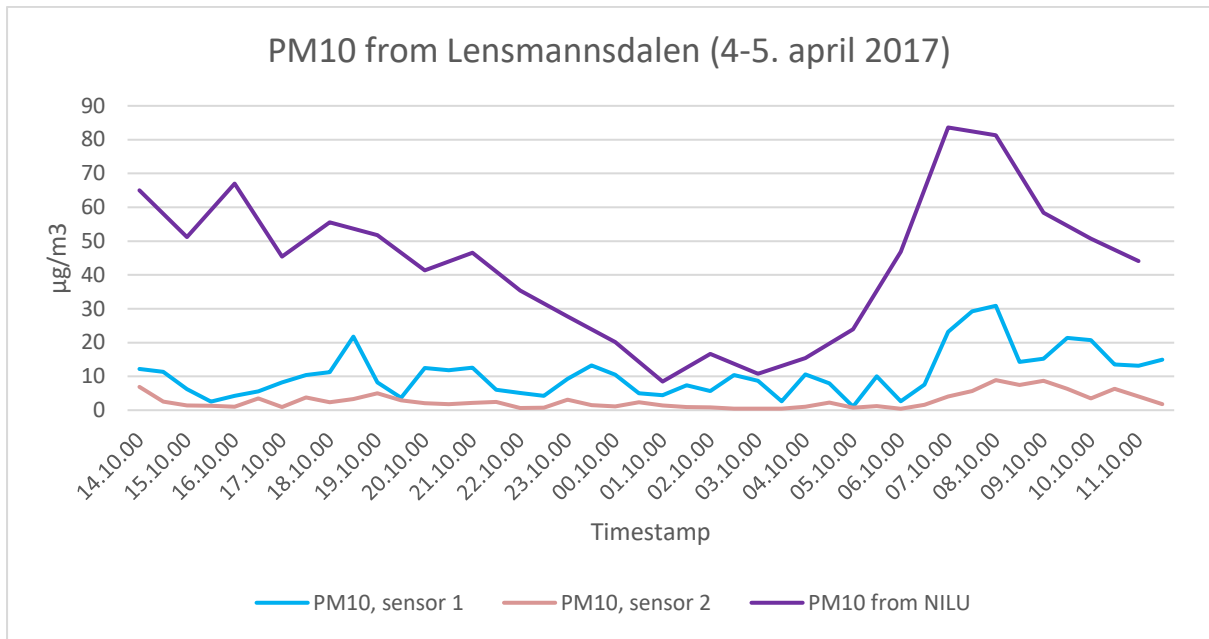


Figure 7.12 PM₁₀ measurements

As Figure 7.12 shows, the readings from the prototype deviate more from NILU's data when the concentration is greater, and is more accurate when it falls below 20 µg/m³. However, it is possible to see a relation in the trends in the time frame from 06:10 – 09:10, the concentration rises for both the prototype and the data from NILU. This can be explained by the increased traffic volume during these hours.

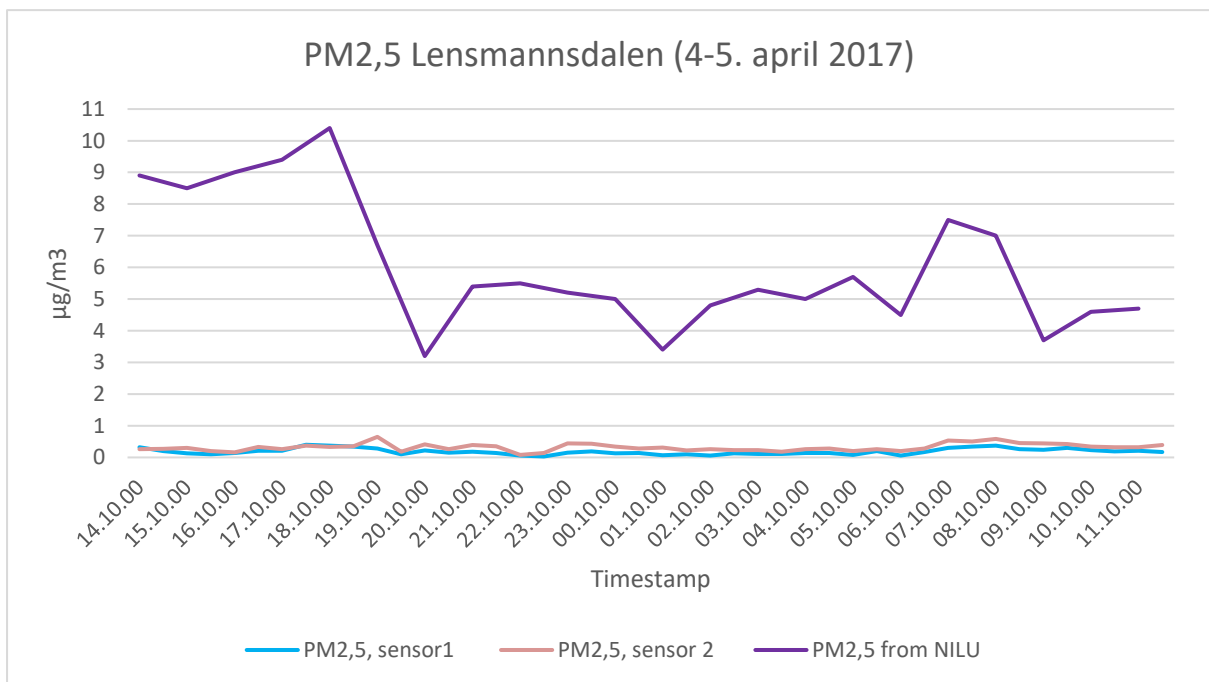


Figure 7.13 PM_{2,5} measurements

7 Mobile Measuring Station Prototype - Solution

Figure 7.13 shows that the $PM_{2.5}$ measurements are less conclusive than the ones for PM_{10} . The reason for such low $PM_{2.5}$ measurements can be many, one theory is the accuracy of the sensors. It is hard to see any meaningful trend, and all measurements from the prototype lies under $1\mu g/m^3$.

What is also interesting to see is that the two sensors behave differently. Finding a solution for this has not been a priority, but one theory can be their position inside the air channel (see Figure 7.9), and their calibration from factory [31].

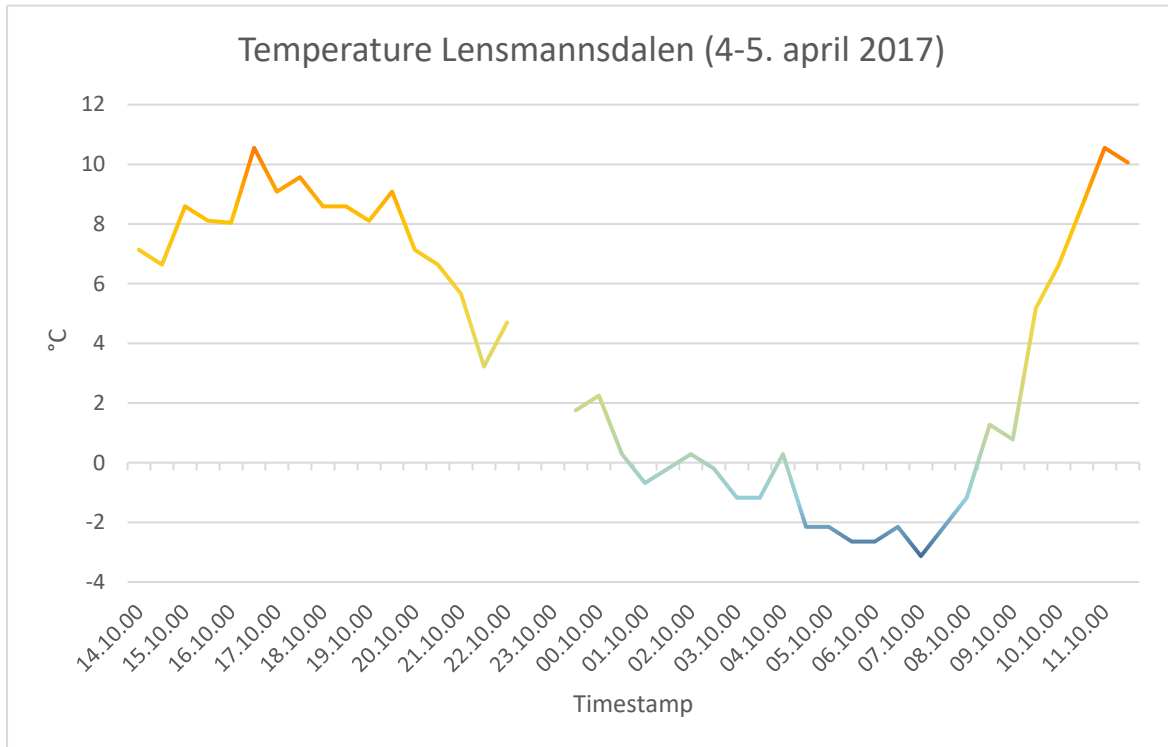


Figure 7.14 Temperature measurements

There was a gap in the temperature measurements where 1 hour of data was not sent to the database, see Figure 7.14. The reason for this is unclear, but can derive from code being processed wrong. The temperature graph however shows an expected trend, that does not deviate far from what is normal around this date.

To summarize, the field test shows that the measurements are not accurate enough compared to the measurements gathered from NILU. The deviation between the measurements are too great, and cannot be used for public information purposes. Possible solutions and improvements for better measurements is described in chapter 8 where this is discussed further.

7.6 Maintenance requirement

The focus from the beginning of the project was to create a simple prototype which did not require a lot of maintenance. However, it is suggested to have a maintenance routine in regards to cleaning the air channel and sensors for any dust that over time will be gathered up inside the channel. This problem cannot be avoided and must therefore be handled.

In terms of software updates and changing the code, no method has been made for doing this without having knowledge about coding. Figure 7.15 illustrates the hard coded parameters for the Arduino microcontroller, to change these permanently the main code must be re-configured.

```

1 #include <GSM.h>
2 #include <ArduinoJson.h>
3 #include <stdio.h>
4 #define GPRS_APN "internet"
5 #define GPRS_LOGIN ""
6 #define GPRS_PASSWORD ""
7 GSMClient client;
8 GPRS gprs;
9 GSM gsmAccess;
10 char apn[] = "internet";
11 char server[] = "luftforurensing.azurewebsites.net";
12 char connectionString[] = "http://luftforurensing.azurewebsites.net/adddata/post HTTP/1.1";
13 char PINNUMBER[] = "1234";
14 int port = 80;
15 boolean notConnected = true; //Variable for knowing if connection is established
16 String stationId1 = "8"; //Micro sensor 1
17 String stationId2 = "9"; //Micro sensor 2
18 String inputPin;
19 String inputInterval;
20 int pm10id = 4;
21 int pm25id = 5;
22 int tempid = 8;
23
24 unsigned long starttime;
25 unsigned long sampletimeMs = 1800000; //Update interval, 30min

```

Figure 7.15 Hard coded parameters for the Arduino Uno

7.7 End-user functionality

Initialization and pre-configuration is needed. Due to the time frame of the project, several features and solutions for the user interface were not implemented. These topics are discussed further in chapter 8.

The Raspberry Pi functions as a user interface for the prototype. It communicates with the Arduino via serial communication over a USB cable. To access the GUI, a monitor must be connected to the Raspberry Pi with a HDMI cable, see Figure 7.16.

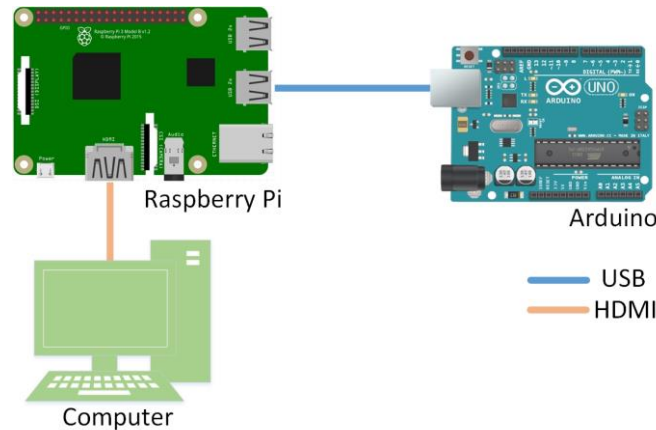


Figure 7.16 Connection between the main components

The GUI is a Universal Windows Platform (UWP) application, which can run on any windows-based device. The application consists of a login window and a main page where parameter changes can be done. In addition, when connected to the GSM network, it is possible to see the status of the Arduino and the last measured values. Details regarding the use of this GUI is described in Appendix K. Figure 7.17 and Figure 7.18 shows the different windows of the GUI, and the design encompasses functions to change parameters, as well as user credentials to access these.

Bruker-ID:

Passord:

Første logg inn.
Bruk default bruker-ID og
passord

Logg inn

Figure 7.17 Log in page for GUI

7 Mobile Measuring Station Prototype - Solution

To ensure that no unauthorized personnel has access to the parameters, a log in feature is implemented as shown in Figure 7.17.

Bruker-ID:	<input type="text" value="admin"/>	Passord:	<input type="text" value="admin"/>
PIN-kode:	<input type="text" value="1234"/>	<input type="button" value="Koble til Arduino ..."/>	
Stasjon-ID: 1	<input type="text" value="8"/>	Sensor 1	PM10: <input type="text"/> µg/m³
Stasjon-ID: 2	<input type="text" value="9"/>	Sensor 2	PM2,5: <input type="text"/> µg/m³
Sendingsinterval:	<input type="text" value="30"/>	min	Temperatur: <input type="text"/> °C
<input type="button" value="Oppdater endringer"/>		<input type="button" value="Logg ut"/>	

Figure 7.18 Main page for GUI

In Figure 7.18, the page for changing parameters is shown. The station-ID's here represents an ID for each of the two air quality sensors installed. It is also possible to monitor the measured values that are retrieved from one of the air sensors, and the temperature as well.

8 Discussion

This chapter gives an insight into the discussion of the project solution carried out by the project group. The project group has reflected on the solution and will mention recommendations for future work on the project in the following chapter.

The project group used the Scrum method for managing project tasks and the overall project progress. The individual project tasks were initially based on the Gantt chart, and WBS diagram. The Scrum tool in VSTS was used frequently during the first period, but during the project it became clear that this was not an ideal method for a project that includes more than just software development. The Scrum tool was used more like a checklist and was used less towards the end. The project group believes the Scrum tool would work better for a project focused on just programming and software development, it is too time consuming compared to the benefits of using it.

A decision had to be made on whether to continue working on the web application and database developed by the master group mentioned in chapter 1. Since the web application they created was in the early stages of development and was created using PHP, it was decided to scrap it and start over with the developing tools described in chapter 1. A choice was made to further develop the database the master group created.

The project group has developed a solution from which the project can be developed further. This has been done by further developing the work from previous groups, as well as developing the MMSP and a new dynamic web application. However, as assumed in the project planning, the measurement of air quality is not accurate enough to be used for public and scientific information, but the system in general works as intended. The web application has a simple and informative design which shows the users data presented in graphs. The navigation between pages is done by using the navigation bar, where the user can select different measurement stations.

Another topic worth mentioning is the time it took to assemble the MMSP. Too much time was spent waiting on funding before parts could be ordered. This, combined with long delivery time on some parts delayed the assembly of the prototype which left little time for field testing and data analysis. Therefore, future thesis assignments at HSN which requires parts to be bought should have funding ready to use at the start of the assignment.

The project partners wanted the graph shown in Figure 6.10 to have an indicator showing the limit value for each pollutant shown in the graph. Various methods and designs for having the indication in form of lines on the graph were tried, but having the indication shown like this might cause confusion. With the time frame given this ended up not being prioritized, but it is something a future project group can consider. A possible solution is described in chapter 8.1.8.

