

Raising awareness of citizens by interactively providing environmental data

Pilot of a static sensor network in Delft

Final report 30-06-2017

Synthesis Project
MSc Geomatics TU Delft
SensingTheCity@tudelft.nl

N. Beelaar
C. M. Kleijwegt
R. W. E. Meulmeester
G. Michailidou
N. Salheb
E. L. D. Vaissier



Sensor City Delft

DID-C1 Final Report

By:

| | | |
|----------------------|---------|--------------------------------------|
| Niek Bebelaar | 4619609 | N.Bebelaar@student.tudelft.nl |
| Cathelijne Kleijwegt | 4218469 | C.M.Kleijwegt@student.tudelft.nl |
| Roeland Meulmeester | 4175417 | R.W.E.Meulmeester@student.tudelft.nl |
| Gina Michailidou | 4624785 | G.Michailidou@student.tudelft.nl |
| Nebras Salheb | 4622715 | N.Salheb@student.tudelft.nl |
| Noortje Vaissier | 4206479 | E.L.D.Vaissier@student.tudelft.nl |

Coaches:

| | |
|---------------------|---------------------------------|
| Stefan van der Spek | S.C.vanderSpek@tudelft.nl |
| Wilko Quak | C.W.Quak@tudelft.nl |
| Teun Verkerk | T.J.Verkerk@tudelft.nl |
| Rob Braggaar | R.C.Braggaar@student.tudelft.nl |
| Sabine de Milliano | S.DeMilliano@spingsmart.nl |

Preface

During the MSc Geomatics for the Built Environment at the TU Delft a Synthesis Project is conducted. This is a group project which brings together all the knowledge of the first year of the programme. The Academic Year 2016/2017 has two main subjects for the synthesis project: Internet of (every)Things (IoT) and Point Clouds. Three groups are part of the point cloud subject and two groups of the IoT subject among which this project. This is the 'static' IoT group, where the focus is on doing measurements with a static sensor network; the other group is the 'dynamic' group, that uses sensors on moving platforms to do measurements. The projects takes place at the end of the first year of the Master programme and takes 9 weeks all together.

This project is called Sensor City Delft and is a pilot of a sensor network in the city center of Delft. The aim is to raise local environmental awareness by interactively providing environmental data using a sensor network.

N. Beelaar, C. M. Kleijwegt, R. W. E. Meulmeester,
G. Michailidou, N. Salheb, E. L. D. Vaissier
Delft, June 2017

Acknowledgements

During the last nine weeks we had an intensive time working on this project. We had ups and downs, some setbacks but also some big highlights. At the end we are very pleased with what we accomplished and proudly present the outcomes of the research.

Many people contributed to the completeness of the Sensor City Delft project with their support and guidance at critical moments being of great value. As a team and individually, we would like to thank all the people who supported us over the last 9 weeks. We would like to thank the supervisors and mentors of the Synthesis Project of MSc Geomatics for the Built Environment of TU Delft, Dr. Ir. Stefan van der Spek and Drs. Wilko Quak, for giving useful suggestions and remarks through the whole project and for offering us the opportunity to broaden our knowledge on the IoT field.

In addition, Teun Verkerk and the Science Center Delft for providing us with all the hardware, equipment and space needed to work on this project and also for teaching us how to use them. Special thanks to Rob Braggaar and Aidan Wyber for their valuable help on the technical parts of this project.

We are also pleased to acknowledge the cooperation and support of Sabine de Milliano from Sping Smart. Her input during the first days of our project when our decision-making process took place, but also her support for the rest of the time until the completion of this project were highly important and appreciated. Additionally we want to thank Dr. Ir. Martijn Meijers of TU Delft for helping us on the technical aspect of the project and for being available when that was needed.

Last but not least, we would like to thank the stakeholders, the Municipality and citizens of Delft for supporting this project by giving their feedback and embracing the sensors in the implementation stage.

Table of contents

| | |
|--|------------|
| Preface | 3 |
| Acknowledgements | 4 |
| Table of contents | 5 |
| Reading Guide | 7 |
| Executive summary | 8 |
| Chapter 1 – Introduction | 13 |
| 1.1 Research Goal | 13 |
| 1.2 Project Plan | 14 |
| 1.3 Research Question | 17 |
| Chapter 2 – Literature research | 19 |
| 2.1 Internet of Things (IoT) | 19 |
| 2.2 Scenarios & Personas | 23 |
| 2.3 Conclusion literature research | 26 |
| Chapter 3 – Method | 27 |
| 3.1 Hardware | 27 |
| 3.2 Area of study: location and sensor placement | 59 |
| 3.3 Casing | 61 |
| 3.4 Data | 65 |
| 3.5 Feedback mechanism | 71 |
| Chapter 4 - Analyzing Data & Results | 76 |
| 4.1 Locations | 76 |
| 4.2 Temperature and humidity | 79 |
| 4.3 Noise measurements | 89 |
| 4.4 Air quality | 91 |
| Chapter 5 - Conclusion | 101 |
| Chapter 6 – Recommendations | 103 |
| References | 105 |
| List of figures | 110 |
| List of tables | 113 |
| List of maps | 114 |
| List of abbreviations | 115 |
| Appendices | 117 |
| Appendix A: Organizational Breakdown Structure | 117 |
| Appendix B: Work Breakdown Structure | 119 |
| Appendix C: Work Package Description (WPD) | 120 |
| Appendix D: Schedule of the GSP (GANTT chart) | 122 |
| Appendix E: Media outreach strategy | 129 |
| Appendix F: Stakeholders and experts | 135 |
| Appendix G: MoSCoW priority management | 136 |
| Appendix H: Boundary Conditions | 139 |
| Appendix I: Functional, non-functional and killer requirements | 141 |
| Appendix J: MicroPython script main.py | 145 |
| Appendix K: Sensor locations – facades | 147 |
| Appendix L: Sensor locations – trees and lanterns | 154 |
| Appendix M: Casing | 161 |
| Appendix N: System overview | 162 |
| Appendix O: LoRa JSON message | 163 |
| Appendix P: Sensor billboard | 165 |
| Appendix Q: Sensor flyer | 166 |

| | |
|--|-----|
| Appendix R: Twitterbot code | 167 |
| Appendix S: LoRa encryption | 174 |
| Appendix T: Test script MAX9814 | 176 |
| Appendix U: Test script AM2302 | 177 |
| Appendix V: Graphs and tables for AM2302 measurements | 178 |
| Appendix W: Graphs and tables for air quality measurements | 200 |
| Appendix X: Node-RED node codes | 215 |

Reading guide

This document provides an overview of the main elements of the GSP that have been accomplished. It describes both the organizational and technical features of the project. The first part of this report is the executive summary.

Chapter 1 provides an introduction to the present research. Furthermore, the research question together with the sub questions are presented. It then focuses on the project plan including the management strategy, planning, media outreach, the involved stakeholders, requirements, and boundary conditions.

Chapter 2 outlines the literature research that is done regarding Internet of Things (IoT). After this research, three distinct scenarios are introduced. For every subject that will be investigated a scenario is described. These scenarios function as primary assumptions of the project.

Chapter 3 elaborates on the method that is applied during the project. This chapter describes the hardware, the casing, and the locations that are chosen for placing the sensors. The data management is also explained in this chapter. Finally, the feedback mechanism is described.

Chapter 4 provides the data analyses together with a discussion on the measurement results. For every subject, there is an in-depth discussion on the meaning of the data and the implications of these results.

In Chapter 5 the conclusions regarding the research question and the sub-questions are stated. Finally, there are recommendations made for future research.

Chapter 6 contains the practical recommendations for researchers who will continue this project. These - mainly practical - recommendations are stated for future student groups, who want to build further on this pilot project

Throughout the document there will be referred to the citizens of Delft and the visitors of the city as: the citizens of Delft.

Sensor City Delft: An interactive static sensor network in the center of Delft

N. Salheb, C. Kleijwegt, R. Meulmeester N. Bebelaar, N. Vaissier, G. Michailidou

June 30, 2017

Abstract

This synthesis project is focused on implementing an Internet of Things (IoT) network to measure environmental data in the city of Delft. This network consists of sensor platforms that are placed in the urban environment. Each sensor platform is mounted on fixed locations and it is not moved during the measurement time. The aim is to raise community's environmental awareness to improve the quality of the environment.

Recent developments in technology made it possible to fabricate small, efficient, and reliable sensors boards which are the base of these sensors platforms and making them efficient and reliable. Sensor boards like Arduino, Raspberry Pi, and LoPy are some examples of these small sensor boards. In this project, the LoPy is used which is a sensor board that is equipped with Bluetooth Low Energy, Wifi and a LoRa radio. This last one is a communication technology that makes longer communication distances possible.

The sensor network measures four different environmental indicators that will be distributed to the public: temperature, humidity, noise and air quality. The network then communicates via LoRa this data to one centralized server where the data is stored, processed and sent back to the citizens. This data is made publicly accessible to academia, citizens and the stakeholders alike. The network is also made interactive, people who pass by can interact with the sensors and request specific environmental data in real time.

The sensor network has been build and deployed in the city. During the uptime of the network it succeeded to provide the data to the citizens via the feedback mechanisms: a website with a dashboard and an automated twitter account. Local differences have been measured with temperature and humidity sensors. With regard to the noise sensor and air quality sensors no definitive conclusions could be drawn.

1 Introduction

Sensor City Delft is a project conducted by six students of the TU Delft MSc Geomatics track for the duration of 9 weeks. This research project focused on raising local environmental awareness by interactively providing environmental data using a static sensor network. The sensor network has been installed in the City Center of Delft. The specific area of study is the Choorstraat, the Voldersgracht and the Oude Langendijk (see figure IV). These three streets are chosen because they have different profiles and are all located in the vicinity of the stakeholders. The project has been conducted in partnership with the Municipality of Delft, Science Center, SpingSmart and TU Delft.

This research serves as a pilot project of Internet of Things (IoT), the deployed sensor network serves as a proof of concept. The research has to be continued by future students or graduates. The innovative part of the project is that the network is fine-grained, i.e. it has a relatively high density of low cost sensors. This can provide a detailed picture of environmental indicators on a large scale. Moreover, the sensor platforms that are created can be deployed relatively easily in the built environment. Finally, an innovative aspect of the project is that it is interactive with citizens and actively involves them.

2 Aim

The aim of the Sensor City Delft research is to raise awareness about three main topics:

Smart City: A smart city is defined as a city that engages its citizens and connects its infrastructure electronically. A smart city has the ability to integrate multiple technological solutions, in a secure fashion, to manage the city's assets [1]. To inform citizens about this concept in a practical way, visible sensor platforms are deployed throughout the city center of Delft.

Local environment: A fine grained network of sensor platforms connected to the Internet enables research on local differences in the environment. Usually, this type data is generated from interpolation from distant platforms.

TU Delft academia vs Citizens: About 20% of the population of Delft is affiliated with the TU Delft [2]. To show what the TU Delft researched, and that the research also can benefit citizens, this project raises awareness of the presence of the TU Delft in Delft.

Raising awareness about these three topics at the same time has been done by making the network visible in the city and encouraging the citizens to interact with it. Therefore the research question of this paper is:

How to raise local environmental awareness by interactively providing environmental data using a static sensor network?

This research has been conducted with a case study which is the pilot of deploying a static sensor network in the city center of Delft. To be able to answer the research question the following sub questions are formulated:

1. How to make a static sensor network interactive?
2. What environmental data to collect and how to process it?
3. What environmental data is to be provided to the citizens to raise awareness?
4. Where to install the sensors?
5. How to build the sensor platforms?
6. How to ensure data quality?

By answering the sub questions, a complete picture of this project is shown.

2.1 Interactivity and visibility

Online and offline awareness is raised using visible sensor platforms, billboards, flyers, a website, local media and social media. The sensors were placed in eyesight on lamp posts and trees in the City Centre of Delft. Billboards and flyers were distributed in the nearby area, informing passersby about this project and how to retrieve data. The sensor platform is made interactive in two ways: Firstly, by equipping the billboards with QR-codes which link to the website www.scdelft.nl. This website contains general information on the project, news and a dashboard with the latest readings of the sensors and time series of the data coordinated per street (see figure I).

The second way of interactivity is using a twitterbot. This is a computer program that uses Twitter (@SensorCityDelft) and responds to users who tweet at it. The twitterbot is capable of reading tweets and distinguish between different environmental data that is requested by users. By looking at hashtags with both the street name and the environmental characteristic, a reply is given (see Figure II).

2.2 Environmental Data

The gathered data consists of temperature, humidity, noise levels and air quality. These were chosen because they are known to have serious health effects on people. These data are measurable with low cost sensors and are of most interest to all stakeholders.



Figure I: Dashboard of the latest readings



Figure II: Exemplary Twitterbot response

2.3 System Design

For the Sensor City Delft project, sensor platforms were created; each one having three types of sensors mounted on them. The air quality sensors are both the PMS5003 and PPD42NS, the noise sensor is the MAX9814 and the temperature and humidity sensor is the AM2302.

After a couple of test rounds, it seemed that not from every type of sensor reliable information could be expected. The PMS5003 and AM2302 can be regarded as reliable, but the PPD42NS is wind-sensitive and the MAX9814 sensor is in fact an indoor microphone, instead of a decibel meter. With these remarks taken into account, the sensors are used though.

The communication protocol that is used to send the measured values from the sensor boxes to a database on a server is LoRaWAN. The underlying modulation technique is LoRa, which allows the sender to send a message with low power, also when noise is high and distances are big [3]. In the deployed solution, every platform in the implemented sensor network, measures the environmental values once in every 15 minutes. Then, these values are sent to an application server over the KPN LoRa network. These intervals are chosen due to limitations of the LoRaWAN network and energy saving on the sensor platform.

The measured data is sent from the sensor platforms to an application server which makes use of Node-RED and PostgreSQL. Node-RED is a browser-based flow editing programming interface for connecting Internet of Things devices. PostgreSQL is an open source object-relational database tool. The Node-RED environment serves as the central location to manage data flows.

The data flows origin from the sensor platforms. The messages are sent encrypted over the LoRaWAN network and decrypted on the application server. After decryption the data is parsed and processed to make it ready for storage. The data is stored in a PostgreSQL database.

Two other important aspects of the system are also originating from the application server: the twitterbot and the dashboard on www.scdelft.nl. The twitterbot retrieves stored data from the database and so does the dashboard. Node-RED includes easy setup for data dashboards, the website embeds these dashboards. An overview of the system as described in the former paragraph can be seen in figure III.

2.4 Implementation

The sensor platforms placed in the City Center of Delft were measuring for one week. During this week, the twitterbot replied to several people who requested data. The website has been providing live data on a dashboard in this time. Historic data is also shown on the website.

2.5 Ensuring Data Quality

Ensuring data quality is important to draw reliable conclusions. At first, the sensors need to be reliable and of good quality. For this reason, they are tested in an indoor environment, where weather influences are minimized. After this, being critical on the data output is very important:

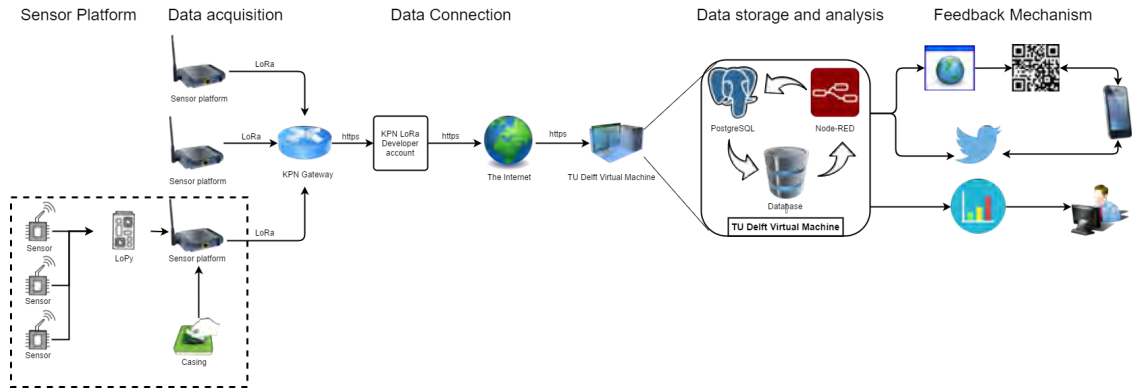


Figure III: System Overview

outliers, systematic errors or decay of hardware are still possible. Placing the sensors at a fixed location which does not influence the measurements is also important for good data quality.

Furthermore, in this project in total five sensor platforms are placed; two streets with two devices and one street with one. To draw better and more reliable conclusions, redundancy of data is needed, therefore more sensor platforms should be placed.

2.6 Sensor Locations

To ensure a maximum amount of people is being reached, the locations are chosen because they are visited by many citizens and tourists. Likewise, the different areas of interest have different patterns and environments to detect differences if present. Therefore, the Voldersgracht, Oude Langendijk and Choorstraat are good locations to place the sensors.

The sensors are placed on trees and lanterns in these streets, out of reach for people to prevent theft or vandalism (see figure IV). This way of placing the sensors has affected the measurements, which is caused by exposure of direct sunlight and wind.

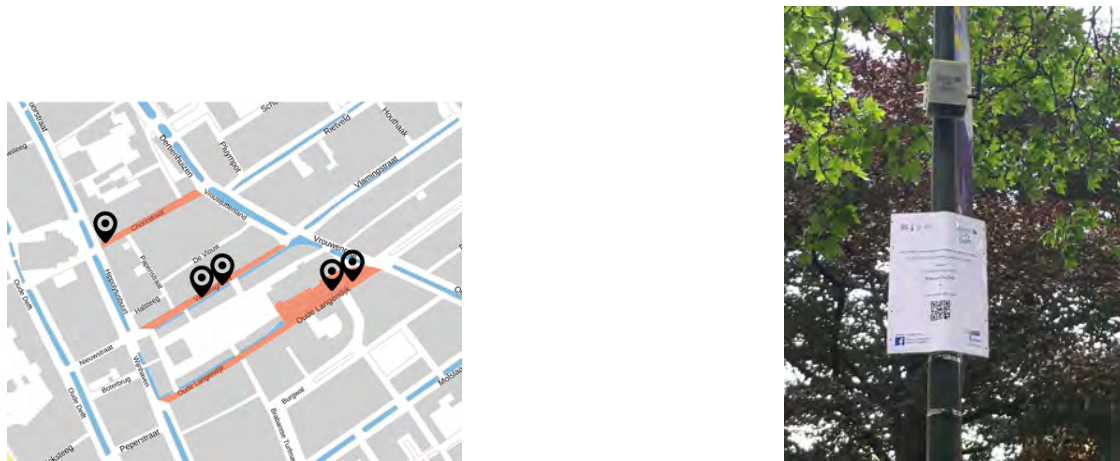


Figure IV: Sensor locations (left), sensor and billboard (right)

2.7 Data Analyses

The measured data in the streets showed local differences in humidity and temperature. These differences have been accredited to the fact that some street were receiving more sun than others. The noise levels had no viable conclusions due to coding errors and the type of the sensor: it is a microphone. One type of the air quality sensors was receptive for wind which led to skewed results in air quality measurements.

3 Conclusions

The main aim of this project of raising the environmental awareness was done by successfully installing an interactive static sensor network that provides environmental data. Measuring the amount of awareness that has been raised among citizens is hard however. But the entire technical system has been set up in this project, and serves as a proof of concept for future projects on interactive static sensor networks. The sensor network has been built, it gathered data and was interactive.

Shortcomings of not having data of high quality did not affect the concept of the project. The sensor network was working as intended, and if high quality sensors would have been used, the measurements would have also been of higher quality. Moreover, the experiences with the LoRaWAN communication protocol is that it is a reliable communication technique. That is mainly because the LoRa network that is used is relatively saturated in the study area.

Relative to the short span of the project it reached citizens. Via Facebook 60 people are reached (with ‘likes’), the project is covered in two local newspapers, the sensor billboards have been displayed in the streets of Delft for a week, and the project is pitched in front of a crowd of (international) students. Multiple citizens also interacted with the system using twitter and the website and were able to acquire the environmental data they asked for.

3.1 Future Research

To improve and continue this research, the following steps have to be taken. First, the measuring time should be extended over longer period, for example 1 year. During this period, the measurements should be more reliable. Furthermore, the number of sensor platforms should be raised to detect more local differences. Finally, the data should be communicated and processed according to an (open) standard; the OGC Sensor Web Enablement standard.

To be able to measure for a year, there are improvements in the hardware and casing needed. This means that the battery should last longer, which can be done by installing the solar panels on the sensor platforms, or by connecting the sensor platforms to the electrical grid of the city. Another way is to reduce power consumption by the microcontroller itself. In the case of the LoPy this would mean using a ‘deepsleepshield’: a component that was not yet available during the course of the project time.

To make the measurements more reliable, a longer testing and calibration phase of the sensor is needed. Systematic errors of the sensors will then be detected. Moreover, the casing needs an improvement by adding a separate component to the sensor casing. With this extra component the air quality data will be more reliable since it is not affected by wind.

Finally, the psychological effect of implementing a sensor network can be researched. The current situation of environmental awareness of the citizens should be measured before implementing the sensor network (ex ante). Then, after implementing the sensor network in the city the environmental awareness should be evaluated, to see the effect of the sensor network (ex post). To conclude, different city centres of different cities can also result in interesting findings about the environmental issues of the cities.

References

- [1] Musa, M., 2016. Smart Cities: A road-map for development. Accessed at 19/06/2017 via http://www.academia.edu/21181336/Smart_City_Roadma1
- [2] TU Delft, 2017. De studentinstroom aan de TU Delft is dit jaar licht gestegen. Nieuws 26 oktober 2016. Accessed at 19/06/2017 via <https://www.tudelft.nl/2016/tu-delft/instroom-tu-delft-stijgt-licht/>
- [3] Lora Alliance, 2017. LoRa Alliance™, 2017. LoRa Alliance™ Technology. Accessed at 19/05/2017, via <https://www.lora-alliance.org/What-Is-LoRa/Technology>

Chapter 1 – Introduction

Collecting environmental data from the city can be done in many ways. One option for this is to create a 'smart city'. This means that a sensor network is applied to a city and many elements can communicate with each other, does measurements and save these to a relational database. This smart city way of working using Internet of Things (IoT) is developing at the moment.

This report informs about a project from the master of Geomatics at the TU Delft. Internet of Things and collecting data using a sensor network is the central subject of this project. The sensor network will be static (i.e. located in one place in the city center and not moving) and interactive. This last element means that the citizens can request the data of the sensors.

1.1 Research goal

The project has two parts from different sources that both have their own goals. The first one is the goal of the course from the MSc Geomatics programme, which is, as described in the Course Catalogue of the university:

"The Synthesis project aims at the application of knowledge and skills gained in the core programme.

During the Synthesis project students have to:

- *define an innovative project, developing new knowledge on novel geomatics subjects;*
- *apply the knowledge from the core courses in a project, from data acquisition, processing, storage and validation to visualization, analysis and conclusions and show controlling the subject;*
- *be able to systematically report on the conducted research, covering the research process and the outcomes;*
- *show the ability to work together in a project team."*

The other part of the project that have goals set is the stakeholder part. The stakeholders invest time and money in this project, have certain expectations of the outcomes of it and reasons on why to be involved in this project. The goals from the stakeholders are:

- Get evidence of the environmental issues and status of the city center.
- Bridge the gap between the TU Delft and the citizens of the city Delft i.e. communicate the research and development of the university to the citizens.

1.2 Project plan

Each project has an organisational element in it. Because of this element, the actors know what to do, appointments are met and the goal will be reached. The different elements of the organisational aspect of this project are described below.

1.2.1 Overall approach, management and planning, media outreach

As the synthesis project is part of the MSc Geomatics programme there is a limited amount of time, this is related to the academic calendar and the amount of time reserved for the project. The project will run over a period of 9 weeks, i.e. from the 21st of April 2017 to the 23rd of June 2017. For the successful completeness of the project, this time needs to be spent efficiently and all group members should know what to do and when deadlines are set. These management aspects are described later in this document.

Besides time as an important aspect of this project, all team members will have roles and tasks assigned to them at the beginning of the project. While the project proceeds roles can be changed and tasks can be added. The project plan consists of a couple of documents which the team members agreed upon. The different roles of the team members are described in the Organizational Breakdown Structure (OBS) (see Appendix A). Next to that, in order to provide a clear overview of the different tasks a Work Breakdown Structure (WBS) is constructed (Appendix B). The Work Package Description (WPD) (Appendix C) is a detailed description of the tasks that are mentioned in the WBS. The description of the project planning can be found in Appendix D, where a GANTT chart is included. All work that is done on the synthesis project will be documented in the log that is part of the DIDs (Deliverables Items Descriptions). To have a fast glance at the project planning, figure 1 below is a simplified workflow of the project, to show the parallel processes.

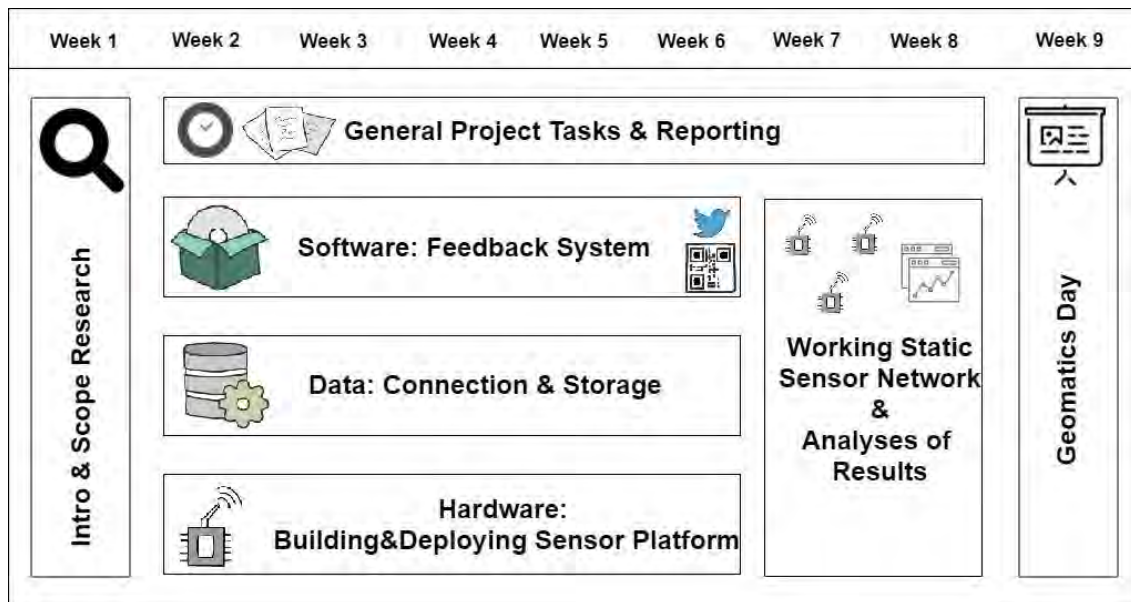


Figure 1: Aggregated workflow

Another managerial task is the media outreach, which is a major task in the project since the goal of the Sensor City Delft project is to raise environmental awareness of the citizens of Delft by proactively providing environmental data. Therefore, the citizens must be informed via media. A strategy for reaching the media is given in Appendix E.

1.2.2 Stakeholders, boundary conditions and requirements

A project is a continuous cycle that can change while it is on the run. The stakeholders and experts who are involved in this project are briefly described in Appendix F. The priority management plan ('MoSCoW') is placed in Appendix G. MoSCoW is a project management technique that aids in understanding and managing priorities (Verbree, 2017; Agile Business Consortium, 2017). It is an acronym for *Must*, *Should*, and *Could* have, and *Will not* have this time, and helps the project group staying focused on the most important tasks regarding the project.

In appendix H are boundary conditions of this project described: it focuses on the context in which this group project takes place. Appendix I is an exhaustive description of the top down requirements for the Sensor City Delft project. These requirements are derived from the project description of the programme, the request of both the actors and stakeholders, and from the goals of the project team. It is the basis upon which the project is built and describes the agreement with the stakeholders. In general the following requirements and boundaries are set:

- The project should be interactive

- The project should be completed within the set timespan (nine weeks)
- A maximum amount of sensors is set to 40
- Only data of interest will be processed
- As much as possible different data will be collected

The Project Logic Diagram in figure 2 shows the overview of the to be delivered product in one image. To express the centrality of the sensor placement and usage, a house is the center of the image. Around this house, three elements of the project are described: the hardware, software, and the connection.

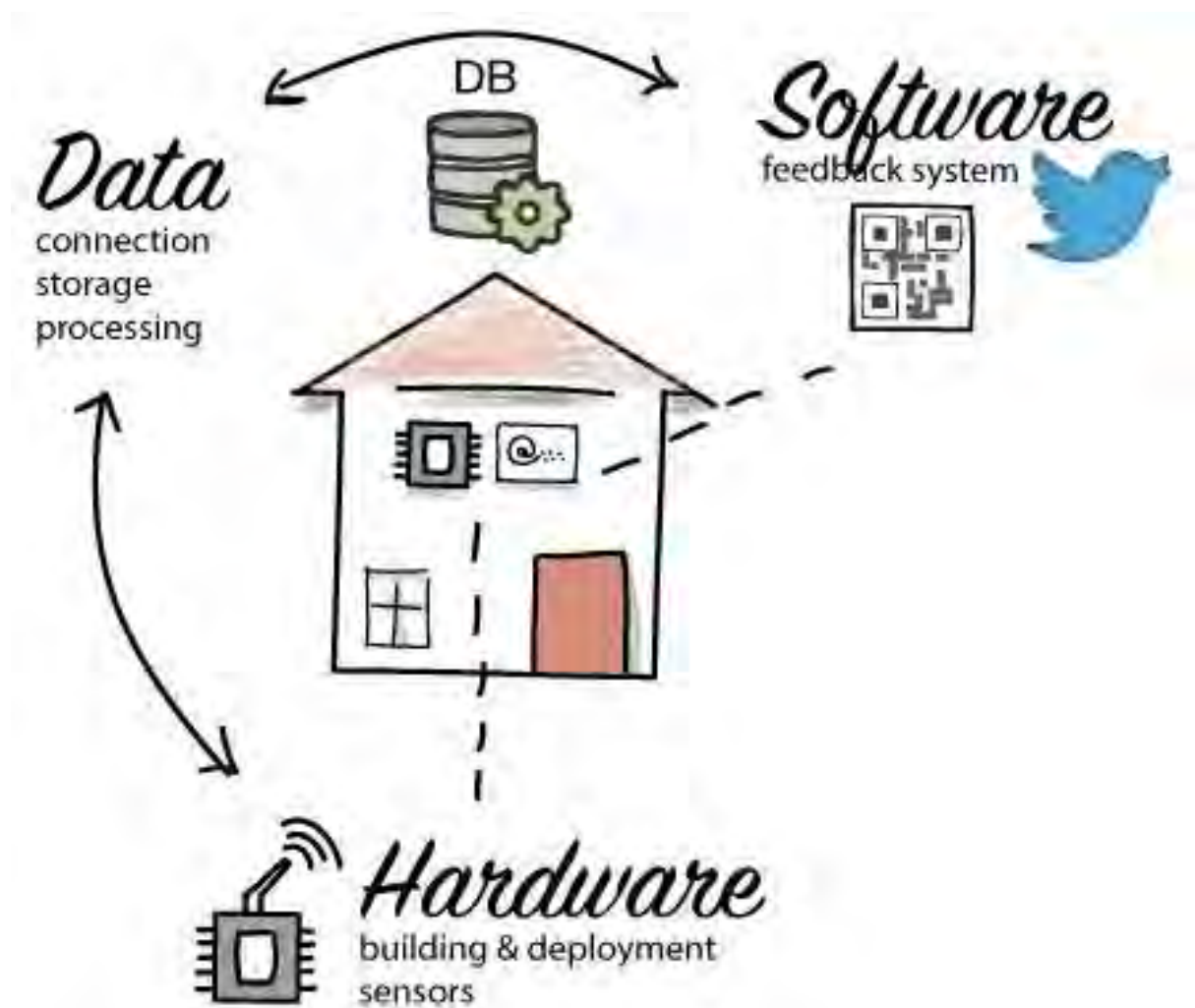


Figure 2: Project Logic Diagram

1.3 Research questions

After having described the research goals and considered the aims of the stakeholders the following research question can be derived:

How to raise local environmental awareness by interactively providing environmental data using a static sensor network?

Case study: a pilot in the city center of Delft

To be able to answer this research question the next six sub questions are formulated:

1. How to make a static sensor network interactive?

To make a sensor network interactive many things have to be researched. First of all is how to reach the target audience. And secondly how to provide the collected information to the audience in an approachable and meaningful way. The storage of the data is also an important aspect that is covered with this sub question.

2. What environmental data to collect and how to process it?

Since there are too many phenomena to be sensed all at once in the city, decisions have to be made about what to sense exactly. These environmental values have to be measurable with low cost equipment. Besides technical aspects of measuring, the interests of stakeholders also play an important role. During a planned stakeholder meeting their points of interest can be defined.

3. What environmental data is to be provided to the citizens to raise awareness?

A good way to define what data to collect is to create scenarios that will describe different personas that are part of the target audience. The scenarios will be based on the interests of the stakeholders, but should also be relatable to the average person in the city center of Delft.

4. Where to install the sensors?

Sensor platforms will be placed at locations within the city center of Delft that have different profiles. Therefore different results are expected within the area of study and local differences might be present. Since the sensor network is interactive, the locations of the sensors need to be visible to the target audience.

5. How to build the sensor platforms?

Research has to be conducted on several components of the sensors platforms. This includes the sensors, the microcontroller, the casing and the means of communication of the data.

6. How to ensure data quality?

Exhaustive testing and calibration of the sensors have to be conducted before deployment in order to provide reliable data. Sensor placement might also affect the quality of data.

Chapter 2 – Literature research

This chapter covers the literature research that is carried out for this project. The first paragraph focuses on the Internet of Things concept. In the second paragraph is argued for three scenarios that are relevant with respect to the Sensor City Delft project.

2.1 Internet of Things (IoT)

Internet of Things (IoT) is considered as a technological and economic wave in the global information industry (Chen et al, 2014). The aim of IoT is to achieve intelligent identifying, locating, tracking, monitoring, and managing of things (Stankovic, 2014). This will expand an Internet-based network from human-human to human-things and things-things communication, promoting the harmonious interaction between humans, societies, and smart things (Guo et al, 2013; Zhong et al, 2013; Chen et al, 2014). The availability of different types of data, collected by a pervasive urban IoT, may be exploited to increase the transparency and promote the actions of the local government toward the citizens, enhance the awareness of people about the status of their city, stimulate the active participation of the citizens in the management of public administration, and also stimulate the creation of new services upon those provided by the IoT (Zanella & Vangelista, 2014).

2.1.1 Elements

As described by Gubbi et al (2013), three components are required for Internet of Things:

- 1) hardware, i.e. the sensors, actuators and embedded communication hardware;
- 2) software, i.e. the on demand storage and computing tools for data analysis, and
- 3) presentation, i.e. the visualization and interpretations tools which should be designed by different applications and can be widely accessed on different platforms (Gubbi et al, 2013).