Another request from the project partners, was some sort of functionality allowing them to print either a custom report or a picture of the graph. After some thought, it was decided that spending time on a function that downloads a picture of the graph would be poor time management since this can be done using either print screen or other free Microsoft/Apple software. The custom report feature was not prioritized in this project. During a meeting with the project partners, it was decided that the web application should be focused towards the public to begin with. More on this feature in chapter 8.1.9.

After deciding what type of information to display on the web application, the next step was to find reliable sources for this information. After doing research and getting some guidance from project partners, it was decided to use the sources listed in chapter 5.2. The easiest way to access and use data from external databases is through an API. Unfortunately, data from the web applications chosen are not often used in the way it is in this project. This means that most of them do not have an API. In this case only NILU's database has one, whilst the NRPA has one in development. Since NILU has an open API, their data is used directly on the web application and copied to the local database. The same can be done with the traffic data from NRPA once their API is ready for public use (see chapter 8.1.11). The remaining sources are linked to on the web application.

When the MMSP was finished, it appeared that some of the solutions implemented could have been done differently. The overall goal was to make a small and cheap prototype that proves the concept of a MMSP in a compact form. For the prototype to gather data and transmitting them to the web application, several decisions regarding equipment and functions were made. Nevertheless, the prototype can be used together with the existing measuring stations as a complete system, but the measurements from the prototype must be regarded as too inaccurate compared to the existing stations. The prototype contains two microcontrollers as described earlier in chapter 7.6. A Raspberry Pi for GUI handling and an Arduino that handles the sensors and the GSM shield. For future works regarding the prototype, several features and solutions are described in detail in chapter 8.1.

8.1 Future Work

As this project solution is part of a larger project lead by Tel-Tek on behalf of Porsgrunn municipality, the bachelor group believes that it is important to include a chapter on what can be done to further develop the solution in future projects. The following subchapters gives an overview of work that can be done on the prototype and the web application.

8.1.1 Improvements on the MMSP's air channel

How the air flows through the air channel and through the sensors has an important impact on how the sensors would register particles. No work has been done regarding optimization of the air flow. This should be looked further into in terms of getting more accurate sensor data. Sensor positioning, type of fan and air vents should be considered, as well as the shape of the air channel. The initial plan was to use a circular plastic pipe, but this was not done due to limited time and resources.

8.1.2 MMSP mounting solution

A solution for how the prototype should be mounted to objects such as lamp posts and house walls should be developed. It should be fast and easy to install/remove with a limited number of tools.

8.1.3 Sensor calibration

Calibration methods and mathematical functions related to sensor accuracy, have not been prioritized. By doing several field tests for data gathering it should be possible to create a

spread sheet to see how the data diverts from the reference data. NILU can be a good source to contact regarding this subject.

A second type of air quality sensor, which is not tested on the prototype, was bought. This sensor should be tested further to see if it is more accurate than the sensors already in use [37].

8.1.4 MMSP power source

The prototype is currently powered by a connection to electric infrastructure. Another power source that should be considered is a battery that is being charged by a solar panel.

Depending on weather conditions and power consumption, this option could potentially power the prototype just as reliably as the current method. Using a solar panel will make it independent of electrical infrastructure at the deployment site.

8.1.5 MMSP memory

A disadvantage with the microcontrollers is the memory. The Arduino has a volatile memory which means that if the power disappears, all the changes made by the operator will be lost and settings will be set back to start-up settings. The optimal solution for this problem is sending a query to the database that retrieves the settings, it should be possible to alter these settings from the web application.

8.1.6 GPS module

The MMSP should be equipped with a GPS module. Currently, an operator must plot the location where the prototype is placed. A better solution would be a GPS module that finds the location and transmits the coordinates to the web application automatically.

8.1.7 Advised solutions for MMSP

This chapter presents two alternative solutions for the MMSP. Each solution is based on a single microcontroller.

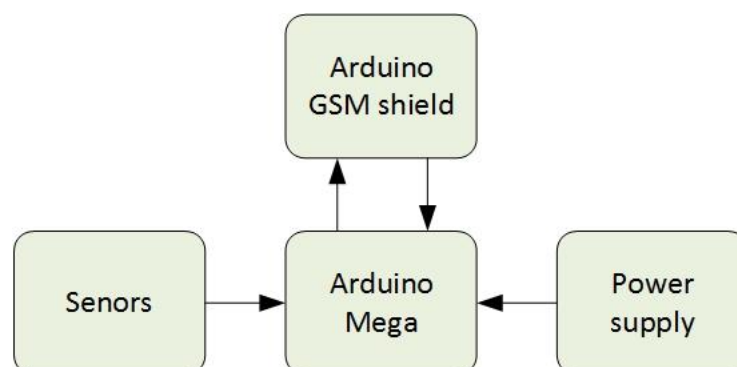


Figure 8.1 Arduino solution

8 Discussion

The Arduino solution is the simplest in terms of hardware. The Arduino Mega is recommended because it has up to four serial communication ports, which can be helpful in terms of process monitoring and initial configuration [38]. In addition, it has more GPIO's than the Arduino Uno and therefore more equipment can be connected. The Arduino also has analog I/O's that can be directly connected to equipment that uses analog signals. As Figure 8.1 illustrates, the GSM shield in this solution would have to transmit and receive data. The idea is that the prototype can be remotely configured over the GSM network, and store parameters in a database rather than locally in the prototype. This must be done in the Arduino code, and a web application would act as a GUI.

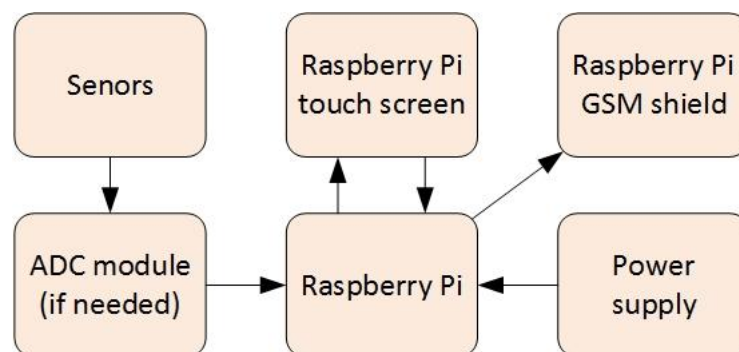


Figure 8.2 Raspberry Pi solution

The Raspberry Pi solution requires more components as the GUI is based on a touch screen mounted on the prototype as seen in Figure 8.2. In addition, the Raspberry Pi does not have analog I/O's and must therefore have an ADC-module if the sensors use analog signals. The sensors currently used for air quality have digital signals, and could be used with the Raspberry Pi. However, the temperature sensor used has an analog signal, and would not work directly with the Raspberry Pi. The Raspberry Pi has a connection for HDMI-based screens, and this is ideal for a touch screen that would work as a GUI. By using the Windows IoT OS the GUI can be developed in Visual Studio with the C# language. In terms of GSM communication, it is advised to find a GSM shield that can be directly mounted on the Raspberry Pi.

It is also possible to have remote configuration with the Raspberry Pi solution. In this case, a touch screen is not needed and the GSM shield would both send and receive data. Configuration would be done in a web application.

8.1.8 Indication for amount of times the limit values have been exceeded

As mentioned in chapter 8.1, the project partners wanted some form of indication for the amount of times the limit values for the different pollutants are exceeded. Figure 8.3 illustrates an example of this.





Stoff				
PM 10	3	0	5	7
PM 2.5	9	6	9	0
SO2	0	9	1	6
NO2	14	12	0	16

Figure 8.3 example of limit value exceeded graph

Since the web application has an option in the admin control panel for setting limit values for the different pollutants in the database, implementing a function for automatic counting will not be difficult. The different colour codes can be found in NILUs API.

8.1.9 Downloading automatically generated reports from the web application

A report would ideally consist of some standard text with values that get their data from the database, the exceeded limit values info from chapter 8.1.8 and the graph in Figure 6.10. The data from the database could be retrieved using models in the code, doing it this way would make it possible to let the user choose a date range to get values from.

8.1.10 Rerouting from the home page map

Having the option to click the names of the stations shown in Figure 6.15 and being redirected to that stations information site would improve the usability of the site. This is a small feature that has not been prioritized. The ID parameter mentioned in chapter 6.3.2 is created with this in mind.

8.1.11 Data from NRPA

Data from the NRPA is currently not used on the web application, or copied to the local database due to them not having a public API. Both solutions can be implemented as soon as this API is done. Being able to compare traffic data with the amount of pollution in different areas would be an improvement to web application. Traffic data should be presented in the same graph as measured pollution.

8.1.12 Alert function - power failure in the MMSP

An improvement for the mobile measuring station would be a function that sends a text or email to a specified phone number or email address, if something was to happen to the station. There are two different ways to solve this depending on whether the system is sending a text or email. Sending a text message requires the use of the GSM module, this means that the mobile station needs a battery in case of a power outage. Sending an email can be done by implementing code to the web application. Configuration options for this should also be added to the admin page on the web application to easily change the email or phone number of the recipient.

9 Summary

- The bachelor thesis, web application and Mobile Measuring Station Prototype adds up to a solution that meets the primary goal defined by the group at the start of the bachelor project.
- A compact and low-cost MMSP has been designed and built. The MMSP can be placed anywhere GPRS coverage and electrical infrastructure is available. More accurate measurement data can be achieved by investing in better air quality sensors, this will have a large impact on the price of the build.
- Data from the MMSP is sent automatically to the database over the GPRS network.
- The web application presents air pollution data in an informative and easy to grasp manner using map and graph functions.
- Data from existing air quality stations is gathered automatically from NILU's API.
- The database designed by an earlier project has been modified for use with the web application developed in this project.
- Cost estimates of local website hosting compared to a cloud solution have been performed. The cost difference between local and cloud hosting is small, but the chance of unforeseen expenses is greater in a local solution.

References

- [1] *Forskrift om begrensning av forurensning*, 2004
- [2] Hydro. (2016). *1987: Industriens skyggeside*. Available: <http://www.hydro.com/no/hydro-i-norge/Om-Hydro/Var-historie/1978---1990/1987-Industriens-skyggeside/>
- [3] T. Bakke, G. Borgersen and B. A. Beylich, “Monitoring in the Grenland fjords 2012: Sediments and bottom fauna”, NIVA, Oslo, Norway, M-9/2013, nov.13, 2013.
- [4] E. Eek, M. Schaaning, G. Cornelissen, “Thin layer capping of contaminated sediments - Monitoring of four test fields in the Grenlandfjords”, NGI. NIVA. Univ. Stockholm, Oslo, Norway, M-219/2014, Oct.27, 2014.
- [5] E. Haugen, “Helsefarlig luft I Grenland I dag morges,” *Varden*, 28. November 2016, [Online]. Available: <http://www.varden.no/nyheter/helsefarlig-luft-i-grenland-i-dag-morges-1.1599151>. [Retrieved 30.03.2017]
- [6] E. Edvardsen and H. Solum, “Nå har Grenland den giftigste lufta I hele landet,” *Telemarksavisa*, 23. November 2015, [Online]. Available: <https://www.ta.no/nyheter/skien/porsgrunn/na-har-grenland-den-giftigste-lufta-i-hele-landet/s/5-50-149724>. [Retrieved 30.03.2017]
- [7] Luftkvalitet.info. (2017). *Frontpage*. Available: <http://www.luftkvalitet.info/home.aspx>
- [8] Norskeutslipp.no. (2017). *Landbasert industri*. Available: <http://www.norskeutslipp.no/no/Landbasert-industri/?SectorID=600>
- [9] A.Bartonova. (2017). *Det er noe i luften*. Available: <http://www.nilu.no/Portals/0/Files/News/NILU-mikrosensorseminar-jan2017-allepres-small.pdf>
- [10] S. Bratland, B. R. Bay, N. Landvik and S. Guttu, “Guidance to pollution regulations chapter 7 about ambient air quality”, NEA, Oslo, Norway, M-413, 2015.
- [11] L. Tisopulos. (2014). *Air Quality Sensor Performance Evaluation Center (AQ-SPEC)*. Available: <http://www.baaqmd.gov>
- [12] A. Z. Gjerset and L. Ang, “Development of a database system for environmental and public health information”, HSN, Porsgrunn, Norway, SIV-14-16, Nov.25,2016.
- [13] Information management system for environmental and public health information. (2017). *Grenland Grønn*. Available: <http://luftforurensing.azurewebsites.net/>
- [14] Miljødirektoratet. (2017). *Vannmiljø*. Available: <http://vanmiljo.miljodirektoratet.no>
- [15] Miljødirektoratet. (2017). *Grunnforurensing*. Available: <http://grunnforurensning.miljodirektoratet.no>
- [16] Lenovo. (2017). *Lenovopress*. Available: <http://lenovopress.com/lp0553-lenovo-thinkserver-rs160>
- [17] Komplet Services AS. (2017). *Komplett*. Available: <http://www.komplett.no/product/916455/pc-nettbrett/server-tilbehoer/server/serve/lenovo-thinkserver-rs160#>

- [18] Komplet Services AS. (2017). *Komplett*. Available: <http://www.komplett.no/product/834289/pc-nettbrett/server-tilbehoer/servere-harddisker/lenovo-thinkserver-35-1tb-72k#>
- [19] w3schools.com. (2017). *ASP.NET Razor – Markup*. Available: https://www.w3schools.com/asp/razor_intro.asp
- [20] Bootstrap. (2017). *Designed for everyone, everywhere*. Available: <http://getbootstrap.com/>
- [21] Open Source Initiative. (2017). *The MIT License*. Available: <https://opensource.org/licenses/MIT>
- [22] Open Source Initiative. (2017). *The 3-Clause BSD License*. Available: <https://opensource.org/licenses/BSD-3-Clause>
- [23] Creative Commons. (2017). *Attribution 3.0 Unported*. Available: <https://creativecommons.org/licenses/by/3.0/legalcode>
- [24] Microsoft. (2017). *Bing maps for enterprise*. Available: <https://www.microsoft.com/maps/faq.aspx>
- [25] Google. (2017). *Google Maps APIs*. Available: <https://developers.google.com/maps/pricing-and-plans/#details>
- [26] Telia. (2017). *Dekningskart*. Available: <https://telia.no/dekningskart>
- [27] R. Johnsen, P.B. Andersen and G. Stette. (2009). “GPRS” in *Store norske leksikon*. Retrieved from: <https://snl.no/GPRS>
- [28] T. Hansen. (2016). “GSM” in *Store norsk leksikon*. Retrieved from: <https://snl.no/GSM>
- [29] Arduino. (2017). *Arduino GSM shield V2*. Available: <https://www.arduino.cc/en/Main/ArduinoGSMShield>
- [30] Adafruit. (2012). *Using a temp sensor*. Available: <https://learn.adafruit.com/tmp36>
- [31] Elfa distrelec. (2017). *Grove – Støvsensor, Arduino, Raspberry Pi, Beaglebone, Edison, Launchpad, Mbed, Galiel*. Available: <https://www.elfadistrelec.no/en/grove-dust-sensor>
- [32] Arduino pollution. (2016). *Arduino air quality*. Available: <http://arduinoairpollution.altervista.org/progetto/>
- [33] The Shinyei experiment. (2017). *Real-time Air Quality readings from Beijing*. Available: <http://aqicn.org/sensor/shinyei/>
- [34] J. Arling, K. O’Connor and M. Mercieca. (2010). *Air Quality Sensor Network for Philadelphia*. Available: <http://www.fijnstofmeter.com/documentatie/Data-Validation.pdf>
- [35] Biltema. (2017). *MC-Batteri, HD 12V 30AH*. Available: <http://www.biltema.no/no/Bil---MC/MC/Reservedeler/Batteri/MC-BATTERI-HD-12V-30AH-2000036063/>
- [36] H. Dvergstad. (2017). “Internett” in *Store norske leksikon*. Retrieved from <https://snl.no/Internett>
- [37] Aqicn. (2017). *The Plantower PMS3003 Air Quality Sensor experiment*. Available: <http://aqicn.org/sensor/pms3003/>
- [38] Arduino. (2017). *Arduino mega*. Available: <https://www.arduino.cc/en/Main/arduinoBoardMega>

Appendices

Appendix A Project assignment.