Hardware

The hardware consists of two main technologies: the sensors and the ways of communication. The most used way of communication is Radio Frequency IDentification (RFID), for the sensors these are Wireless Sensor Networks (WSN). The former is well established for low cost identification and tracking. The latter, however, brings IoT applications more capabilities for sensing, providing low-cost data acquisition and actuation. Today, WSN's cover a wide range of applications, mainly used to monitor physical or environmental conditions. A combination of WSN and RFID can result to a better tracking of the status of locations, temperature, movements etc. (Lee & Lee, 2015).

Software

Two types of IoT software are used for the deployment of successful products and services: the middleware and cloud computing. The middleware is a software layer interposed between software applications that allow developers to perform in easier ways, that handles communication and controls input/output of devices. The middleware is useful when the simplification of the development of an IoT applications is required, for example in a complex distributed infrastructure of IoT with several heterogeneous devices (Lee & Lee, 2015). The cloud computing model provides an ideal back-end solution for the processing of massive amounts of data streams in real-time.

Presentation

Today, only a small number of mainly human-centered IoT applications allow both the visualization of information and the interaction with the environment. With the introduction of the touch screen technologies and the use of smartphones and tablets, this interaction became intuitive, easy to understand and meaningful for the end users (Gubbi et. al., 2013). However, the cases where IoT applications where the visualization and interaction of the data are both critical and required are limited.

2.1.2 Challenges

In a hyper-connected world, the lack of appropriate standardization of applications, security and privacy, as well as the implementation of complex communicating systems can result to chain reactions with unwanted or even disastrous results (Lee & Lee, 2015). This part provides the four main challenges that need to be concerned by enterprises in the IoT development. These are the data mining, data management, privacy, and policy challenges. According to Gartner (2014); *“the enormous number of devices, coupled with the sheer volume, velocity and structure of IoT data, creates challenges, particularly in the areas of security, data, storage management, servers and the data center network, as real-time business processes are at stake”* (Gartner, 2014). The four challenges are further discussed below.

Data mining

IoT sensors are generating massive amounts of data that need to be processed and stored. However, the current architecture of data centers is not able to support the heterogeneous nature and the volume of the personal and enterprise data (Gartner, 2014). Following this, the enterprises should be able to invest in data storage in order to support all the IoT data collected from their networks (Lee & Lee, 2015). In addition to this data centers should improve processing efficiency and response time for the

IoT devices that widely used and consume more bandwidth.

Sensor platform OGC Standardization

The number of internet-connected sensors is rapidly growing around the world (Ericsson AB, 2016). To be able to combine and compare the data from these sensors, standardization is a key requirement. The Sensor Web Enablement (SWE) standards framework by the Open Geospatial Consortium (OGC) meets this requirement. Implementing this standard in the project will make it easy to integrate information regarding sensor locations into geospatial applications that implement the OGC's other standards (OGC 2017a). This will also make the network discoverable, accessible and useable via the Web (OGC 2017b). An example of the OGC Standards in the SWE framework is the SOS standard. Implementing SOS will make the sensors data manageable in an interoperable way. This standard can define a Web service interface which allows querying observations, sensor metadata, as well as representations of observed features. Further, this standard defines means to register new sensors and to remove existing ones. Also, it defines operations to insert new sensor observations (SOS, OGC 2017c).

Data management

In addition to the demand of investment in data storage and processing, the massive amount of data requires the use of data mining tools. The data is a combination of discrete and streaming data associated with location, movement, temperature, humidity, among others. These unstructured data need to be analyzed and understood using mathematical and computer models, that the traditional data mining techniques are unable to provide. Furthermore, both the addressing of operational issues and the information of the enterprises about the changes in customer's preferences require advanced data mining tools and expertized data analysts (Lee & Lee, 2015).

Security and privacy

Security and privacy is becoming one of the biggest challenges for the success of IoT (Babar et. al, 2010). Wireless devices are expected to be the platform of choice for launching attacks targeting the Internet. Furthermore, problems like confidentiality, authenticity, and integrity of data sensed (such as address and names) and exchanged by "things" have become apparent. The most important parameter associated with wireless networks is location and so location information provided by the network should be trustworthy (Hu & Wang, 2006). According to the 2016 edition of the Vormetric Data Threat Report, the protection of sensitive data and the privacy of things has ranked as the top concern of the most enterprises (Vormetric, 2016).

2.1.3 Applications

Between the numerous applications of IoT, 'smart cities' is one of the most promising application fields (smart-cities.eu, 2015). However, the Smart City market has not taken off due to political, technical and financial barriers. According to the smart cities European project, five service sectors of smart cities can be introduced, the Smart Governance, Smart Mobility, Smart Utilities, Smart Buildings and Smart Environment. These service sectors can be defined as indicators for ranking the 'smartness' of European cities (smart-cities.eu, 2015).

Several services can be defined regarding smart cities: the maintenance of historical buildings, waste management, air quality and noise monitoring, traffic congestion, city energy consumption, smart parking, smart lighting, and automation of public buildings (Zanella & Vangelista, 2014). The practical realization of most of these services is not hindered by technical barriers but by the lack of a widely accepted communication architecture that can provide harmonized access (Zanella & Vangelista, 2014). In the following parts are some examples of implementations of the smart city concept described.

Padova Smart city project

The Padova Smart city is a web-based project implemented on the island of Padova. The project was interconnected with the data network of the city municipality. The aim was to discuss a general reference framework for the design of the IoT services, as well as related protocols and technologies. In addition, it was aimed to provide a discussion on both the suitability of the proposed framework for a Smart City environment and the solutions regarded technical issues of the application (Cenedese et al, 2014).

SmartSantander project

SmartSantander is one of the projects developed by the European Commission, in the context of the Future Internet Research and Experimentation initiative. The project represents a city-scale experimental research facility. It is aimed to provide both a description of the testbed architecture and the deployment issues and experiences of its implementation.

For this project, four different uses cases were considered. The Parking Space Management use case aimed on the development and deployment of a service that enables monitoring the occupancy of outdoor parking spaces on the streets of Santander city center. The data could be subscribed by mobile applications. The Precision Irrigation was aimed to monitor the plants' requirements in water, augmenting the existing irrigation system along the parks and gardens. The Augmented Reality use case was aimed to provide essential location information such as Points of Interest (POI) to citizens and

tourists, using NFC tags. The Participatory Sensing use case was aimed to give the ability to users to monitor location or environmental data by transforming their smartphones into sensors (Sanchez et al, 2013).

Conclusion IoT

Internet of Things (IoT) is one of the most promising fields in the global information industry, targeting to the intelligent identification, location, tracking, monitoring and management of things. For the implementation of an IoT system, 3 different components are required: the hardware, the software and the presentation / visualization of the data. Among the numerous advantages for the society and the economy, an IoT system could lead to the transparency of the governance and improve the awareness and communication between citizens and the city (Crump & Brown, 2013). In this way, an IoT system could expand the current Internet-based network from humans-humans to humans-things and things-things communication. However, challenges like the lack of the appropriate 1) standardization, 2) privacy, and 3) security of these applications, together with 4) the lack of tools for mining and analyses of the big data could act as barriers in this success. Furthermore, IoT systems could be applied to cities, leading to the notion of the 'smart cities'. 'Smart cities' is one of the most rapid growing application topic, with the most small- or large- scale applications being related to a city's infrastructure systems and energy fields.

2.2 Scenarios & Personas

As described in the previous section, it becomes clear that within the scope of IoT there are a lot of different subjects to do research on. To limit these options and define some experiments three scenarios are written for this particular project. The scenarios are measuring air quality, noise levels, temperature and humidity. On these scenarios several decisions are made like types of sensors and interval of measuring. To make the scenarios more 'alive', there are three fictional 'stories' described in the following parts.

2.2.1 Air quality at Oude Langendijk

Peter owns a lunch-cafe at Oude Langendijk, in the center of Delft (figure 3). As time passes by, Peter is getting more and more concerned about the health of his small business. His cafeteria attracts less customers and, thus, Peter decided to learn why. After discussions with customers and several experiments on the menu and internal decoration, Peter realized that the main reason why customers did not choose his cafeteria anymore was the crowded street where it was located. Buses and scooters

were producing lots of noise and odor. The quality of the outside environment was not the same as it used to be. Peter is now trying to find out whether the daily traffic affects the air quality of his street and those nearby (Choorstraat, Voldersgracht). However, he understands he does not have the knowledge and permission to do that.

Peter

Social, cost-driven character



"People's satisfaction is the most important thing in our business. Once having this, success is guaranteed"

Profile

Peter spends most of his time at the center of Delft, in a lunch-café he owns. He has not introduced any means of technology at his business life yet. However, if an application would help him gain more money he would be highly motivated to use it.

Peter enjoys watching movies and listening to music. For 4 years now, Peter is the drummer of "Kind Giants", a rock group that he and his friends created.

Peter loves his family and he's always trying to find more time to spend with them.

Personal Information

Age: 53

Location: Delft, Netherlands

Knowledge of IoT: Low

Profession: Owner of business

Hobbies: Music, Movies

Favorite Movie: Fast & Furious

Personality: Social, Hard-working

Internet use

Level of daily use: Low to Medium

Figure 3: Personality of Peter

Air quality will be expressed in a measure called Particulate Matter (PM), which is categorized in three types: PM1, PM2.5 and PM10. Particles that belong to the PM10 category are smaller than 10 micrometer and larger than 2.5 micrometer. These are large organic particles like coarse dust, sand, leaves or hairs. PM2.5 particles include particles such as pollen and spores. These particles are smaller than 2.5 micrometer and larger than 1 micrometer. PM1 particles are the smallest particles: smaller than 1 micrometer and larger than 0.3 micrometer. To this category belong particles such as dust, combustion particles, bacteria and viruses. Fine particles such as PM2.5 and PM1 have been found to play a significant role in health issues, climate change and pollution problems (Vecchi et al., 2004).

2.2.2 Noise level at the Choorstraat

Eline has been working at the municipality of Delft for 3 years (figure 4). Being responsible for the external communication, Eline is communicating with citizens to be updated about how citizens perceive their daily lives in the city of Delft. She decided to start this research when she received complaints about the loud noise produced by a bar/restaurant at Choorstraat. Citizens reported that

the problem was even bigger when parties or other events were taking place at this bar. Eline had to take actions to solve this problem but she did not know if the magnitude of noise was dangerous for health of residents (Haines et al, 2001). She also did not know how to find the data she needed to answer her questions or compare the situation in the Choorstraat to other streets in the neighborhood.

Eline

Always informed, social, seeks innovation



"Design for people and everyone loves you, users or companies"

Profile

For Eline, personal evolution is one of the most important aims in life. Eline loves her job! As a social person, she enjoys meeting with different people and getting to know more from and about them. More important, she wants to use all acquired knowledge at her job with the view to create something innovative. Eline is open to all ideas which can transform Delft into a most attractive place for visiting or living.

She is always updated about recent innovations because she likes to attend related conferences or discuss with passionate people who can help her learn and improve.

Personal Information

Age: 38
Location: Rotterdam, Netherlands
Knowledge of IoT: Medium
Profession: Marketing in Delft Municipality
Hobbies: Reading, Movies
Favorite Movie: 12 Angry Men
Personality: Sociable, engaging but critical

Internet use

Level of daily use: High

Figure 4: Personality of Eline

2.2.3 Temperature and humidity at Voldersgracht

Richard was always curious about environmental issues (figure 5). Recently, Richard went shopping at the Choorstraat, in the center of Delft. On his way back home, Richard passed by the Voldersgracht where he immediately felt a temperature drop and rise in humidity levels. He found this strange but he thought that such differences might be normal and caused by the presence of canals and trees along the street. He continued walking. At the end of the street he had to turn right into the Oude Langendijk to get the bus from the bus stop. After a few minutes of walking at the Oude Langendijk he unzipped his jacket. He immediately realized that he did this without thinking, just because he felt again a rise in temperature. It was true that the street was generally very busy, with busses and cars driving up and down, but his actual feeling was not wrong. The air was warmer and less humid, Richard thought. This shopping day at the center of Delft was the first trigger for him to look further into the temperature and

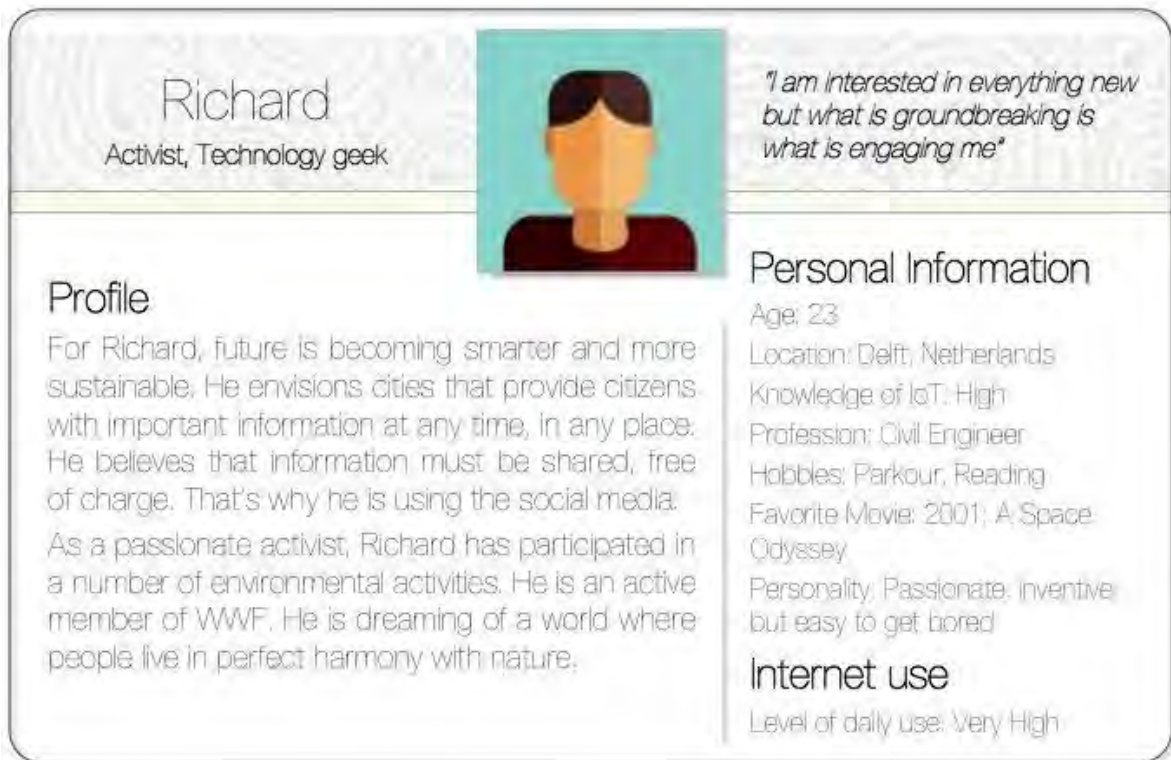


Figure 5: Personality of Richard

humidity differences which are found at certain streets and search for factors which might cause them. Is it possible that the traffic or the canals or the trees affect the temperature of certain areas so much? If yes, to what extend and when in time?

2.3 Conclusion literature research

In this chapter a literature research was presented regarding Internet of Things. In this context, the definitions, elements, challenges and IoT applications of Smart Cities were discussed. Furthermore, three different scenarios and personas were introduced with an important role in the project's decision-making process. As can be read in the literature research there are many topics that will be covered in this research. In Chapter 4 the three scenarios will be analyzed. But first, in the next chapter a design for this project is researched and elaborated, where the middleware and cloud computing are covered.

Chapter 3 – Method

This chapter describes the technical system of the project. It is subdivided into 8 sections, each of which is described in a separate paragraph. The flow and accessory paragraphs can be seen in figure 6. The sensor platform is described in paragraphs 3.1, 3.2 and 3.3. This is subdivided in the internal hardware of the sensor platform, the placement of the platforms in the city, and the casing of the sensor platform. Paragraph 3.4 describes the data storage and analysis part of the system. Finally, the feedback mechanism is treated in paragraph 3.5.

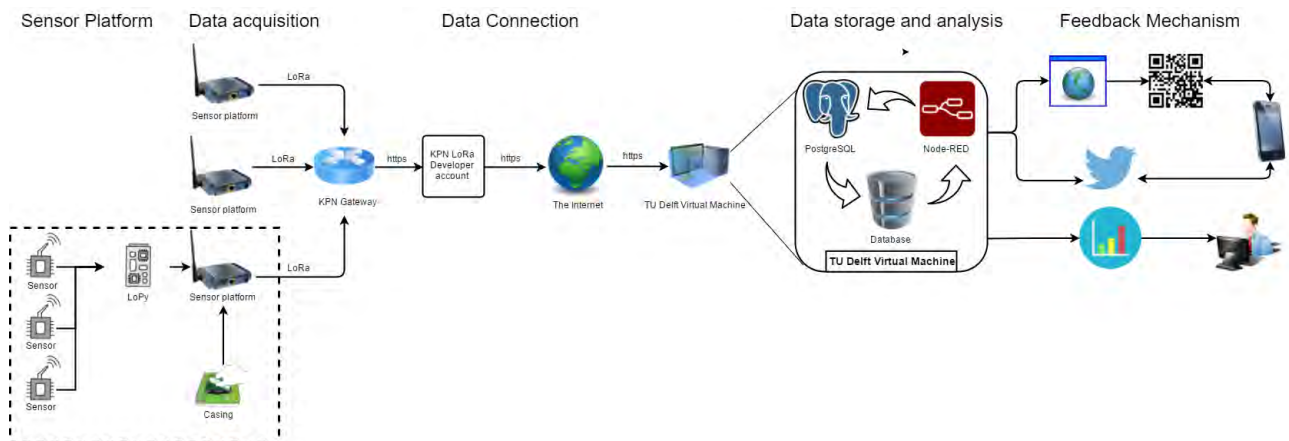


Figure 6: Flow of the technical system

3.1 Hardware

In this paragraph is the hardware that is going to be used for the Sensor City Delft system described. The hardware parts of the sensor platforms are the sensors themselves, development boards (including communication over LoRa), and the charging of the sensor platforms. They are first described separately, and then the assembly of the sensor platforms is described.

3.1.1 Introduction sensors

The sensors are the devices that pick up the environmental data in the first place. The sensors that will be used are the sound sensor MAX9814, the air quality sensors PMS5003 and PPD42NS, and the temperature and humidity sensor AM2302. Scripts are created for reading the data from the sensors which are connected to the LoPy development board. These scripts are included in Appendix T, U and F of this report. The scripts can also be found in the SensorCityDelft GitHub repository (<https://github.com/NiekB4/SensorCityDelft>).

3.1.2 Air quality sensor 1 (PMS5003)

To gather data for the first use case is an air quality sensor needed, which will be the PMS5003 sensor, see figure 7. The PMS5003 is a digital particle concentration sensor. It can be used to obtain the number of dangling particles in the air, and therefore it can give an indication of the concentration of particles (Yong, 2016).



Figure 7: PMS5003 air quality sensor (source: Yong, 2016)

The laser scattering principle is used for this sensor. This technique produces scattering by using a laser that radiates dangling particles in the air, then the scattering light is collected, and consequently the curve of scattering light change with time is obtained. After that, the number of particles with a certain particle diameter per unit volume are calculated with a microprocessor mounted on the PMS5003 (Yong, 2016). For an overview of the different modules in the PMS5003, see figure 8, image left.

The output of the sensor is in digital format, so it needs a digital input on the LoPy. An UART connection protocol is used to connect this sensor to the LoPy. This protocol requires at least two available Pins on the LoPy. For the connection circuit to the LoPy, see figure 8, image right.

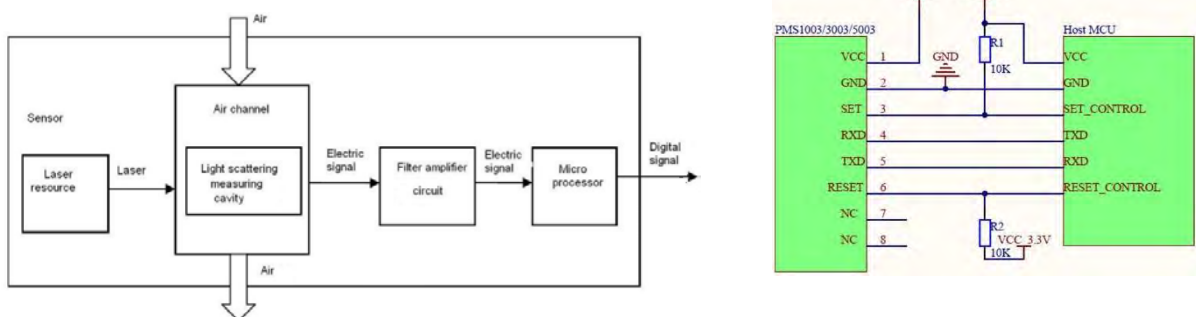


Figure 8: PMS5003 schematic overview (left) and circuit (right) (source: Yong, 2016)

The MicroPython language features an UART library to read the data from the sensor (MicroPython, 2017).

Volt booster

Since the PMS5003 air dust sensor needs an input voltage of 5 volts, while the LoPy is able to give only

3.3 volts output, a step-up converter ('volt booster') is needed. The MT3608 is an appropriate one, which is able to work with input voltages ranging from 2 volts to 25 volts (Aerosemi Technology Co, 2017). The output of the booster is set at 5 volts. Next to powering the air quality sensors is the step-up converter a useful device to extend the duration of the battery charge: the battery output can drop to 2 volts minimum and the devices will still work.

PMS5003 quality sensor 1: wire connections

This sensor is connected to the perfboard with 6 cables. See table 1 for the cables and the related PINS of the LoPy module.

| PIN Perfboard | PIN air quality | Abbreviation | Cable color | Description |
|----------------|-----------------|---------------|-------------|---------------------------|
| Power + | PIN 1 | VCC | Red | Positive power 5V |
| Power - | PIN2 | GND | Black | Negative power |
| P11 | PIN3 | SET | Yellow | Set PIN |
| P8 | PIN4 | RX | Green | Serial port receiving PIN |
| P9 | PIN5 | TX | Blue | Serial port receiving PIN |
| P10 | PIN6 | RESET | White | Module reset signal |
| - | PIN7/8 | No connection | - | - |

Table 1: PMS5003 pins and connections

VCC is connected via the volt booster module, GND is connected to both the GND of the volt booster as well as the GND of the perfboard (which is connected to the battery GND).

Air quality sensor 1: data calibration

The PMS5003 sensor provides data about particles in the air. The particles that are measured are PM1.0, PM2.5 and PM10 (PM = Particulate Matter). See the literature research for a discussion on Particulate Matter. PM data is provided under normal circumstances and under atmospheric environment. Therefore, there are six types of PM data in total, they are all provided per cubic foot (CF). Next to PM data is also the number of particles of a certain diameter per 0.1 Liter of air provided by the PMS5003, however they are not send to the database. See figure 9 for an overview of the data provided by the PMS5003.

Default baud rate: 9600bps Check bit: None Stop bit: 1 bit

32 Bytes

| | | |
|--------------------------|-------|--|
| Start character 1 | 0x42 | (Fixed) |
| Start character2 | 0x4d | (Fixed) |
| Frame length high 8 bits | | Frame length=2x13+2(data+check bytes) |
| Frame length low 8 bits | | |
| Data 1 high 8 bits | | Data1 refers to PM1.0 concentration unit μ g/m3 (CF=1, standard particle) * |
| Data 1 low 8 bits | | |
| Data2 high 8 bits | | Data2 refers to PM2.5 concentration unit μ g/m3 (CF=1, standard particle) |
| Data2 low 8 bits | | |
| Data3 high 8 bits | | Data3 refers to PM10 concentration unit μ g/m3 (CF=1, standard particle) |
| Data3 low 8 bits | | |
| Data4 high 8 bits | | Data4 refers to PM1.0 concentration unit * μ g/m3 (under atmospheric environment) |
| Data4 low 8 bits | | |
| Data5 high 8 bits | | Data 5 refers to PM2.5 concentration unit μ g/m3 (under atmospheric environment) |
| Data5 low 8 bits | | |
| Data6 high 8 bits | | Data 6 refers to concentration unit (under atmospheric environment) μ g/m3 |
| Data6 low 8 bits | | |
| Data7 high 8 bits | | Data7 indicates the number of particles with diameter beyond 0.3 μ m in 0.1 L of air. |
| Data7 low 8 bits | | |
| Data8 high 8 bits | | Data 8 indicates the number of particles with diameter beyond 0.5 μ m in 0.1 L of air. |
| Data8 low 8 bits | | |
| Data9 high 8 bits | | Data 9 indicates the number of particles with diameter beyond 1.0 μ m in 0.1 L of air. |
| Data9 low 8 bits | | |

| | | |
|----------------------------|-------|--|
| Data10 high 8 bits | | Data10 indicates the number of particles with diameter beyond 2.5 μ m in 0.1 L of air. |
| Data10 low 8 bits | | |
| Data11 high 8 bits | | Data11 indicates the number of particles with diameter beyond 5.0 μ m in 0.1 L of air. |
| Data11 low 8 bits | | |
| Data12 high 8 bits | | Data12 indicates the number of particles with diameter beyond 10 μ m in 0.1 L of air. |
| Data12 low 8 bits | | |
| Data13 high 8 bits | | Data13 Reserved |
| Data13 low 8 bits | | |
| Data and check high 8 bits | | Check code=Start character1+ Start character2+.....+data 13 Low 8 bits |
| Data and check low 8 bits | | |

Figure 9: PMS5003 data output (source: Yong, 2016)

A ventilator inside the sensor sucks the air into the sensor case. The internal microcontroller of the PMS5003 measures the time that the laser reflector is emitted and the time that it is not emitted. The time that the reflector is emitted by light results in the value for high bits; the time that the reflector is not emitted by light results in the value for the low bits. This data needs to be processed to bytes and then integers by the microcontroller of the user of the PMS5003, i.e. the LoPy.

Data calibration PMS5003

Before the PMS5003 is deployed, the sensor is tested in three environments: inside a 'clean' room (windows wide open), inside a 'dirty' room (windows closed and smoke), and outside. The sensor is tested with the script in Appendix Y. See the table 2 and figure 10 for the results for the inside ('clean') room.

| Inside (Science Center, windows wide open) 'clean' air, 09-06-2017 afternoon | | | | | | | | | | | | |
|--|-----|-------|------|-----------|-----------|----------|--------|--------|--------|---------|---------|---------|
| | PM1 | PM2.5 | PM10 | PM1.0 env | PM2.5 env | PM10 env | Data 7 | Data 8 | Data 9 | Data 10 | Data 11 | Data 12 |
| #1 | 3 | 5 | 5 | 3 | 5 | 5 | 930 | 256 | 46 | 2 | 0 | 0 |
| #2 | 4 | 10 | 10 | 4 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | |
|------|-----|-----|-----|-----|-----|-----|-------|-------|------|-----|-----|-----|
| #3 | 3 | 6 | 7 | 3 | 6 | 7 | 855 | 250 | 44 | 3 | 0 | 0 |
| #4 | 3 | 6 | 7 | 3 | 6 | 7 | 858 | 252 | 40 | 4 | 2 | 1 |
| #5 | 2 | 5 | 6 | 2 | 5 | 6 | 813 | 223 | 42 | 4 | 1 | 1 |
| #6 | 3 | 5 | 5 | 3 | 5 | 5 | 924 | 249 | 41 | 1 | 1 | 0 |
| #7 | 4 | 8 | 10 | 4 | 8 | 10 | 897 | 248 | 59 | 11 | 3 | 2 |
| #8 | 3 | 5 | 6 | 3 | 5 | 6 | 945 | 246 | 33 | 7 | 2 | 1 |
| #9 | 3 | 7 | 8 | 3 | 7 | 8 | 987 | 274 | 55 | 8 | 0 | 0 |
| #10 | 3 | 6 | 6 | 3 | 6 | 6 | 918 | 251 | 41 | 2 | 0 | 0 |
| Avg: | 3,1 | 6,3 | 7,0 | 3,1 | 6,3 | 7,0 | 812,7 | 224,9 | 40,1 | 4,2 | 0,9 | 0,5 |

Table 2: PMS5003 data calibration, inside, 'clean' air

And the graph, where only PM1, PM2.5 and PM10 values are plotted:

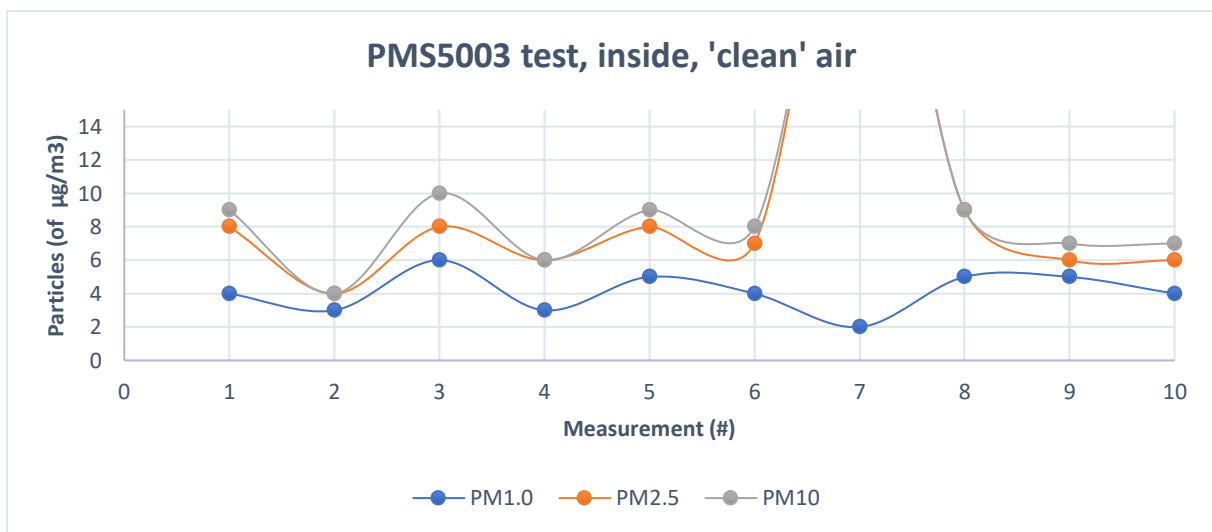


Figure 10: Plot of the PMS5003 test, inside, 'clean' air

The values can be considered relatively stable. Only the seventh measurement can be considered as an outlier and should therefore be removed from the dataset. During the seventh measurement were the values for Data 7 to Data 12 all NULL. So when this happens, the PM values may be unreliable. The software should raise an exception when this happens.

However, the software in the deployed platforms does not take account when this exception happens, so when the received PM values are remarkably high they can be considered as outliers due to this problem. A recommendation is therefore to implement a script that only sends the messages over LoRa when the measurements for values for Data 7 to Data 12 are not NULL. When these values are NULL, the PMS5003 measurements should be redone.

For comparison, the Particulate Matter in the air in a room with 'dirty' air is measured. The air can be considered 'dirty' since for the testing the windows are closed the whole day and there was smoke in the room. After a couple of measurement the windows were opened, an event that can be recognized in the graph. The values can be found in table 3 and are plotted in figure 11.

| Inside (apartment, windows whole day closed, smoke) 'dirty' air, 08-06-2017 late evening | | | | | | | | | |
|--|--------------------------------|--------------------------------|-------------------------------|--------------------------------|-------|--------------------------------|-------|-------------------------------|-------|
| | PM1.0 $\mu\text{g}/\text{m}^3$ | PM2.5 $\mu\text{g}/\text{m}^3$ | PM10 $\mu\text{g}/\text{m}^3$ | PM1.0 $\mu\text{g}/\text{m}^3$ | env | PM2.5 $\mu\text{g}/\text{m}^3$ | env | PM10 $\mu\text{g}/\text{m}^3$ | env |
| #1 | 174 | 281 | 317 | | 115 | | 187 | | 211 |
| #2 | 164 | 254 | 358 | | 108 | | 168 | | 238 |
| #3 | 175 | 284 | 335 | | 116 | | 189 | | 223 |
| #4 | 179 | 286 | 332 | | 119 | | 190 | | 220 |
| #5 | 170 | 274 | 314 | | 113 | | 182 | | 209 |
| #6 | 177 | 280 | 318 | | 118 | | 186 | | 211 |
| #7 | 144 | 256 | 300 | | 94 | | 170 | | 198 |
| #8 | 172 | 272 | 313 | | 114 | | 181 | | 208 |
| #9 | 66 | 110 | 130 | | 54 | | 88 | | 114 |
| #10 | 84 | 126 | 134 | | 62 | | 94 | | 116 |
| Average: | 150,5 | 242,3 | 285,1 | | 101,3 | | 163,5 | | 194,8 |

Table 3: Plot of the PMS5003 test, inside, 'dirty' air

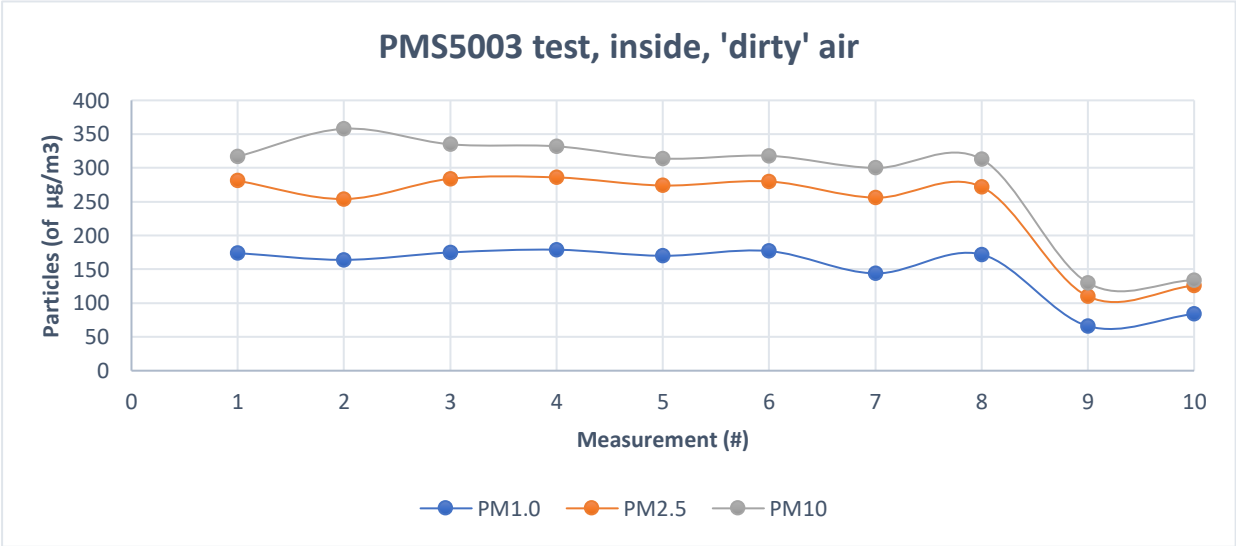


Figure 11: Plot PMS5003 test, inside, 'dirty' air

From the PM values of the second testing round can be concluded that the measurement location was indeed a 'dirty' location. In fact, the European PM2.5 norm (25 µg/m³) on this place is violated nearly ten times (European Parliament and of the Council, 2008).

The third test round with the PMS5003 sensor was in an outside environment. For this test the stabilization of the sensor was investigated. The first measurement was outside, and during the second measurement the sensor was exposed to (cigarette) smoke. The third measurement gave an even higher PM value, although there was no new smoke added, so it needs time before the smoke left the sensor. In the next three rounds was the sensor stabilized again. Then, the PMS5003 sensor was kept (very) close to the exhaust of a car (gasoline). After this measurement, the car removed and the PM values dropped relatively fast (see table 4 and figure 12).

| Outside, two events, 09-06-2017, afternoon | | | | | | |
|--|-------|-------|------|-----------|-----------|----------|
| | PM1.0 | PM2.5 | PM10 | PM1.0 env | PM2.5 env | PM10 env |
| #1 | 3 | 6 | 7 | 3 | 6 | 7 |
| #2 (smoke) | 129 | 366 | 498 | 85 | 243 | 331 |
| #3 | 171 | 643 | 1187 | 113 | 428 | 791 |
| #4 | 33 | 67 | 84 | 27 | 48 | 66 |
| #5 | 13 | 23 | 32 | 13 | 23 | 32 |
| #6 | 13 | 20 | 25 | 13 | 20 | 25 |
| #7 (exhaust car) | 514 | 734 | 794 | 342 | 488 | 528 |
| #8 (exhaust car) | 17 | 26 | 28 | 17 | 26 | 28 |
| #9 | 11 | 20 | 23 | 11 | 20 | 23 |
| #10 | 13 | 20 | 25 | 13 | 20 | 25 |

Table 4: Plot of the PMS5003 test, outside

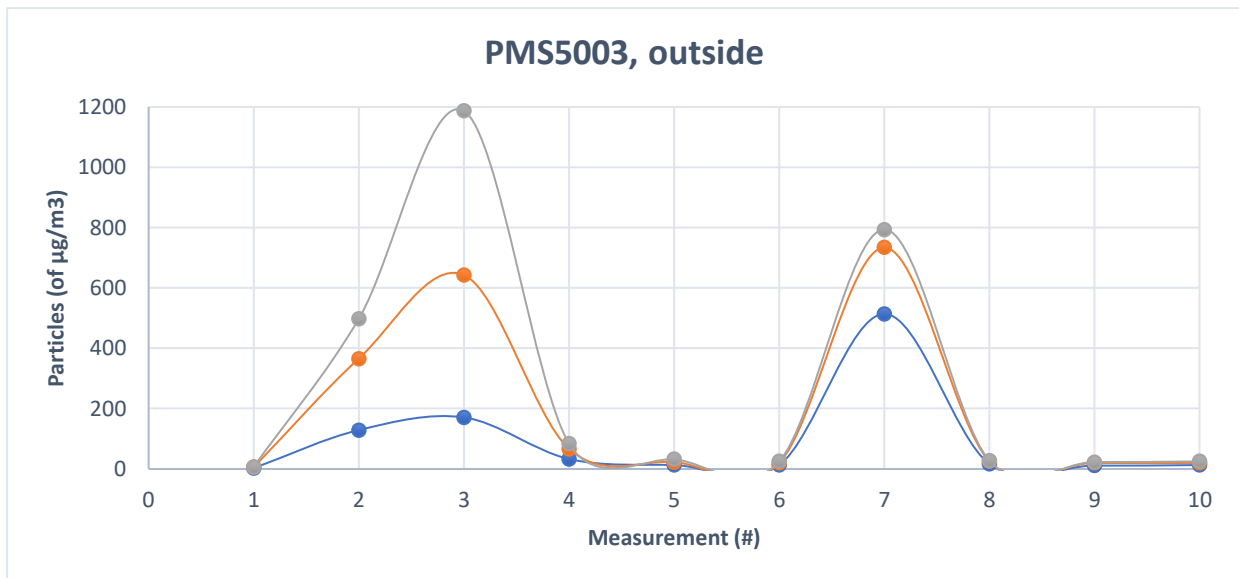


Figure 12: Plot PMS5003 test, outside

Conclusion PMS5003

To conclude, the PMS5003 sensor can detect cigarette smoke and fine dust particles from car exhausts. The sensor is able to react to the values relatively fast. Therefore, the sensor will be implemented on the sensor platform.

3.1.3 Air quality sensor 2 (Sharp Dust Sensor)

Next to the PMS5003 there is another dust sensor that can be used: the Sharp GP2Y1010AU0F, also known as the 'Sharp Dust Sensor' (figure 13). It is a dust sensor that uses the optical sensing technique. With this technique, particles in the air are emitted with an infrared emitting diode and a phototransistor. The first former emits the light and shines upon the particles, the latter receives the reflected light (Sharp Corporation, 2006). With a 100msec interval are the dust particles measured. When compared to the PMS5003, this device does not feature a ventilator that suction the air into the device, so this device must be placed outside of the sensor platform box. According to the product specification, the objects that this device is able to detect are house dust and cigarette smoke (Sharp Corporation, 2006). It should be tested if it is able to detect vehicle emission.



Figure 13: Sharp Dust Sensor

The output of the Sharp Dust Sensor is in digital format and shows a voltage proportional to micrograms per cubic meter. To connect this device to the LoPy there are a resistor of 150Ω and a capacitor of 220μF needed. The connection to the LoPy will be according to the scheme in figure 14.

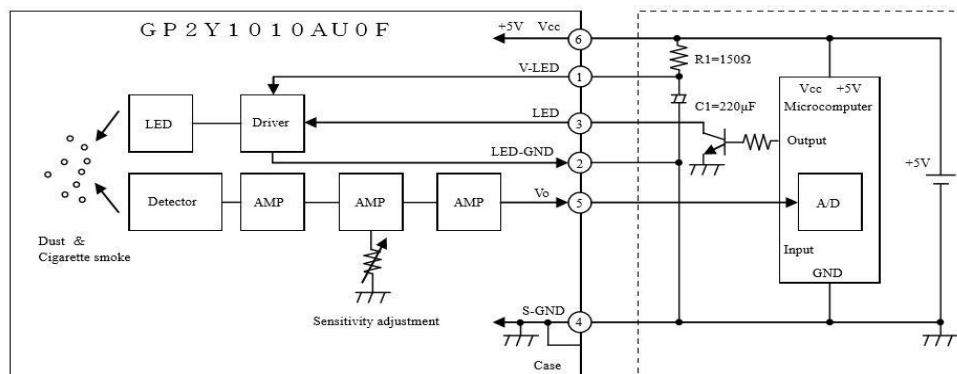


Figure 14: Sharp Dust Sensor connection to LoPy (source: Sharp Corporation, 2016)

After the reflected light from the particles, and thus the amount of dust, is calculated, the device gives an analog output to the LoPy. This analog signal value needs to be transformed in a digital value on the LoPy. The output value has a small delay: 0.28ms after the LED in the device is turned on. So the producer of the device recommends to configure the microcontroller so that it reads the output 0.28ms after the LED emission too (Sharp Corporation, 2016). Next to that, the producer specifies that the time before the device is ready to detect dust is 1 second. The particles in the air are shown as a Voltage value (ΔV):

$$\Delta V = V_o - V_{oc}$$

Where V_o is the monitor value, and V_{oc} is output voltage at no dust. The V_{oc} value is stored in the memory of the LoPy, and if this value is relatively low for a long time, then the V_{oc} value is replaced by this new V_{oc} .

Conclusion Sharp Dust Sensor

The LoPy features an analog to digital conversion Pin, which can be used to read analog voltages and then convert them to a digital value. The MicroPython library for doing this is the 'machine.adc' module (George & Sokolovsky, 2017). Although there is a Sharp Dust Sensor available, the sensor is not tested nor used, since there were several bad reviews about the device and due to time limitations. Instead, the air quality sensor 3 is used as an alternative to the PMS5003 sensor.

3.1.4 Air quality sensor 3 (Shinyei PPD42NS)

A third alternative is the PPD42NS dust sensor, manufactured by Shinyei. The light-scattering principle is also used. This sensor works with an electrical 100mΩ resistor placed inside that warms up and 'attracts' the air with particles in it and then these particles are counted. The resistor is driven at 5 Volts, with a current of 50 milliamperes, so it uses 0.25 Watt (Allen, 2013). Within the sensor the particles are directed through a place that is emitted with an infrared LED light beam. Light that is scattered by the particles is picked up by a photodiode. The 5 Volt power supply must be very stable, since fluctuations in the power supply voltage will directly translate into fluctuations in light output from the LED, which will then affect the measurements. Therefore, the MT3608 step-up converter is used and set at 5 Volts.

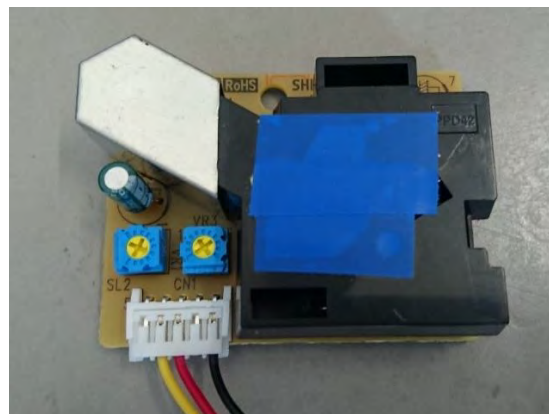


Figure 15: PPD42NS air dust sensor

The two blue pots on the front of the sensor (figure 15) are factory-calibrated. The part that is covered with the blue stickers is where the particle flow will be directed through.

The output voltage of the sensor is normally high (above 4 Volts), since in a normal situation the air flow will be 'clean'. When a particle passes the sensing component, the voltage 'goes low'. The amount of time that the voltage is low is tracked, and expressed in the so-called 'Lo Pulse Occupancy (LPO)'. From the LPO the PM concentration can be determined (Tan, 2014; Fonolossa, 2016). So in short, LPO is proportional to particle count concentration. With this technique, the PPD42NS is able to detect particles which have a minimum size of 1 micrometer (μm). Data from this sensor is sent over one digital

connection. The modulated output unit of the sensor is pcs/0.01cf (cf = cubic foot) which relates to pieces/283ml. The number that is shown is then the number of particles with a minimum size of 1 μ m in the air (Seeed, 2017).

A voltage divider is needed to make sure that the LoPy device does not receive too much volts from the PPD42NS. When the 5 Volts from the sensor are directed to a data input Pin on the LoPy this Pin can be damaged. The LD1117AV33 voltage divider is used, which brings the voltage down to 3.3 Volts (ST Microelectronics, 2013). The voltage divider has the pin connections as can be seen in figure 16.

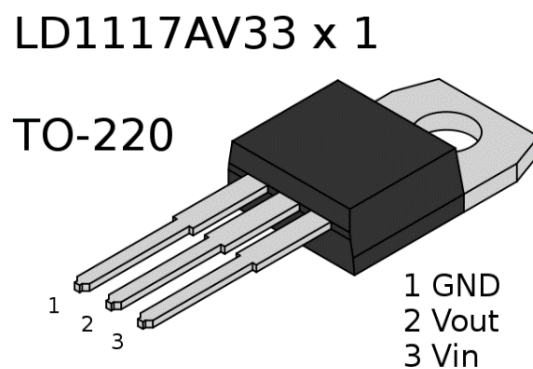


Figure 16: LDD1117AV33 voltage divider (source: ST Microelectronics, 2013)

Since the PPD42NS has no built-in option to switch the device off, the sensor will be activated the whole time by default when it is connected to a battery. The 100m Ω resistor will always use 50 milliamperes, which drains the battery. Moreover, since for the application of this project the measurement will be done with a 15 minutes interval, there is no need for the sensor to be kept on. Therefore, a transistor is added which turns the device on when needed and off when measurements are done. The 2N3004 NPN transistor in combination with a 1000 Ω resistor is used, and the transistor is connected to output Pin 3 on the LoPy device to switch the PPD42NS on and off.

Data calibration PPD42NS

Findings from the testing by Tan (2014) are that the sensor is able to detect cigarette smoke, that the sensor goes gradually down when the conditions change, so the device can be regarded as reliable. However, Tan (2014) did not test the sensor outside.

With the PPD42NS dust sensor three tests are done. The sensor is tested inside under stable circumstances (with a closed box), to make sure that the data fluctuates not too much. Then is it tested inside with an open box, so inside air flows can affect the measurements, and then finally the sensor is tested outside in a green environment. See table 5 and figure 17 for the results.

| Inside and outside, 13-06-2017, afternoon. Unit: pcs/0.01cf (cf = cubic foot) | | | |
|---|------------------|--------------------|---------------------------|
| | Inside, open box | Inside, closed box | Outside, park environment |
| #1 | 1,7409 | 3,1764 | 2,1306 |
| #2 | 5,0959 | 0,9364 | 3,4783 |
| #3 | 1,4387 | 25,0958 | 2,1025 |
| #4 | 3,8665 | 2,3745 | 3,7201 |
| #5 | 1,5640 | 9,2586 | 6,2140 |
| #6 | 1,8296 | 3,5771 | 4,6176 |
| #7 | 1,2923 | 6,3345 | 2,7359 |
| #8 | 2,2838 | 3,4172 | 7,8645 |
| #9 | 1,3807 | 6,4268 | 2,9212 |
| #10 | 1,4175 | 10,6047 | 3,5185 |
| #11 | 1,5928 | 11,7328 | 3,5880 |
| #12 | 1,3183 | 28,1408 | 7,0021 |
| #13 | 1,8700 | 16,3216 | 10,0172 |
| #14 | 1,1048 | 9,6844 | 3,1045 |
| #15 | 2,1596 | 3,9915 | 6,3850 |
| #16 | 0,8694 | 29,0757 | 2,4866 |
| #17 | 1,3720 | 8,1929 | 2,1293 |
| #18 | 0,9622 | 4,4224 | 1,3649 |
| #19 | 1,2199 | 8,3300 | 6,5895 |
| #20 | 0,8111 | 3,1753 | 2,3503 |
| #21 | 0,6547 | 1,8521 | 9,0970 |
| #22 | 1,3523 | 3,0262 | 1,8006 |
| #23 | 1,3098 | 4,8078 | 5,6989 |
| #24 | 1,9368 | 3,2657 | 4,2180 |
| #25 | 2,6682 | 2,3903 | 5,5103 |
| #26 | 2,2324 | 1,5360 | 7,9867 |
| #27 | 1,3372 | 4,0016 | 3,1580 |
| #28 | 2,1509 | 1,8826 | 3,0841 |
| #29 | 2,1975 | 5,0593 | 4,9903 |
| #30 | 1,6344 | 4,2861 | 5,7389 |
| average | 1,7555 | 7,5459 | 4,5201 |

Table 5: PPD42 tests

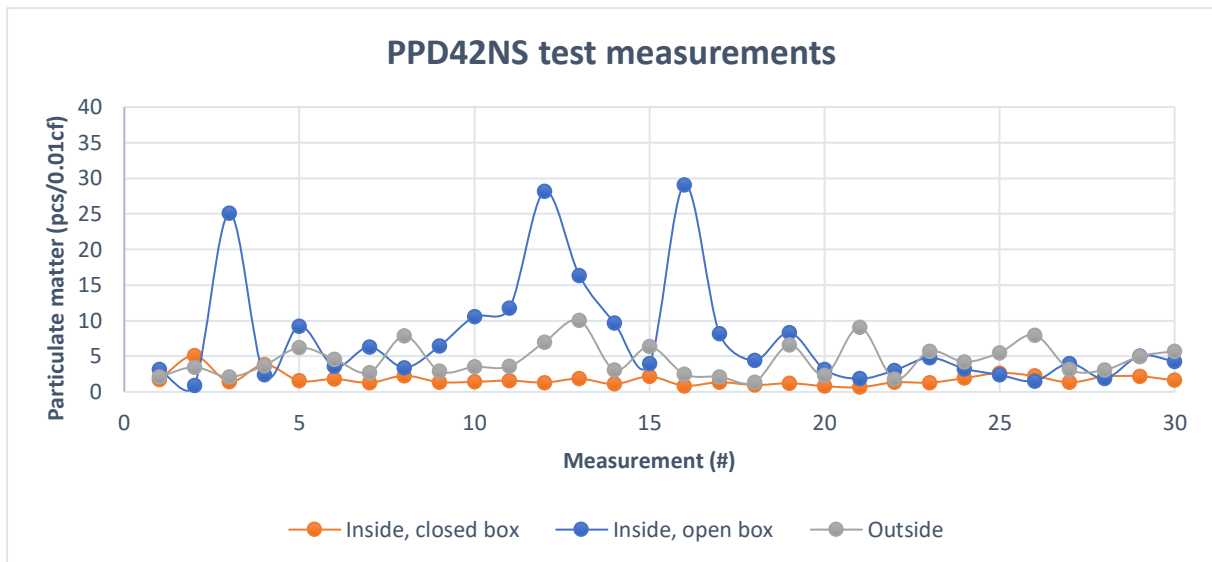


Figure 17: PPD42NS tests plot

The first peak in the second measurement round (inside, open box) is because the device was moved slightly, the second and third peaks occurred because there were relatively hard blows of wind into the room.

Conclusion PPD42NS

From the test results of the PPD42NS can be concluded that the values will fluctuate relatively much due to weather conditions (wind). Therefore, precautions should be taken with the design of the box: the sensor should be protected against strong wind currents and the whole sensor box should not be placed in a relatively windy area. When these precautions are taken, the results can be regarded as reliable. Therefore, the PPD42NS will be implemented on the sensor platform.

In the previous sub-paragraphs are three possible air dust sensors described, which all can be used to measure the air quality which is needed for the first use case that this report focuses on. In the project only device 1 and 3 will be implemented, i.e. the PMS5003 and the PPD42NS sensors.

3.1.5 Sound sensor: MAX9814

For the second use case that is described information about the sound levels in the streets needs to be achieved. This will be done with the MAX9814 sound sensor, see figure 18.



Figure 18: MAX9814 sound sensor

According to the manufacturer, the MAX9814 is a “low-cost, high-quality microphone amplifier that has automatic gain control (AGC) and a low-noise microphone bias” (Maxim Integrated, 2016). The output amount of decibels of the sensor can be configured by the user, by connecting the preamplifier and output amplifier.

Further, this sensor will be connected to the LoPy according to the OneWire protocol, since the sound level that is measured is transferred to the development board over a single channel (mono). A digital input Pin of the LoPy is used to connect this sensor to. Between the output of the sensor and the digital input pin of the LoPy is a capacitor of 1 micro Farad (1 μ F) needed (Adafruit, 2015). For a schematic overview of the connections on the sensor see figure 19. The MicroPython programming language provides a OneWire library, which will be used to read the sensor data (Pycom, 2016).

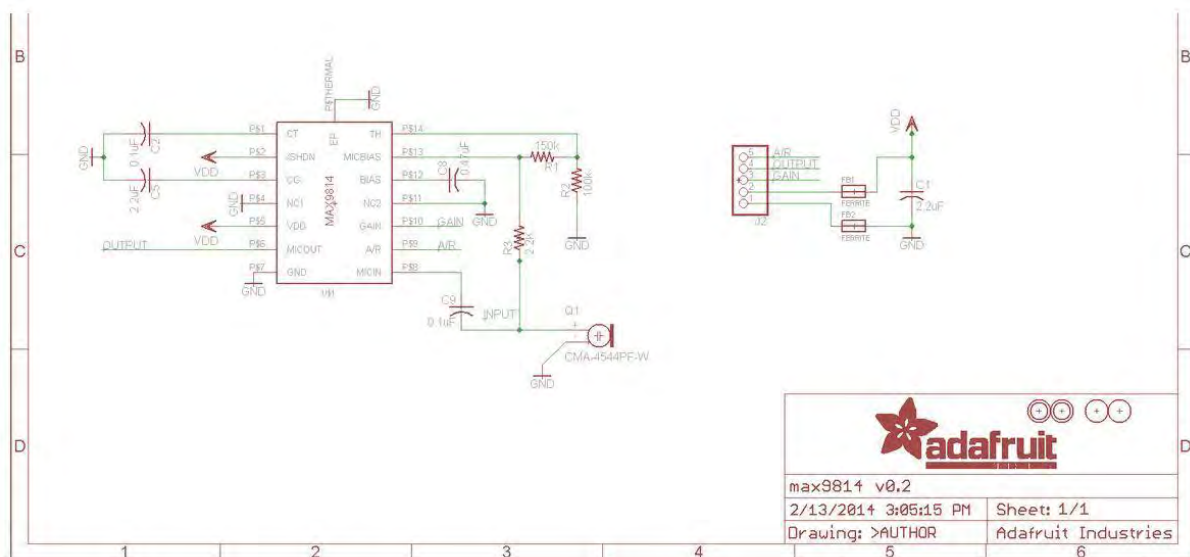


Figure 19: Schematic overview of the MAX9814 sound sensor (source: Adafruit, 2016)

MAX9814 data calibration

Although the MAX9814 gives an indication of the intensity of noise at a certain moment, it is not really

a device meant for detecting sound levels. The purpose of this sensor is to pick up a sound and amplify it. It therefore has a built-in Automatic Gain Control chip, that 'locks onto' the input signal, and makes sure that gradual changes in the amplitude of the input signal has minimal effect on the output signal. The amplification of the sound is not a purpose of this project, however the picked up sound intensity can result in a useful indication of sound levels at a location.

In the graph of figure 20 are the results of a test with this sensor shown. The test is executed at six different types of locations, as can be seen in the legend of the graph. For every 10 seconds are the average and median values plotted in the figure. The thick lines represent the outside measurements, near a road where there was a relatively continuous flow of cars. There is a difference visible in the data, so, in theory, it should be possible to see these patterns in the city too.

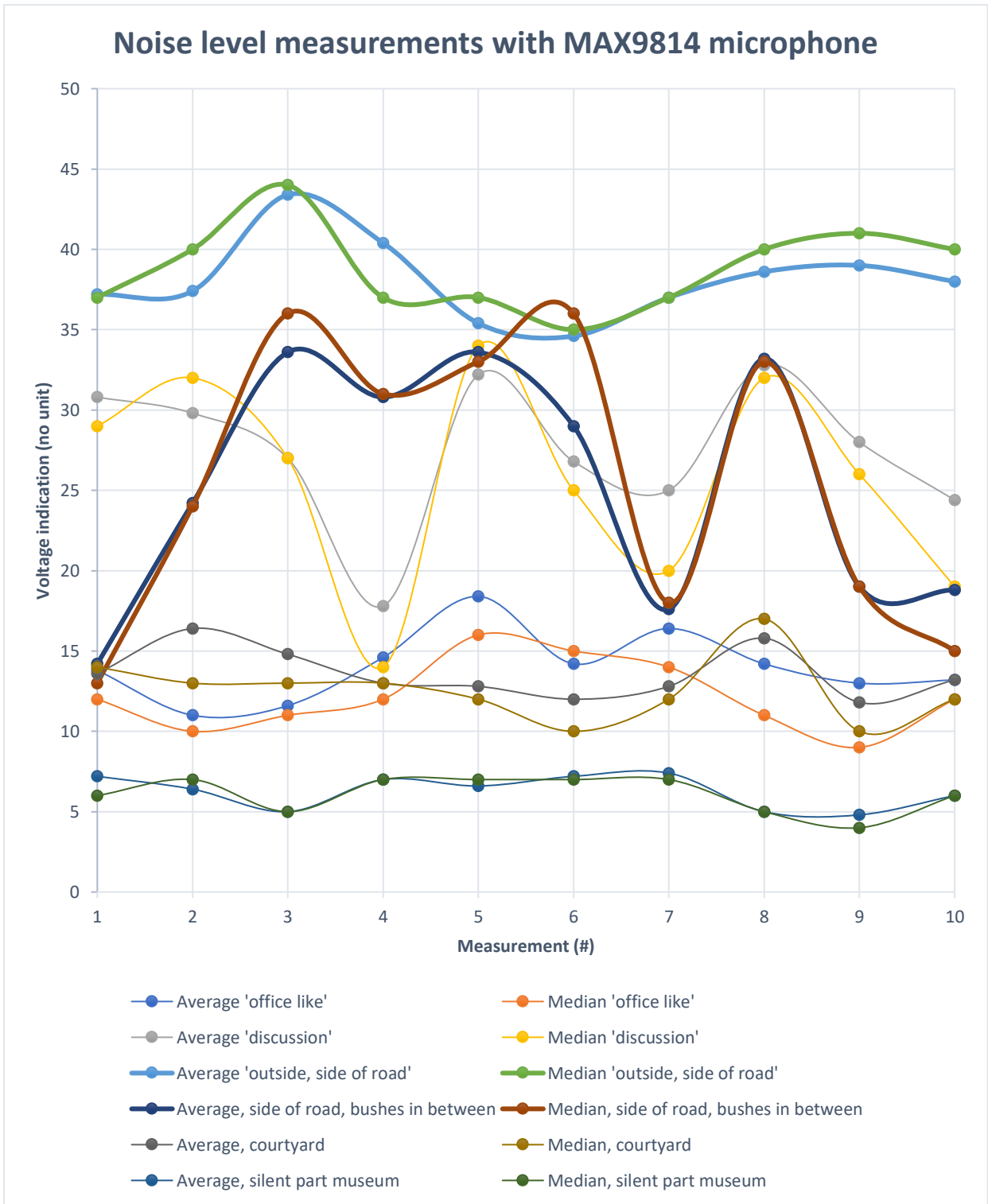


Figure 20: Graph of test measurements with the MAX9814

The test code can be found in Appendix T. Although it is possible to see patterns, the deployed sensor platforms have a script that sends only one value for every 15 minutes (not averaged). This is because the LoRaWAN communication protocol that will be used does not allow to send big amounts of data.

Moreover, output of the MAX9814 will be a voltage level that relates to the loudness of a phenomenon. The way the sound sensor is used in this project seemed not to be useful to be able to answer the research question. A recommendation is to change the software such that more measurements are executed, whereby maximum and minimum values are stored as well as an average sound level. This can give more insight in the 'loudness' on a location than only one single measurement.

3.1.6 Temperature & humidity

The third use case that is described focuses on the temperature distribution on the measurement locations. Temperature is calculated with the basic, low-cost digital temperature and humidity sensor AM2302 (figure 21). The temperature is measured with a thermistor that measures the surrounding air, and gives a calibrated digital output signal on the data pin, so there are not analog input pins needed (Liu, year unknown).



Figure 21: AM2302 temperature and humidity sensor

An 8 bit microcontroller does the conversion from the measured temperature value to the digital output of the device. An exhaustive description of the single-bus communication protocol that is scripted on the microprocessor of the AM2302 can be found in Liu (year unknown). For now it is important that the temperature error as specified by the producer is shown in figure 22, so with the Dutch weather conditions an error of around 0.25 degrees Celsius can be expected.

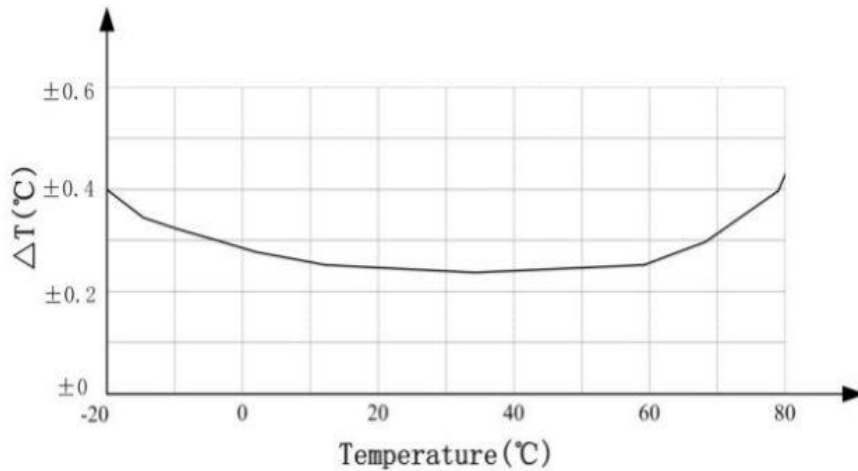


Figure 22: Maximum temperature error (source: Liu, year unknown)

Further, the device is connected to the LoPy according to the OneWire protocol (see figure 23) MicroPython features libraries that can read data from digital temperature and humidity sensors such as the AM2302. The MicroPython library is in the 'dht' module (George & Sokolovsky, 2017).

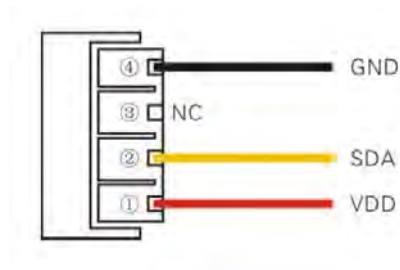


Figure 23: AM2302 pin assignment (source: Liu, year unknown)

AM2302 tests

On the afternoon of 27-05-2017 the temperature and humidity sensor is tested on five different locations: inside in a living room, in a hobby room, and in a cellar; and outside on a terrace in the shadow of a tree and on a location exposed to the sun. With the *temp_hum.py* script (see Appendix U) the values for the locations measured. In this script are for each 5 seconds the temperature and humidity values given. When moving to another location, the sensor is first placed on that location for a minimum of 5 minutes to adapt to the environment. Then the measurement are conducted during 5 minutes. See table 6 for the results.

| Location | Temperature (in degree C) | Humidity (in %) |
|------------------------------------|---------------------------|-----------------|
| Inside: living room | 23.9 | 55.0 |
| Inside: cellar | 14.2 | 87.1 |
| Inside: hobby room | 17.9 | 87.4 |
| Outside: terrace, shadow of a tree | 29.4 | 40.2 |
| Outside: terrace, full sun | 36.2 | 27.8 |

Table 6: AM2302 tests, five locations

In addition to these measurements two devices are located close to each other and compared to an Auriol temperature and humidity measuring device. For a period of 15 minutes are the values calculated, and the results are shown in table 7.

| Location | AM2302 #1 | | AM2302 #2 | | Comparison device Auriol | |
|--------------------|-------------|----------|-------------|----------|--------------------------|----------|
| | Temperature | Humidity | Temperature | Humidity | Temperature | Humidity |
| Average 15 minutes | 22.3 | 57.6 | 21.7 | 60.3 | 22.0 | 55.0 |
| Average 15 minutes | 22.3 | 58.2 | 21.7 | 61.6 | 22.0 | 55.0 |

Table 7: AM2302 calibration

So the accuracy of the device is about 0.3 degrees Celsius for temperature and 5% for humidity (when the Auriol benchmark device is considered as being reliable).

The testing of the AM2302 concludes the description of the separate sensors that are used. The next section elaborates on the development board.

3.1.7 Development board

The development board or microcontroller is the processing unit to which all sensors are connected. This 'platform' also includes software that allows to read the data from the sensors, save it temporary, preprocesses it by doing calculations, and sends it to a gateway (from where it is send to a database). In this part the microcontroller that will going to be used is described: the Pycom LoPy.

Pycom LoPy

The PyCom LoPy is a microcontroller that features data communication over three networks: WiFi, Bluetooth Low Energy, and LoRa. It has an on-board flash memory, features relatively high processing speed, has a decent amount of RAM, when compared to its competitors (see table 8). The specifications are retrieved from the specsheets of the products (Pycom, 2017; Arduino Corporation, 2017; RaspberryPi, 2017).

Since the LoPy features a high amount of communication options, an onboard flash memory, and relatively high processing power, and combines this with low energy usage, the LoPy is chosen. Moreover, the LoPy features the LoRa communication technique, of which producers proclaim is being a good solution for Internet of Things applications (LoRa Alliance, 2017).

| | PyCom LoPy 1.0 | Arduino Uno | Raspberry Pi 3 |
|---------------------------|----------------------------------|--|---|
| Chipset | Espressif ESP32 | ATmega328P | ARMv8 |
| CPU | 160 MHz | 16 MHz | 1.2 GHz |
| RAM | 512 KB | 2 KB | 1 GB |
| Flash memory | 32 MB | 32 KB | No flash, only micro-sd |
| Communication | WiFi, Bluetooth Low Energy, LoRa | WiFi, (LoRa possible with Marvin LoRa development board) | WiFi, Bluetooth Low Energy, Bluetooth 4.1 |
| Scripting language | MicroPython | C# | Python |

Table 8: Comparison of Pycom LoPy, Arduino Uno, and Raspberry Pi microcontrollers

3.1.8 LoRa and LoRaWAN

The LoPy features the LoRa modulation technology, a technique owned by Semtech that enables Internet of Things by offering a cheap, long range and low power national network (KPN, 2017). Gateway devices that receive the LoRa signal are very sensitive because the Chirp Spread Spectrum (CSS) communication technique is used. CSS allows the sender to send message with low power, also when noise is high and distance is big. See figure 24 for an schematic overview of how the LoRa message is modulated over the signal. For a more detailed description of the LoRa technique and test results, we refer to LoRa Alliance (2017), Petäjälä et al (2015) and Aref & Sikora (2014).

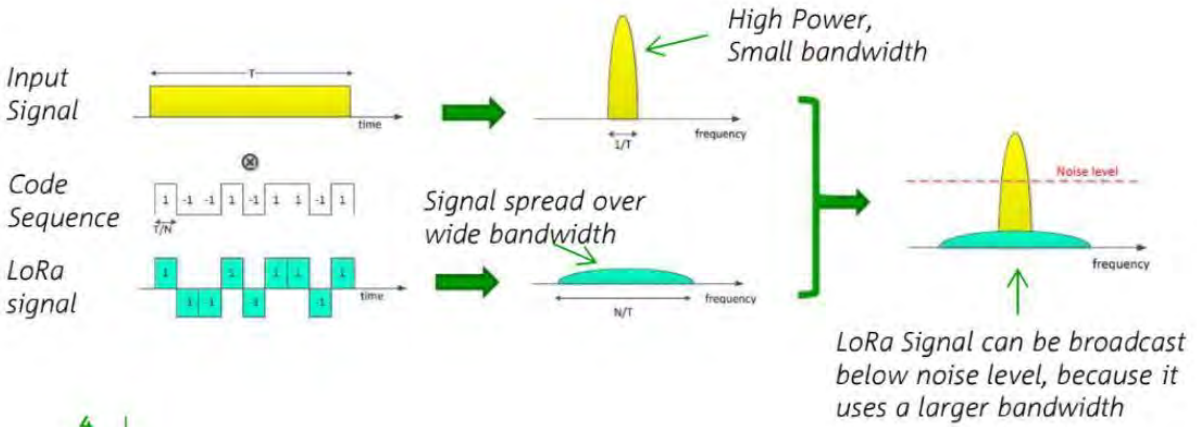


Figure 24: LoRa modulation (source: KPN, 2017)

The LoRaWAN protocol sends small messages efficiently by using LoRa modulation: the size of such a message has a maximum of 54 bytes. As opposed to the TCP/IP protocol, the LoRaWAN protocol allows radio modules to sleep most of the time, because the connection does not have to remain open to be synced in the LoRaWAN network. LoRaWAN enables devices to send uplink (from device to application service) and downlink (from application server to device) messages.