Appendix B Project goals.

Appendix C WBS

Appendix D Gantt chart

Appendix E Traffic data mail chain

Appendix F Raspberry pi specifications

Appendix G Arduino Uno specifications

Appendix H Arduino GSM shield 2 specifications

Appendix I Grove dust sensor

Appendix J Electrical wiring diagram

Appendix K User manual mobile measuring station

University College of South East Norway

PRH612 Bacheloroppgaven

Title: Information Management System for Environmental and Public Health Information

Supervisors: Hans-Petter Halvorsen, Nils-Olav Skeie

External Partners: Tel-Tek, Porsgrunn kommune, Sykehuset Telemark, Avdeling for Arbeidsmedisin

Task Description:

Utvikling av infrastruktur og datasystem for innhenting, registrering og analyse av miljø- og folkehelseinformasjonsdata (English: Environmental and Public Health Information) for Porsgrunn og Grenlandsdistriktet.

Typiske data er for registrering og analyse er luftforurensning og luftkvalitet, utslipp i forbindelse med industri, utslipp i forbindelse med biler, antall biler, m.m. Design og utvikling av målestasjoner i forbindelse med dette, eventuelt integrasjon med eksisterende målestasjoner.

Systemet vil være sentralt i forbindelse med utvikling av ny industri, forskning og utvikling.

Opgaven omfatter følgende sentrale elementer:

- Kartlegging av data er for registrering og analyse
- Planlegging og systemdesign
- Databasemodellering, design og implementering
- Webdesign og webutvikling, e.g., ASP.NET 5/ASP.NET MVC 6 and C# Programming
- Big data – innsamling og analyse av store datamengder
- Datainnsamling og forvaltning. Utvikling av moduler fleksible import- og eksportmuligheter, inkludert REST APIer, m.m.
- Analyse, statistikk og presentasjon av data. Generering av rapporter i forbindelse med analyse og rapportering av data til myndigheter og andre oppdragsgivere. I tillegg til ferdigdefinerte rapporter, bør det være mulig for brukeren å lage egne skreddersydde rapporter.
- Design og utvikling av målestasjoner i forbindelse med dette, eventuelt integrasjon med eksisterende målestasjoner.
- Infrastruktur, servere og nettverksinfrastruktur
- Utvikling av relevante beslutningsstøtteverktøy
- Vurdering av ulike former for lagring, som SQL databaser og NoSQL databaser, m.m.
- Hosting av dataene, lokalt eller i skyen?
- Sikkerhetsaspekter (datasikkerhet, backup, lastbalansering, redundans, osv.)

Dette er et foreløpig utkast til innhold og oppgaver, mer konkrete punkter vil bli laget frem mot prosjektstart. GUI skal i utgangspunktet være på norsk, evt. mulighet for å bytte språk.

Project goals

Primary goals

Develop a complete system for automatic data retrieval of environmental air pollutants and presentation of the data in an informative and easy to grasp manner. The system should contain a web page for displaying information, a database for storing data, and measuring stations for data gathering. A mobile measuring station prototype will be developed. Automatic data gathering from the measuring stations should be the preferred solution.

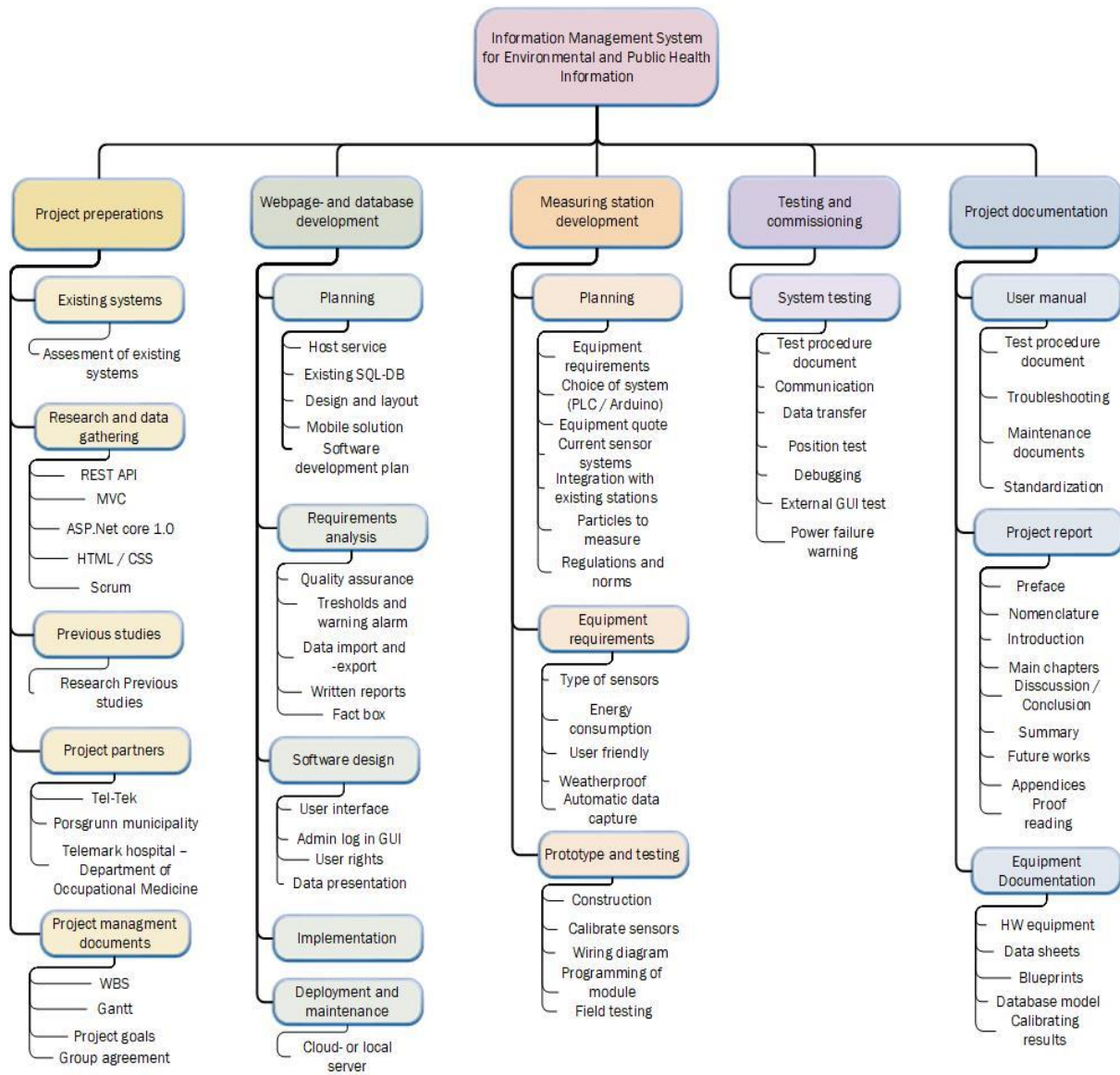
The group will consider cost aspects of data and website hosting and maintenance, as well as the cost of building the mobile measuring station prototype

Competence goals

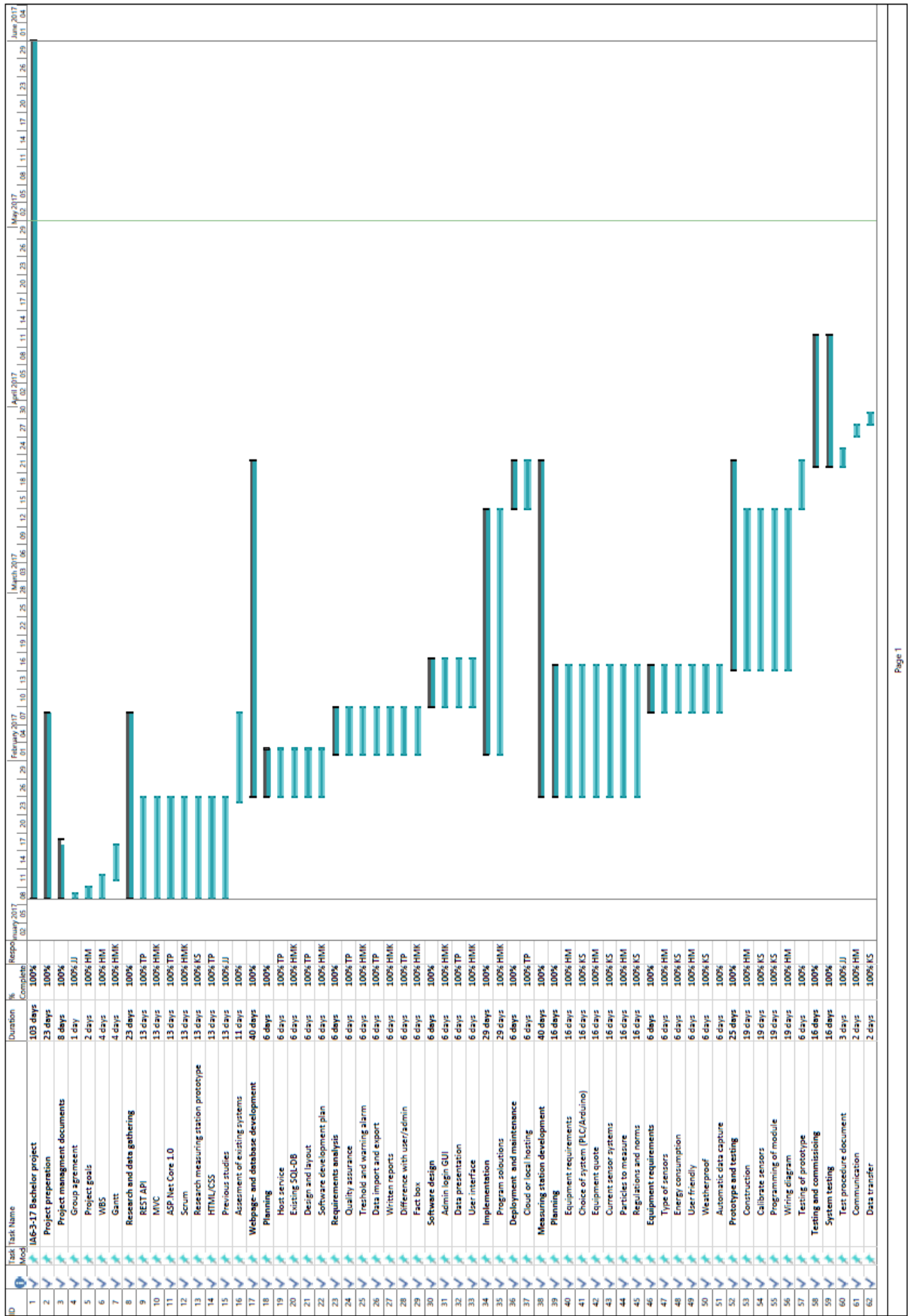
Increased knowledge with the ASP.Net core 1.0 framework and web-based applications. The project group must also learn how to work by using the Scrum project management method. An insight in sensor technology and measuring methods of air pollutants must also be acquired.

The project group must gain some understanding of existing measurement systems, and look into the possibilities of implementing newer technology on existing installations. It must also be considered which type of database solution that has most benefits in terms of vulnerability, cost efficiency and data storage. A user-friendly GUI designed for non-technical users must be implemented.

Appendix C WBS



Appendix D Gantt chart



Mail chain with the NPRA about live traffic data



IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

"live" data av trafikk i Grenland.

10 e-poster

IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>
Til: trafikdata@vegvesen.no

9. februar 2017 kl. 09:20

Hei.

Vi er en bachelorgruppe i fra Høgskolen i Sørøst-Norge, som har fått i oppgave å lage en nettside for visning av (hovedsaklig) luftforurensning i Grenland. Vi har også blitt spurt om å hente inn data, som kan vise grunner til at luftforurensningen er som den er, og vise disse på denne nettsiden.

Er det noen mulighet for å kunne hente inn data "live"? Så det sto at det var kontinuerlig registrering ved trafikkregistreringspunktene. Evnt. hvor ofte blir disse dataene gjort tilgjengelig?

Er det noen mulighet for at vi kunne fått data ifra disse trafikkregistreringspunktene vha. REST API eller annen måte?

Mvh.
IA6-3-17
v. Hans Martin Kristensen

Trafikkdata <trafikdata@vegvesen.no>
Til: IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

9. februar 2017 kl. 09:38

Hei

Vi har ikke sanntidsdata for trafikdata ut til publikum enda.

Der jobbes med å få dette til, men antar ikke tidsnok til deres oppgave, da jeg regner med det vil ta noen år før vi har dette på plass.

Data fra kontinuerlige registreringspunkter er tilgjengelig etter den 10. i påfølgende måned.

Med hilsen
Iiril Helen Ulvøen

Seksjon: Geodataseksjonen
Postadresse: Statens vegvesen Region vest, Askedalen 4, 6863 LEIKANGER
Besøksadresse: Nygårdsgaten 112, BERGEN
Mobil: +47 98872292 **e-post/Lync:** iril.ulvoen@vegvesen.no
www.vegvesen.no **e-post:** firmapost-vest@vegvesen.no

Tenk miljø - spar papir. Trenger du å skrive ut denne e-posten?

Fra: IA6-3-17 Bachelor [<mailto:bachelor.ia6.3.17@gmail.com>]
Sendt: 9. februar 2017 09:21

Til: Trafikkdata <trafikkdata@vegvesen.no>

Emne: "live" data av trafikk i Grenland.

[Sikkert tekst skjult]

IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

9. februar 2017 kl. 10:04

Til: Trafikkdata <trafikkdata@vegvesen.no>

Hei, og tusen takk for svar!

Må disse dataene da manuelt hentes ut fra <http://www.vegvesen.no/fag/Trafikk/Trafikkdata/Trafikkregistreringer?>

Og ligger de kun som PDF?

Eller finnes det en måte man kan hente ut disse dataene på, f.eks vha. REST API?

Mvh.

IA6-3-17

v. Hans Martin

[Sikkert tekst skjult]

Trafikkdata <trafikkdata@vegvesen.no>

9. februar 2017 kl. 10:21

Til: IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

Hei

Ja, data må hentes manuelt ut har ikke mulighet for REST API

Vi har format csv, excel el. pdf – for rådata bare excel.

Med hilsen

Iiril Helen Ulvøen

Seksjon: Geodataseksjonen

Postadresse: Statens vegvesen Region vest, Askedalen 4, 6863 LEIKANGER

Besøksadresse: Nygårdsgaten 112, BERGEN

Mobil: +47 98872292 **e-post/Lync:** iril.ulvoen@vegvesen.no

www.vegvesen.no **e-post:** firmapost-vest@vegvesen.no

Tenk miljø - spar papir. Trenger du å skrive ut denne e-posten?

Fra: IA6-3-17 Bachelor [mailto:bachelor.ia6.3.17@gmail.com]

Sendt: 9. februar 2017 10:04

Til: Trafikkdata <trafikkdata@vegvesen.no>

Emne: Re: "live" data av trafikk i Grenland.

[Sikkert tekst skjult]

IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

9. februar 2017 kl. 10:30

Til: Trafikkdata <trafikkdata@vegvesen.no>

Okei, takk for det.

Har du en link til hvor vi kan finne på excel format?

Fant bare PDF på den siden jeg linka.

Mvh.
IA6-3-17
v. Hans Martin

[Sitert tekst skjult]

Trafikkdata <trafikkdata@vegvesen.no>
Til: IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

9. februar 2017 kl. 10:32

Hei

Beklager, jeg må ta ut data til dere om dere skal ha annet enn pdf.

Hvilke punkter ønsker dere data fra?