The parameter that shows the quality of the received signal is the Spreading Factor (SF), where SF7 is close to a gateway ('good') and SF12 is far away from a gateway ('bad'). Placing a LoRa device outside on a façade is beneficial for a higher SF, because there is higher chance of line of sight between the device and a LoRaWAN gateway. LoRa uses more spreading, and repeating messages, when there is a weak signal or a big amount of noise. The ideal SF is chosen by LoRaWAN itself, by using the Adaptive Data Rate (ADR) technique. With ADR, first a message is send on SF12, where after the network measures the signal strength, and then replies to the device which SF it should use from then onwards. See figure 25 for SF and their characteristics.

| Spreading Factor | Bitrate | Range in free field | Time on Air (ms)* |
|------------------|----------|---------------------|-------------------|
| SF7 | 3470 bps | 2 km | 50 ms |
| SF8 | 3125 bps | 4 km | 100 ms |
| SF9 | 1760 bps | 6 km | 200 ms |
| SF10 | 980 bps | 8 km | 400 ms |
| SF11 | 440 bps | 10 km | 800 ms |
| SF12 | 290 bps | 14 km | 1600 ms |

Values are indicative and approximated, because they depend on environmental conditions and hardware conditions
 *for typical payload around 10 bytes

Figure 25: LoRaWAN SF characteristics (source: KPN, 2017)

LoRa uses the license-free frequency spectrum in the 868 MHz band, which is an ISM band. This band is divided in sub-bands ranging from 863 to 870 MHz. The relevant frequency bands are then the F, G, H, and K1 bands, as defined by the Dutch Government (Agentschap Telecom, 2014). For LoRaWAN, the G band is mostly used, ranging from 865.0 to 868.6 MHz. Users have to obey restrictions determined and enforced by the government for these frequencies, such as a maximum power output of 14 dBm and a limited allowed time-on-air (duty-cycle) of maximum 1% of an hour (KPN, 2017).

The messages (keys, payloads and addresses) that are broadcasted over LoRaWAN are encoded in hexadecimal strings. To secure these messages, LoRa uses the Advanced Encryption Standard (AES). A key of 128 bits, which is the same as 32 hexadecimal characters, is used for this encryption. Next to that, network information is encrypted with a Network Session Key (NwkSKey), which needs to be shared between the device and the KPN LoRa core network. Furthermore, the payload information is encrypted

with an Application Session Key (AppSKey), that must be shared between the device and the customer application server. KPN delivers a Device Address (DevAddr) to the developer, which is a unique identifier within a specific LoRa network. An End Device Unique Identifier (DevEUI) is defined globally. It is unique for all LoRaWAN networks, and is administrated by the IEEE (KPN, 2017).

LoRa in the Netherlands: KPN versus The Things Network

Two initiatives that provide communication over a LoRa network are available in the Netherlands. One initiative is by KPN, which covers most of the Netherlands with a network of more than 1000 LoRa Gateways (KPN, 2017). Another initiative is by The Things Network, a community-driven initiative that is building a global Internet of Things data network (The Things Network, 2016). In table 9 advantages and disadvantages are described (source: authors). Thereafter, the LoRa networks of KPN and The Things Network are compared.

| | KPN | The Things Network |
|----------------------|--|--|
| Advantages | <ul style="list-style-type: none"> + nationwide coverage (coverage in study area) + good web-debugger, where message payloads can be retrieved + big community and online forum | <ul style="list-style-type: none"> + free to use (unlimited number of devices can be added) + possibility to add own LoRa gateway + two types of message decryptions (OTAA and ABP) + big community and online forum |
| Disadvantages | <ul style="list-style-type: none"> - limited number of devices can be added for free (maximum 10) - problem with ABP message decryption, after sending over LoRa | <ul style="list-style-type: none"> - no coverage in study area (can be overcome by adding LoRa gateways) - no web-debugger to see payloads |

Table 9: KPN and The Things Network LoRa comparison

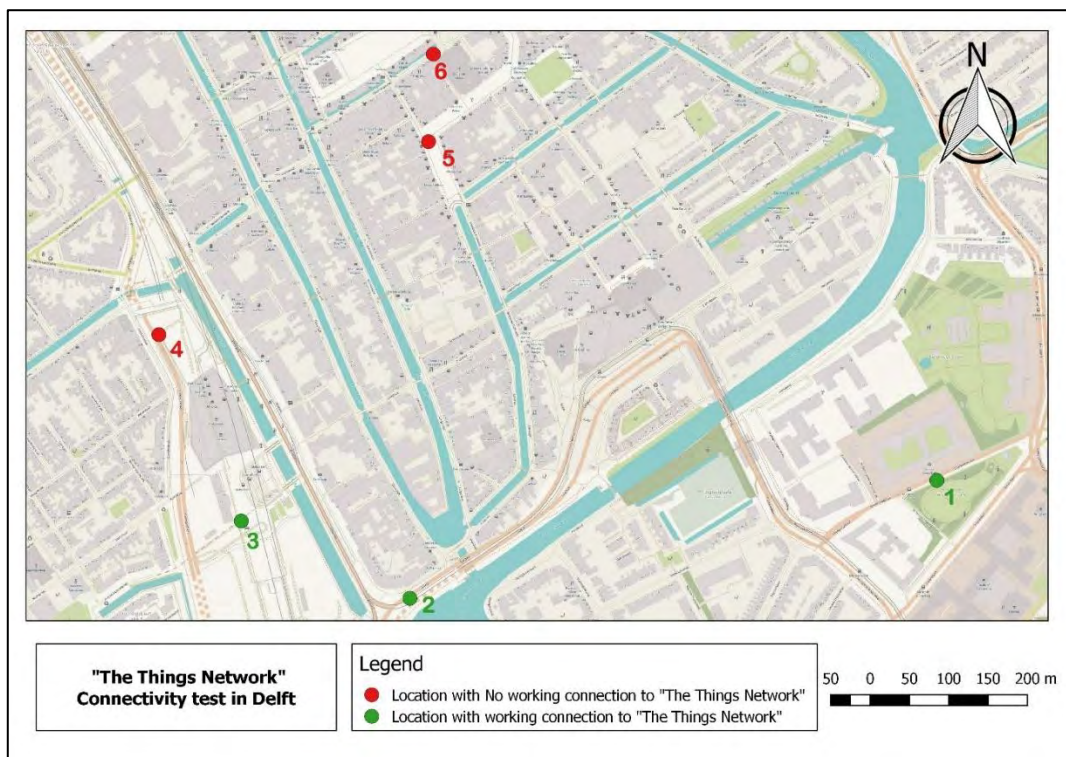
The Things Network research

The coverage of The Things Network (TTN) is tested in the study area. The closest active TTN gateway is located at the TU Delft campus, at the Faculty of Electrical Engineering, Mathematics and Computer Science, as shown in map 1.



Map 1: The Things Network coverage in Delft (source: The Things Network, 2017 (edited))

There is no coverage in the study area because only locations that have direct view of the above-mentioned gateway have coverage (map 2). This might be due to the low energy of this gateway.

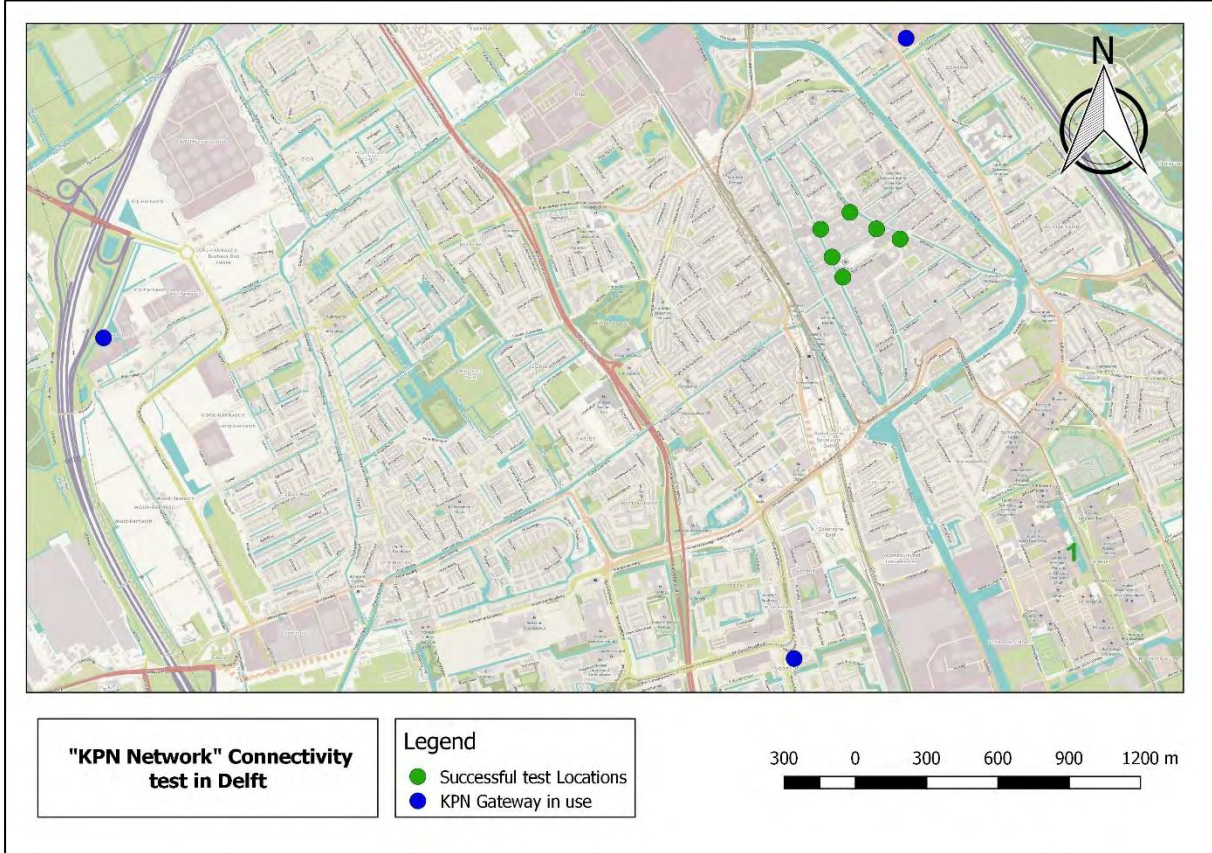


Map 2: The Things Network coverage test in the study area

The possibility of deploying Nano-gateways to extend the things network to cover the study area was researched. These gateways are connected to the internet via DSL and uses LoPy to receive the LoRa signal. However, these gateways were not used eventually because a large amount of them needed to create an obstructed view with the sensors. Moreover, multiple collaborative citizens are needed to place these gateways at their places and connect it to their internet. The possibility to use different hardware was also not possible because of high expenses that are not possible within the context of this project. Moreover, KPN provides a free account that was sufficient for the project and deploying these Nano-gateways will cost unnecessary time and money.

KPN LoRa network research

KPN coverage is first tested in the study area, there were no issues found regarding the coverage. LoRa signal was sent from the study area and received by KPN gateways, the study locations and the receiving KPN gateways are shown in map 3.



Map 3: KPN LoRa connectivity test in the study area

When the message of LoRa via KPN is received it will be encrypted. Node-RED is used to decrypt the message, which will be discussed in paragraph 3.4. The full decryption code is found in the Appendix S.

3.1.9 Software on the LoPy

LoPy software environment

The LoPy board works with Micropython, which is a Python 3.5 implementation optimized to work on microcontrollers. There is a flash folder on the LoPy where it is possible to add and remove files. The available RAM capacity is 512 KB for Python code while the storage capacity is 32 MB. When booting, two files are executed automatically:

1. Boot.py: This is the first file to be executed automatically (as can be seen in figure 26), it contains necessary information for the LoPy to boot.
2. Main.py: Is executed automatically after Boot.py. Here, the main.py file is programmed to access the data collected from different sensors, process the data and send it through LoRa to the gateway.

| | | |
|---------|-------|-------------|
| cert | | File folder |
| lib | | File folder |
| sys | | File folder |
| boot.py | 185 | PY File |
| main.py | 1,287 | PY File |

Figure 26: the default folder structure in the flash folder of LoPy

File structure of the LoPy sensors platform

The goal of the software architecture of the board is to efficiently acquire the data from different sensors and send this data using to a LoRa Gateway using the LoRa component in the LoPy. To achieve this goal the folder structure of the flash folder looks as following (figure 27).

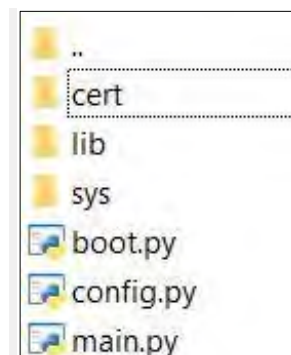


Figure 27: the folder structure in the flash folder of LoPy for each LoRa platform

These MicroPython files will have the following functions:

- boot.py: left almost unchanged. Only one line is added to it to change the password of the LoPy to prevent other people from accessing and making changes to the software;

- main.py: It will import the data from all the sensors and send the data using LoRa ABP (Activation by Personalization) protocol;
- config.py: will add the following an ABP authentication parameters:
 - o DEV_ADDR = Device address
 - o NWKS_KEY = network shared key
 - o APPS_KEY = applications shared key
- the folder *lib* contains the library related to the temperature and humidity sensor.

Software architecture of main.py

The main.py program is where most of the software functions will happen. Two versions of this file are used depending on the AQ sensor used: PMS5003 or PPD42NS (see appendix J for the whole program).

When the main.py program is executed the following workflow as shown in figure 28 will run.

In the workflow are first all the important libraries for the program to function properly loaded. Most of these libraries are already defined on the LoPy. Other libraries are added to the flash folder which are:

- *config*: which contains the unique authentication parameters, and
- *DTH* function from the *dth* library which will read the temperature and humidity.

Next the watchdog timer (*WDT*) is set to 32 minutes to restart the program if it failed to completely run within this time frame. Then the send with LoRa parameters are defined. Afterwards, the different read from sensors functions are defined. These read from sensors functions contains which Pin provides the sensor data and other sensor specific functionalities. Next, a *While True* loop will run indefinitely if the hardware is running properly. In this loop, the readings from sensors functions will run and will give output values.

During the while loop, the output values are formatted to be efficient for sending by LoRa. This is done by converting the values into hexadecimals. And then splitting the values into bytes, so that every value will not take more than one byte. For example, in temperature and humidity the numbers on the left of the comma are sent separately from the numbers on the right of the comma. These values are then combined back in Node-Red.

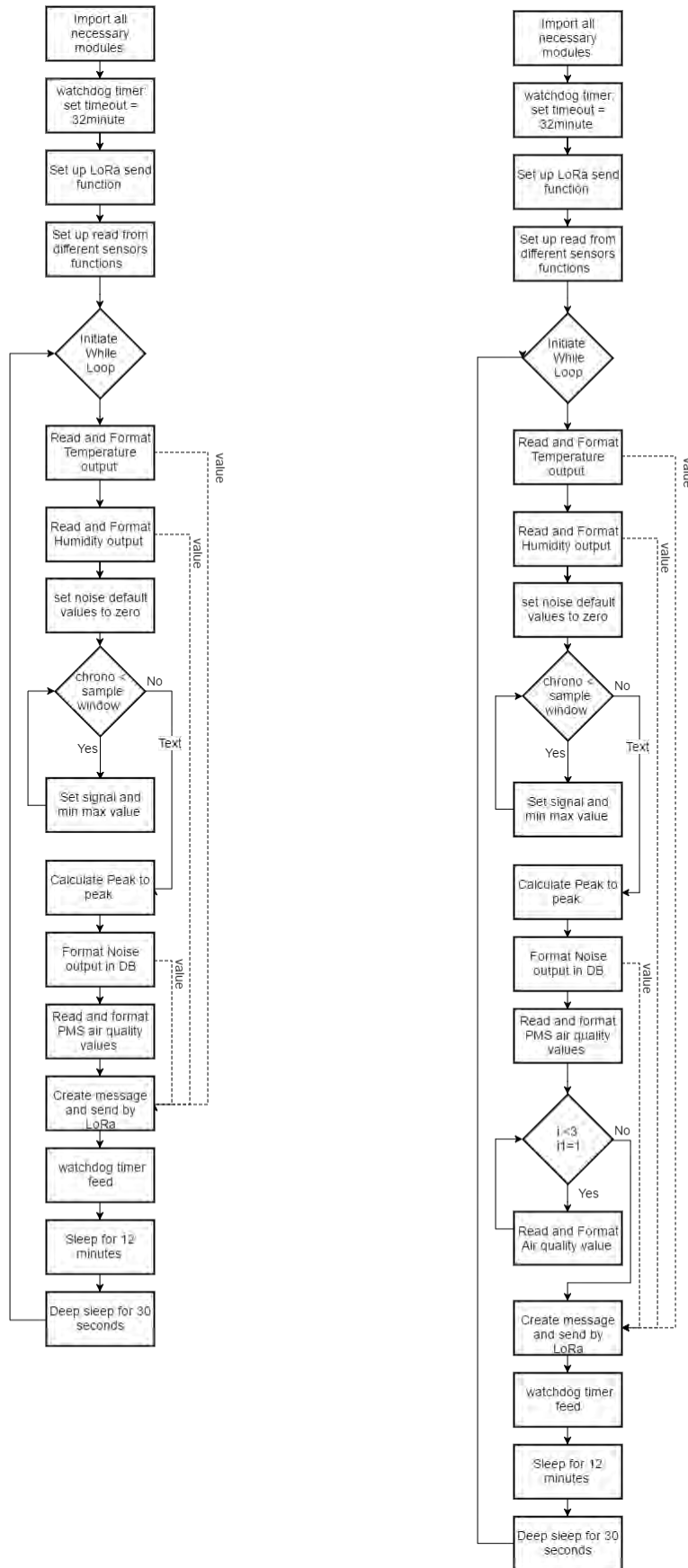


Figure 28: Workflow of main.py (PMS5003 on the left and PPD42NS on the right)

After formatting the values, they are combined in one list of 13 bytes/value and this list is sent as a LoRa message. Next a WDT is fed which will inform that the program is running properly. Next the program will be placed to sleep for 12 minutes and then for a deep sleep for 30 seconds, where after the program is started again.

Program design

The above-mentioned design was made in a way that will fulfil the aim of the sensor platform in a simple and efficient way. It also takes into consideration the limitation of LoRa and LoRa regulation policies as mentioned described in paragraph 3.1.3, which limits the data each end-device can send.

To decrease the size of the LoRa message, there has been decided not to include a unique id of the device and not to add a timestamp to the sensed data. Because when a LoRa message is sent and received by the database, the message will automatically be time stamped by the gateway and a unique id of the device will be also included. This information will later be included in the database. Therefore, the time of receiving the message will be considered the same as the time of collecting the data.

Temperature and humidity readings do not need to be as frequent as the noise and AQ readings. However, according to the LoRa policy mentioned above it is not possible to send the high frequency and big size messages that noise and AQ ideally requires. Moreover, creating different readings frequencies has the following disadvantages:

- it will require different time stamping;
- it will create an unnecessary complexity in the database, and
- more energy consumption.

Therefore, a high frequency has been chosen for all the sensors. And for every cycle, a reading is taken from all sensors and sent to the gateway.

3.1.10 Charging of the sensor platform

Types of batteries, the charging module, and the solar panel charging solution are described in this paragraph.

Batteries and charging

The sensor platforms get energy from batteries with a capacity of 5000 mAh. In the first four sensor platforms only one battery is placed, while in two new sensor platforms there are two batteries placed

(in parallel).

The batteries are the GEB 905085 lithium-polymer batteries. These batteries have a capacity of 5000mAh and deliver around 3.7 Volts. When the batteries are totally charged, they deliver 4.17 Volts. The batteries are rechargeable. These batteries are able to work between temperatures of -20°C until 60°C (General Electronics Battery Co, 2017). The batteries are shipped with a female connection cable, only for the connection of this battery to the sensor perfboard is a lipo battery 'male' JST connector needed with a pitch distance of 2 millimeter.

For the charging of the batteries are the Adafruit Lipo chargers used that is able to charge with a current of 500mA. So charging a completely empty battery will take around 10 hours.

Solar panel system

To overcome the challenge of battery drainage the possibility of using solar panels system as a sustainable charging method is explored. This system is then connected to the sensors platform which provide sufficient electrical current for a single sensors platform. The components of the system are shown in table 10.

| Quantity | Name | Model | Specifications |
|----------|--|------------|---|
| 8 | solar panels | 70X35 | 2V , 35MA |
| 1 | lithium-polymer battery | GEB 905085 | 5000mAh |
| 1 | Integrated System Load Sharing and Battery Charge Management | MCP73871 | Simultaneously Power the System and Charge the Li-Ion Battery |

Table 10: components for the solar panels system

To optimize the solar panels system, every two solar panels are connected as series. The resulting four doubled solar panels are then connected on parallel to form one combined solar panel that has a default specifications of 4v and 140mA. That is sufficient to power the sensors platform with high energy consuming sensors such as the PPD42NS.

The resulting solar panel is then connected to the Load Sharing and Battery Charge Management system which will charge the battery and provide a current of regular 3.7 V to the sensors platform (see figure 29 and figure 30).

The system is then tested on a sensor platform equipped with a PPD42NS which has a high-power consumption. This platform worked for a maximum 16 hours when connected to one 5000mAh battery. The same sensors platform worked for a maximum of 20 hours when connected to the solar panel system that is equipped with the same battery. The increase of

work time is then 20%. However, this test is done while the solar panels system is inside in the shadow. A higher result will be achieved if the system was under direct sun light.

Due to time limitations, further optimization was not performed on the solar panel system. Based on the results so far, it is possible to create a self-sufficient sensors platform by using solar panels system that has more solar panels and batteries capacity.

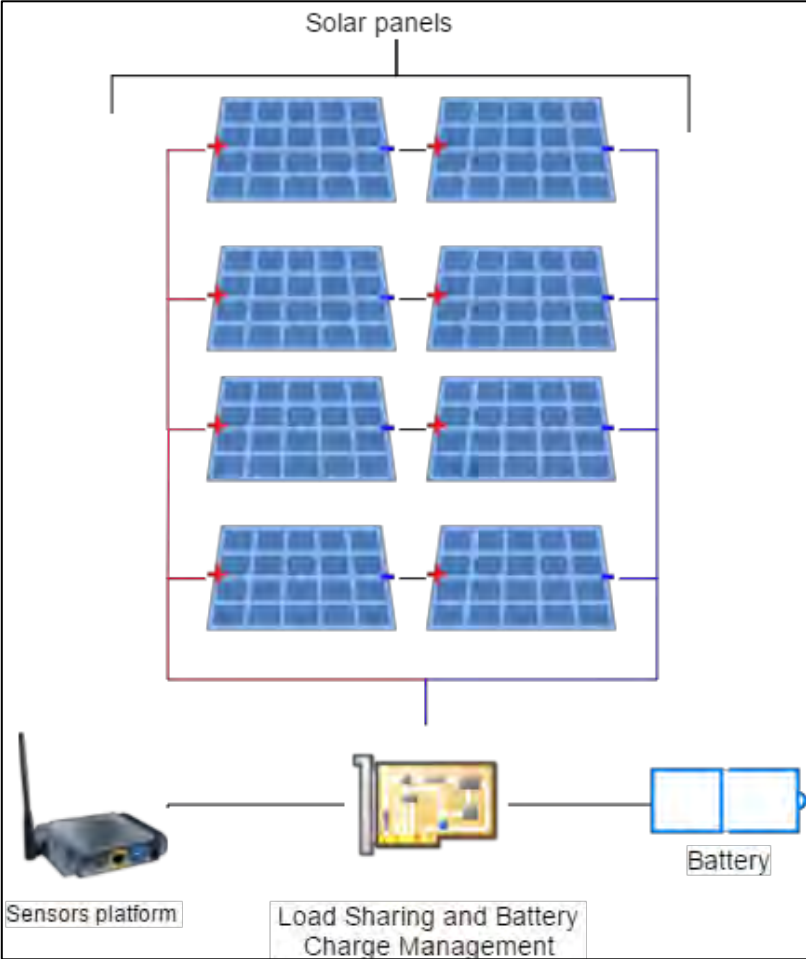


Figure 29: Solar panel system design

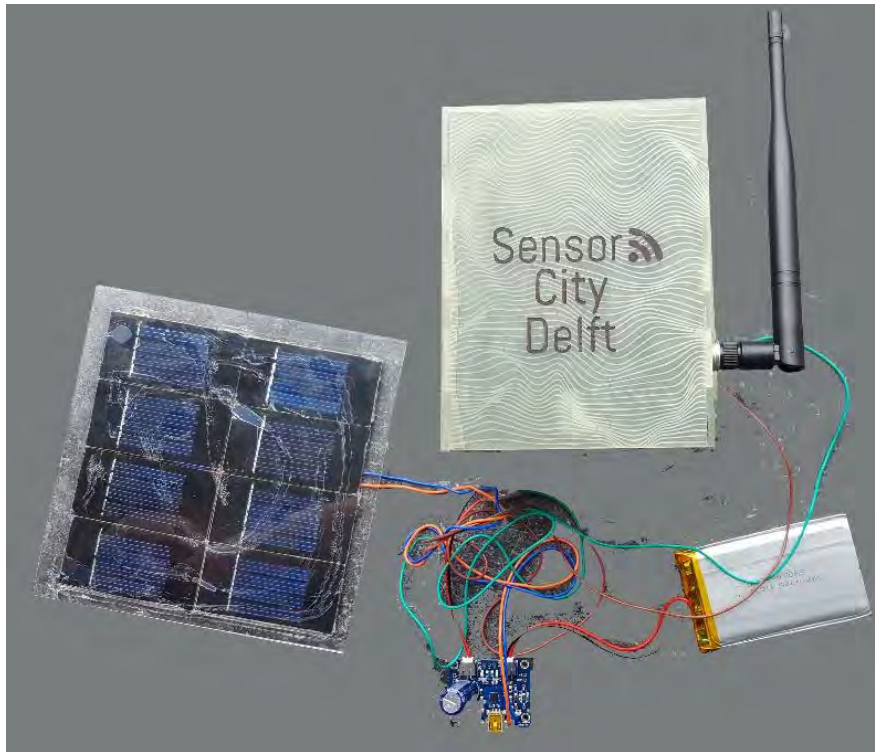


Figure 30: Charging the LoPy with solar panels

3.1.11 Creating the prototype

In the previous paragraph all the separate parts of the sensor platform are described. How they are integrated on the sensor platform is described in this paragraph.

A perforated board ('perfboard', see figure 31) forms the basis of the sensor platform. They are chosen instead of Printed Circuit Boards or Breadboards, because the number of available pins to connect sensor cables to it is high. Connections between sensor pins and relevant holes on the perfboard can be created easily, just by soldering and/or scratching away connections. Furthermore, the sensors can be removed relatively easy from the perfboard since they are connected via 'headers'. That is advantageous for maintenance and re-use of sensors. Finally, perfboards are relatively cheap.

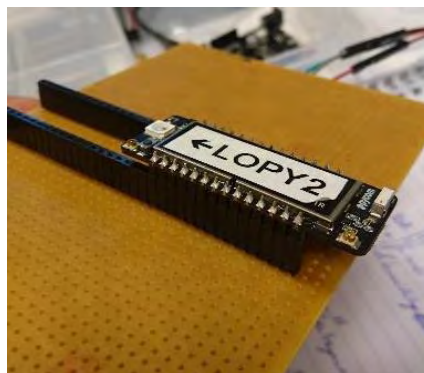


Figure 31: perfboard, with headers and a LoPy

Then, the LoPy, charging circuits, connection cables, and batteries are placed onto and close to the perforated board. The sensors are not placed on the perforated board, since they need to measure the ‘outside’ air and are thus placed outside of the heavier protected compartment of the box.

3.1.12 Producing more sensor platforms

The final part of the hardware section now describes how more sensor platforms are produced. The schematic design according to which the other sensor platforms are created is according to figure 32. In this figure is shown which of the Pins on the LoPy are connected to Pins on sensors.

| |
|---|
| GND BAR – MAX9814 (-), AM2302 (-), PMS5003 (-), PPD42NS (-), BATTERY GND BAR |
| 3.3V BAR – MAX9814 (+), AM2302 (+) |

| | | | |
|-----------------|-------------|-------------|----------------------|
| | RST | 5V | Volt Booster - OUT + |
| | P0 | GND | GND BAR |
| | P1 | 3.3V | 3.3V BAR |
| | P2 | P23 | |
| AM2302 (DATA) | P3 | P22 | |
| PMS5003 (SET) | P4 | P21 | |
| | CLK | P20 | |
| | MOSI | P19 | |
| | MISO | P18 | |
| PMS5003 (RESET) | P8 | P17 | |
| PMS5003 (TXD) | P9 | P16 | MAX9814 (DATA) |
| PMS5003 (RXD) | P10 | P15 | |
| PPD42NS (DATA) | P11 | P14 | |
| | P12 | P13 | |

| | | | |
|------------------------------------|--------------|--------------|-----------------|
| LoPy (5V), PMS5003 (+), PPD42NS(+) | OUT + | OUT - | GND BAR |
| Battery 3.7V BAR | IN + | IN - | Battery GND Bar |

| |
|--|
| BATTERY GND BAR – Volt Booster (IN-) |
| BATTERY 3.7V BAR – Volt Booster (IN+) |

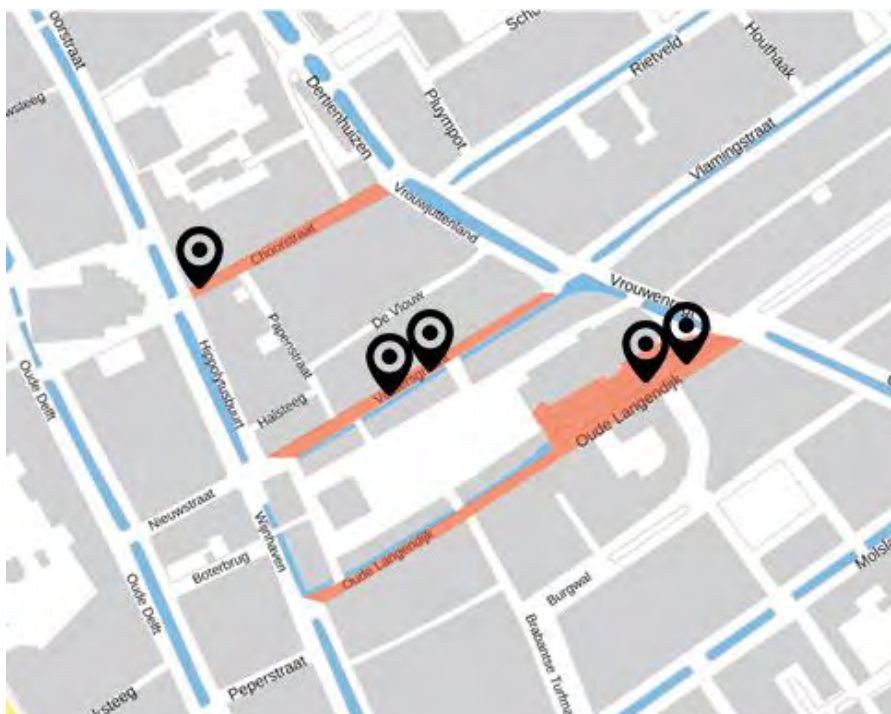
Figure 32: Schematic overview of the sensor platform

3.1.13 Conclusion hardware

In this paragraph the different aspects of the hardware of the Sensor City Delft project are described. The sensors themselves, the LoPy development board, communication over LoRa, software on the LoPy, charging of the sensor platforms, and integrating all the components on the perfboard are covered. The next paragraph focuses on the study area: where in the city center the sensor platforms are going to be placed and how exactly.

3.2 Area of study: location and sensor placement

The sensor platforms are deployed in the city center of Delft. The streets for placing the sensors are the Choorstraat, Voldersgracht and the Oude Langendijk (see map 4). Each street has its own profile. All streets have shops, restaurants and cafes. The Choorstraat is the smallest of the tree streets. It is a shopping street where mainly pedestrians and cyclist come. The Voldersgracht is meant for local traffic; pedestrians and cyclist are also allowed. One side of the street has buildings while and on the other side the canal is located. The Oude Langendijk is the busiest of the three streets as it allows pedestrians, cyclists, cars, and public transport.



Map 4: The study area in red, scale 1:2381

3.2.1 Requirements for placing the sensor boxes

For the sensors placement in the streets several requirements are determined. Firstly, the sensors boxes have to be mounted at a height at which people cannot reach it. This prevents people from taking or

breaking the sensor. Secondly, the location of the sensor boxes should be visible to pedestrians walking by, in order to enable interactivity. Thirdly, the location of the sensor should make it possible to make repairs when a part of the sensor box is broken.

Several options are investigated to which the sensor boxes can be attached. For attaching the sensors to the facades research is conducted into suitable facades. In this research was looked at attributes sticking out of the facade and overhangs at which the sensor boxes can be placed, this research can be read in Appendix K.

Another option for sensor placement are the lanterns and trees in the streets. The specific trees and lanterns that are considered for hanging the sensor boxes to are shown in appendix L. For attaching the sensor boxes tie rips are used. For this option it was necessary to get in touch with the municipality to ask for permission to place the sensor boxes. The project group contacted the Centre Manager of Delft (Stichting Centrum Management Delft, 2017), who gave permission to hang the sensors on tree and lanterns in the designated study area. In addition to the sensor boxes, also billboards and flyers are distributed though the city center of Delft and in shops in order to raise awareness among citizens of this project.



Figure 33: Deployed Sensor and Billboard at the Oude Langendijk (left) and Voldersgracht (right)

3.2.2 Conclusion

Five sensors have been deployed in the city center of Delft. Two at the Voldersgracht and the Oude Langendijk, and one at the Choorstraat. The sensor platforms were tie ripped to both streets and lanterns, out of reach of passersby. This was possible because the project got permission from the Municipality of Delft to hang them there. See map 4 for the final sensor deployment locations and figure 33 for deployed sensors with billboards.

3.3 Casing

This paragraph presents the design requirements regarding the casing, i.e. the design of the sensor boxes. The casing consists of two different parts: the inner box and the casing. Each of these parts is used for a different purpose and, thus, it has distinct guidelines and design requirements. These design requirements are introduced below, together with a discussion on the decisions taken. Finally, the end product of this project is presented.

3.3.1 Casing design requirements

The casing is used for the protection of both the sensors and the hardware. The hardware and the batteries were placed inside an inner box with the aim to ensure safety of the sensitive parts. The inner box is a part of the casing which, as mentioned above, has distinct requirements that are further described in a separate paragraph. For a successful design of the casing, it is important to ensure good ventilation of the sensors and the inner box.

Furthermore, a possible obstruction of the sensors might affect the measurements taken and lead to unwanted results. On the other hand, a complete exposure to the unexpected weather conditions of the Netherlands could harm the sensors. The material of the casing needs to be durable in order to withstand the various weather conditions of the whole implementation period. In addition to this, the selected material(s) has to be corrosion-resistant and watertight. Based on these requirements, three materials were introduced; acrylic, polypropylene and EVA cast. Other materials with IP65, IP67 & IP68 should also be considered for the particular project. Notably, the material selected had to be consistent with the manufacturing technology.

To stimulate the interaction with the citizens and for aesthetic purposes, the design of the sensor needs to be noticeable and appealing to the citizens. At the same time, the colors of the casing must not affect

the measurements of the temperature sensors. The logo of the project should also be included in the casing design. Finally, essential contact information has to be shown on the casing, to avoid any possibility that the sensor box gets stolen or lost. For a design of the 3D model that obeys the before mentioned requirements, see figure 34.