Mvh Iril

Fra: IA6-3-17 Bachelor [mailto:bachelor.ia6.3.17@gmail.com]
Sendt: 9. februar 2017 10:31

[Sitert tekst skjult]

[Sitert tekst skjult]

IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>
Til: Trafikkdata <trafikkdata@vegvesen.no>

9. februar 2017 kl. 10:38

Åja!

Vi skulle gjerne hatt all data på målestasjonene i Grenland(Skien/Porsgrunn), hvis det hadde vært mulig.

Hvis vi f.eks skulle hatt denne dataen en gang i mnden, når den blei klar, måtte vi da ha sendt deg en epost for å spørre om siste mndes data?

Mvh.
Hans Martin
[Sitert tekst skjult]

Trafikkdata <trafikkdata@vegvesen.no>
Til: IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

13. februar 2017 kl. 10:57

Hei

Her kommer MDT og ÅDT fra kontinuerlige trafikkregistreringsstasjoner Skien og Porsgrunn.

Med hilsen
Iiril Helen Ulvøen

Seksjon: Geodataseksjonen

Postadresse: Statens vegvesen Region vest, Askedalen 4, 6863 LEIKANGER

Besøksadresse: Nygårdsgaten 112, BERGEN

Mobil: +47 98872292 **e-post/Lync:** iril.ulvoen@vegvesen.no

www.vegvesen.no **e-post:** firmapost-vest@vegvesen.no

Tenk miljø - spar papir. Trenger du å skrive ut denne e-posten?

Fra: IA6-3-17 Bachelor [mailto:bachelor.ia6.3.17@gmail.com]

Sendt: 9. februar 2017 10:38

[Sikkert tekst skjult]

[Sikkert tekst skjult]

IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

13. februar 2017 kl. 11:42

Til: Trafikkdata <trafikkdata@vegvesen.no>

Hei

Glemte du å legge ved vedleggene? Fikk ingenting.

Mvh

IA6-3-17

v. Hans Martin Kristensen

[Sikkert tekst skjult]

Trafikkdata <trafikkdata@vegvesen.no>

13. februar 2017 kl. 12:04

Til: IA6-3-17 Bachelor <bachelor.ia6.3.17@gmail.com>

Obs, beklager her kommer filen

Mvh Iiril

Fra: IA6-3-17 Bachelor [mailto:bachelor.ia6.3.17@gmail.com]

Sendt: 13. februar 2017 11:42

[Sikkert tekst skjult]

[Sikkert tekst skjult]

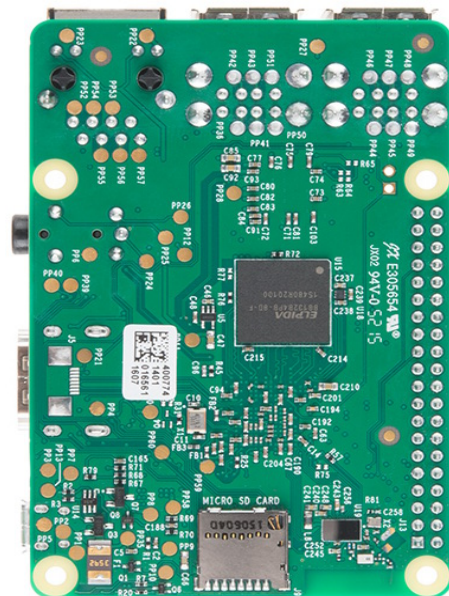
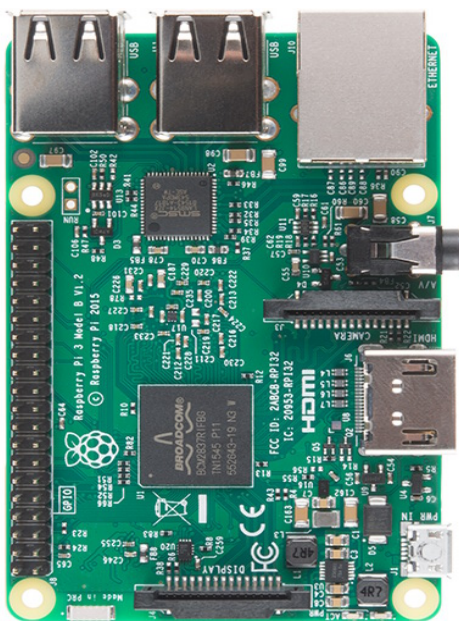
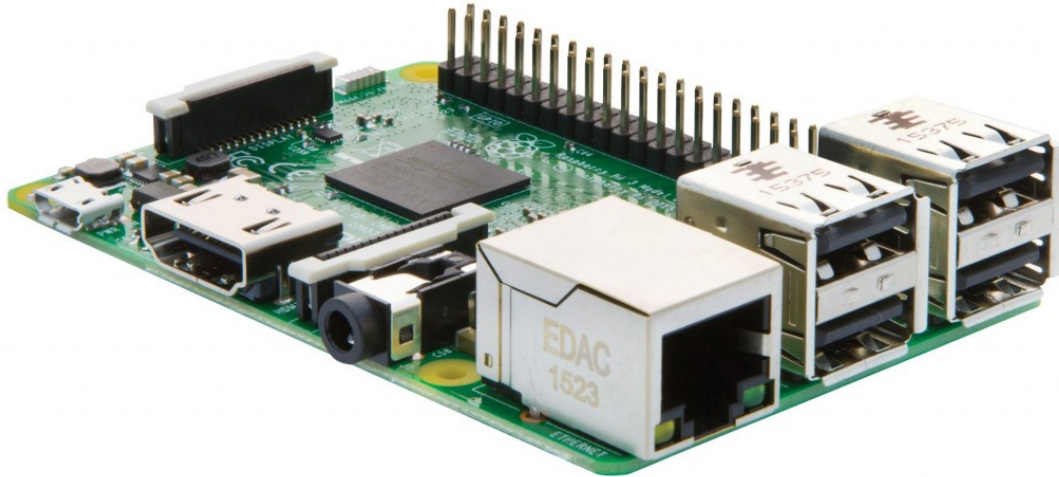


ÅDT og MDT Porsgrunn - Skien fra kontinuerlige registreringspunkter.xls

64K

Raspberry Pi specifications sheet

RASPBERRY PI 3 MODEL B



Product Name: RASPBERRYPI3-MODB-1GB

Technical Specification:

Processor

- Broadcom BCM2387 chipset.
- 1.2GHz Quad-Core ARM Cortex-A53 (64Bit)

802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)

- IEEE 802.11 b / g / n Wi-Fi. Protocol: WEP, WPA WPA2, algorithms AES-CCMP (maximum key length of 256 bits), the maximum range of 100 meters.
- IEEE 802.15 Bluetooth, symmetric encryption algorithm Advanced Encryption Standard (AES) with 128-bit key, the maximum range of 50 meters.

GPU

- Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode.
- Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure

Memory

- 1GB LPDDR2

Operating System

- Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT

Dimensions

- 85 x 56 x 17mm

Power

- Micro USB socket 5V1, 2.5A

Connectors:

Ethernet

- 10/100 BaseT Ethernet socket

Video Output

- HDMI (rev 1.3 & 1.4)
- Composite RCA (PAL and NTSC)

Audio Output

- Audio Output 3.5mm jack
- HDMI
- USB 4 x USB 2.0 Connector

GPIO Connector

- 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
- Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines

Camera Connector

- 15-pin MIPI Camera Serial Interface (CSI-2)

Display Connector

- Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane

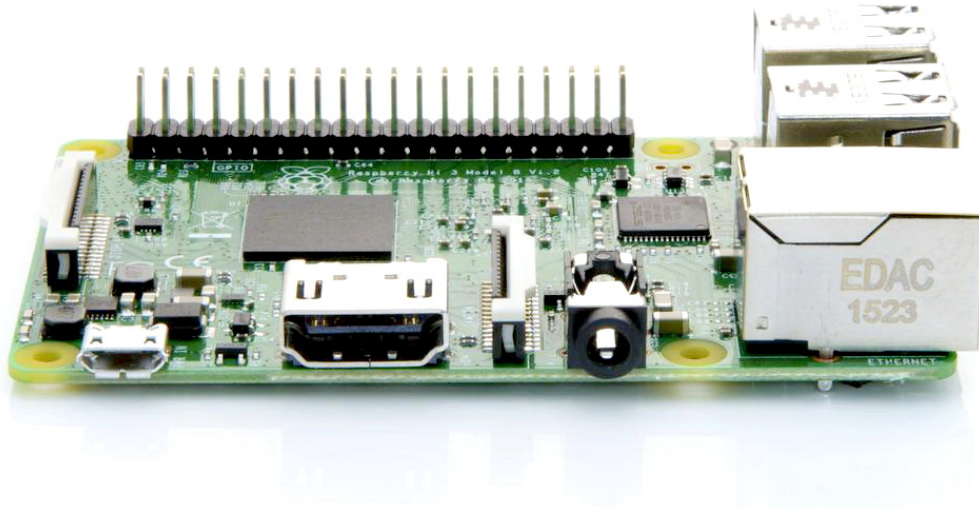
Memory Card Slot

- Push/pull Micro SDIO

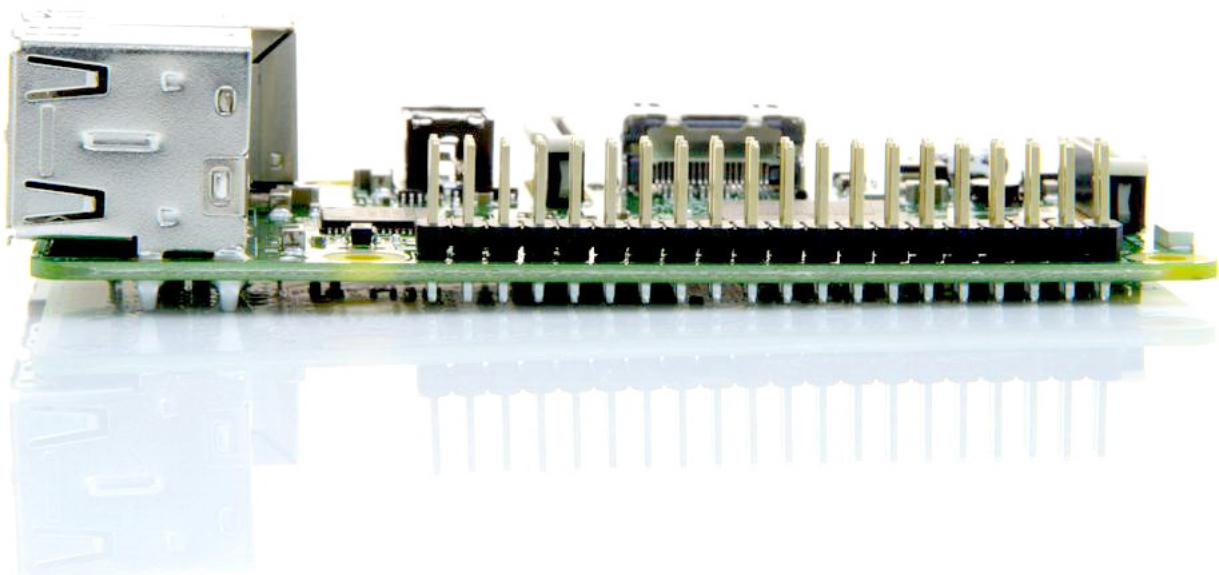
<http://uk.farnell.com/buy-raspberry-pi>

<http://www.newark.com/buy-raspberry-pi>

The GPU provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode and is capable of 1Gpixel/s, 1.5Gtexel/s or 24 GFLOPs of general purpose compute. What's that all mean? It means that if you plug the Raspberry Pi 3 into your HDTV, you could watch BluRay quality video, using H.264 at 40Mbits/s



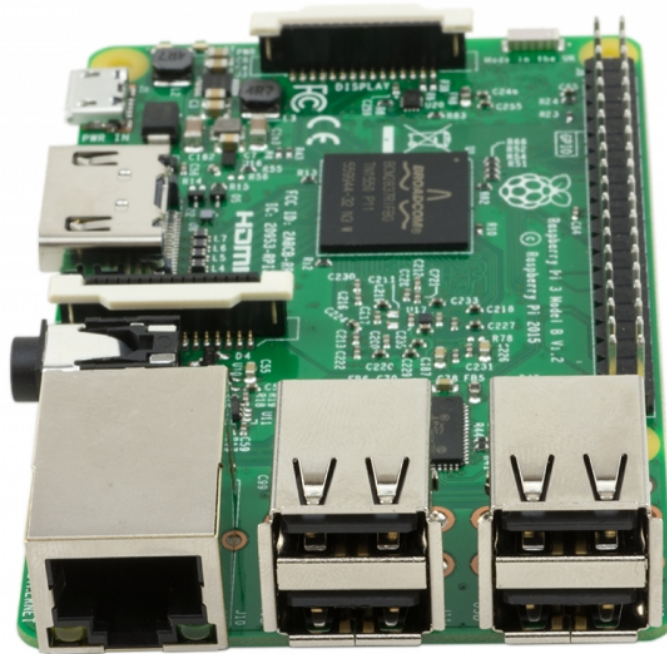
The biggest change that has been enacted with the Raspberry Pi 3 is an upgrade to a next generation main processor and improved connectivity with Bluetooth Low Energy (BLE) and BCM43143 Wi-Fi on board. Additionally, the Raspberry Pi 3 has improved power management, with an upgraded switched power source up to 2.5 Amps, to support more powerful external USB devices.



<http://uk.farnell.com/buy-raspberry-pi>

<http://www.newark.com/buy-raspberry-pi>

The Raspberry Pi 3's four built-in USB ports provide enough connectivity for a mouse, keyboard, or anything else that you feel the RPi needs, but if you want to add even more you can still use a USB hub. Keep in mind, it is recommended that you use a powered hub so as not to overtax the on-board voltage regulator. Powering the Raspberry Pi 3 is easy, just plug any USB power supply into the micro-USB port. There's no power button so the Pi will begin to boot as soon as power is applied, to turn it off simply remove power. The four built-in USB ports can even output up to 1.2A enabling you to connect more power hungry USB devices (This does require a 2Amp micro USB Power Supply)



On top of all that, the low-level peripherals on the Pi make it great for hardware hacking. The 0.1" spaced 40-pin GPIO header on the Pi gives you access to 27 GPIO, UART, I²C, SPI as well as 3.3 and 5V sources. Each pin on the GPIO header is identical to its predecessor the Model B+.

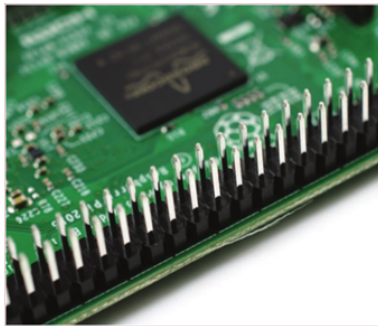
SoC

Built specifically for the new Pi 3, the Broadcom BCM2837 system-on-chip (SoC) includes four high-performance ARM Cortex-A53 processing cores running at 1.2GHz with 32kB Level 1 and 512kB Level 2 cache memory, a VideoCore IV graphics processor, and is linked to a 1GB LPDDR2 memory module on the rear of the board.



GPIO

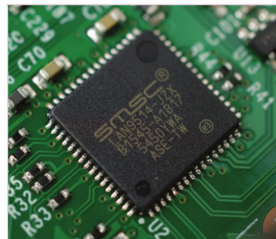
The Raspberry Pi 3 features the same 40-pin general-purpose input-output (GPIO) header as all the Pis going back to the Model B+ and Model A+. Any existing GPIO hardware will work without modification; the only change is a switch to which UART is exposed on the GPIO's pins, but that's handled internally by the operating system.