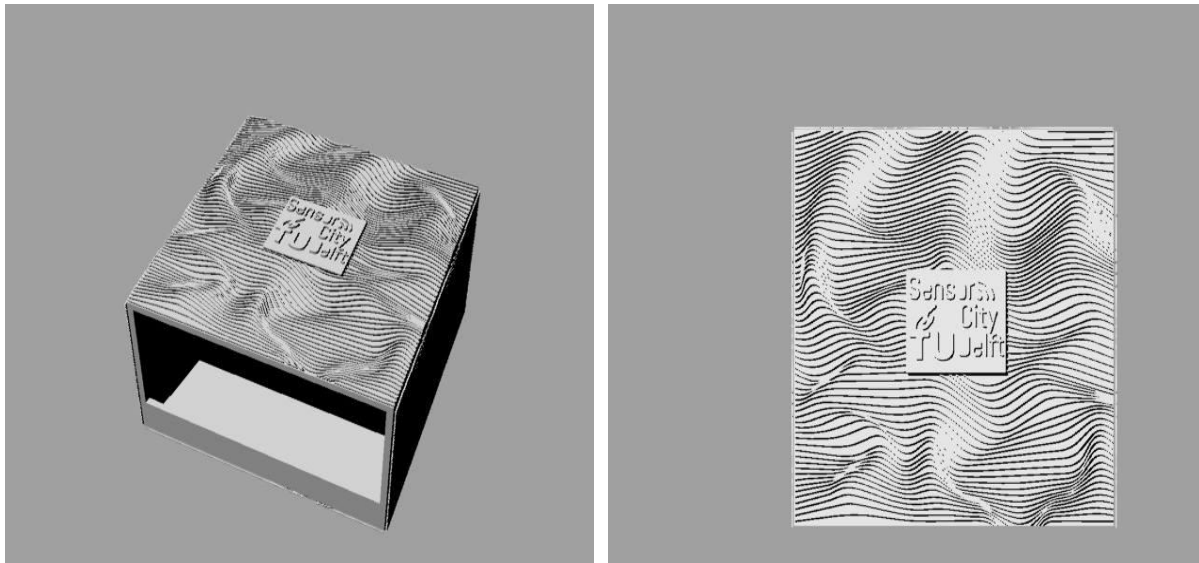


Figure 34: Different views of a casing design concept

Inner box design requirements

Considering the sensitivity of the hardware to the weather conditions, the most important requirement of the casing design is the inner box. The inner box houses all the hardware, i.e. the perforated board, the batteries, and the LoPy. Important for a successful design and implementation is the protection of the hardware from the rain and other weather conditions that could damage or destroy the hardware. This protection could be achieved by placing the hardware in a watertight box. For making a watertight box, two main things had to be taken into consideration, the material(s) of which the box is manufactured and the way the inner box closes.

Furthermore, the inner box should contain enough space for the cables, since a connection between the hardware and the sensors needs to be made. In addition, an opening should be included for the connection between the LoPy and the LoRa antenna. Another factor that should be taken into consideration is the cost of the inner boxes.

Regarding the box size selection, an appropriate size should be no larger than the one required for the placing of the hardware and the cables. More space might result to a costly and time-consuming process and, in the end, to the manufacturing of a bulky casing that could be difficult to be attached to the trees or lanterns. Finally, a critical factor for the inner box selection is the need to open the inner box easily

in order to maintain the perforated board and replace the batteries when needed.

Considering these requirements and the time restrictions, two alternate inner boxes were taken into consideration; a cable box and a lunch box. The inner box (together with the sensors) is placed inside the casing for protection by the rain, wind and/or other weather conditions. Table 11 summarizes all the above described requirements for the casing (including the inner box).

| Inner box | Casing |
|--------------------------------|---|
| Watertight | Watertight |
| 1 compartment (battery & LoPy) | Corrosion-resistant |
| Can be opened and closed | Material (Perspex, polypropylene, EVA cast, IP) |
| Several holes for cables | Attachment holes |
| LoRa antenna hole | Tie-rips |
| Coating | LoRa antenna hole |

Table 11: casing requirements

3.3.2 Manufacturing method and materials

Both the lunch box and the cable box were appropriate choices for an inner box. A lunch box as this was easier to open without harming the batteries or cables.

The laser cutting technology was used as the main manufacturing technology of the casing. However, the use of this technology comes with restrictions regarding the materials that could be used. More specifically, toxic materials such as polypropylene or EVA cast, were not allowed. For that reason, it was decided to use the acrylic perspex. The advantages of using perspex are: a. the price is reasonable, b. it is available in different colors which improves the design and, c. fulfills the pre-defined requirements. Five different colors were used for the casing; lemon, light blue, blue, white mat and white transparent. All 15 casings were manufactured (laser cut and assembled) within 28 hours.

For the ventilation pieces, a 3D model was created in Rhinoceros 5 and, later got 3D printed. Plastic was used as the main material for the ventilation pieces, coming in three colors; grey, white and dark blue. The total amount of time spent for the ventilation 3D printing was 8 hours.

3.3.3 End product

The final casing is presented in figure 35. Take into account the details of the casing, such as ventilation pieces and icons on the right side of the sensor box. Figure 36 shows the cable box solution (left) and the lunch box solution (right) for the inner box. Appendix M shows in detail the 2D view of the casing and the dimensions for the manufacturing.



Figure 35: Final casing in 2 colors (left image). Ventilation piece and icons of measurements on the right side of the sensor box (right image)

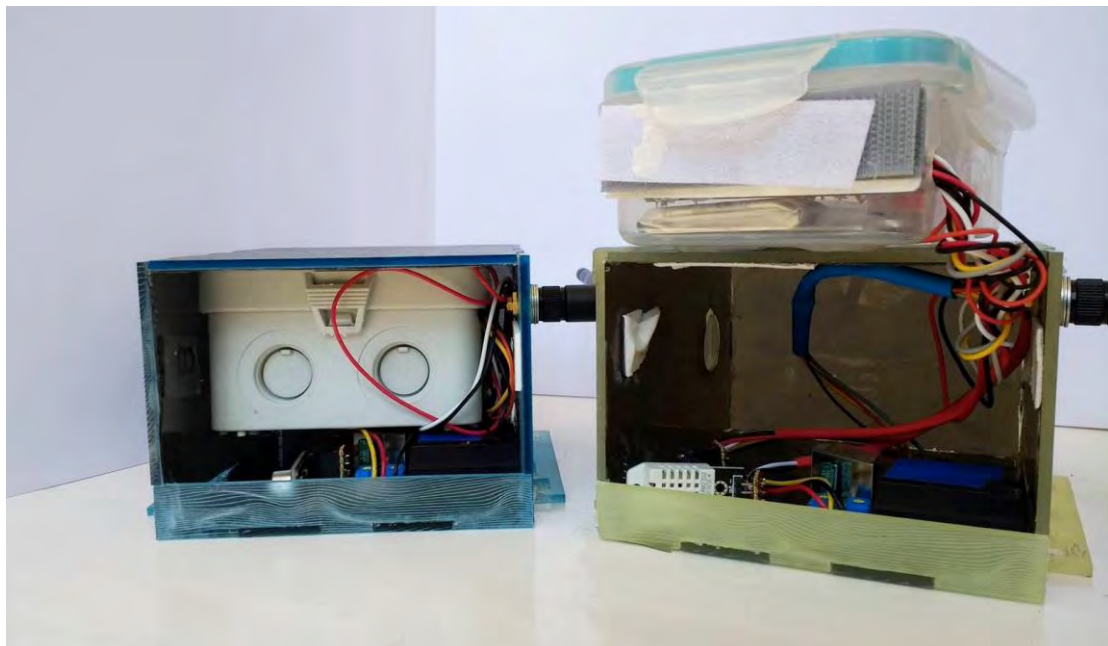


Figure 36: Cable box (left in image) and a lunch box (right in image) as inner boxes, and the sensors used in the final casing

3.3.4 Conclusion casing

The casing of the sensors was an experimental process that was completed within one week. Perspex and plastic were the main materials used for the manufacturing of the casing and the ventilation pieces, respectively. Two different technologies were used for the completeness of the product design: the laser cutting and the 3D printing. Lunch boxes were used as inner boxes after applying the required actions in order to be functional. The parts that were created to support the inner box were also manufactured with laser cutting and glued inside the casing. Additional changes in the final casing were required and were relevant to the way that the various parts were assembled. In practice, the design of the casing achieved its primary target to be appealing to the public, easy recognizable for the project and to constitute part of the city center. After a week of deployment the sensor casings were still in mint condition, so they can be reused.

3.4 Data

The previous paragraphs focused on the components of the Sensor City Delft system that are located in the city. From this paragraph onwards, the components are located on the campus: on the virtual machine of the TU Delft. In this paragraph the data storage in a database is described. The used tools are Node-RED and PostgreSQL and will be introduced in paragraph 3.4.3. The data will also be processed in the Node-RED environment, which is explained in the last subparagraph (3.4.4).

3.4.1 Data processing and storage

Database plan

This database plan is written as a guide to implement the database. It also acts as a functional specification overview of the database. The amount of detail and the complexity of the database design is subsidiary to the complexity of the data. In general a database plan follows the following steps:

1. Gather information.
2. Identify the objects.
3. Model the objects.
4. Identify the types of information for each object.
5. Identify the relationships between objects.

1. Gather information

What is expected from the functionality of the database?

- Store (raw) data of sensors
- Retrieve data for twitter bot

- Retrieve data for website
- Retrieve data for post-analysis

2. Identify the objects

What elements will be part of the database?

- Air quality data
- Noise data
- Temperature data
- Humidity data
- Device addresses
- Timestamps

3. Modelling the objects

After identifying the objects in the system they should be modelled, i.e. represent them visually. This could be used as a reference to and visual presentation of the database.

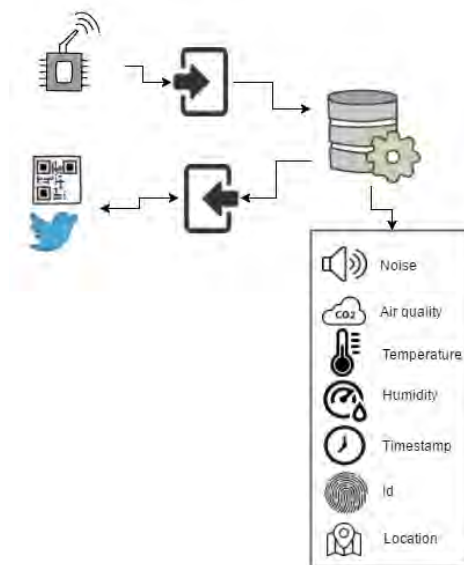


Figure 37: Database system visualized

4. Identifying the types of information of each object

After defining the objects of the database it is important to identify the types of information that must be stored into the database. These will eventually be the columns of the tables. There are four common types of information defined:

1. Table of raw data

- All data collected by the sensors is send in JSON format to the database (in string format). This JSON object contains all information of the send message, including the

information of the measurements, the device address and other information. See Appendix L for an exemplary JSON object which is send to the raw data table.

2. Categorical columns

- Temperature (float)
- Humidity (float)
- Air Quality (integer)
 - Air quality table with single measurement (PPD42NS sensor)
 - Six measurements (PM1.0 etc.) (PMS5003 sensor)
- Noise level (float)

3. Identifier columns

- Every row in every table gets a unique ID based on the time of insertion

4. Relational or referential columns

- Device address (Variable Character Field)

The table in which the data is saved contains 12 columns as can be seen in table 12. When the PPD42NS sensor is implemented on the sensor platform the ‘Air Quality’ column will have a value and the PM values will have values 0. When the PMS5003 sensor is implemented the columns with ‘PM’ in the heading will have a value, and the ‘Air Quality’ will not.

| Device address | Temp. | Humi-dity | Noise | Air Quality | PM1.0 | PM2.5 | PM10 | PM1.0 (atm) | PM2.5 (atm) | PM10 (atm) | Timestamp |
|----------------|-------|-----------|-------|-------------|-------|-------|------|-------------|-------------|------------|---------------------|
| 14203210 | 23.8 | 45.2 | 22 | 237 | 0 | 0 | 0 | 0 | 0 | 0 | 2017-06-09 18:11:09 |
| 1420366D | 23.7 | 63.7 | 10 | 0 | 4 | 14 | 14 | 4 | 14 | 14 | 2017-06-09 21:38:04 |

Table 12: Database table with device addresses, measurements and timestamps

The raw data is saved in a table with one column (table 13), since the JSON object is saved as a string as a single entry.

| Raw Data |
|---|
| <pre> {"DevEUI_uplink":{"Time":"2017-06-08T18:14:36.985+02:00","DevEUI":"0059AC00001807B8","FPort":"2","FCntUp":"1","MType":"2","FCntDn":"1","payload_hex":"641a40d3bdf08b","mic_hex":"473e1648","Lrcid":"0059AC02","LrrRSSI":"-105.000000","LrrSNR":"9.000000","SpFact":"7","SubBand":"G1","Channel":"LC2","DevLrrCnt":"1","Lrrid":"FF010226","Late":"0","LrrLAT":"52.019737","LrrLON":"4.361882","Lrrs":{"Lrr":{"Lrrid":"FF010226","Chain":"0","LrrRSSI":"-105.000000","LrrSNR":"9.000000","LrrESP":"-105.514969"}},"CustomerID":"100006356","CustomerData":{"alr":{"pro":"SMTC/LoRaMote","ver":"1"}}, "ModelCfg":"0","InstantPER":"0.000000","MeanPER":"0.000000","DevAddr":"142037B9"} </pre> |

Table 13: Database table raw data, see Appendix L for more information

5. Identifying the relationship between objects

Every sensor has its own device address; this is a given address by KPN. Based on these device addresses, different sensors can be identified. All the measurements of the sensors are put in two tables, the raw data table and the measurements table. The geographical location of the sensors is not stored in the database, since the sensors remain stationary after deployment and their location is known.

3.4.2 Inserting data in the database

To store the data collected by the sensors in a database it is send through LoRa as is described in chapter 3.1. This data is received in a programming tool called Node-RED. This program runs on the virtual machine that is available on the Faculty of Architecture of the TU Delft (<https://geo1101.bk.tudelft.nl/iot/static/>). Uptime of this domain after the project is not guaranteed. It has 20 GB of storage which will be shared with the Dynamic IoT group of the GSP. Also, it has software installed (like the Node-RED environment and PostgreSQL) and more processing power than usual desktops or laptops. On this virtual machine the Node-RED process will be ran, and the collected data will be stored on in a database using PostgreSQL.

3.4.3 Node-RED & PostgreSQL

What is Node-RED?

Node-RED is a visual programming tool that can wire the Internet of Things. This could be hardware devices, APIs and online services. It is a browser-based editor that contains flows of nodes. These nodes are predefined codes in JavaScript and contain certain, often used, functions. Examples of these functions are http request, a trigger, join or PostgreSQL. Some of those functions are by default part of the Node-RED environment, some functions have to be imported (as libraries) before use is possible.

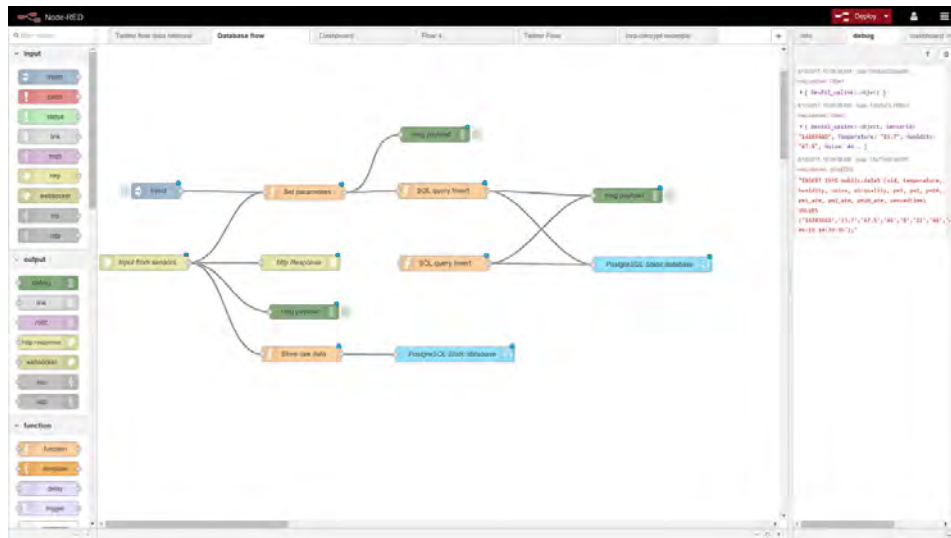


Figure 38: User Interface of Node-RED environment

What is PostgreSQL?

PostgreSQL is used to store the collected data. PostgreSQL is a relational database and open source. It supports (foreign) keys, joins, views, triggers, stored procedures and different datatypes. This database system uses SQL queries to store and retrieve data. The SQL language is standardized according to the ANSI-SQL:2008 standard.

The workflow

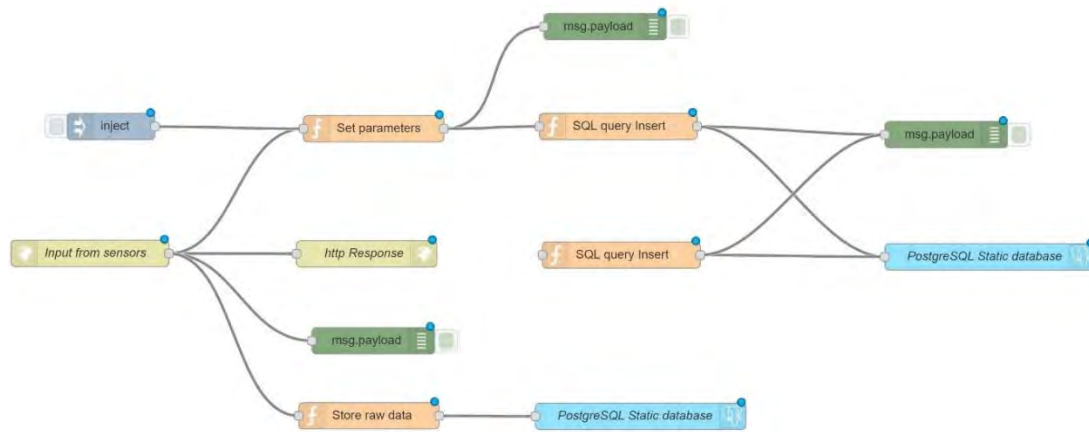


Figure 39: An overview of the workflow in Node-RED

The data collected by the sensors is sent to the KPN gateway, from this gateway the data can be received by Node-RED through an HTTP request. In the Node-RED environment there is a node that takes care of this retrieval, the endpoint address for the receiving of the messages is <https://geo1101.bk.tudelft.nl/iot/static/inputport>. The request will be a POST method since KPN will send the data in the body of the message. By doing a POST request not only the header of the message will be returned but also the body.

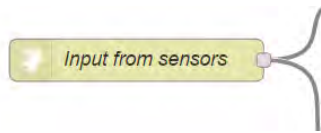


Figure 40: HTTP request to KPN gateway

After sending an HTTP request, an HTTP response needs to be sent as well to inform the gateway that the message was sent correctly. Node-RED provides this function by default. The message that is received from the HTTP request contains the data and metadata of the measurement as a JSON object. This object is stored in classes and objects, and can be extracted by a JavaScript code. The code is written in a function block that updates the message that is send to the next block. This function block also contains processing of the data before it is inserted into the database. Another import aspect of the ‘Set parameters’ block is the decryption of the payload; the actual message containing measurements of the sensors. KPN encrypts the payload of the JSON object. To decrypt this payload a ‘decrypting’ library is installed in the Node-RED environment. This decryption needs information provided by KPN, more information about this can be found in Appendix S.

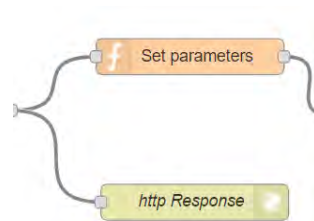


Figure 41: HTTP response and function block for extracting data

To insert the data into the database, an SQL query has to be formulated. This can be done in the ‘Set parameters’ block, and the query will insert it into the right columns by sending the query to a PostgreSQL block. The entire JSON object is also stored in the ‘raw data’ table in the database in the ‘Store raw data’ block.

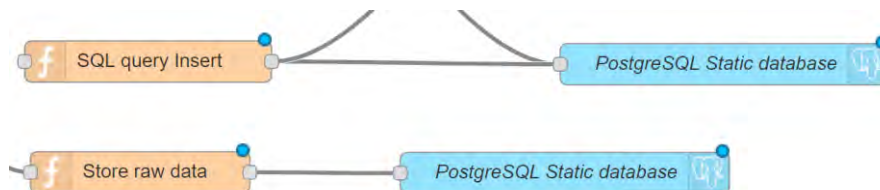


Figure 42: Function blocks for SQL queries

Other blocks used in the workflow are for monitoring and testing, such as the ‘Debug’ nodes that visualize the output of every block. An ‘Inject’ block with a dummy LoRa message is also used in the development to test and debug the workflow without using the LoRa network. The entire code of every block can be found in Appendix X.

3.4.4 Data processing

Most sensor values send over LoRa require one byte, since it contains decimals between 0 and 255. In the air quality, sometimes numbers bigger than 255 can occur. These values are therefore split on the LoPy. The values are glued back together in the Node-RED environment. For example: the number 4096 for example is send as 40 and 96 separately. In the Node-RED environment a string operation is carried out to glue $40 + 96 = 4096$ and not as $40 + 96 = 136$.

Another processing activity is sending floating point numbers such as temperature and humidity. These values are split on the LoPy in the number before the point and the number after the point. A value of 27.1 will therefore be send as 27 and 1, and put back together as 27.1 in the 'Set parameters' block (for code see Appendix X).

3.5 Feedback mechanism

This section will cover the feedback system of the sensor network. The first part describes the Twitter feedback mechanism and the second part covers the Sensor City Delft website. Both systems assume all required data is stored in an online accessible PostgreSQL database. Several other possibilities exist to implement a feedback system. This project focuses on Twitter and a website.

3.5.1 Twitter

Twitter is the second most popular social network. It is a real-time, public microblogging network known for the 140-character limit (Moreau, 2017). In this project, Twitter will be used as the social media medium to convey data. The data can be requested by tweeting at [@SensorCityDelft](#). This paragraph describes how the twitterbot works.

User experience and information

The idea of this feedback system is that the user tweets to [@SensorCityDelft](#) with a specific hashtag. These hashtags are the streets of the study area: [#Voldersgracht](#), [#Choorstraat](#) and [#Oudelangendijk](#) in Delft. The Twitter account and hashtags are communicated to the user with billboards (Appendix P), social media (Facebook account) and flyers (Appendix Q) distributed in the study area. The response of this tweet is the latest available environmental data present in the database. These are the noise level, temperature and humidity. The air quality is not tweeted, because this sensor started working in the last two weeks of the project.

The values which are tweeted to the requestor from a single sensor. Averaging out multiple sensors per street is a possibility, but for the majority if the project time only one sensor per street was deployed.



Figure 43: Twitterbot responds to the user

Technical System

Twitter has an Application Programming Interface (API) which can be used to program the twitterbot. This Twitter API is fully integrated in the Node-RED environment: only credentials of the Twitter account have to be provided in order to connect Node-RED to Twitter. Several authentication steps are taken care of by Node-RED, which makes the programming more user friendly.

The twitter account is run autonomously by a JavaScript code. This code is written in function blocks on Node-RED, the output of this block goes to the Twitter block. A specific flow in the Sensor City Delft instance on Node-RED is called 'Twitter Flow', and takes care of the so called 'twitterbot'. The flow exists of six blocks (figure 44): they 'listen' for tweets, parse tweets, retrieve data from the database and create tweets to send back to the data requester.



Figure 44: Node-RED flow of twitterbot

Listening block

The Node-Red listening block searches all public tweets directed to the [@SensorCityDelft](#) account. When a message is found, it is passed to the next block as a full tweet object as defined by Twitter (Twitter Developer Documentation, 2017)

Parse Tweet & Get Data

This function block parses the tweets using a JavaScript code. This code will look for the specific hashtags of locations and text in the tweets. When one of the hashtags is encountered, or certain words like 'temperature', 'noise' or 'humidity' are part of the tweet, it will generate a text for the reply tweet. The latest values of these measurements are fetched with SQL queries in the code of the block.

PostgreSQL block

A PostgreSQL block is used for retrieving the latest data from the database. The queries to fetch the data stem from the 'parse tweet and get data' block.

Create Tweet block

This block creates the full text that will be sent on twitter. The main text is already generated by previous blocks, but the tweet has to be sent to the requester. This is done in the 'create tweet block'.

Twitter block

This block sends out the tweet on twitter using the Twitter API (integrated in Node-RED).

Drawbacks

The twitterbot can only search for public tweets. When a user has protected tweets, the bot will not be able to respond. Another drawback is that people do not always use the right hashtags, the twitterbot responds with a feedback telling the requester that his or her tweet was not correctly formulated. The code of every block of the workflow can be found in Appendix R.

3.5.2 Website

The billboards and distributed flyers contain a QR-code which leads to www.scdelft.nl. This website contains information on the project, such as news, the project team members, measurements and measurement locations (see figure 45).

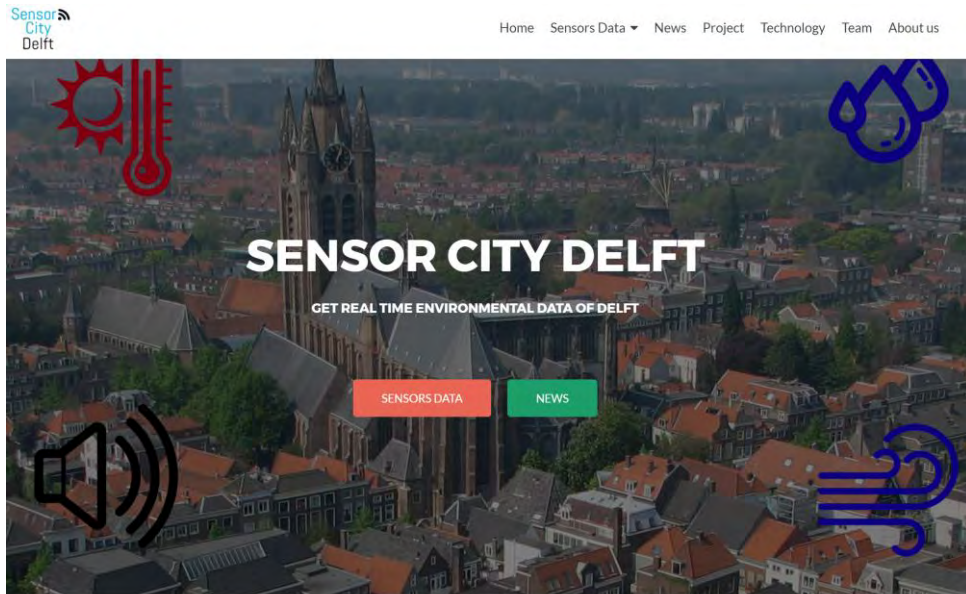


Figure 45: Welcome page of the project website (www.scdelft.nl)

The measurements on the website are shown in a dashboard. This dashboard (figure 46) is created in the Node-RED environment.

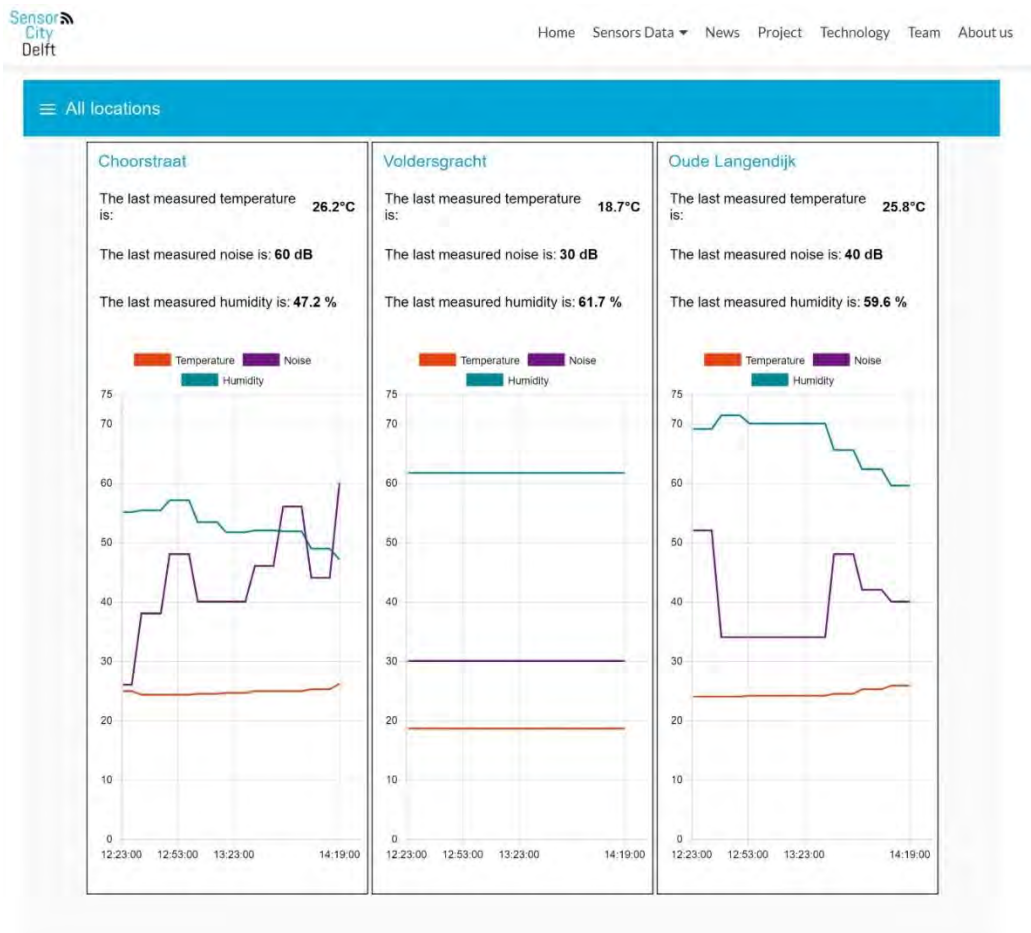


Figure 46: Dashboard on www.scdelft.nl

This dashboard shows live data. After the sensors are taken down, historic data will be shown (figure 47). See Appendix R for the dashboard in the Node-RED environment.

Time series of the last readings is shown below:



Figure 47: Historic data on www.scdelft.nl

Chapter 4 - Analyzing Data & Results

The sensors of this project are located as can be seen in figure 49. All sensors are equipped with a temperature and humidity sensor and a noise sensor as described in chapter 3.1. Sensors 66D (Oude Langendijk) and 981 (Choorstraat) are also provided with a PMS5003 sensor as described in chapter 3.1. The three locations in which the sensors are located have different profiles and the situation of each sensor is different. All three locations are briefly explained in the first paragraph of this chapter. In the next paragraphs the scenarios that are introduced in chapter 2 will be analyzed and answered. All graphs that are found in this chapter can also be found on a bigger scale in appendices V and W. Also some information about the data like maximum, minimum, mean and standard deviation per time period of four hours can be found in this appendix.

4.1 Locations

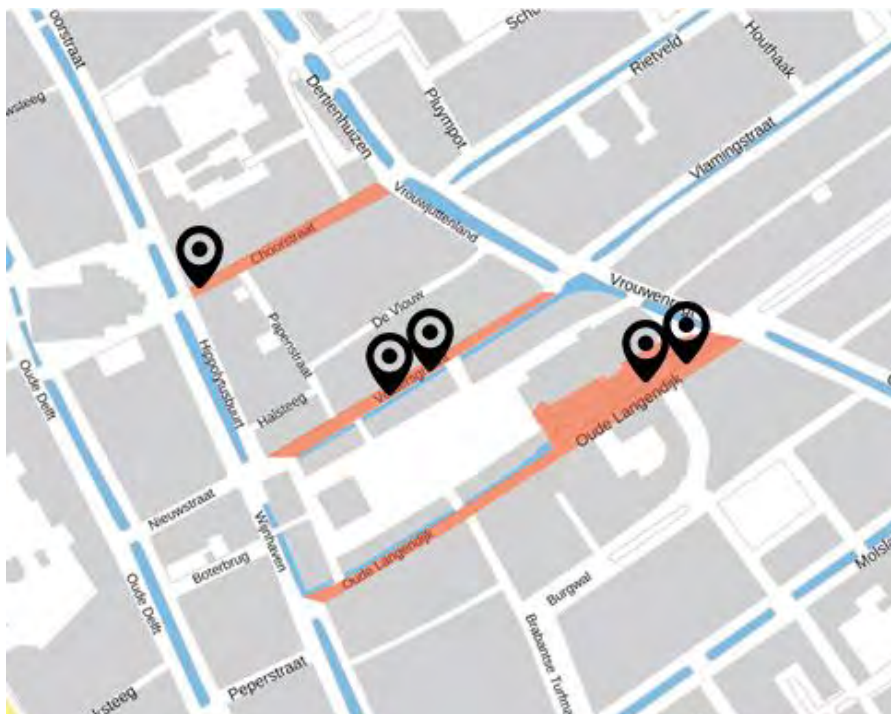


Figure 48: Locations of sensors in the City center of Delft

4.1.1 Voldersgracht

The Voldersgracht is a small street and has a canal next to it. In this street no cars are allowed, only pedestrians, bikes and scooters. In this street sensor 073 and sensor 210 are mounted. The sensors are

mounted to trees which do not have many leaves and therefore do not cause much shade for the sensors. Since this street is east/west oriented there are two periods of direct sunlight in the street. According to SunCalc (<http://suncalc.net>) in the morning the sun will shine in the street between 5:30 and 8:30 o'clock. In the afternoon, there will be direct sunlight between 15:30 and 16:30 (figure 50).



Figure 49: Location of sensor 073 (left) and location of sensor 210 (right) at the Voldersgracht



Figure 50: Voldersgracht in SunCalc

4.1.2 Choorstraat

Choorstraat

At the Choorstraat only one sensor is deployed, this is sensor 981. The sensor is located at the far west end of the street. The sensor is mounted to a tree and is mostly covered by the leaves of this tree as can be seen in figure 51. Just like the Voldersgracht this street only allows pedestrians, bikes and scooters to pass through and no cars are allowed. Different from the Voldersgracht it does not have a canal next to it. This street is oriented east/west too, therefore it has two moments of direct sunlight. For the morning sun, this is between 5:30 and 8:00 o'clock and the afternoon sun between 14:30 and 16:45 o'clock (figure 52).



Figure 51: Location of the sensor 981 (left) in Choorstraat. The sensor is covered by leaves of the tree (right)



Figure 52: Choorstraat in SunCalc

4.1.3 Oude Langendijk

The Oude Langendijk is the street that is most busy of the three streets investigated. Cars and busses are allowed to pass and it is expected that they cause a lot of noise, heat and pollution. But then again, this street has a lot of big trees which cause a lot of shadow. Sensor 66D and 772 are located at this street and are mounted to lamp posts (figure 53). Also this street is oriented east/west and therefore has two periods of direct sunlight. The morning sun is in the street between 5:30 and 6:30 o'clock and the afternoon sun between 15:00 and 17:15 o'clock (figure 54).