Pin#	NAME		NAME	Pin#
01	3.3v DC Power	●	DC Power 5v	02
03	GPIO2 (SDA1, I2C)	●	DC Power 5v	04
05	GPIO3 (SCL1, I2C)	●	Ground	06
07	GPIO4 (GPIO_GCLK)	●	(TXD0) GPIO14	08
09	Ground	●	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	●	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	●	Ground	14
15	GPIO22 (GPIO_GEN3)	●	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	●	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	●	Ground	20
21	GPIO9 (SPI_MISO)	●	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	●	(SPI_CEL_N) GPIO18	24
25	Ground	●	(SPI_CEL_N) GPIO17	26
27	ID_SD (I2C ID EEPROM)	●	(I2C ID EEPROM) ID_SC	28
29	GPIO15	●	Ground	30
31	GPIO16	●	GPIO12	32
33	GPIO13	●	Ground	34
35	GPIO19	●	GPIO16	36
37	GPIO26	●	GPIO20	38
39	Ground	●	GPIO21	40

USB chip

The Raspberry Pi 3 shares the same SMSC LAN9514 chip as its predecessor, the Raspberry Pi 2, adding 10/100 Ethernet connectivity and four USB channels to the board. As before, the SMSC chip connects to the SoC via a single USB channel, acting as a USB-to-Ethernet adaptor and USB hub.



Antenna

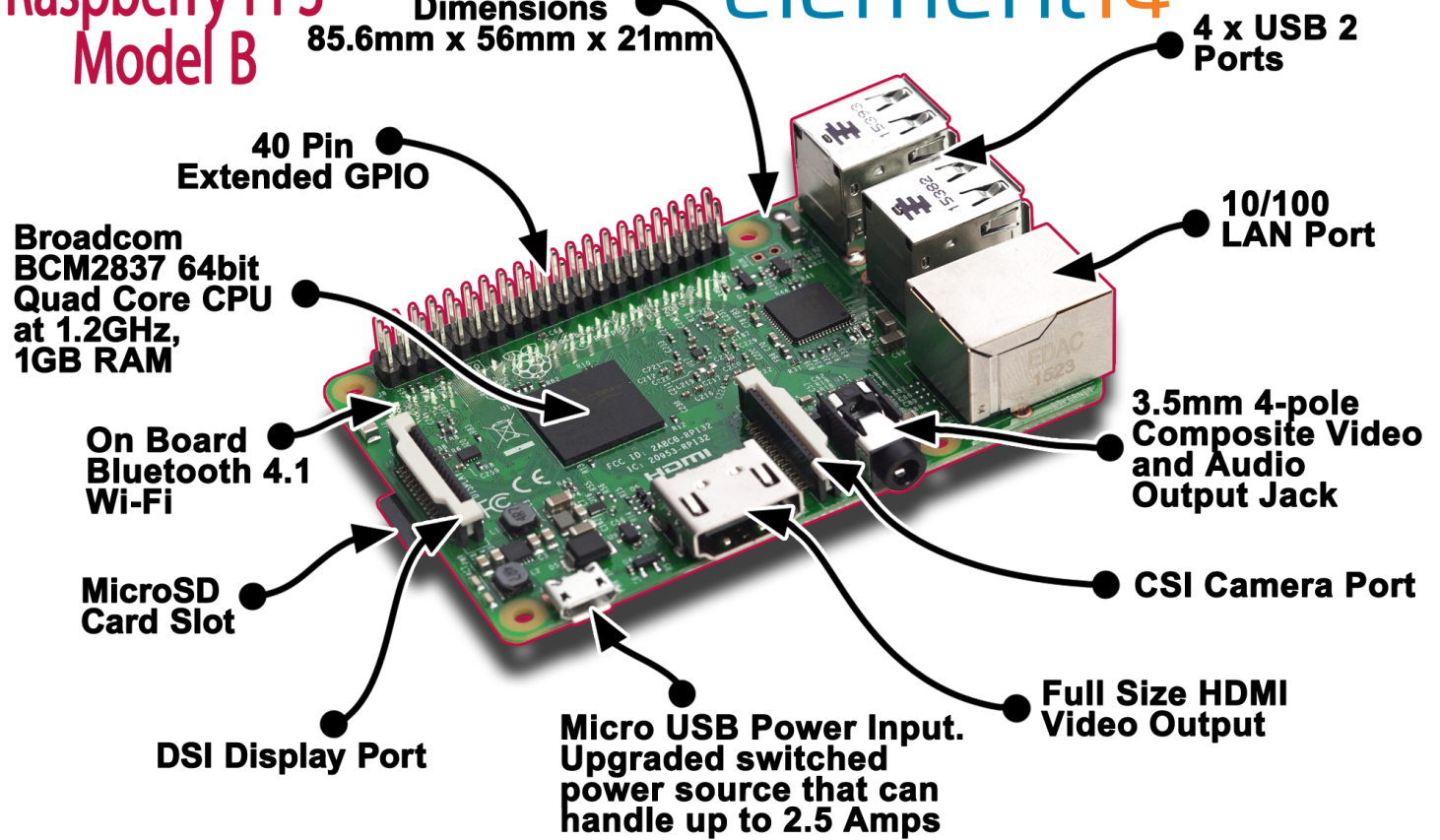
There's no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board, in order to keep the size of the device to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.



Raspberry Pi 3 Model B

Dimensions
85.6mm x 56mm x 21mm

element14



Key Improvements from Pi 2 Model B to Pi 3 Model B:

- Next Generation QUAD Core Broadcom BCM2837 64bit ARMv7 processor
- Processor speed has increased from 900MHz on Pi 2 to 1.25Ghz on the RPi 3 Model B
- BCM43143 Wi-Fi on board
- Bluetooth Low Energy (BLE) on board
- Upgraded switched power source up to 2.5 Amps (can now power even more powerful devices over USB ports)

The main differences are the quad core 64-bit CPU and on-board Wi-Fi and Bluetooth. The RAM remains 1GB and there is no change to the USB or Ethernet ports. However, the upgraded power management should mean the Pi 3 can make use of more power hungry USB devices

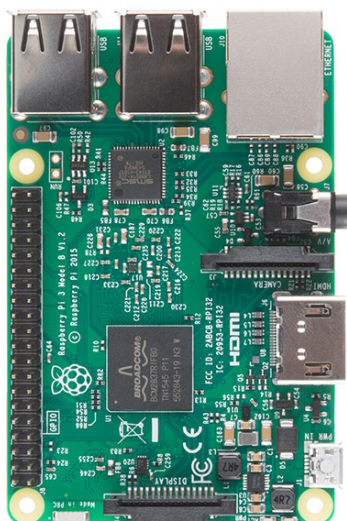
For Raspberry Pi 3, Broadcom have supported us with a new SoC, BCM2837. This retains the same basic architecture as its predecessors BCM2835 and BCM2836, so all those projects and tutorials which rely on the precise details of the Raspberry Pi hardware will continue to work. The 900MHz 32-bit quad-core ARM Cortex-A7 CPU complex has been replaced by a custom-hardened 1.2GHz 64-bit quad-core ARM Cortex-A53

In terms of size it is identical to the B+ and Pi 2. All the connectors and mounting holes are in the same place so all existing add-ons, HATs and cases should fit just fine although the power and activity LEDs have moved to make room for the WiFi antenna.

The performance of the Pi 3 is roughly 50-60% faster than the Pi 2 which means it is ten times faster than the original Pi.

All of the connectors are in the same place and have the same functionality, and the board can still be run from a 5V micro-USB power adapter. This time round, we're recommending a 2.5A adapter if you want to connect power-hungry USB devices to the Raspberry Pi.

Raspberry Pi 3 Model B

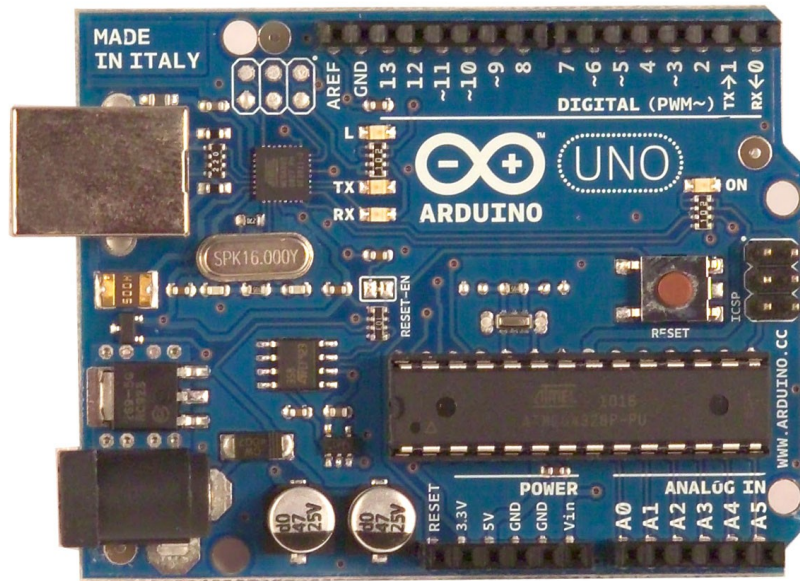


Raspberry Pi 2 Model B



Arduino Uno specifications sheet

Arduino UNO



Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 ([datasheet](#)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the [index of Arduino boards](#).

Index

Technical Specifications

Page 2

How to use Arduino
Programming Environment, Basic Tutorials

Page 6

Terms & Conditions

Page 7

Environmental Policies
half sqm of green via Impatto Zero®

Page 7



radiospares

RADIONICS



Technical Specification

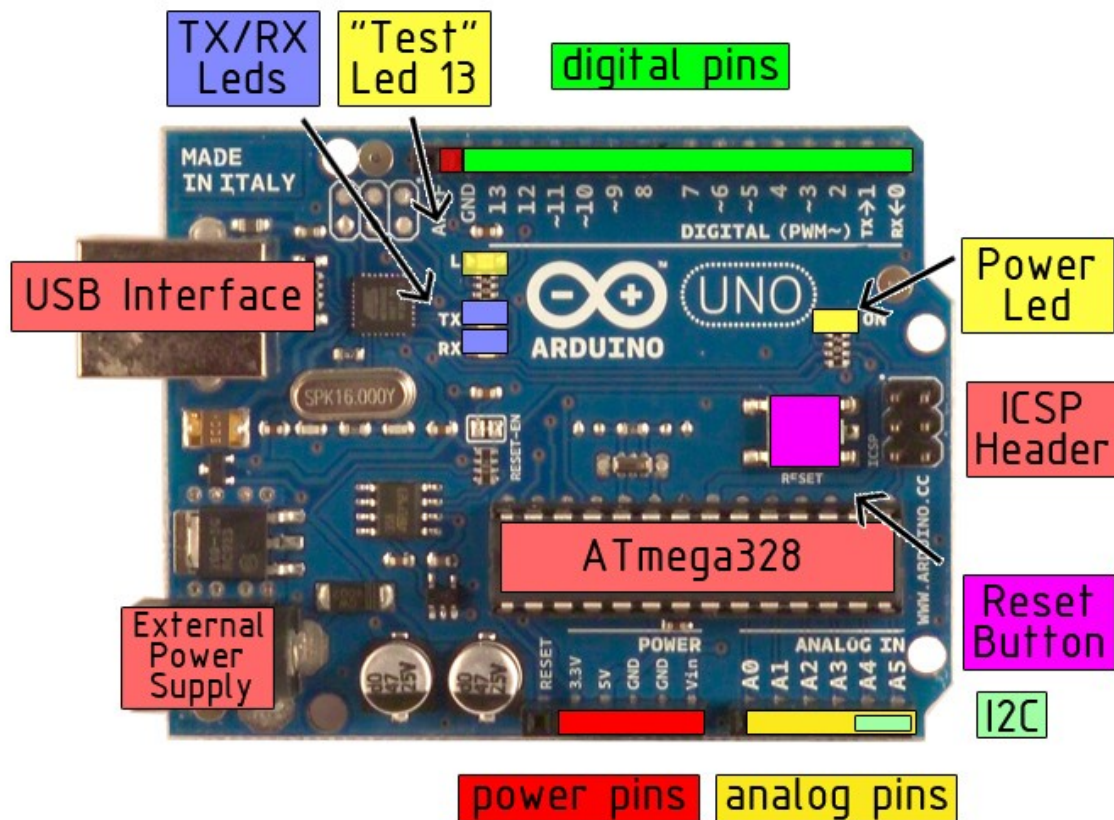


EAGLE files: [arduino-duemilanove-uno-design.zip](#) Schematic: [arduino-uno-schematic.pdf](#)

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



radiospares

RADIONICS



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.



radiospares

RADIONICS



The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the [analogReference\(\)](#) function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the [Wire library](#).

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with [analogReference\(\)](#).
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the [mapping between Arduino pins and Atmega328 ports](#).

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](#) allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation](#) for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software ([download](#)). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the [reference](#) and [tutorials](#).

The ATmega328 on the Arduino Uno comes preburned with a [bootloader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](#) for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use [Atmel's FLIP software](#) (Windows) or the [DFU programmer](#) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).



RADIOSPARES

RADIONICS



Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

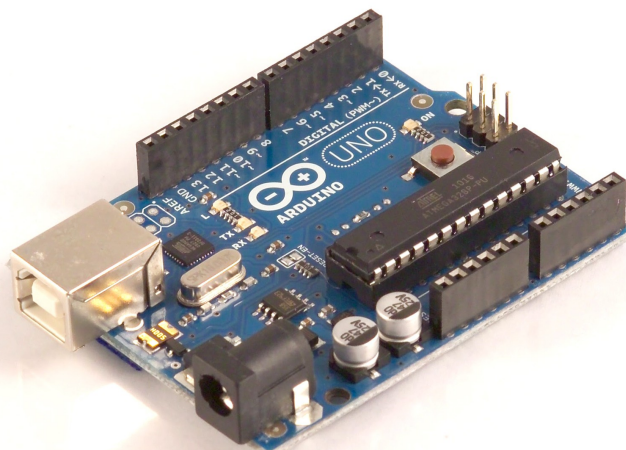
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.



RADIOSPARES

RADIONICS



How to use Arduino



Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the [Arduino programming language](#) (based on [Wiring](#)) and the Arduino development environment (based on [Processing](#)). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platform program. You'll have to follow different instructions for your personal OS. Check on the [Arduino site](http://arduino.cc/en/Guide/HomePage) for the latest instructions. <http://arduino.cc/en/Guide/HomePage>

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

**File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink**

Once you have your sketch you'll see something very close to the screenshot on the right.

In **Tools>Board** select

Now you have to go to **Tools>SerialPort** and select the right serial port, the one arduino is attached to.

```
int ledPin = 13; // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup() {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH); // set the LED on
  delay(1000);                // wait for a second
  digitalWrite(ledPin, LOW);  // set the LED off
  delay(1000);                // wait for a second
}
```



Done compiling.

Press Compile button
(to check for errors)



Upload



TX RX Flashing



Blinking Led!