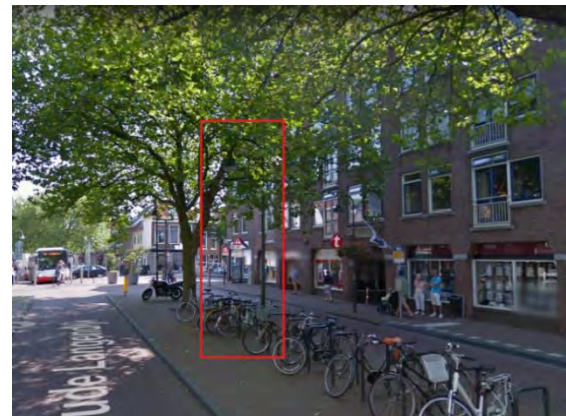


Figure 53: Location of sensor 772 (left) and location of sensor 66D (right) at Oude Langendijk

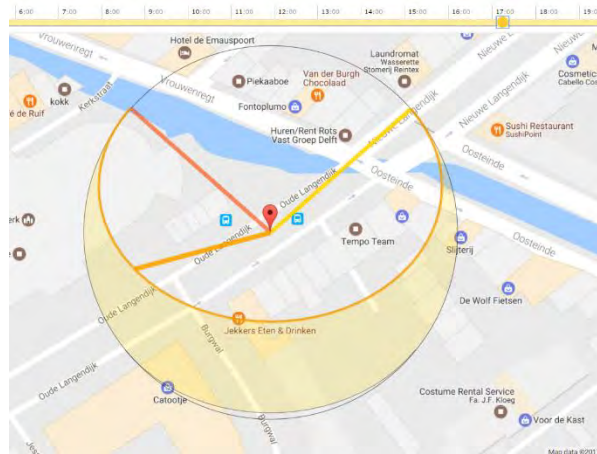


Figure 54: Oude Langendijk in SunCalc

4.2 Temperature and humidity

4.4.1 Introduction

The scenario introduced in sub-paragraph 2.2.3 describes a difference in temperature and humidity between the three streets of interest. The profile of Richard describes the fact that he notices a difference in temperature after doing his grocery shopping at the Choorstraat. His findings are that the Voldersgracht is cooler than the Choorstraat and that the Oude Langendijk is even hotter than the Choorstraat. In this paragraph, these findings are put to the test by analyzing the data that is collected.

4.4.2 Temperature

In general, for all streets, a pattern of day and night can be seen in the graphs of the data. Besides that, some sensors had other patterns or noticeable events that will be discussed and declared in this paragraph.

Voldersgracht

Like described above the pattern of day and night can be seen in the graph underneath (figure 55). During the night the street cools down and during the day it heats up. The biggest difference between the two sensors of this street can be seen between 16:00 and 18:00 o'clock. Just before this moment the sensors had direct sunlight in the street so they could heat up. But the temperature of the sensor 210 continues to increase. This might have to do with the fact that this sensor has a longer exposure of direct sunlight than the other one. It takes some time for the data to actually increase after an exposure of heat and this is why the peak is just after the sun has shined upon the sensor platform.

At the Voldersgracht the maximum temperature is reached around 18:00 o'clock, for the first day this is 31,3 °C and the second day 33,9 °C (both from the sensor 210 which has more direct sunlight). The minimum temperature is reached at 6:00 o'clock, for the first day this is 15,2 °C and the second day 18,2 °C (also both from sensor 210).

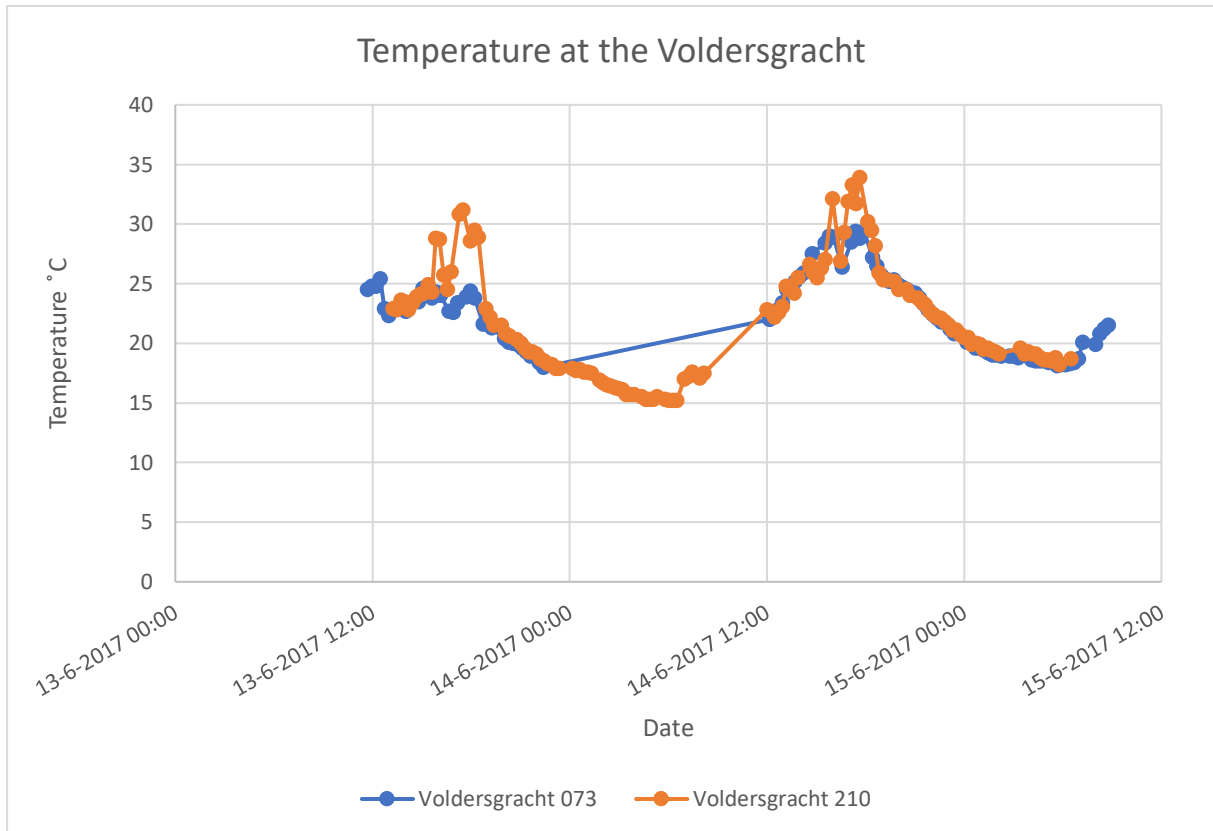


Figure 55: Graph of temperature at the Voldersgracht

Oude Langendijk

In general, at the Oude Langendijk both sensors describe the same pattern for the measured days (figure 56). Nevertheless, during the period between 7:00 and 10:00 o'clock a bigger increase of temperature is seen for sensor 772. This is caused by the fact that this sensor platform is located at the most east part of the street and therefore there is no shadow during the early morning sun. The other sensor (66D) is in between trees from both east and west side and thus has shadow during the morning and afternoon sun. After this the temperature increases less and this results in the values being around the same of the other sensor.

The highest temperature measurements differ for the two days. The first day is measured at 15:00 o'clock (21,9 °C) and for the second day it is measured at 18:00 o'clock (25,9 °C). The moment of measuring the lowest temperature is the same as at the Voldersgracht: 6:00 o'clock. On the 14th of June this lowest temperature is 13,7 °C and at the 15th of June 17,0 °C.

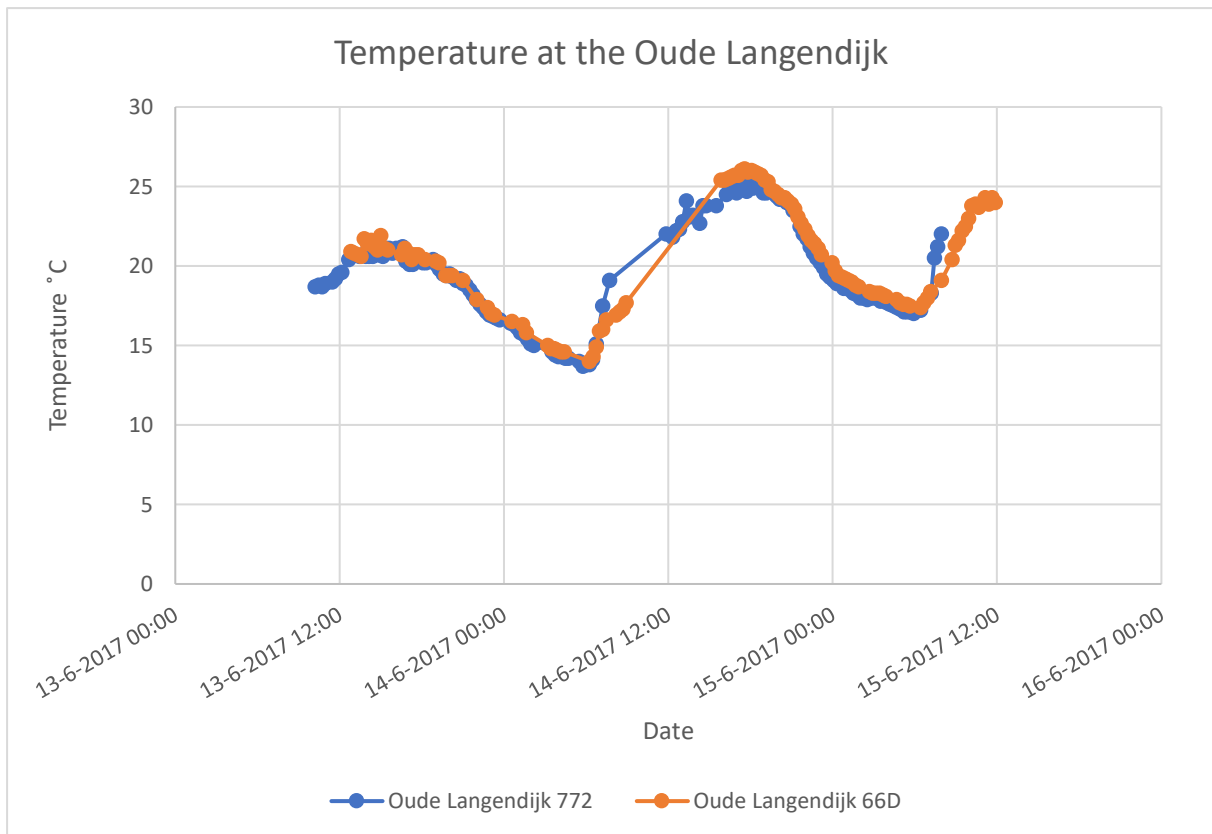


Figure 56: Graph of temperature at Oude Langendijk

Choorstraat

Only one sensor station has measured the values of the Choorstraat, so no differences between different sensor platforms in one street can be drawn. This street has the same kinds of patterns as the two streets described before (graph 57). At 18:00 o'clock the temperature is at its maximum (22,9°C at the 13th of June and 28,9°C the 14th). Also, the minimum temperature is reached at 6:00 with a value of 15,3°C and 19,1°C. This sensor was most of the time in the shade of the tree. But during the period where the maximum amount was reached the sunlight probably reached the sensor box between the leaves since this value is rather high.

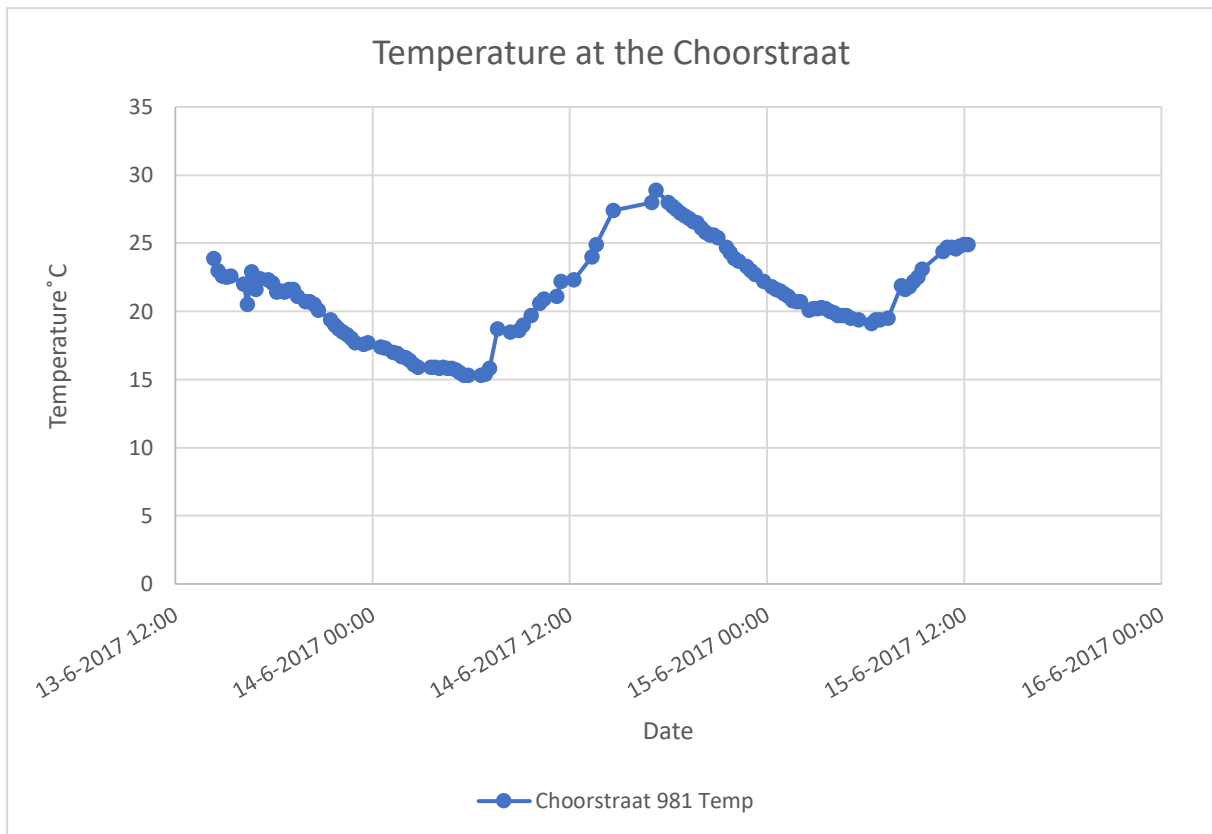


Figure 57: Graph of temperature at Choorstraat

4.4.3 Humidity

Just like the temperature, humidity also shows a pattern for day and night. If the night falls, the temperature drops and the humidity increases. During the day the temperature rises and the humidity decreases. The location of the sensors and the condition of the direct environment has an influence on the data. The remarkable incidents are discussed and clarified in this paragraph.

Voldersgracht

The two sensors that measured the humidity at the Voldersgracht show a same kind of pattern as they did while measuring the temperature, but now inverse (graph 58). At the moment the 210 sensor showed an increase of temperature after direct exposure to the sun, the humidity dropped.

The only difference is the moment of highest humidity. These differ slightly from the minimum value of temperature and also have 4 hours of difference between the two measuring days. For the 14th of June the highest humidity is measured at 6:00 o'clock (79,5%) which is the same moment of the highest temperature. In contrast to this, the highest humidity measured at the 15th of June is at 2:00 o'clock (72,2%) which is 4 hours prior to the lowest temperature. For the second day this is the case for both sensors, so there is a consistency. For the first day there are only measurements of one sensor. The

early peak might be caused by the fact that between 4:00 and 7:00 o'clock the temperature is quite constant and deviates only 0,5 °C and at 2:00 o'clock the temperature is 1 °C warmer than the lowest value. Since a low temperature is reached at 2:00 o'clock the humidity is high as well. The lowest humidity is measured, as expected, at 18:00 o'clock (27,6% and 23,6%).

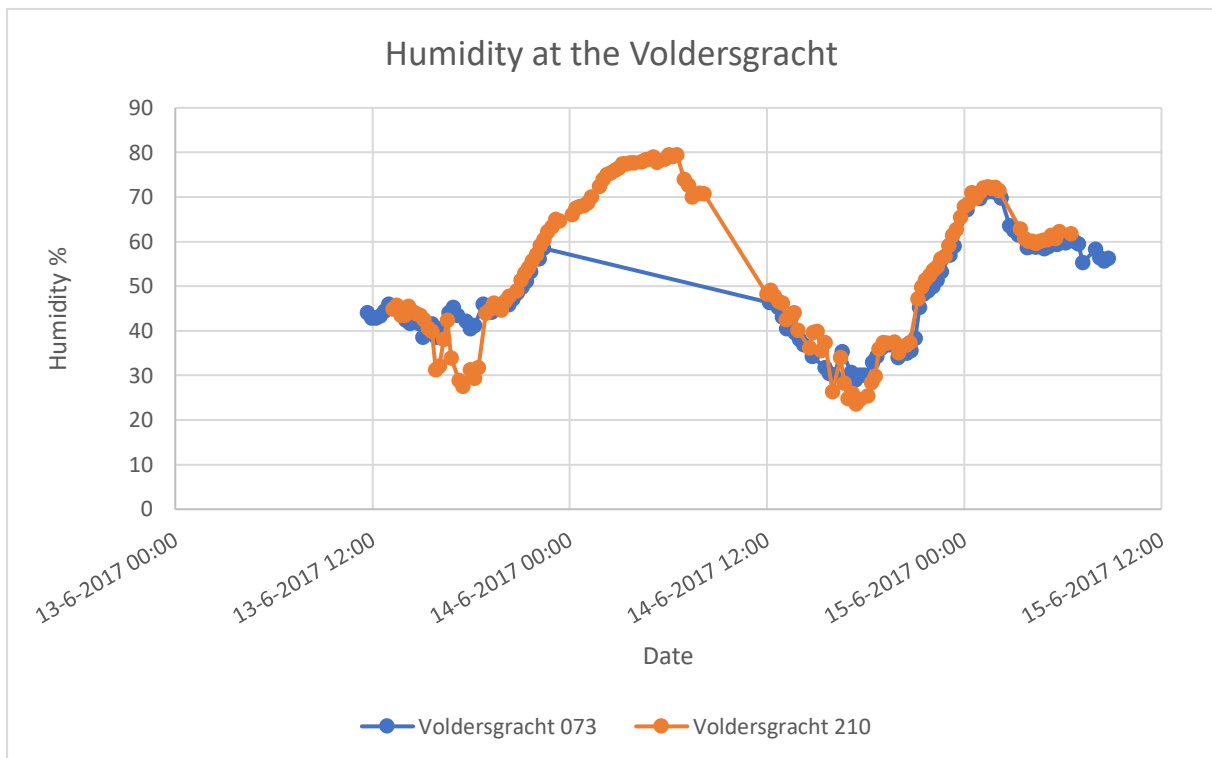


Figure 58: Graph of humidity at Voldersgracht

Oude Langendijk

Since the Oude Langendijk has a lot of big trees which cause shade and therefore the temperature is lower, this street has a higher humidity value. The pattern of the humidity value at the Oude Langendijk can be found in graph 56. Sensor 66D has a higher humidity value than sensor 772. This is caused by the same circumstance why sensor 66D reported lower temperature values in the morning than sensor 772. Sensor 66D is in between trees and 772 is at the corner of the street. Trees do not only cause shade but also a higher humidity value. At one moment in time the humidity value of 66D is very high (93%) the explanation for this phenomenon is that there could have been condensation or fog at the sensor.

The lowest humidity value for the 13th of June is measured at 14:00 o'clock (50%) and at the 14th of June at 20:00 o'clock (39%). This is very remarkable since these two values differ 6 hours. But this can be explained by the fact that the temperature peak of the sensor for the 13th is at 15:00 o'clock and the 14th it is at 18:00 o'clock. The highest values of humidity at the Oude Langendijk are also not at the same time for the two days. For the first day it is at 6:00 o'clock (93,2%) and the second day it is at 2:00 o'clock

(87,2%). This deviation from the minimum temperature can be explained the same way as for the Voldersgracht: the temperature is quite constant at that time. Between 2:00 and 6:00 o'clock the temperature difference is only 1 °C (figure 59).

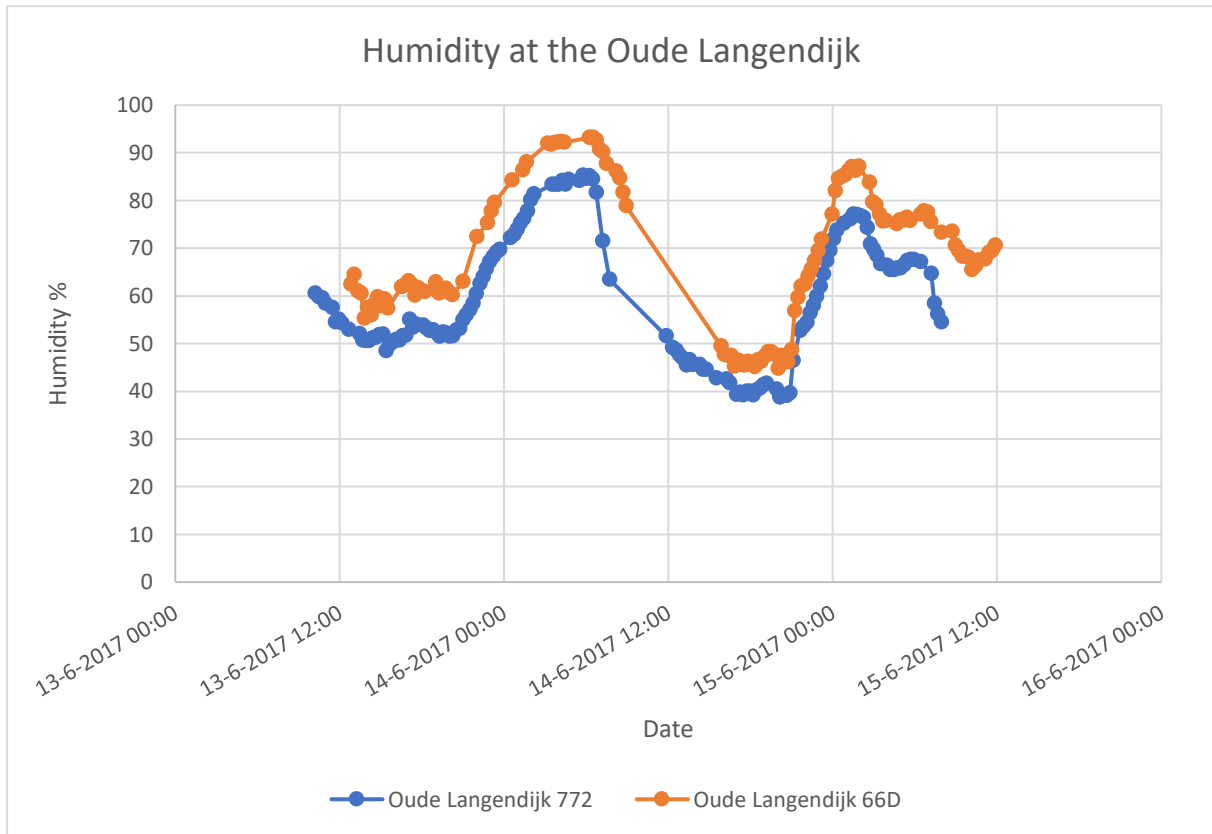


Figure 59: Graph of humidity at Oude Langendijk

Choorstraat

The sensor that is located at the Choorstraat shows the same kind of patterns as the humidity sensors at the Voldersgracht and Oude Langendijk (figure 60). For the first night the maximum humidity is reached around 6:00 o'clock (79,4%) and the second night around 2:00 o'clock (64,8%). This is for the first night around the lowest temperature value but for the second night a couple of hours before that. This is, like with other sensors, caused by the minimum spread of temperature values between 2:30 and 6:30 o'clock. The lowest humidity values are observed at 14:30 (42,2%) and 18:00 o'clock (31,6%). For the 13th of June the deviation of the highest temperature at that day can be explained by the small temperature deviation during this time period. The temperature is quite constant and therefore the minimum humidity value is reached at this moment.

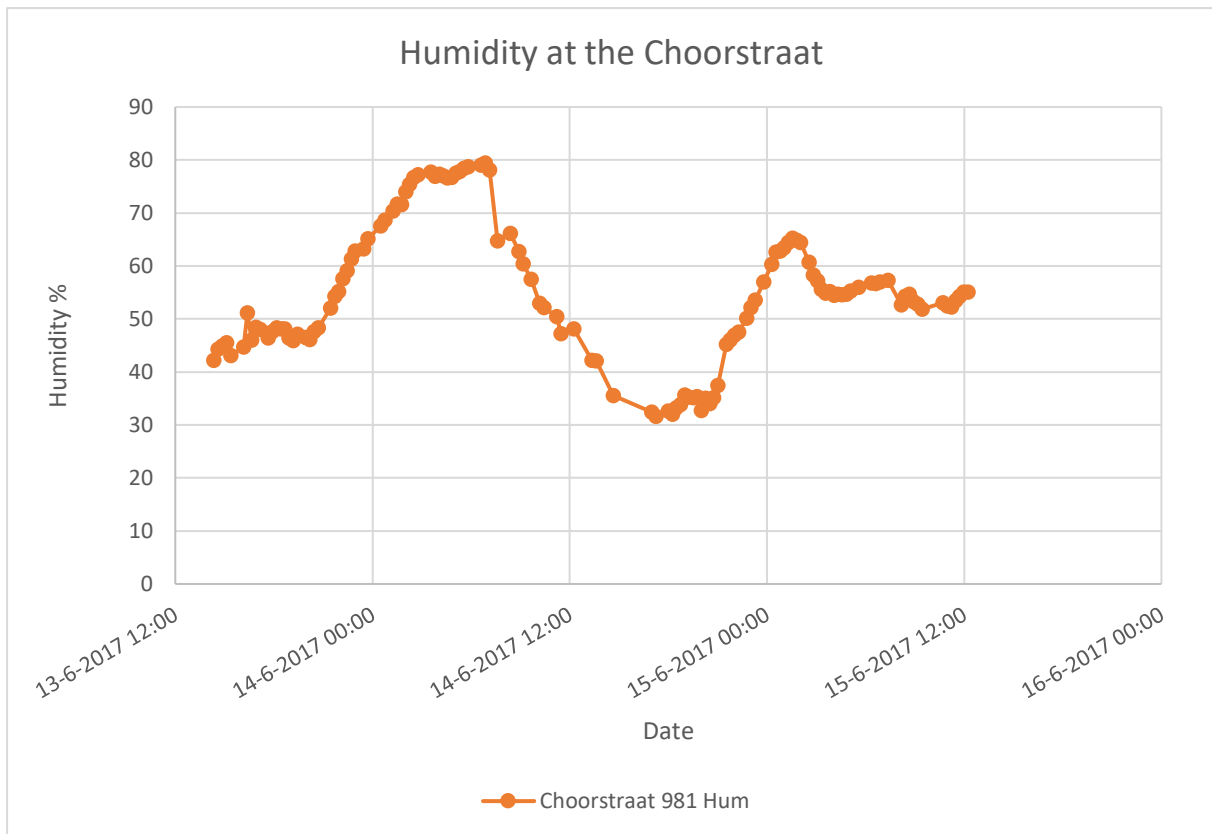


Figure 60: Graph of humidity at Choorstraat

4.4.4 Conclusion temperature and humidity

After plotting the temperature data of all sensors in one graph as can be seen in figure 61 the main differences are in the values and not in the pattern. The peaks fall all around the same time besides the maximum temperature of the Oude Langendijk, which is prior to all other maximum values. This value is detected by sensor 66D which is the one to the west (in between trees). After examining the data it can be concluded that this was an early peak due to direct sunlight exposure. The sensor is placed in between trees but has some gaps between the leaves and the street is wider than the other ones so the sun reaches the sensor earlier than in the other streets.

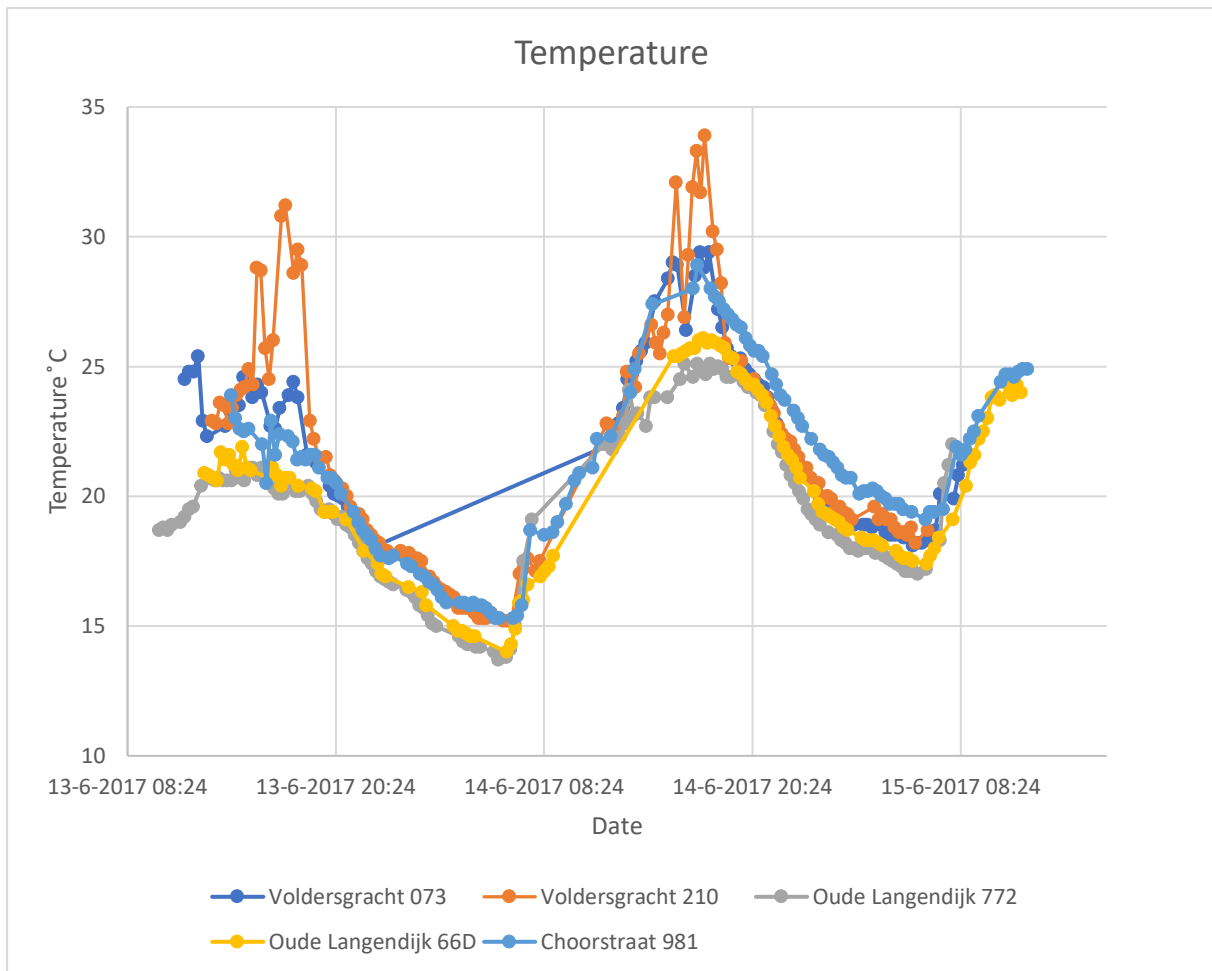


Figure 61: Temperature of all five sensors (note: the y-axis starts with 10 degrees C)

Examination of the numbers of the data as in table 14 exposes the consistency that for all streets the second day was hotter than the first one. Focused on the maximum temperature it shows that the Choorstraat has increased the most in temperature, after this the Oude Langendijk and the Voldersgracht has increased the less in temperature in the two days.

| | Voldersgracht | Oude Langendijk | Choorstraat |
|------------------------|-----------------|-----------------|-----------------|
| Temperature max | 18:00 - 31,3 °C | 15:00 - 21,9 °C | 18:00 - 22,9 °C |
| | 18:00 - 33,9 °C | 18:00 - 25,9 °C | 18:00 - 28,9 °C |
| Temperature min | 6:00 - 15,2 °C | 6:00 - 13,7 °C | 6:00 - 15,3 °C |
| | 6:00 - 18,2 °C | 6:00 - 17,0 °C | 6:00 - 19,1 °C |

Table 14: Table with temperature values

Plotting all humidity data results in figure 62. This graph shows similar patterns for all streets like the temperature did. Also like the temperature graph the main differences can be found in the values of the measurements. Some peaks are found at different time slots but on average

they are aligned. It is very clear that sensor 66D has the highest humidity overall. This one had the lowest temperatures too and is located in between trees and therefore more humidity is produced around the sensor.