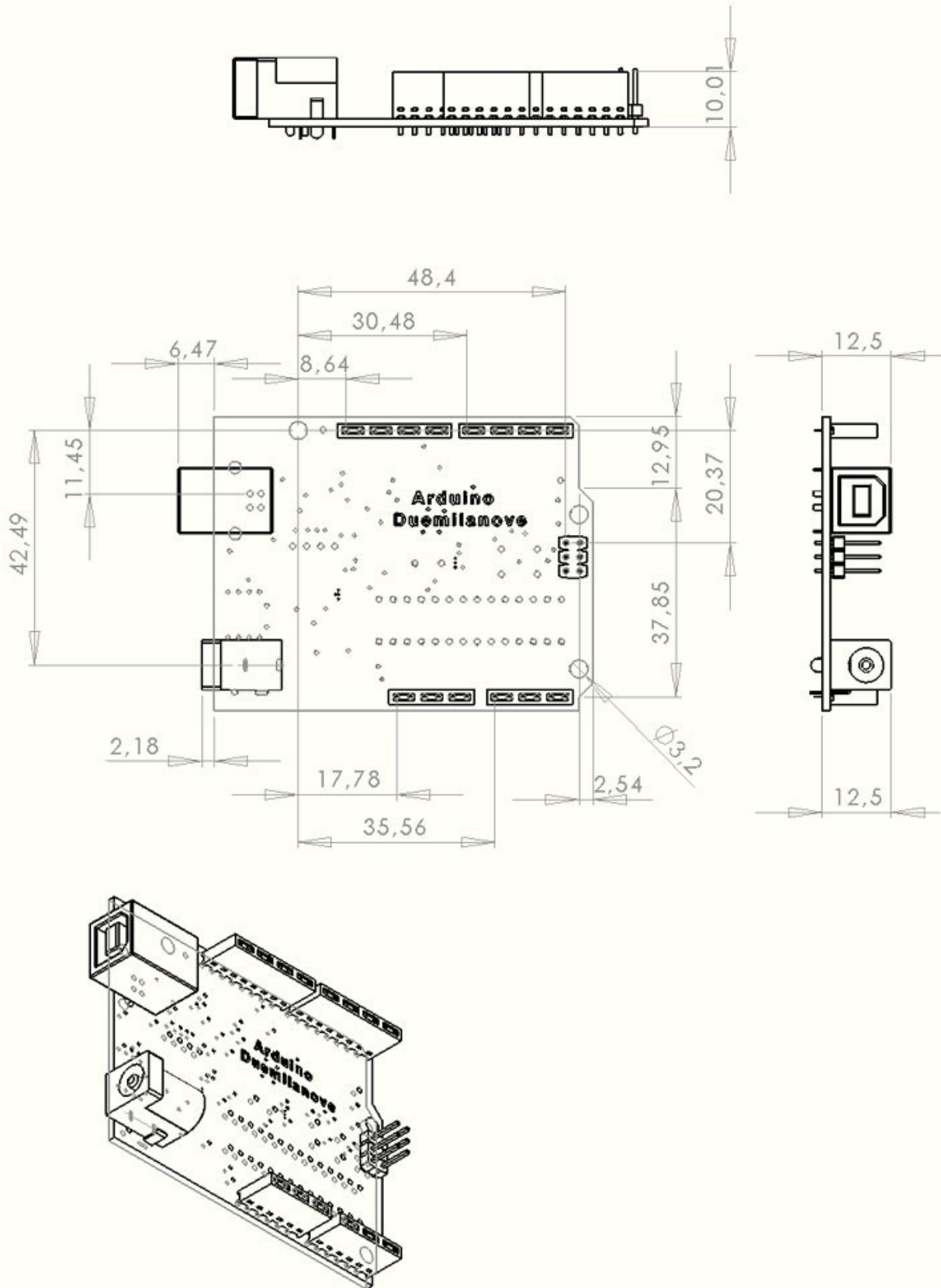


radiospares

RADIONICS



Dimensioned Drawing



radiospares RADIONICS



Terms & Conditions



1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.



Environmental Policies



The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.



radiospares

RADIONICS



Arduino GSM Shield 2

With integrated antenna



With antenna connector



Schematic & Reference Design

Download: PDF of GSM shield schematic

(<http://download.arduino.org/products/GSMSHIELD/Arduino-GSM-Shield2-Rev3.2-SCH.pdf>), Reference design

(<http://download.arduino.org/products/GSMSHIELD/Arduino-GSM-Shield2-Rev3.2-reference-design.zip>)

The GSM library is included with Arduino IDE 1.0.4 and later.

Overview

The Arduino GSM Shield connects your Arduino to the internet using the GPRS wireless network. Just plug this module onto your Arduino board, plug in a SIM card from an operator offering GPRS coverage and follow a few simple instructions to start controlling your world through the internet. You can also make/receive voice calls (you will need an external speaker and microphone circuit) and send/receive SMS messages.

As always with Arduino, every element of the platform – hardware, software and documentation – is freely available and open-source. This means you can learn exactly how it's made and use its design as the starting point for your own circuits. Hundreds of thousands of Arduino boards are already fueling people's creativity all over the world, everyday. Join us now, Arduino is you!

- Requires an Arduino board (not included)
- Operating voltage 5V (supplied from the Arduino Board)
- Connection with Arduino Uno on pins 2, 3 (Software Serial) and 7 (reset).

Description

The Arduino GSM Shield allows an Arduino board to connect to the internet, make/receive voice calls and send/receive SMS messages. The shield uses a radio modem M10 by Quectel (datasheet (http://download.arduino.org/products/GSM_SHIELD/Arduino-GSM-Shield2-Rev3.2-SCH.pdf)). It is possible to communicate with the board using AT commands. The GSM library has a large number of methods to communicate with the shield.

The shield uses digital pins 2 and 3 for software serial communication with the M10. Pin 2 is connected to the M10's TX pin and pin 3 to its RX pin. The modem's PWRKEY pin is connected to Arduino pin 7.

The M10 is a Quad-band GSM/GPRS modem that works at the following frequencies: GSM850MHz, GSM900MHz, DCS1800MHz and PCS1900MHz. It supports TCP/UDP and HTTP protocols through a GPRS connection. GPRS maximum data downlink and uplink transfer speed is 85.6 kbps.

To interface with the cellular network, the board requires a SIM card provided by a network operator.

The most recent revision of the board uses the 1.0 pinout on rev 3 of the Arduino Uno board.

The micro to nano SIM adapter is included in the box.

Power requirements

It is recommended to power the board with an external power supply that can provide between 700mA and 1000mA. Powering an Arduino and the GSM shield from a USB connection is not recommended, as USB cannot provide the required current when the modem is in heavy use.

The modem can pull up to 2A of current at peak usage, which can occur during data transmission. This current is provided through the large orange capacitor on the board's surface.

On board indicators

The shield contains a number of status LEDs:

- On: shows that the Shield is getting power.
- Status: switches on when the modem is powered and data is being transferred to/from the GSM/GPRS network.
- Net: blinks when the modem is communicating with the radio network.

On board interfaces

The shield comes with a on-board audio jack as well, and it can be used for both microphone and line inputs. It is also possible to make voice calls. You don't need to add a speaker and microphone.

There are two small buttons on the shield. The button labeled "Reset" is tied to the Arduino reset pin. When pressed, it will restart the sketch. The button labeled "Power" is connected to the modem and will power the modem on and off. For early versions of the shield, it was necessary to press the power button to turn on the modem. Newer versions of the board will turn the modem on automatically.

If you have an early version of the shield, and it does not turn on automatically, you can solder a jumper to the CTRL/D7 pad on the reverse side of the board, and it will turn on when an attached Arduino receives power.

Several of the modem pins are exposed on the underside of the board. These provide access to the modem for features like speaker output and microphone input. See the datasheet for complete information.

Product Code

A000105 (with integrated antenna)

A000106 (with antenna connector)



Grove - Dust Sensor User Manual

Release date: 2015/9/23

Version: 1.0

Wiki: [http://www.seeedstudio.com/wiki/Grove - Dust sensor](http://www.seeedstudio.com/wiki/Grove_-_Dust_sensor)

Bazaar: <http://www.seeedstudio.com/depot/Grove-Dust-Sensor-p-1050.html>

Document Revision History

Revision	Date	Author	Description
1.0	Sep 23, 2015	Jiankai.li	Create file

Contents

Document Revision History	2
1. Introduction	2
2. Features	3
3. Application Ideas	4
4. Cautions	5
5. Specification	6
6. Usage	7
7. Resources	10
8. Reference	11
9. Related Projects	12
9.1 Air Quality Box	12
9.2 Share Your Awesome Projects with Us	12

Disclaimer

For physical injuries and possessions loss caused by those reasons which are not related to product quality, such as operating without following manual guide, natural disasters or force majeure, we take no responsibility for that.

Under the supervision of Seeed Technology Inc., this manual has been compiled and published which covered the latest product description and specification. The content of this manual is subject to change without notice.

Copyright

The design of this product (including software) and its accessories is under tutelage of laws. Any action to violate relevant right of our product will be penalized through law. Please consciously observe relevant local laws in the use of this product.

1. Introduction

This Dust Sensor measures the Particulate Matter level in air by counting the Lo Pulse Occupancy time(LPO time) in given time unit. LPO time is in proportion to PM concentration. This sensor can provide you pretty reliable data for air purifier system because it's still responsive to particulates whose diameter is 1um. **Note:** This sensor use counting method to test dust concentration but not weight method, and the unit is pcs/L or pcs/0.01cf.



2. Features

- Highly responsive
- Reliable
- ROHS/PEACH compliant

Note: New version updates output Hi Voltage from Approx. over 4.0V change to Approx over 4.5V.

3. Application Ideas

- Dust emission monitor
- [Air Quality Monitoring](#)

4. Cautions

- Please keep it upright.
- 3 min preheat time is required when used at the first time.
- Arbitrary operation may cause unexpected damage.
- Pins VR1 and VR2 come preset. **Please DON'T change the default configuration.**

5. Specification

Items	Min	Norm	Max	Unit
VCC	4.75	-	5.25	V
Standby Current Supply	-	90	-	mA
Detectable range of concentration	-	0~28,000 / 0 ~ 8000	-	pcs/liter / pcs/0.01cf
Operating Temperature Range	0	-	45	° C
Output Method	Negative Logic, Digital output,Hi over 4.0V(Rev.2) Lo: under 0.7V			
Detecting the particle diameter	>1 um			
Dimensions	59(W) × 45(H) × 22(D) [mm]			
Humidity Range	95%rh or less			

6. Usage

Here is a demo to show you how to obtain PM concentration data from this Grove - Dust Sensor.

1. Plug the dust sensor into digital port D8 on the [Grove - Base Shield](#). It can only be D8, because the operation of this sensor involves sampling, a function only can be achieved by D8, the capture input pin of Atmage328P, on Arduino/Seeeduino.

Also you can connect Grove - Dust sensor to Arduino UNO without Base Shield :

Arduino UNO	Dust Sensor
5V	Red wire
GND	Black wire
Digit 8	Yellow wire

2. Copy and paste the demo code below to a new Arduino sketch.

```
/* Grove - Dust Sensor Demo v1.0
Interface to Shinyei Model PPD42NS Particle Sensor
Program by Christopher Nafis
Written April 2012

http://www.seeedstudio.com/depot/grove-dust-sensor-p-1050.html
http://www.sca-shinyei.com/pdf/PPD42NS.pdf

JST Pin 1 (Black Wire) => Arduino GND
JST Pin 3 (Red wire)   => Arduino 5VDC
JST Pin 4 (Yellow wire) => Arduino Digital Pin 8
*/

int pin = 8;
unsigned long duration;
unsigned long starttime;
unsigned long samptime_ms = 30000;//sampe 30s ;
unsigned long lowpulseoccupancy = 0;
float ratio = 0;
float concentration = 0;

void setup() {
  Serial.begin(9600);
  pinMode(8, INPUT);
  starttime = millis();//get the current time;
}
```

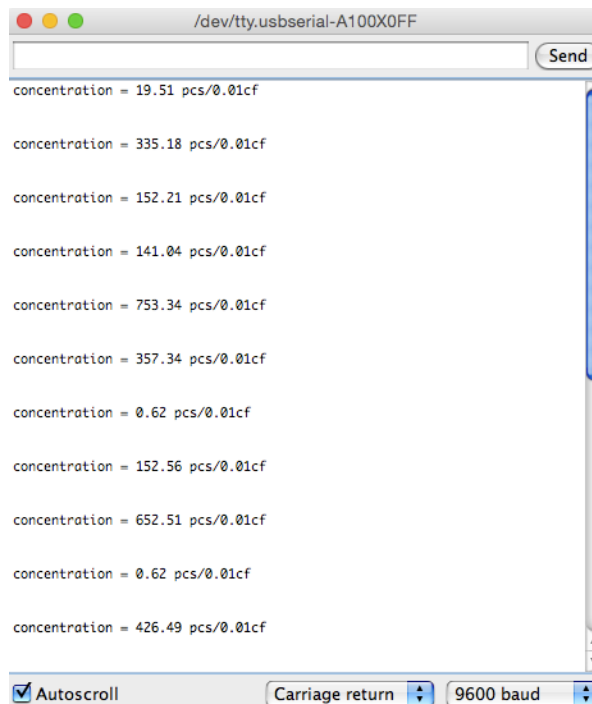
```

void loop() {
  duration = pulseIn(pin, LOW);
  lowpulseoccupancy = lowpulseoccupancy+duration;

  if ((millis()-starttime) > sampletime_ms)//if the sampel time == 30s
  {
    ratio = lowpulseoccupancy/(sampletime_ms*10.0); // Integer percentage 0=>100
    concentration = 1.1*pow(ratio,3)-3.8*pow(ratio,2)+520*ratio+0.62; // using spec sheet curve
    Serial.print("concentration = ");
    Serial.print(concentration);
    Serial.println(" pcs/0.01cf");
    Serial.println("\n");
    lowpulseoccupancy = 0;
    starttime = millis();
  }
}

```

In this program, the seeeduino samples the total duration of "logic low" in 30s, and this duration illustrates the dust density of environment. Open serial monitor, we can read air quality's value detected by sensor from pc serial port.

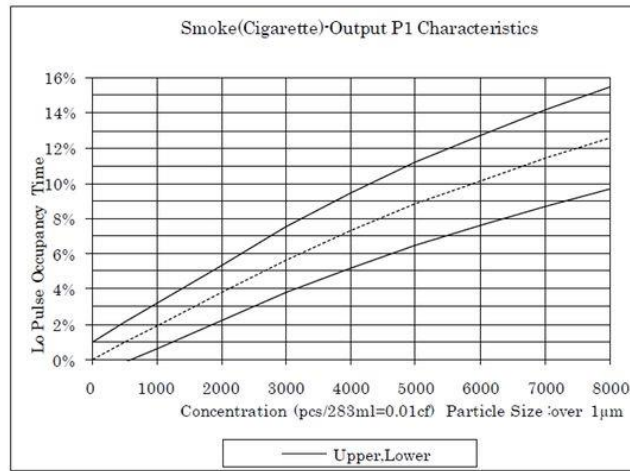


The result above consists of three parts: lowpulseoccupancy, ratio and concentration.

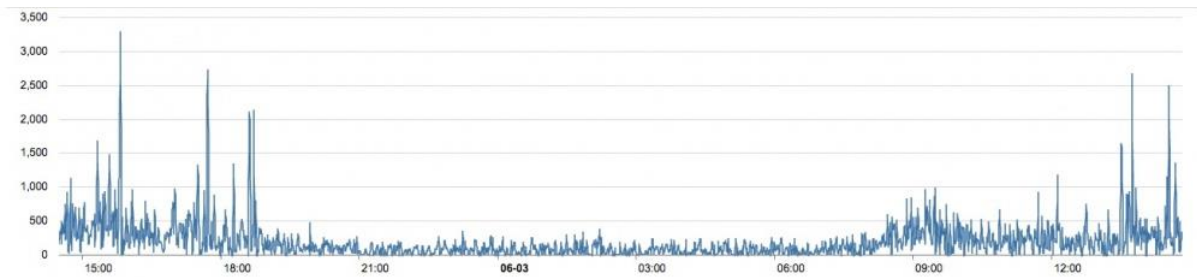
"lowpulseoccupancy" represents the Lo Pulse Occupancy Time(LPO Time) detected in given 30s. Its unit is microsecond.

"ratio" reflects on which level LPO Time takes up the whole sample time.

"concentration" is a figure that has physical meaning. It's calculated from the characteristic graph below by using the LPO time.



Here is a figure we tested dust concentration in office :



We can see the concentration of dust is very low in the evening, but it's higher in the afternoon. Maybe you can set a threshold value while the concentration is beyond a value. Also, if you wanna set the sensor more sensitive you can add a fan on the sensor, and add a 10k resistor between the Pin5 and Ground. More information please visit the blog of A.J.