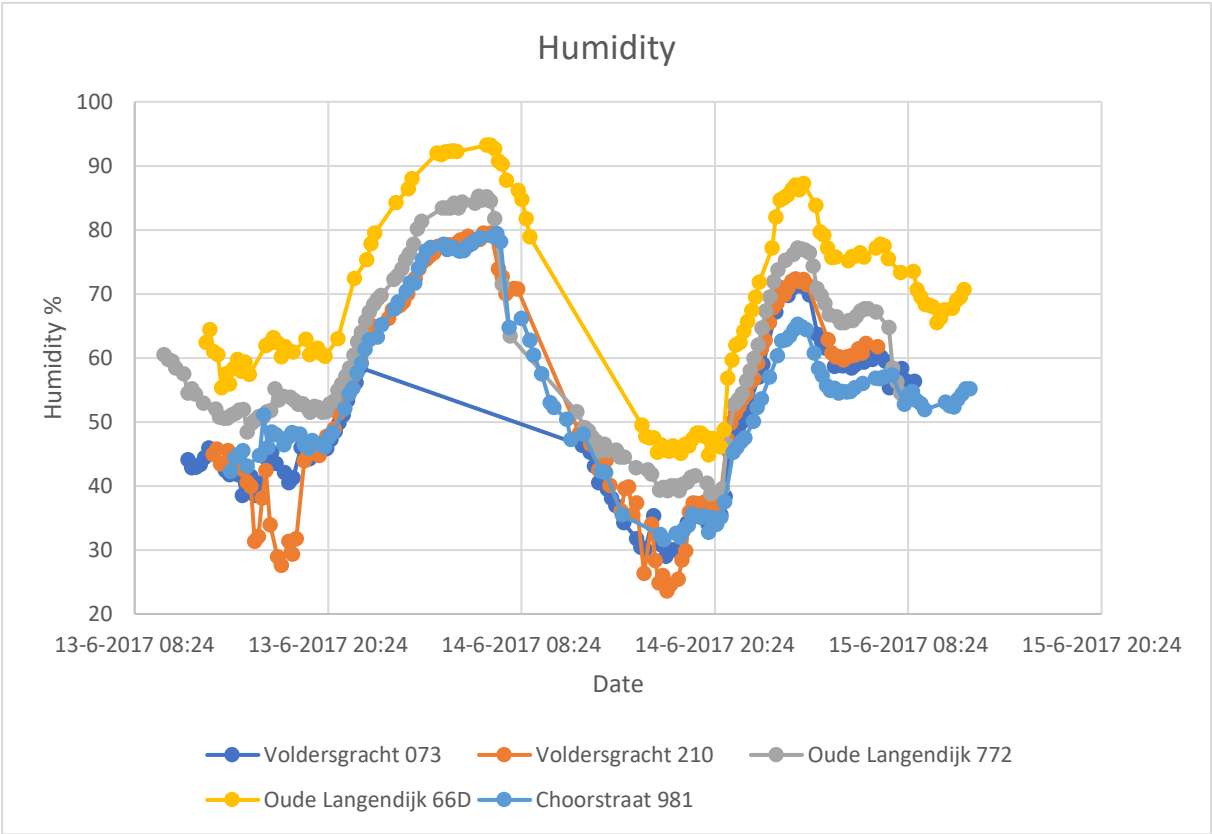


Figure 62: Humidity of all five sensors (note: the y-axis starts with 20%)

Examination of the data of humidity as presented in table 15 shows the maximum and minimum values of the humidity. The moment of the maximum values for the 14th of June are all at the same time, but for the 15th of June this maximum value is measured at the Voldersgracht and Oude Langendijk four hours earlier which is declared by the fact that the temperature between 2:00 and 6:00 o'clock is fairly constant at those streets. The deviation for the minimum values is also explained by this phenomenon of a period of fairly constant measurements around this time.

| | Voldersgracht | Oude Langendijk | Choorstraat |
|---------------------|---------------|-----------------|---------------|
| Humidity max | 6:00 - 79,5% | 6:00 - 93,2% | 6:00 - 79,4% |
| | 2:00 - 72,2% | 2:00 - 87,2% | 6:00 - 64,8% |
| Humidity min | 18:00 - 27,6% | 14:00 - 50% | 14:30 - 42,2% |
| | 18:00 - 23,6% | 20:00 - 39% | 18:00 - 31,6% |

Table 15: Table with humidity values

After ordering the five sensors in the three streets from highest to lowest temperature overall and taking into account the humidity of the streets the following list comes out:

1. Voldersgracht 210
2. Voldersgracht 073
3. Choorstraat 981
4. Oude Langendijk 66D
5. Oude Langendijk 772

This list indicates a different order than Richard expected. Richard expected the Voldersgracht to be the coolest and Oude Langendijk to be the hottest which is completely opposite from this list. A clarification for this unexpected outcome is that the surroundings of the sensors were not the same for the different streets. The sensors at the Oude Langendijk were most of the time in the shade of the big trees that are in that street and the sensor of the Choorstraat was in the shade of the tree as well. In contrast, the trees on which the sensors at the Voldersgracht were placed did not have that many leaves and therefore did not cause that much shade for the sensors. For this reason, the Voldersgracht came out much higher than expected. For further research to make more reliable measurements all sensors should be placed in the shade without any direct sunlight.

4.3 Noise measurements

In contrast to the temperature, humidity and air quality data analysis, the noise data analysis failed to give a clear insight of the noise levels at the three streets chosen. This was mainly a result of the way that the sound sensor works. Similarly to the temperature/humidity and air quality sensors, the sound sensor were recording data every 15 minutes for 50 msec, being in deep sleep for the rest of the time. This resulted to the generation of graphs with values that deviated from high picks of 60 to low picks of 10 after 15 minutes. Thus, the amount and quality of data was not sufficient to recognize noise patterns during the time of implementation of a sensor. Furthermore, since the sound sensor constitutes a low-cost microphone, the data received by the sound sensor was only an indication of dBa, it was not directly measured in dBa units. All these together, including the difficulty to correlate the data to a real sound, make it impossible to come to a conclusion related to which street and when in time has higher noise levels.

Table 16 shows the results of data collected at various time points from the three selected streets. Note the small differences in standard deviations (St. dev), average values (Lmean), maximum (Lmax) and minimum (Lmin) values between the data obtained from each of the three streets.

| | Category | N | Lmean | Lmax | Lmin | St.dev |
|--------------------|-----------------|-----|-------|------|------|--------|
| Total | | 884 | 34,9 | 60 | 2 | 19,16 |
| Street | Choorstraat | 206 | 25,7 | 60 | 2 | 13,01 |
| | Voldersgracht | 279 | 42,3 | 60 | 2 | 12,13 |
| | Oude Langendijk | 399 | 29,87 | 60 | 2 | 12,6 |
| Time of Day | 06:01-10:00 | 77 | 37,53 | 60 | 4 | 14,2 |
| | 10:01-14:00 | 104 | 33,64 | 56 | 2 | 13,2 |
| | 14:01-18:00 | 138 | 35,5 | 60 | 2 | 15,06 |
| | 18:01-22:00 | 203 | 34,62 | 60 | 2 | 14,36 |
| | 22:01-02:00 | 204 | 30 | 60 | 2 | 13,28 |
| | 02:01-06:00 | 153 | 29,47 | 60 | 2 | 13,4 |

Table 16: Noise data for the three selected streets and during specific time slots

The minor differences observed do not allow us to draw safe conclusions. Based on these data we can only report that the Voldersgracht shows the largest average noise value (average noise: 42.30), which means it is the busiest street during the whole day and especially between 22:00-02:00 (average noise: 43.13, St. dev: 9.22; see table 17), b. the Oude Langendijk is the second busiest street (average noise: 29.87), and c. The Choorstraat is the most quiet street of all (average noise: 25.7). Notably, these results are not consistent to our initial hypothesis (chapter 3) which means that further research and analyses are needed. Quality of data is also an important aspect that has to be taken into account. For example, the sound values from the two sensors placed in the Voldersgracht were very different to each other although the recording time slots were the same (see figure 63). A similar problem is reported for the data obtained from the two other streets.

| Voldersgracht | Category | N | Lmean | Lmax | Lmin | St.dev |
|--------------------|-------------|-----|-------|------|------|--------|
| Total | | 279 | 42,3 | 60 | 2 | 12,13 |
| Time of Day | 06:01-10:00 | 18 | 48,84 | 60 | 30 | 10,72 |
| | 10:01-14:00 | 21 | 33,1 | 50 | 14 | 11,07 |
| | 14:01-18:00 | 43 | 42,76 | 60 | 2 | 12,82 |
| | 18:01-22:00 | 72 | 42,75 | 60 | 10 | 14,08 |
| | 22:01-02:00 | 71 | 43,13 | 60 | 24 | 9,22 |
| | 02:01-06:00 | 54 | 42,33 | 60 | 22 | 10,84 |

Table 17: Noise data of Voldersgracht during specific time slots

As an example, the noise that is sensed by two devices in the Voldersgracht is plotted in a graph (figure 63).

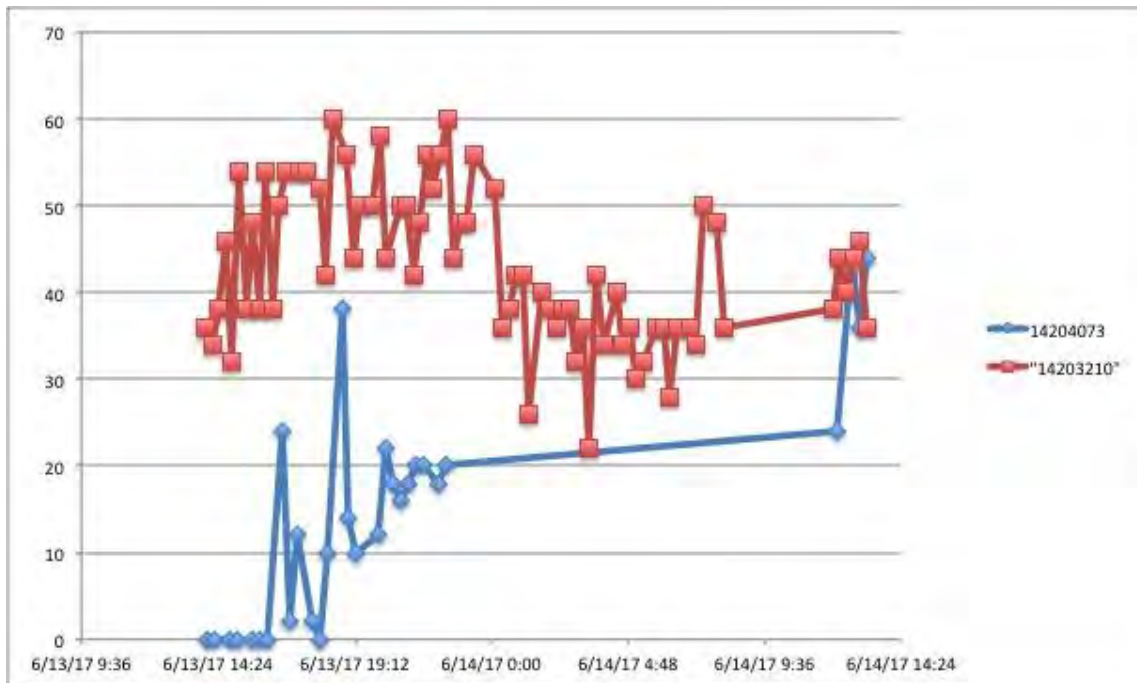


Figure 63: Graph of noise values from sensors in Voldersgracht

4.4 Air quality

4.4.1 Introduction

This section describes the data analysis of the measurements that have been gathered by the PMS5003 dust sensor and the PPD42NS dust sensor at the Oude Langendijk, Choorstraat and Voldersgracht. This data is analyzed to provide an answer to the story concerning the air quality at the Oude Langendijk. In this story Peter, a small business owner, wants to find out whether the daily traffic affects the air quality of his street and those nearby.

In order to provide an answer to this question two sensors were installed: the PPD42NS and the PMS5003 sensor. The Choorstraat had one PMS5003 sensor installed which gathered data from June 13th till June 15th. At the Voldersgracht the PPD42NS sensor was installed which worked from 18:11 on the 9th of June until 10:36 on the 12th of June, conducting in total only 32 measurements. In order to be able to compare the different streets to each other, measurements taken in the same time slot are compared.

First, the measurements of the PMS5003 sensor at the Oude Langendijk and Choorstraat are analyzed and compared. Second, the measurements from the PPD42NS sensor at the Voldersgracht and Choorstraat are discussed. Thirdly, the PMS5003 sensor is discussed. Afterwards, based on the analysis the conclusion formulates the answer to the story.

4.4.2 Measurements with the PMS5003

The Oude Langendijk had two different kind of dust sensors implemented: PMS5003 and PPD42NS. First the outcomes of the PMS sensor are discussed. The sensor platform with the PMS5003 sensor gathered data from June 13th till June 15th. The PMS5003 sensor was used to gather measurements on fine particulate matter (PM). Particulate Matter is already discussed in the theoretical framework (paragraph 2.2.1).

According to the data manual of the PMS5003 sensor the sensor measures three ranges PM10, PM2.5 and PM1. Fine particles such as PM2.5 and PM1 have been found to play a significant role in health hazard, climate change and pollution problems (Vecchi et al., 2004). Therefore the main focus will be on the PM2.5 and PM1 particles, which are emitted by motor vehicles such as buses and scooters (Keogh & Sonntag, 2011).

Measuring air quality in the Oude Langendijk with PMS5003

The results of the measurements of PM2.5, PM1 and PM10 are shown in table 18. Figure 64 shows the PM measurements conducted on the period of June 13th to June 15th at the Oude Langendijk. It shows an increase of PM levels during the course of the night of June 13th to June 14th from 23:17 o'clock until 6.58 o'clock and the night of June 14th to June 15th from 4.54 till 10.24 o'clock in the morning. In the next sections the possible causes for these increases are discussed.

| | Category | N | Mean | Max | Min | Standard deviation |
|------------------------|----------|-----|-------------|-----|-----|--------------------|
| Oude Langendijk | | | | | | |
| Total | PM10 | 116 | 1.145299145 | 12 | 0 | 2.600613181 |
| | PM2.5 | 114 | 14.5 | 34 | 0 | 6.966061346 |
| | PM1 | 116 | 8.47008547 | 34 | 0 | 7.568734064 |
| Choorstraat | | | | | | |
| Total | PM10 | 126 | 2.1015625 | 38 | 0 | 5.391506654 |
| | PM2.5 | 126 | 6.46875 | 24 | 0 | 5.350601556 |
| | PM1 | 126 | 10.5234375 | 116 | 0 | 11.74798682 |

Table 18: Mean, maximum and minimum of PM10, PM2.5 and PM1

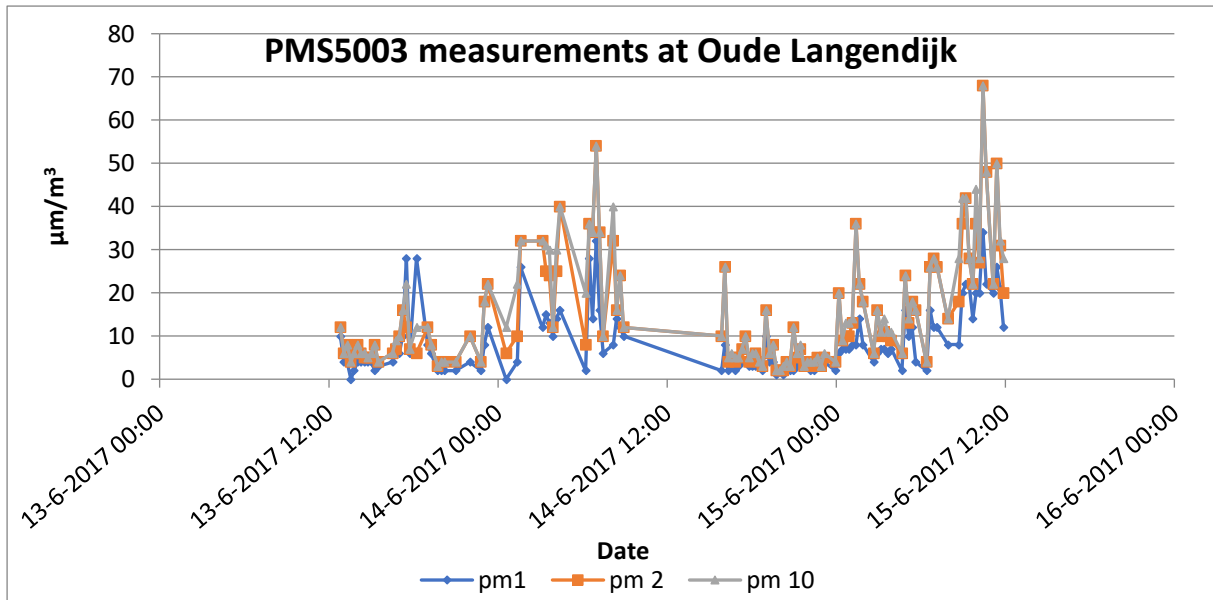


Figure 64: Graph of PMS measurements at Oude Langendijk, 13, 14 and 15 June 2017

Influence of humidity?

When looking at the temperature and humidity graph for this time period, the humidity also increases during the night, as shown in figure 65. Figure 66 displays the PM levels together with the humidity measured at the Oude Langendijk. The humidity level reaches at 6.14 o'clock in the morning of June 14th a peak of 93.2%. Also PM2.5 reaches a peak at 6.58 o'clock in the morning. At 1.24 o'clock in the night on the 15th of June both humidity and PM2.5 reach a peak of respectively 87% humidity and 36 microgram PM2.5. The rise of PM levels and humidity during the night could indicate a correlation between humidity and PM levels.

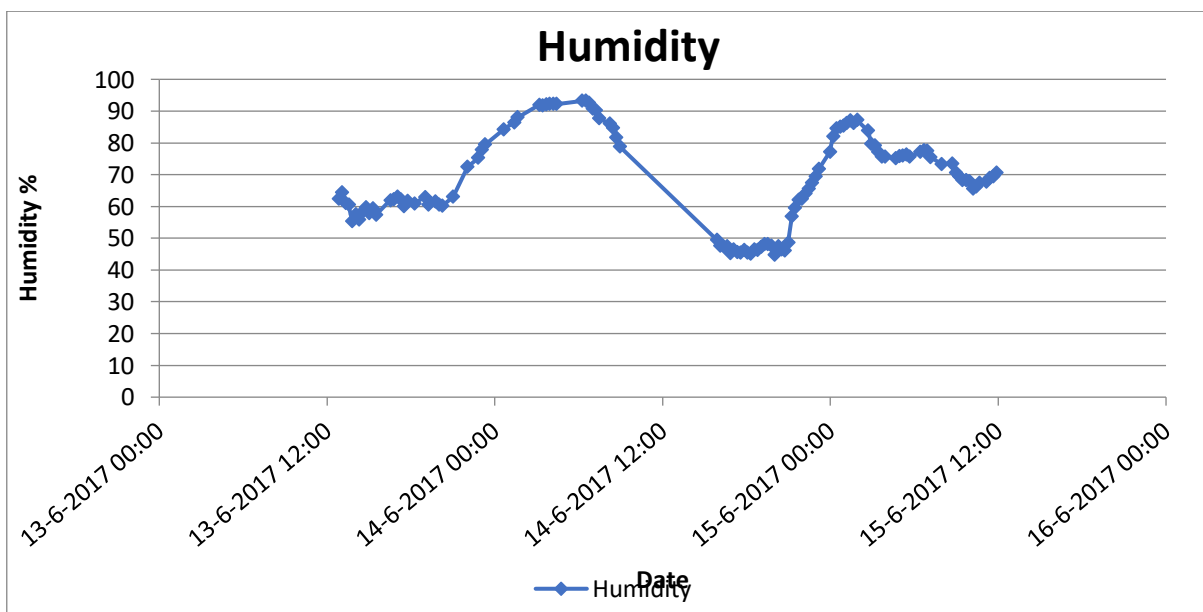


Figure 65: Humidity at Oude Langendijk

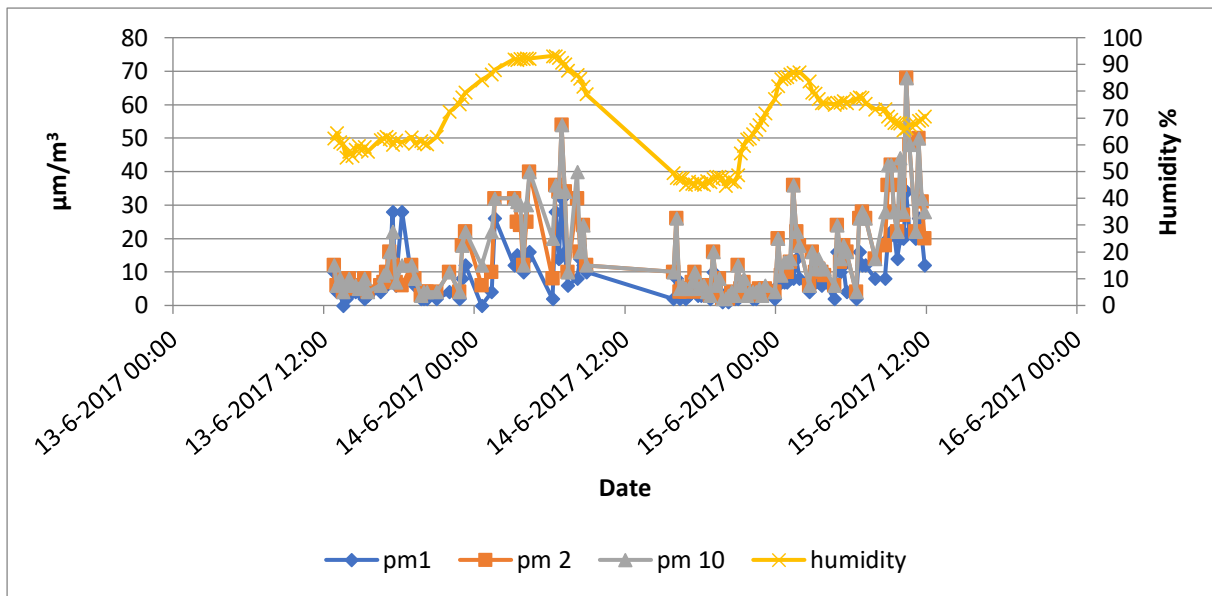


Figure 66: Humidity and PM levels at Oude Langendijk

Wang and Ogawa (2015) conducted research into the connection between meteorological conditions and pollutants. Concerning pollutants and humidity Wang and Ogawa state that when humidity is low, the PM_{2.5} concentration increases because of hygroscopic growth.

When humidity reaches a certain high the particles become too heavy to stay in the air and fall to the ground. The consequence is that the number of particles in the air decreases and the PM_{2.5} concentration also decreases. In addition, Afzali et al. (2015) also state that increased humidity is often preceded by rain and therefore through wash-out processes of the particles the concentration of pollutant in the air is reduced.

So, the existing research into the relationship between humidity and PM levels is not in accordance with the measurements gathered by the sensor platform. In general the research conducted into the correlation between humidity and PM concentrations in the air go more in depth into subjects such as the chemical composition of the particles and meteorological circumstances at a certain location.

However, the research by Afzali et al. (2015) is conducted on a dataset that is collected over two years, so the dataset will be more reliable than the dataset of this project. Next to that, in the scope of our project there is no information available about the chemical composition of the measured particles and, finally, detailed meteorological information at this specific place for the Oude Langendijk is unavailable. Therefore, it is not possible to provide enough evidence for a link between increasing levels of humidity and the increasing PM level.

Influence of wind?

Another cause of the increasing PM levels during the night could be a correlation between wind speed and PM levels. When looking at wind speed measurement in Hoek van Holland, the measurements show that the wind speed was around three to four m/s throughout the day and night of June 13th and 14th. The night of June 15th to 16th the wind speed was between four to five m/s, see figure 67.

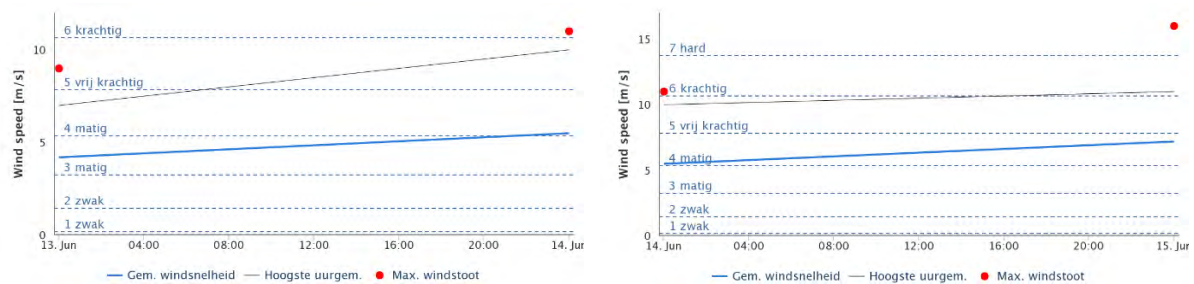


Figure 67: Wind speeds on the 13th and 14th of June

The wind speed increased at a constant rate during both nights, therefore there could be a correlation between the wind speed and PM levels. Wang (2015) states that in the case of low wind speed the wind can blow away pollutants in a limited geographical range, yet when wind speed is high the wind can carry large amounts of pollutants from far away. The effect is that the PM levels go up.

Nevertheless, there is no detailed information about the wind speed neither about the wind direction at the Oude Langendijk: the measurements at Hoek van Holland are way too far away to be reliable, and the local built environment can affect wind speeds and directions very easily. To investigate a relation between wind speeds, wind directions and PM values in the air, a recommendation is to mount a wind meter on the sensor platform. This knowledge is namely critical when making assumptions about the link between wind and PM concentrations

Difference plots

In order to see the distribution of the three types of PM values difference graphs are plotted. In the first difference plot in figure 68 PM_{2.5} is subtracted from PM₁₀ which returns the particles that belong to the PM₁₀ category. The graph shows that 81 out of the 117 measurements appear to have measured PM_{2.5} and no PM₁₀. From this difference plot it can be concluded that PM₁₀ levels are low compared to PM_{2.5} at the Oude Langendijk.

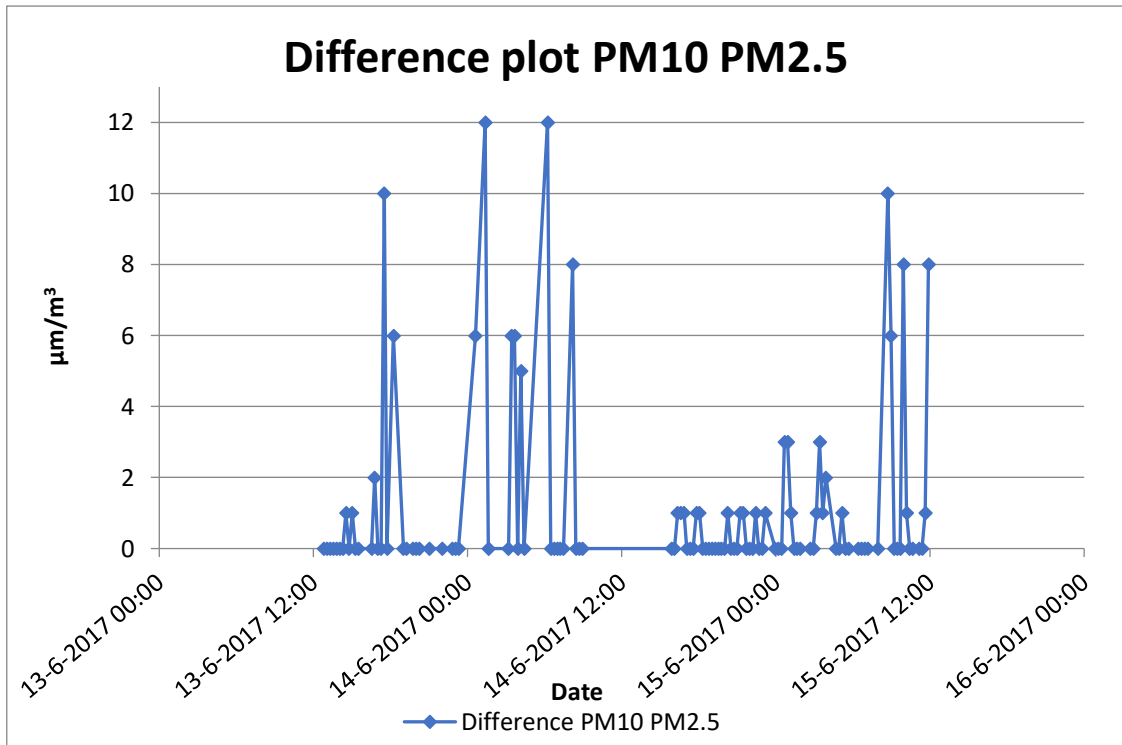


Figure 68: Difference plot PM10 and PM2.5 (PM10 values shown)

The graph in figure 69 shows the difference graph of the particles belonging to PM2.5. PM1 is therefore subtracted from PM2.5. The difference plot shows that only in 3 measurements more PM1 than PM2.5 was measured. It can be concluded that PM2.5 particles are the most present in the air at the Oude Langendijk.

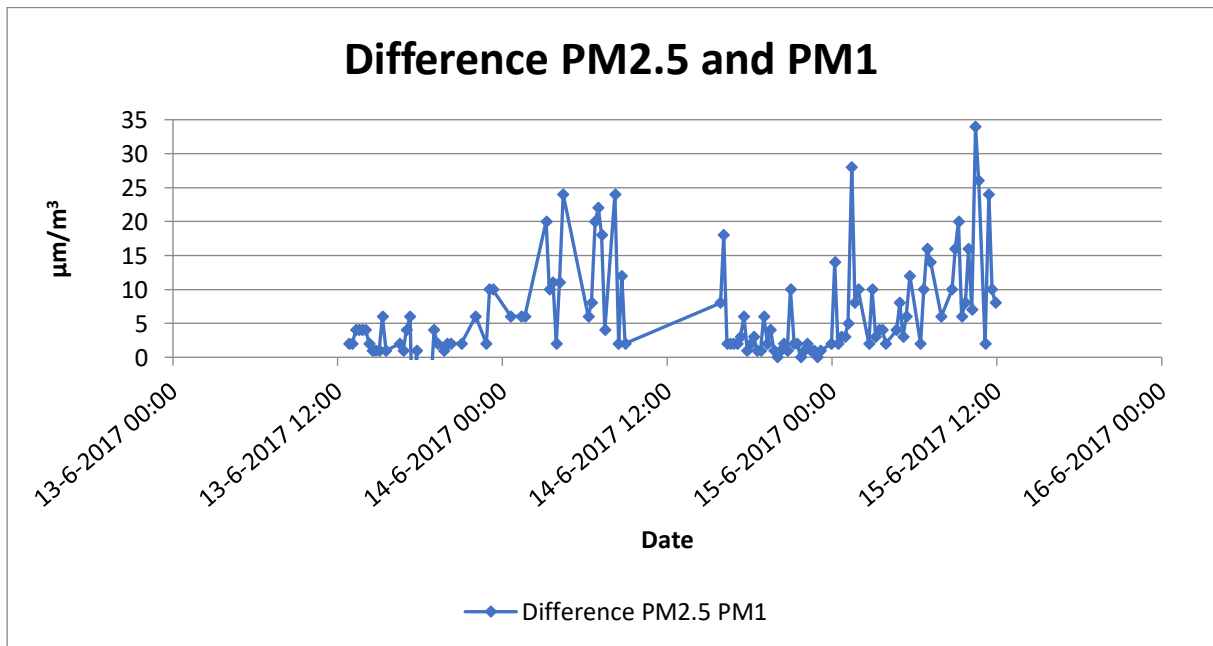


Figure 69: Difference plot PM2.5 and PM1 (PM2.5 values shown)

The graph of figure 70 shows the PM1 values. It is shown that the PM1 values are going up during the day: this can be an effect of the vehicles.

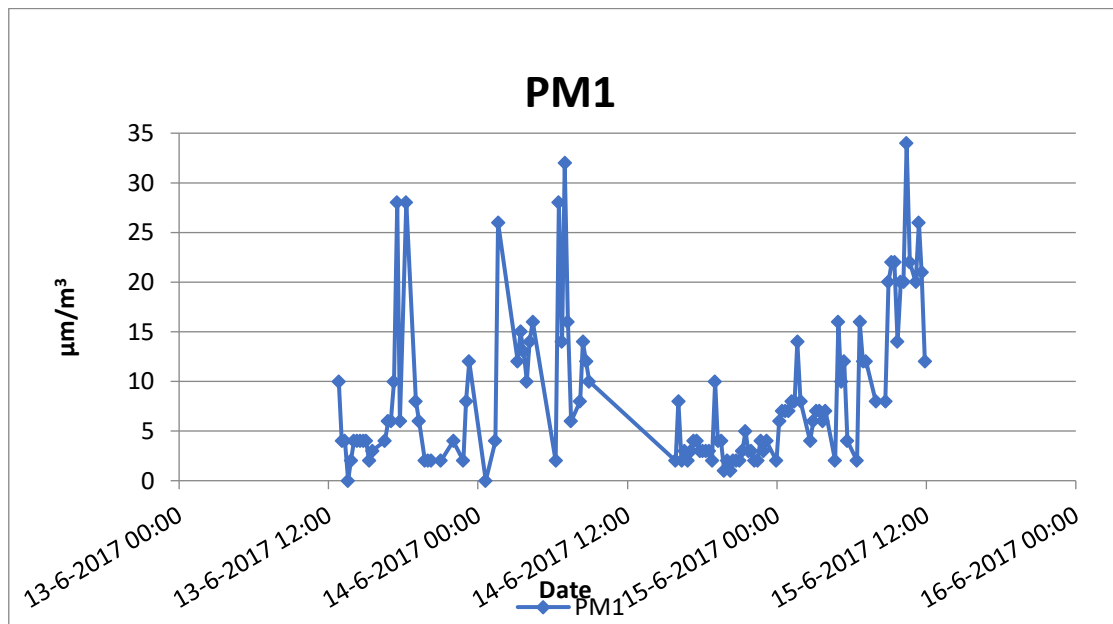


Figure 70: PM1 values

Choorstraat compared to Oude Langendijk

The PM values measured in the Choorstraat increase during the night of 14 to 15 June as can be seen in figure 71. On June 15th a peak is shown at 19.45 o'clock in the evening. Another increase of PM levels can be seen from 15 to 16 June. An outlier can be seen at the 14th of June at 19.45 o'clock in figure 71.

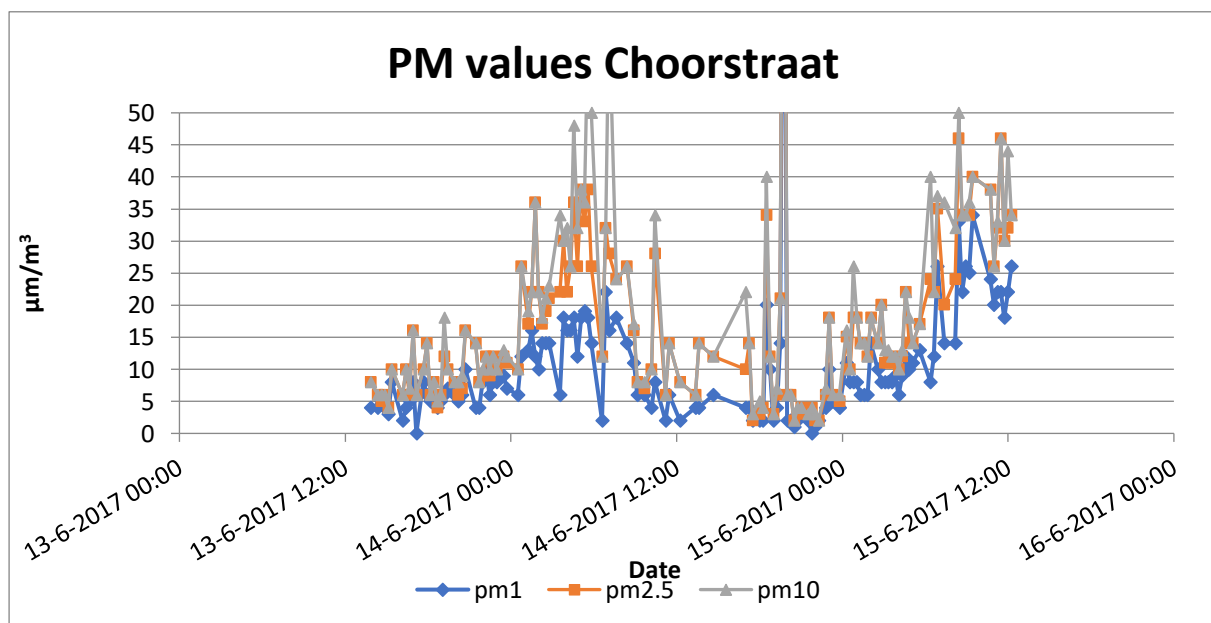


Figure 71: PMS measurements Choorstraat

When comparing the PM values of the Choorstraat to the Oude Langendijk, the assumption was that the Oude Langendijk would have higher PM values than the Choorstraat. In figure 72, 73 and 74 the difference plots of PM1, PM2.5 and PM10 values of the two streets are compared with each other. The graphs show no clear evidence of the Oude Langendijk being more polluted than the Choorstraat.