7. Resources

- [Grove_-Dust_sensor datasheet](#)
- [De-construction of the Shinyei PPD42NS dust sensor](#) *Made by Tracy Allen*

8. Reference

- [Building a low-cost networked PM2.5 monitor](#) -- *Made by A.J.*
- [Measuring the Pickle Jr. – a modified PPD42 with an attached fan.](#) -- *Made by A.J.*
- [Testing the Shinyei PPD42NS](#) -- *Made by darell tan*
- [Air Quality Monitoring](#) -- *Made by Chris Nafis*

9. Related Projects

If you want to make some awesome projects by Dust Sensor, here's some projects for reference.

9.1 Air Quality Box



This is an IoT demo make by Seeeduino and [Grove](#).

More and More concern are being paid to the environmental air quality nowadays because the tiny granule in the around air can badly endanger people' s health. We always get the information of environment from our government department. But it' s the average value of the whole city / section. It can' t reflect the situation near by you accurately.

I want to make it.

9.2 Share Your Awesome Projects with Us

Born with the spirit of making and sharing, that is what we believe makes a maker.

And only because of this , the open source community can be as prosperous as it is today.

It does not matter what you are and what you have made, hacker, maker, artist and engineers, as long as you start sharing your works with others,

you are being part of the open source community and you are making your contributions .

Now share you awesome projects on with us on [Recipe](#), and win a chance to become the Core User of Seeed.

- Core Users, are those who showing high interests and significant contributions in Seeed products.

- We cooperate with our Core Users in the development of our new product, this, in another word, the Core Users will have the chance to experience any new products of Seeed before its official launch, and in return we expect valuable feedback from them to help us improving the product performance and user experience. And for most of cases if our Core Users have any good ideas for making things, we'll offer hardware pieces, PCBA services as well as technical support. Besides, further commercial cooperation with the Core Users is highly possible.

Get more information about Core User please email to: recipe@seeed.cc

Mouser Electronics

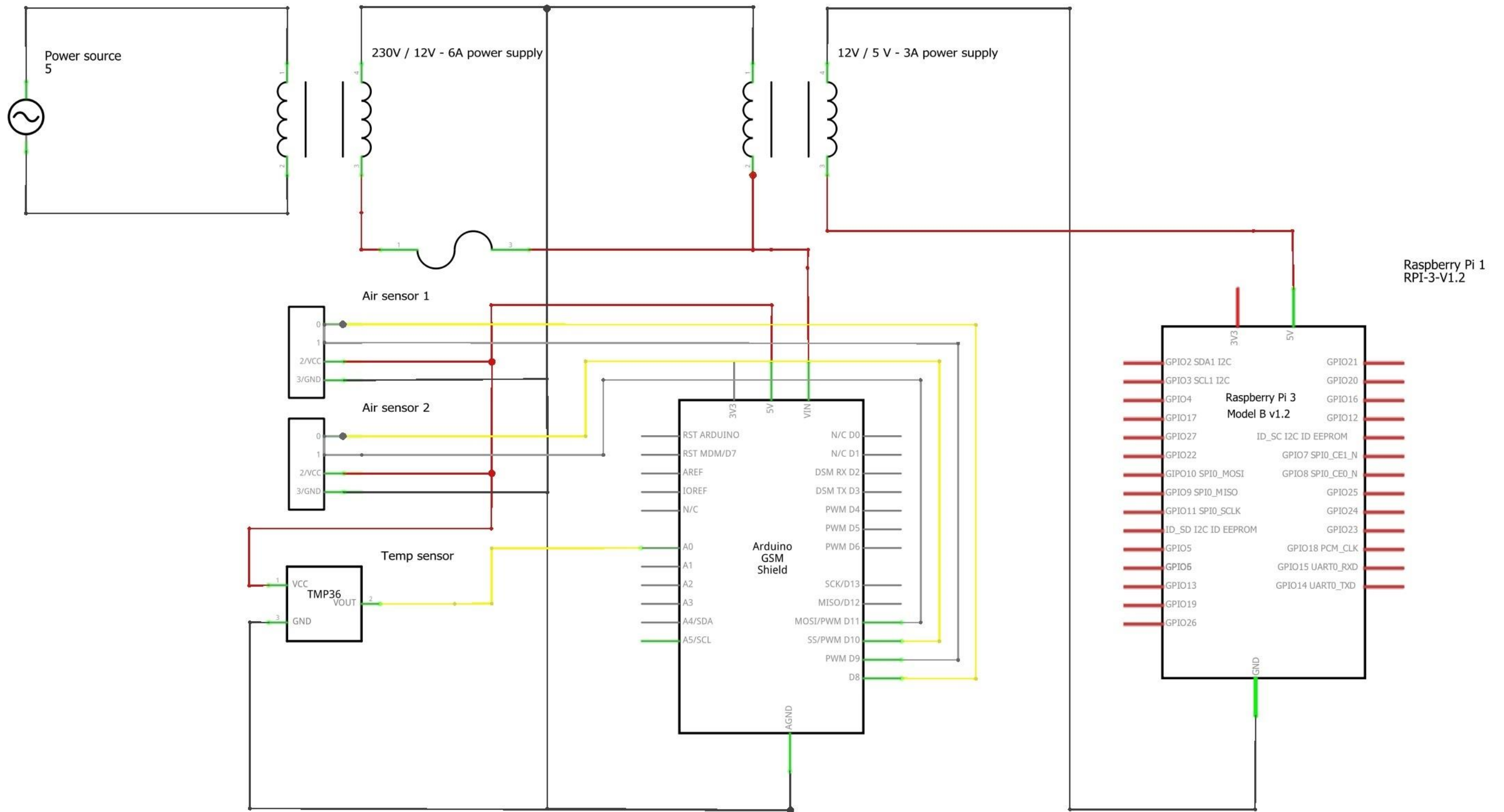
Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[Seeed Studio:](#)

[101020012](#)

Appendix J Electrical wiring diagram





Brukerveiledning

Mobil målestasjon versjon 1.0



Øvrig informasjon

Denne brukerveiledningen inneholder instruksjoner for korrekt bruk av målestasjonen, og tar også for seg sikkerhetsaspekter knyttet til personsikkerhet og forhindring av materielle skader. Sikkerhetsaspekter er markert med symboler i brukerveiledningen, og følgende symboler benyttes for farenivå.

	<p>ELEKTRISK SJOKK</p> <p>Indikerer fare for elektrisk sjokk som kan forårsake alvorlig personskade eller død, samt skade på materiell og utstyr.</p>
	<p>ADVARSEL</p> <p>Indikerer fare for skade på materiell og utstyr ved uforsiktig bruk. Indikerer også at feil bruk kan forårsake at målestasjonen ikke fungerer som tiltenkt.</p>

1.1 Kvalifisert personell

Kun personer med faglig bakgrunn og kjennskap innen elektro kan utføre koblingsarbeid og endringer på elektrisk utstyr knyttet til målestasjon. Dette for å sikre at utførelse blir gjort på en sikker måte og for å unngå fare for liv, helse og materielle skader.

1.2 Øvrig bruk

Personer uten elektrobakgrunn kan montere, sette i drift og utføre vedlikehold av målestasjon. Føringer for bruk knyttet til dette er forklart i kapittel 3 og 4, og må følges for å oppnå sikker bruk.

1.3 Krav til bruker

Det stilles krav til at bruker av målestasjonen har tilstrekkelig teknisk kunnskap for å kunne operere målestasjonen, samt kunne lese og forstå denne brukerveiledningen.

Innholdsfortegnelse

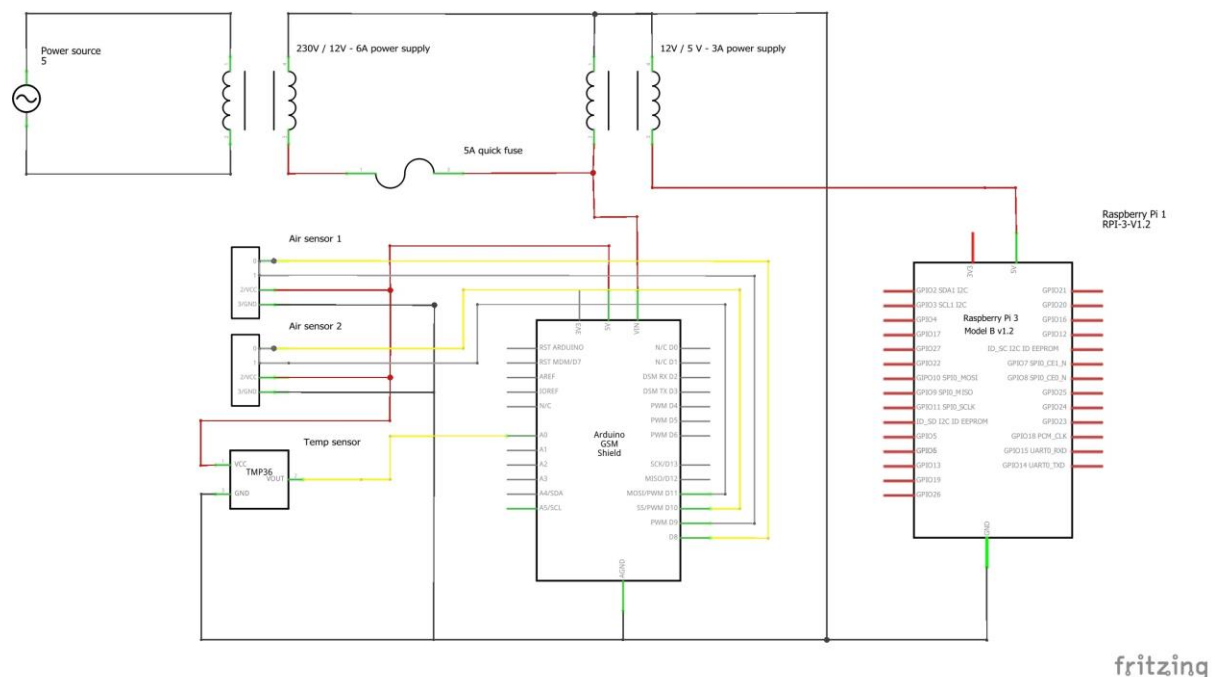
Øvrig informasjon	2
1.1 Kvalifisert personell.....	2
1.2 Øvrig bruk.....	2
1.3 Krav til bruker.....	2
Innholdsfortegnelse	3
2.. Systembeskrivelse.....	4
2.1 Koblingsskjema	4
2.2 Sensorer og luftkanal	5
3.. Montering og igangkjøring.....	6
3.1 Plassering.....	6
3.2 SIM-kort.....	6
3.3 Strømtilførsel.....	7
3.4 Igangkjøring	7
3.5 Feil ved oppkobling	8
3.6 Brukergrensesnitt	8
4.. Drift og vedlikehold.....	10
4.1 Rengjøring	10
4.2 Strømbrydd	10
4.3 Endring av parametere	10
4.4 Lokalt minne	11
4.5 Sikring.....	12

2 Systembeskrivelse

Målestasjonen måler luftkvalitet og temperatur. Det er installert 2 sensor som hver måler konsentrasjon av PM₁₀ og PM_{2,5} i $\mu\text{g}/\text{m}^3$, og 1 temperatursensor som måler i $^{\circ}\text{C}$. Målestasjonen benytter telenettet for overføring av måledata til en database. Målestasjonen sender data over en definert periode som er beskrevet i kapittel 4.3. Den har dreneringshull i bunn for å drenere eventuell kondens og fukt. Kabel for tilkobling til strømforsyning er montert i bunn av målestasjonen.

2.1 Koblings skjema

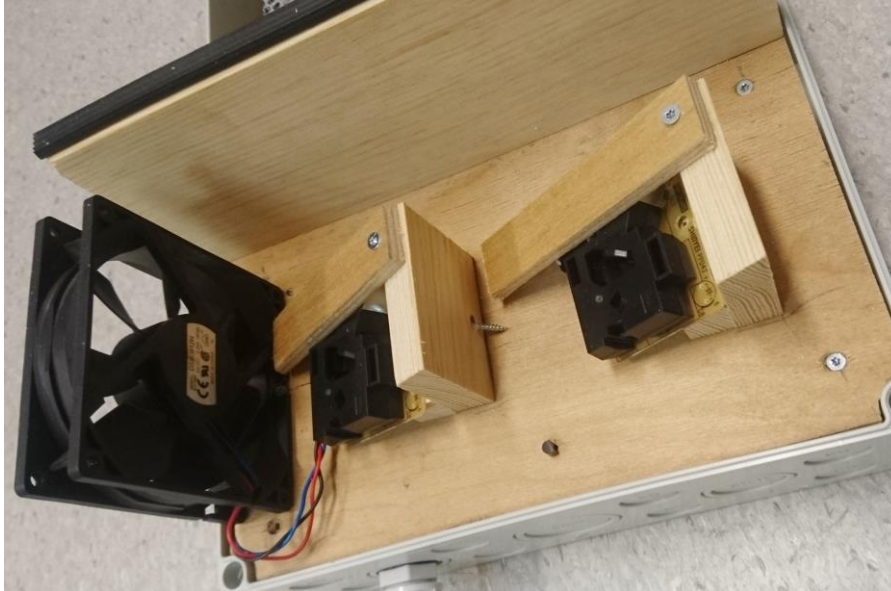
Målestasjonen benytter mikrokontrollere (Raspberry Pi og Arduino Uno) for databehandling. Sensorer og strømforsyning er tilkoblet disse som vist i Figur 2.1. Kommunikasjon mellom mikrokontrollene skjer over USB-kabel. Det er ledige USB-porter og 1 HDMI-port for tilkobling av eksternt utstyr.



Figur 2.1 Koblings skjema

2.2 Sensorer og luftkanal

Luften som skal måles føres gjennom luftkanalen i bunnen av målestasjonen. En vifte sørger for kontinuerlig strøm av luft. Luftkanalen blir tett når lokket settes på. I luftkanalen er det montert 2 sensorer for luftkvalitet (i midten), og 1 mindre temperatursensor (under luftsensor-ene), se Figur 2.2.



Figur 2.2 Luftkanal

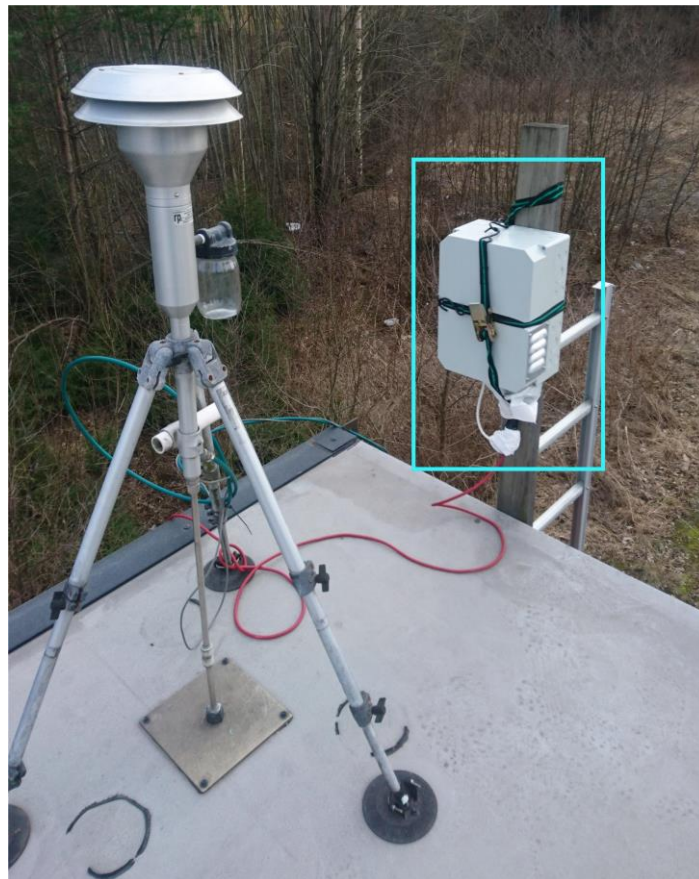
3 Montering og igangkjøring

3.1 Plassering

Målestasjonen plasseres i en høyde hvor det er interessant å måle luftkvalitet. Det er ikke laget noe montasjemulighet på målestasjonen, så dette må tilpasse til hvor målestasjonen er tiltenkt. Figur 3.1 viser en montering ved bruk av jekkestropper.



Påse at målestasjonen monteres med ledningsuttaket på undersiden (slik som vist i Figur 3.1). Dette for å sikre tilstrekkelig tetthet mot inntrenging av vann og fukt.



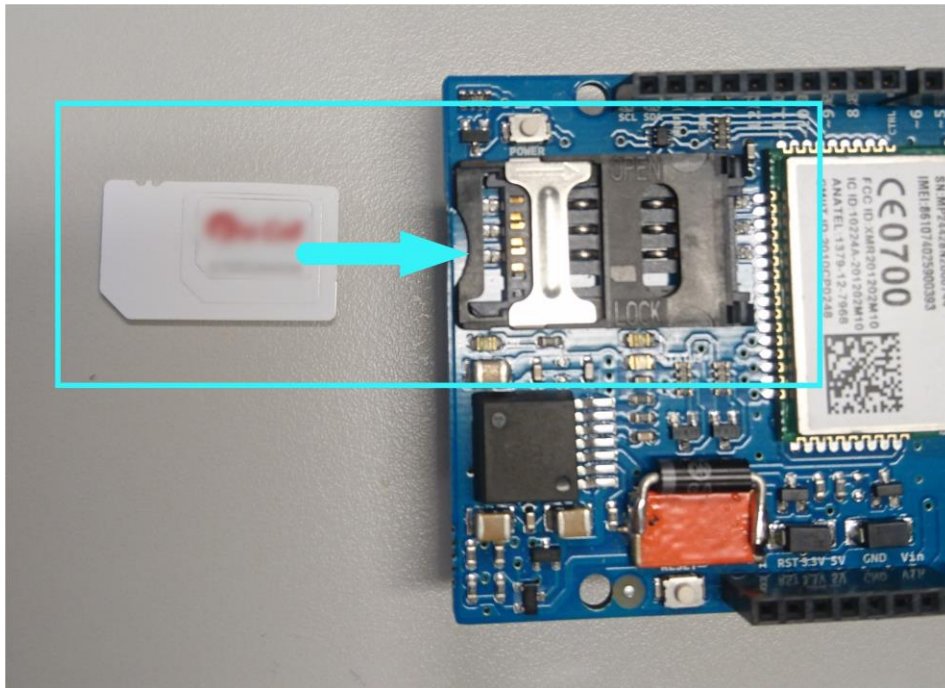
Figur 3.1

3.2 SIM-kort

Et SIM-kort **må** benyttes for å kunne sende måledata til nettserveren. SIM-kortet plasseres i kortholder, som vist i Figur 3.2. SIM-kortet må ha standard størrelse (største rammen). De fleste teleoperatører leverer compatible SIM-kort.



Tilse at SIM-kortet har **PIN-kode: 2010** før det settes i. Hvis ikke vil ikke stasjonen overføre måledata. Dette kan endres for eksempel på en telefon.



Figur 3.2 SIM-kort holder

3.3 Strømtilførsel

Målestasjonen kan tilkobles vanlig strømuttak 230V. På undersiden av målestasjonen er det en kabel med støpsel som kolbes til strømtilførselen. Det er ikke noe batteridrift på målestasjonen om strømmen blir borte.

3.4 Igangkjøring

Igangkjøring av målestasjonen krever kun at den er **tilkoblet** en strømtilførsel og har installert et SIM-kort med PIN-nummer **2010**. Målestasjonen er programmert (kan ikke endres) til å sende data til følgende nettsted:

URL: <http://luftforurensing.azurewebsites.net>

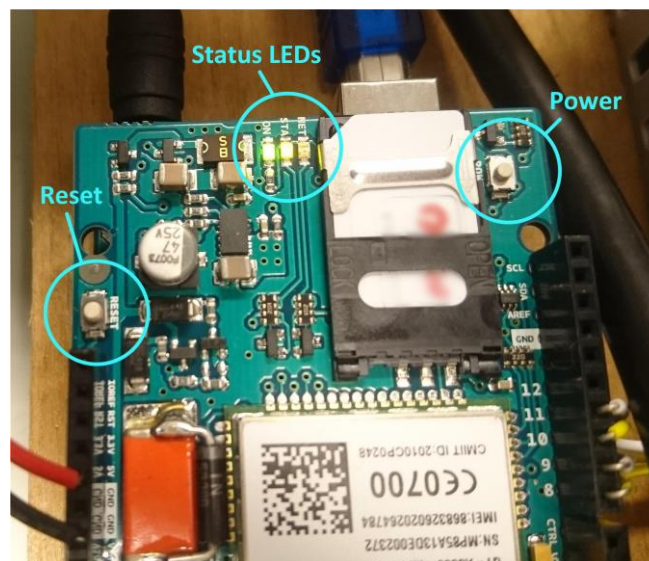
SERVER: <http://luftforurensing.azurewebsites.net/adddata/post>

For å verifisere at system logger data til nettstedet kan dette gjøres ved å vente på at data kommer inn på nettsiden, eller ved å benytte brukergrensesnittet, se kapittel 3.6 og 4.3.

3.5 Feil ved oppkobling

Ved oppstart kan det oppstå problem ved oppkobling til GSM-nettverket. Hvis GSM-kontakt ikke blir opprettet vil «Network» og «Status» LEDene slutte å lyse / blinke, se Figur 3.3. For å foreta et nytt forsøk på oppkobling gjøres følgende:

1. Trykk på «Power» knappen slik at LEDene begynner å lyse igjen
2. Trykk på «Reset» knappen for å restarte kontrolleren
3. Vent ca 15-60 sekunder å se om LEDene slutter å lyse. Hvis de ikke slutter å lyse vil den fortsatt prøve å koble opp, eller den er tilkoblet. Status kan også sees på en skjerm ved å kolbe seg til med HDMI-kabelen, se kapittel 3.6.
4. Hvis LEDene slutter å lyse, koble fra USB-kabelen (toppen på Figur 3.3) og gjenta punkt 1-3. **NB!** Ved å koble ut USB-kabelen vil ikke statusinformasjon vises på skjerm tilkoblet HDMI. Hvis oppkobling ser ut i å være i orden, koble til USB-kabelen igjen.



Figur 3.3 Plassering av knapper og status LEDs.

3.6 Brukergrensesnitt

Brukergrensesnitt kan benyttes for å verifisere oppkobling mot telenettet og for å endre parametere. For å få tilgang til brukergrensesnittet gjøres følgende:

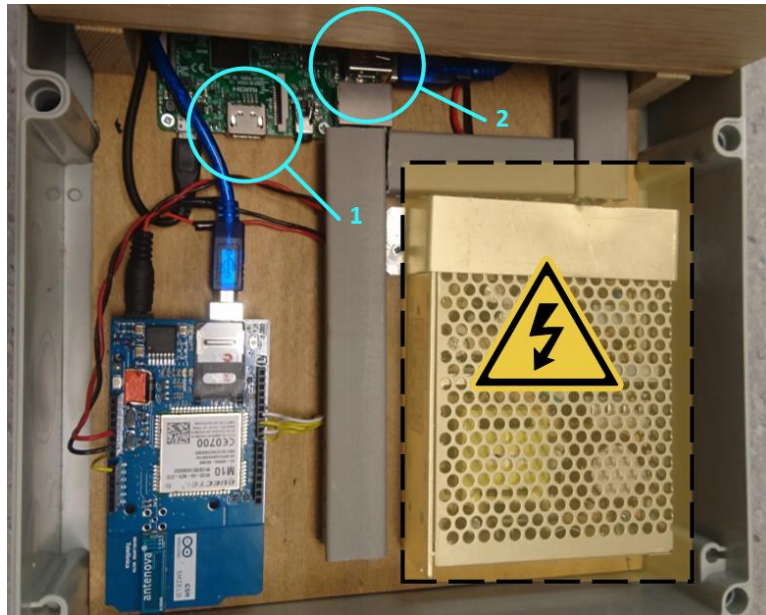
1. Åpne boksen for å få tilgang til HDMI-uttak (1) og USB-porter (2), se Figur 3.4.



Hold avstand fra tilkoblinger for strømforsyning. Disse er tilgjengelige på toppen av strømforsyningen og forsyner målestasjonen med 230V.

2. Koble HDMI-uttaket til en monitor med en HDMI-kabel.
3. Koble til tastatur og mus til USB-portene hvis dette trengs.
4. Påse at skjermen er stilt inn til å motta signal fra HDMI.

5. Vent til logg-inn vindu vises. NB! Ved første logg inn etter at strømmen er slått på må **standard** bruker-ID og passord benyttes. **Bruker-ID: admin** og **passord: admin**. Se Figur 3.5.



Figur 3.4 Plassering av HDMI-uttak og USB-porter.

Bruker-ID:

Passord:

Første logg inn.

Bruk default bruker-ID og passord

Figur 3.5 Logg inn side

4 Drift og vedlikehold

4.1 Rengjøring

Luftkanal med tilhørende utstyr bør rengjøres regelmessig for å oppå mest nøyaktig måleresultat.

4.2 Strømbrudd

Ved strømbrudd vil alt dynamisk minne bli tømt og systemet vil starte opp igjen når strømmen kommer tilbake. Ved oppstart igjen vil parameterne gå tilbake til standard-verdier og foretatte endringer underveis vil ikke fungere. Derfor **anbefales** det å benytte standard-innstillingene for å unngå at målestasjonen ikke sender data.



Benytt standard-innstillinger for å unngå at systemet slutter å sende data. Dette er som følge at systemet ikke har noe innebygd funksjon for å huske endringer ved strømbrudd.

4.3 Endring av parametere

Under normal drift er det mulig å endre parametere ved å benytte brukergrensesnittet. Etter å ha logget på (se kapittel 8) gjøres følgende for å endre parametere:

1. Trykk på «Koble til Arduino» for å kunne kommunisere med mikrokontroller som foretar sending og måling av data.
2. Vent til «Arduino tilkoblet» vises ved tilhørende knapp. Når dette er gjort vil målerverdier vises i tekstboksene for disse.
3. Utfør endringene for de respektive parametere, se figur Figur 4.1.
4. Trykk på «Oppdater endringer» for å oppdatere parametere.
5. Trykk på «Logg ut» for å logge ut. **NB!** Husk å noter bruker-ID og passord hvis disse er endret.

Bruker-ID: Passord:
 PIN-kode: ...
 Stasjon-ID: 1 Sensor 1
 Stasjon-ID: 2 Sensor 2
 Sendingsinterval: min
 PM10: $\mu\text{g}/\text{m}^3$
 PM2,5: $\mu\text{g}/\text{m}^3$
 Temperatur: $^{\circ}\text{C}$

Figur 4.1 Hovedside for endring av parametere

Tabell 1 viser Standard-innstillinger som er anbefalt brukt for målestasjon versjon 1.0.

Parameter	Verdi
PIN-nummer	2010
Sensor-ID 1	8
Sensor-ID 2	9
Sendingsinterval	30min

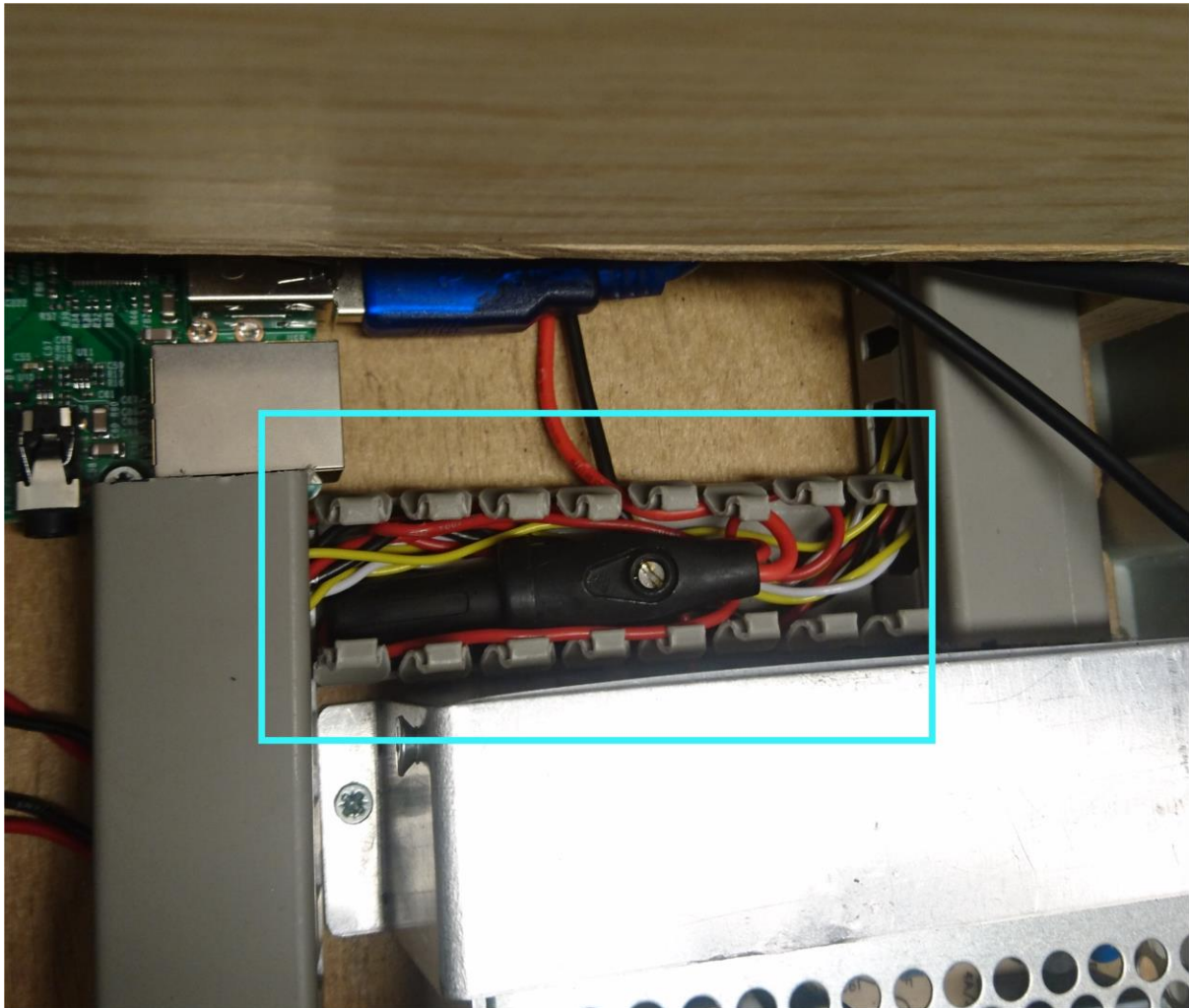
Tabell 1 Standard-innstillinger

4.4 Lokalt minne

Det er ikke laget noe funksjon for lokal lagring. Programvare for brukergrensesnitt lastes opp på et **SD-kort** før det settes inn i målestasjonen. Port for SD-kort er ikke direkte tilgjengelig og må settes i før målestasjonen settes sammen.

4.5 Sikring

Elektriske komponenter i målestasjonen er sikret med en 5A hurtig glass-sikring for å hindre skade på utstyr ved utilsiktet elektrisk feil, se Figur 4.2.



Figur 4.2 Plassering av sikringsholder



Tilse at strømtilførselen er frakoblet først.

For å bytte sikring gjøres følgende:

1. Koble ut strømtilførselen.
2. Ta av lokket på kabelkanalen for å komme til sikringsholderen.
3. Dra ut holderen fra kanalen. Påse at ingen ledninger blir unødvendig strekt.
4. Vri om på holderen for å frigjøre sikringen.
5. Erstatt sikringen og repeter utførte punkter i reversert rekkefølge.