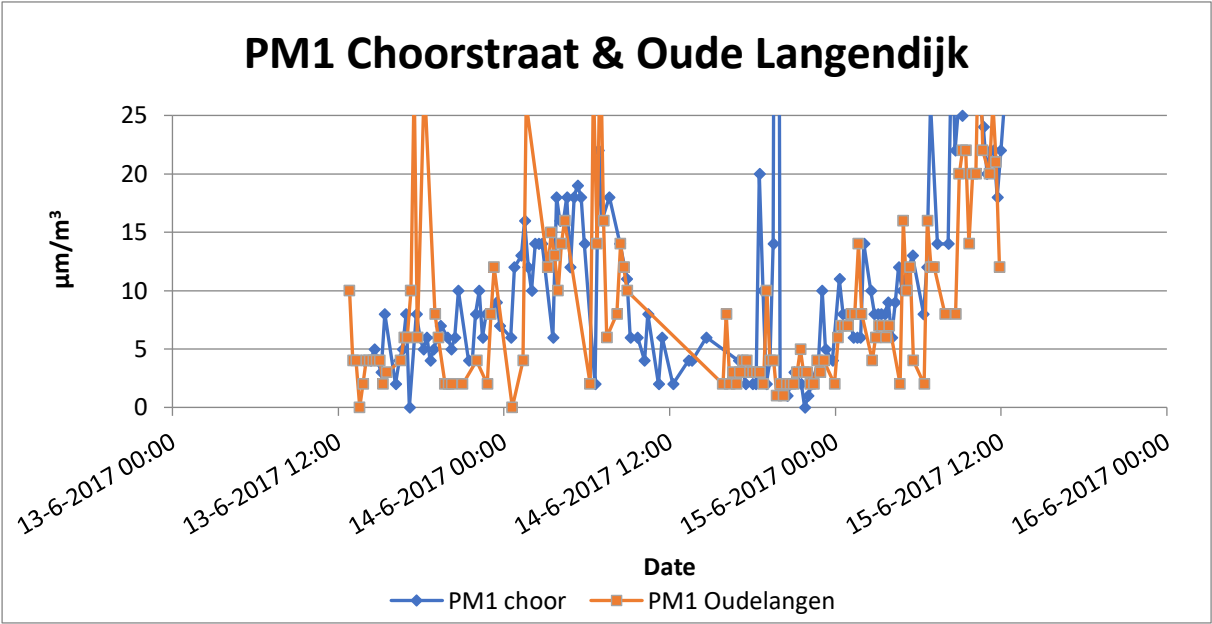


Figure 72: PM1 measurements Choorstraat and Oude Langendijk

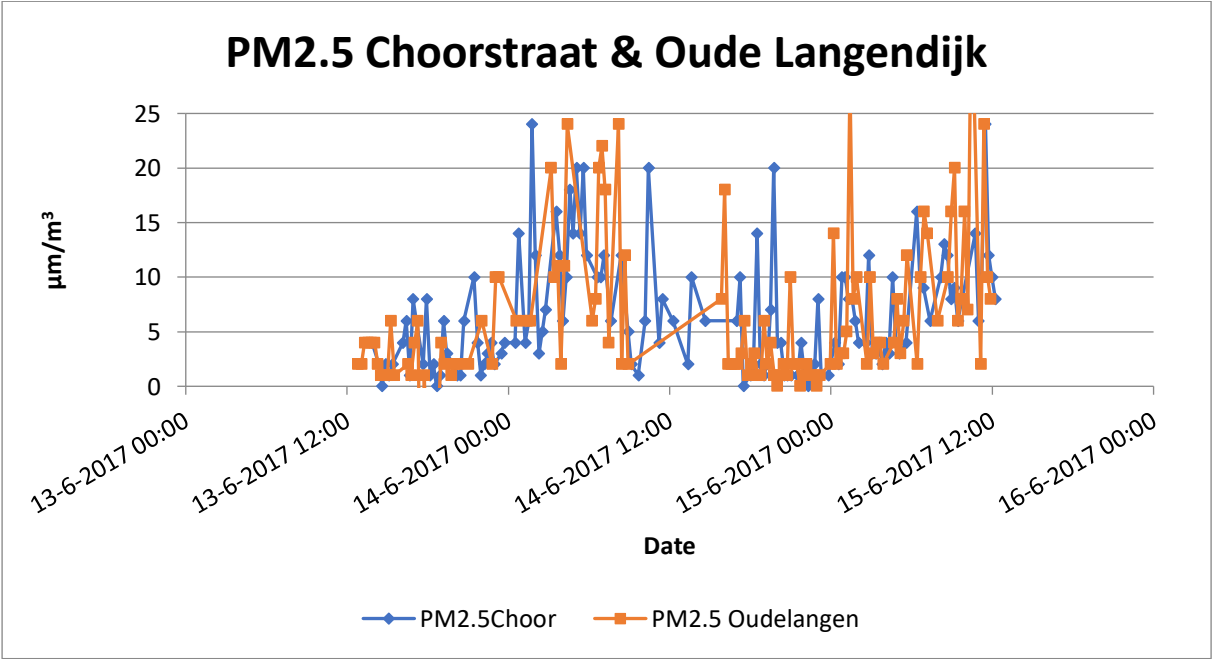


Figure 73: PM2.5 measurements Choorstraat and Oude Langendijk

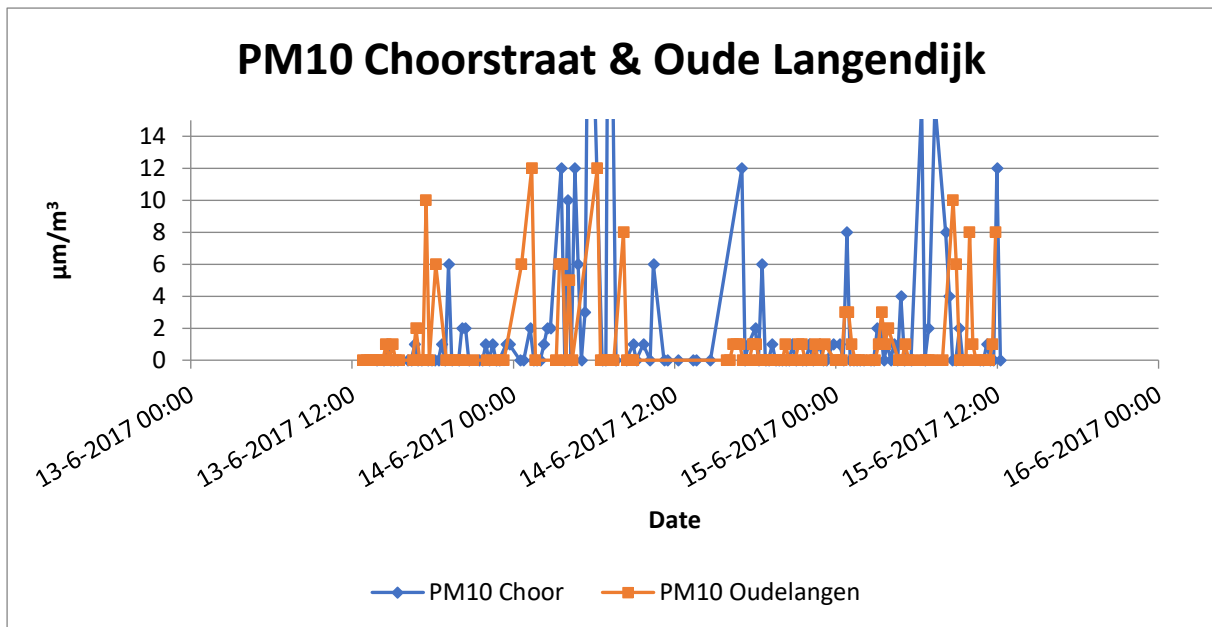


Figure 74: PM10 measurements Choorstraat and Oude Langendijk

Consistency of the PMS5003 sensor

The data manual of the sensor pointed out that after 500 hours of storage in high temperature (70 °C) and humidity (90-95%) or cold storage (-30 °C), the sensors has to be checked for consistency (Yong, 2016). Taking into account that the sensors were delivered after six weeks, it is unknown what circumstances the sensor had to endure. Therefore, also storage could have been of influence on the performance of the PMS5003 sensor. In addition the sensor is meant to be used for indoor use. In this project the sensors where used in outdoor environments. Meteorological conditions can therefore have influence on the measurements.

These remarks, plus the fact that the data is collected for only two days, result in the conclusion that these measurements with the PMS5003 are not reliable enough to be able to answer the question posed by Peter (paragraph 2.2.1).

4.4.3 Measurements with the PPD42NS

The PPD42NS sensor is also meant for indoor use. The test with this sensor, described in chapter 3.1.1, made it clear that the measurements of the PPD42NS sensor were heavily influenced by wind. Therefore the measurements conducted by the PPD42NS sensor are not used in the data analyses. See figure 75 for a plot of the values with the PPD42NS: they differ too much to conclude anything about it.

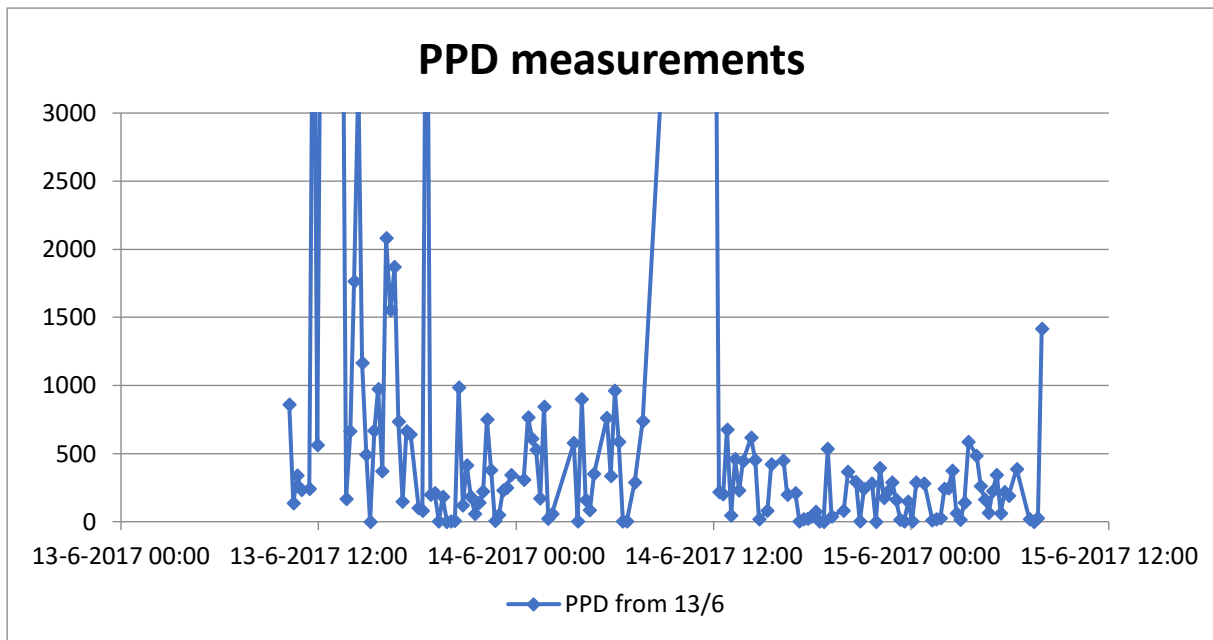


Figure 75: Measurements with PPD42NS in Oude Langendijk

4.4.4 Conclusion air quality

In conclusion, it can be stated that the measured data from both the PMS5003 and the PPD42NS sensor is not sufficient for making assumptions about the causes of the increase and decrease of the PM values at the Oude Langendijk in order to provide an answer to the story. More measurements have to be conducted over a longer period of time. In addition, there should be more detailed information about the meteorological situation at the Oude Langendijk to be able to draw conclusions. It therefore can not be concluded that the buses and scooters are of influence on the air quality at the Oude Langendijk.

Chapter 5 - Conclusion

The IoT world is growing rapidly and therefore research on this topic is done. In the introduction the research question is defined as:

How to raise local environmental awareness by interactively providing environmental data using a static sensor network?

Case study: a pilot in the city center of Delft

To answer this research question six sub questions are defined and will be answered in the following paragraph.

1. How to make a static sensor network interactive?

To increase the awareness of citizens and pedestrians the static sensor network has been made interactive. Two options that are implemented to reach this are a twitterbot that automatically replies with data of the sensors on Twitter and a website with graphs of live data. To make sure the target group is aware of this interactivity a media outreach is done including Facebook, billboards, flyers and articles in local newspapers.

2. What environmental data to collect and how to process it?

For this project noise, air quality, temperature and humidity sensors are chosen. These four datatypes are the most relatable for citizens. Besides that, these type of environmental phenomena are related to health issues. Most stakeholders involved in this project are also interested in this data. The data is processed by making graphs and metadata like mean, minimum and maximum values. These graphs and numbers will be used for analyzing and searching for patterns.

3. What environmental data is to be provided to the citizens to raise awareness?

To raise the awareness of citizens relatable data (like described in sub question two) should be collected and presented. Generally citizens are only interested in the current situation and therefore only the latest data should be presented to them. If interested in more data, the citizen can find this on the website of the project which has historic data.

4. Where to install the sensors?

To ensure a maximum amount of people is being reached, the location should be visited by many citizens and tourists. Likewise, the different areas of interest should have different patterns and

environments to detect differences if present. Therefore the Voldersgracht, Oude Langendijk and Choorstraat are good locations to place the sensors.

The location of the sensors in these streets should be out of reach to prevent vandalism but also visible to support interactivity. In case the sensors are placed on private properties, permission of the owner of the location should be arranged. In this project the sensors are placed on trees or street lanterns in consultation with the Municipality of Delft. For future research it is important to take into account that direct sunlight or sun has a big influence on the measured temperature.

5. How to build the sensor platforms?

Creating a sensor platform starts with deciding on which sensors and other hardware to use. After this the code has to be written that will collect the data through the sensors. Then, a design has to be made for casing and this one has to be build. At last the different elements of hardware and software should be combined and deployed.

6. How to ensure data quality?

Ensuring data quality is important to draw reliable conclusions. At first the sensors themselves need to be reliable and of good quality. They have to be tested and the outcome of the data calibrated. This has to be done in a stable environment with reliable sensors. After this, being critical on the data output is very important; outliers, systematic errors or decay of hardware is still possible. Placing the sensors in a constant location which will not influence the data is important too for good data quality.

These sub questions all together answer the main research question. However concluding on the increase of environmental awareness cannot be answered through this research. Further research on the psychological effect of the implementation of this project has to be done.

Future research

Based on this research, new research can be done to explore the IoT world. Like mentioned above, the psychological effect of implementing a sensor network can be researched. Also it is interesting to increase the scale of the project and research the best way to make measurements of the whole city, for example by using a dynamic sensor network. This can be done by equipping moving vehicles with sensor platforms instead of having them on fixed locations. Comparing different city centers of different cities can also result in interesting findings about the environmental issues of the cities.

Chapter 6 – Recommendations

The Sensor City Delft project was a pilot project whereby a sensor network is deployed in the city center of Delft. This chapter provides a list of practical recommendations for students and other researchers who also want to deploy a similar sensor network.

Hardware

- Solder very careful, one faulty wire can shut down the platform.
- The PPD42NS sensors continuously use power, switch them off with transistors.
- Make sure your noise sensor measures decibels, don't use microphones for measuring decibel levels.
- The LoRa antenna wires are fragile, be careful not to break the wiring by twisting them too much.
- Take enough time for calibration and testing of sensors: in a controlled environment to make sure that the sensors are sensing correctly.
- Use a windmeter (direction and speed) together with the PMS5003 and PPD42NS air dust sensors, to be able to see the relation between wind and particles in the air.
- Extensively check LoPy's network settings. By default it is on USA instead of NL. When installed at USA, Pycom does not (yet) provide an option to change this to the Netherlands.

Casing

- Have an accessible enclosure box: one will have to make changes and do checks on the platform continuously, and make the replacing of the batteries easily possible.
- If you want to assemble a self-made enclosure, make sure to schedule laser cutting or 3D-printing time. Some groups in the Science Center make a reservations of an entire 08.00-22.00 week.

LoRa

- Understand the KPN LoRa encryption, it requires external tools to decrypt payloads (in Node-RED: node-red-contrib-loradecrypt).
- Don't expect KPN's LoRa network to have reception all the time: take into account the placing in the built environment and check reception beforehand multiple times. Even the locations with reception sometimes have no reception.

- When using a KPN developer account: there are limitations on how many messages can be send. Use a paid account which provides more messages per hour. Or use The Things Network, but then make sure to create LoRa Gateways in the study area.

Software

- Put anti-crashing code in the LoPy (WatchDogTimer). The LoPy might be unstable with your code, the WatchDogTimer makes sure it reboots if the LoPy becomes unresponsive.
- Implement exception for PMS5003 sensor. When Data 7 until Data 12 are 0 the PM values are unreliable: then do the measurement again.
- Have noise measure a couple of times, do it a couple of times during an interval for more accurate readings (for example: sample time = 50 ms, measure a couple of times, save values in a list, get minimum, maximum, average and median).

References

- Adafruit, 2015. *Adafruit AGC Electret Microphone Amplifier – MAX9814, Wiring and Test*. Accessed at 17-05-2017 via <https://learn.adafruit.com/adafruit-agc-electret-microphone-amplifier-max9814/wiring-and-test>.
- Aerosemi Technology Co, 2017. MT3608 High Efficiency 1.2 MHz 2A Step Up Converter. Accessed at 13-06-2017 via <https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf>
- Afzali, A., Rashid, M., Sabariah, B., Ramli, M. (2014) PM10 Pollution: Its Prediction and Meteorological Influence in PasirGudang, Johor. 8th International Symposium of the Digital Earth (ISDE8)
- Agentschap Telecom, 2014. *Vergunningsvrije radiotoepassingen*. Accessed at 19-05-2017, via <https://www.agentschaptelecom.nl/sites/default/files/brochure-vergunningsvrije-radiotoepassingen.pdf>.
- Agile Business Consortium, 2017. *MoSCoW prioritisation*. Accessed on 04-05-2017, via <https://www.agilebusiness.org/content/moscow-prioritisation>.
- Allen, T. 2013. *De-construction of the Shinyei PPD42NS dust sensor*. EME Systems LLC. Accessed at 13-06-2017, via http://takingspace.org/wp-content/uploads/ShinyeiPPD42NS_Deconstruction_TracyAllen.pdf
- Arduino Corporation, 2017. *Arduino & Genuino products, Arduino & Genuino UNO*. Accessed at 19-05-2017, via <https://www.arduino.cc/en/main/arduinoBoardUno>.
- Aref, M. & Sikora, A. 2014. Free Space Range Measurements with Semtech LoRa™ Technology, *IEEE International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems*. 2nd edition, pp.19-23.
- Babar, S., Mahalle, P., Stango, A., Prasad, N., & Prasad, R., 2010. Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT), *Springer-Verlag Berlin Heidelberg*. In: Meghanathan, N. et al. (Eds.): CNSA 2010, CCIS 89, pp. 420–429.
- Cenedese, A., Zanella, A., Vangelista, L. & Zorzi, M., (2014). Padova Smart City: An urban Internet of Things experimentation, *IEEE Internet Things J.*, vol. 1, no. 1.
- Chen, S., Xu, H., Liu, D., Hu, B. & Wang, H., 2014. A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective. *IEEE Internet Things J.*, vol. 1, no. 4.
- Chung, L. & Do Prado Leite, J.C.S. 2009. On Non-Functional Requirements in Software Engineering, *Lecture Notes in Computer Science*, volume 5600, pp. 363-379.

Crump, J. & Brown, I. 2013. The societal impact of internet of things. Workshop report by Chartered Institute for IT, 14 February 2013. Accessed via <http://www.bcs.org/upload/pdf/societal-impact-report-feb13.pdf>

European Parliament and of the Council, 2008. Directive 2008/50/EC, *Official Journal of the European Union*, (152) pp 1-44.

EPA (Environmental Protection Agency), 2017. *PM10, PM2.5, PM1 particles*. Accessed at 19-06-2017, via <https://www.epa.gov/particle-pollution-designations>

Ericsson AB, 2016. *Ericsson Mobility Report: on the pulse of the networked society*. Accessed via <https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf>

Fonollosa, J. 2016. *Test of Particle Sensor (PPD42NS, Shinyei) and integration with Arduino boards*. Accessed on 13-06-2017, via https://jordifonollosa.files.wordpress.com/2015/01/dust_sensor.pdf

Gartner, 2014. *Gartner Says the Internet of Things Will Transform the Data Center*. Press release, Gartner: Stamford. Retrieved from <http://www.gartner.com/newsroom/id/2684616>.

General Electronics Battery Co., 2017. *GEB905085 Lithium Polymer Battery, 3.7V 5500mAh lipo battery*. Accessed on 26-05-2017 via http://www.chinaseniorsupplier.com/Electrical_Equipment_Supplies/Batteries/60442165175/rechargeable_905085_3_7V_5500mAh_LED_Li_Po_batetry_with_high_cost_effective.html

George, D.P. & Sokolovsky, P. (and contributors), 2017. *MicroPython Tutorial for ESP8266 -Temperature and humidity*. Accessed at 19-05-2017, via <https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/dht.html>.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7), pp. 1645-1660.

Guo, B., et. al. (2013). Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things. *Journal of networks and computer applications*, vol. 36, Issue 6, pp. 1531-1539.

Haines, M.M., Stansfeld, S.A., Job, R.F.S., Berglund, B. & Head, J. (2001). Chronic aircraft noise exposure, stress responses, mental health and cognitive performance in school children. *Psychological Magazine*, volume 31, pp. 265 – 277.

Hu, Y. & C., Wang, H.J. (2006). Location Privacy in Wireless Networks. In: Proceedings of the ACM SIGCOMM.

Jol, M. (2016a). *Lora Decryption on Application Server*. Forum post on <https://zakelijkforum.kpn.com/lora-forum-16/lora-decryption-on-application-server-8416>.

Jol, M. (2016b). Encryption of LoRa messages. Forum post on <https://zakelijkforum.kpn.com/lora-forum-16/encryption-of-lora-messages-8321>.

KPN, 2016. Beyond Telecom, LoRa. Accessed on 29-04-2017 via <http://www.loranode.com/LoRa/>.

KPN, 2017. *KPN LoRa, advanced workshop*, by Michiel Jol. Restricted access – contact authors.

Law on Noise Pollution, article 82. Accessed on 29-04-2017 via <http://wetten.overheid.nl/BWBR0003227/2016-04-14#HoofdstukVI>.

Lee, I., Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, vol. 58, pp. 431–440.

Liu, year unknown. *Temperature and humidity module. AM2302 product manual*. Accessed on 19-05-2017, via <http://akizukidenshi.com/download/ds/aosong/AM2302.pdf>.

LoRa Alliance™, 2017. *LoRa Alliance™ Technology*. Accessed at 19-05-2017, via <https://www.lora-alliance.org/What-Is-LoRa/Technology>.

Maxim Integrated, 2016. *MAX9814*. Accessed at 15/05/2017, via <https://datasheets.maximintegrated.com/en/ds/MAX9814.pdf>.

MicroPython, 2017. *MicroPython Libraries, class I2C – a two-wire serial protocol*. Accessed at 17-05-2017 via <http://docs.micropython.org/en/latest/wipy/library/machine.I2C.html>.

OGC 2017a, SWE standards, <http://www.opengeospatial.org/domain/swe#standards>

OGC 2017b, Sensor Observation Service (SOS), OGC 2017, <http://www.opengeospatial.org/standards/sos>

OGC 2017c, Sensor Web Enablement (SWE), <http://www.opengeospatial.org/ogc/markets/technologies/swe>

Petäjärvi, J., Mikhaylov, K., Roivainen, A., Hänninen, T. & Pettissalo, M. 2015. On the Coverage of LPWANs: Range Evaluation and Channel Attenuation Model for LoRa Technology. *International Conference on ITS Telecommunications (ITST)*, 14th edition, pp 55-59.

Pycom, 2016. *Docs, Tutorials and examples, One-wire driver*. Accessed at 17-05-2017, via https://docs.pycom.io/pycom_esp32/pycom_esp32/tutorial/includes/onewire.html.

Pycom, 2017. *LoPy 1.0*. Accessed at 19-05-2017, via <https://www.pycom.io/wp-content/uploads/2016/12/lopySpecsheet.pdf>.

RaspberryPi, 2017. *Raspberry Pi 3 Model B*. Accessed at 19-05-2017, via <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.

Sharp Corporation, 2006. *Sharp GP2Y1010AU0F, Compact Optical Dust Sensor*. Accessed at 19-05-2017, via https://www.sparkfun.com/datasheets/Sensors/gp2y1010au_e.pdf.

Smart-cities.eu, 2015. *European Smart Cities 4.0 (2015)*. Accessed at 21-05-2017, via <http://smart-cities.eu/?cid=01&ver=4>.

Sanchez, L., Muñoz, L., Galache, J.A., Sotresa, P., Santana, J.R., Gutierrez, V., Ramdhany, R., Gluhak, A., Krco, S., Theodoridis & E., Pfisterer, D. (2013). SmartSantander: IoT experimentation over a smart city testbed. *Computer Networks*, Volume 61, pp. 217–238.

Seeed, 2017. *Grove – dust sensor*. Accessed at 13-06-2017, via http://wiki.seeed.cc/Grove-Dust_Sensor/

Stankovic, J.A. (2014). Research directions for the Internet of Things, *IEEE Internet Things J.*, vol. 1(1), pp. 3–9.

ST Microelectronics, 2013. *LD1117A - Low drop fixed and adjustable positive voltage regulators*. Accessed on 14-06-2017, via <http://www.findic.us/ld1117av33-datasheet-pdf-en-pejZBaxle.html>

Tan, D. (2014). *Testing the Shinyei PPD42NS*. Accessed at 13-06-2017, via <http://irq5.io/2013/07/24/testing-the-shinyei-ppd42ns/>

The Things Network, (2017). *Limitations: data rate, packet size, 30 seconds uplink and 10 messages downlink per day Fair Access Policy* (forum post). Accessed via: <https://www.thethingsnetwork.org/forum/t/limitations-data-rate-packet-size-30-seconds-uplink-and-10-messages-downlink-per-day-fair-access-policy/1300>.

Twitter Developer Documentation, 2017. *API Overview*. Accessed via: <https://dev.twitter.com/overview/api>

Vecchi, R., Marazzan, G., Valli, G., Ceriani, M., C, Antoniazzi, C. (2004). The role of atmospheric dispersion in the seasonal variation of PM1 and PM2.5 concentration and composition in the urban area of Milan (Italy). *Atmospheric Environment*. Volume 38, Issue 27, p. 4437-4446

Verbree, E. 2017. *Presentation 'Project Management'*. Presentation for the GEO1101 Synthesis Project at TU Delft.

Vormetric, 2016. *Vormetric Data Threat Report, Trends in Encryption and Data Security*. Accessed at 20-05-2017, via http://enterprise-encryption.vormetric.com/rs/480-LWA-970/images/Vormetric_2016_Data_Threat_Report_Global_WEB.pdf

Wang, J., Ogawa S. (2015) Effects of Meteorological Conditions on PM2.5 Concentrations in Nagasaki, Japan. *International Journal of Environmental Research and Public Health* ISSN 1660-4601 p.9090-9101

Yong, Z. 2016. Digital Universal Particle Concentration Sensor, v2.3. Accessed at 17-05-2017 via http://www.aqmd.gov/docs/default-source/aq-spec/resources-page/plantower-pms5003-manual_v2-3.pdf?sfvrsn=2.

Zanella, A. & Vangelista, L. (2014). Internet of Things for Smart Cities. *IEEE Internet Things J.*, vol. 1(1).

Zhong, N. et. al. (2013). Research challenges and perspectives on Wisdom Web of Things (W2T). *Springer, The Journal of Supercomputing*, Volume 64, Issue 3, pp 862–882.

List of figures

| | |
|--|----|
| Figure 1: Aggregated workflow | 15 |
| Figure 2: Project Logic Diagram | 16 |
| Figure 3: Personality of Peter | 24 |
| Figure 4: Personality of Eline | 25 |
| Figure 5: Personality of Richard | 26 |
| Figure 6: Flow of the technical system | 27 |
| Figure 7: PMS5003 air quality sensor (source: Yong, 2016) | 28 |
| Figure 8: PMS5003 schematic overview (left) and circuit (right) (source: Yong, 2016) | 28 |
| Figure 9: PMS5003 data output (source: Yong, 2016) | 30 |
| Figure 10: Plot of the PMS5003 test, inside, 'clean' air | 31 |
| Figure 11: Plot PMS5003 test, inside, 'dirty' air | 32 |
| Figure 12: Plot PMS5003 test, outside | 34 |
| Figure 13: Sharp Dust Sensor | 35 |
| Figure 14: Sharp Dust Sensor connection to LoPy (source: Sharp Corporation, 2016) | 35 |
| Figure 15: PPD42NS air dust sensor | 36 |
| Figure 16: LDD1117AV33 voltage divider (source: ST Microelectronics, 2013) | 37 |
| Figure 17: PPD42NS tests plot | 39 |
| Figure 18: MAX9814 sound sensor | 40 |
| Figure 19: Schematic overview of the MAX9814 sound sensor (source: Adafruit, 2016) | 40 |
| Figure 20: Graph of test measurements with the MAX9814 | 42 |
| Figure 21: AM2302 temperature and humidity sensor | 43 |
| Figure 22: Maximum temperature error (source: Liu, year unknown) | 44 |
| Figure 23: AM2302 pin assignment (source: Liu, year unknown) | 44 |
| Figure 24: LoRa modulation (source: KPN, 2017) | 46 |
| Figure 25: LoRaWAN SF characteristics (source: KPN, 2017) | 47 |
| Figure 26: the default folder structure in the flash folder of LoPy | 51 |
| Figure 27: the folder structure in the flash folder of LoPy for each LoRa platform | 51 |
| Figure 28: Workflow of main.py (PMS5003 on the left and PPD42NS on the right) | 53 |
| Figure 29: Solar panel system design | 56 |
| Figure 30: Charging the LoPy with solar panels | 57 |
| Figure 31: perfboard, with headers and a LoPy | 57 |
| Figure 32: Schematic overview of the sensor platform | 58 |
| Figure 33: Deployed Sensor and Billboard at the Oude Langendijk (left) and Voldersgracht (right) | 60 |
| Figure 34: Different views of a casing design concept | 62 |

| | |
|--|----|
| Figure 35: Final casing in 2 colors (left image). Ventilation (right image) | 64 |
| Figure 36: Cable box (left in image) and a lunch box (right in image) as inner boxes | 64 |
| Figure 37: Database system visualized | 66 |
| Figure 38: User Interface of Node-RED environment | 69 |
| Figure 39: An overview of the workflow in Node-RED | 69 |
| Figure 40: HTTP request to KPN gateway | 70 |
| Figure 41: HTTP response and function block for extracting data | 70 |
| Figure 42: Function blocks for SQL queries | 70 |
| Figure 43: Twitterbot responds to the user | 72 |
| Figure 44: Node-RED flow of twitterbot | 72 |
| Figure 45: Welcome page of the project website (www.scdelft.nl) | 74 |
| Figure 46: Dashboard on www.scdelft.nl | 74 |
| Figure 47: Historic data on www.scdelft.nl | 75 |
| Figure 48: Locations of sensors in the City center of Delft | 76 |
| Figure 49: Location of sensor 073 (left) and location of sensor 210 (right) at the Voldersgracht | 77 |
| Figure 50: Voldersgracht in SunCalc | 77 |
| Figure 51: Location of the sensor 981 (left) in Choorstraat. The sensor is covered by leaves of the tree (right) | 78 |
| Figure 52: Choorstraat in SunCalc | 78 |
| Figure 53: Location of sensor 772 (left) and location of sensor 66D (right) at Oude Langendijk | 78 |
| Figure 54: Oude Langendijk in SunCalc | 79 |
| Figure 55: Graph of temperature at the Voldersgracht | 80 |
| Figure 56: Graph of temperature at Oude Langendijk | 81 |
| Figure 57: Graph of temperature at Choorstraat | 82 |
| Figure 58: Graph of humidity at Voldersgracht | 83 |
| Figure 59: Graph of humidity at Oude Langendijk | 84 |
| Figure 60: Graph of humidity at Choorstraat | 85 |
| Figure 61: Temperature of all five sensors (note: the y-axis starts with 10 degrees C) | 86 |
| Figure 62: Humidity of all five sensors (note: the y-axis starts with 20%) | 87 |
| Figure 63: Graph of noise values from sensors in Voldersgracht | 91 |
| Figure 64: Graph of PMS measurements at Oude Langendijk, 13, 14 and 15 June 2017 | 93 |
| Figure 65: Humidity at Oude Langendijk | 93 |
| Figure 66: Humidity and PM levels at Oude Langendijk | 94 |
| Figure 67: Wind speeds on the 13th and 14th of June | 95 |
| Figure 68: Difference plot PM10 and PM2.5 (PM10 values shown) | 96 |
| Figure 69: Difference plot PM2.5 and PM1 (PM2.5 values shown) | 96 |
| Figure 70: PM1 values | 97 |
| Figure 71: PMS measurements Choorstraat | 97 |
| Figure 72: PM1 measurements Choorstraat and Oude Langendijk | 98 |

| | |
|---|-----|
| Figure 73: PM2.5 measurements Choorstraat and Oude Langendijk | 98 |
| Figure 74: PM10 measurements Choorstraat and Oude Langendijk | 99 |
| Figure 75: Measurements with PPD42NS in Oude Langendijk | 100 |
| Figure 76: APPENDIX A - Organogram of the Organizational Breakdown Structure | 117 |
| Figure 77: APPENDIX B - Work Breakdown Structure | 119 |
| Figure 78: APPENDIX D - GANTT chart of the project | 122 |
| Figure 79: APPENDIX D - The aggregated workflow | 127 |
| Figure 80: APPENDIX E - Using Twitter as a feedback mechanism | 130 |
| Figure 81: APPENDIX E - news article about the Sensor City Delft project on AD.nl/delft | 132 |
| Figure 82: APPENDIX E - news article about the Sensor City Delft project in Delft op Zondag | 133 |
| Figure 83: APPENDIX L - Possible locations on trees | 154 |
| Figure 84: APPENDIX L - Possible locations in Voldersgracht | 156 |
| Figure 85: APPENDIX L - Possible locations in Oude Langendijk | 159 |
| Figure 86: APPENDIX M – Casing | 161 |
| Figure 87: APPENDIX N - Sensor City Delft system overview | 162 |
| Figure 88: APPENDIX P - Sensor billboard | 165 |
| Figure 89: APPENDIX Q - Flyers for the Sensor City Delft project | 166 |
| Figure 90: APPENDIX R - Node-RED flow of twitterbot | 167 |
| Figure 91: APPENDIX O - Data plotting interface in Node-RED | 172 |
| Figure 92: APPENDIX R - Dashboard flow in Node-RED | 173 |
| Figure 93: APPENDIX S - Device information KPN Developer Portal (screenshot from KPN website) | 174 |
| Figure 94: APPENDIX S - Decryption scheme example KPN LoRa message (Source: Jol, 2016a) | 175 |
| Figure 95: APPENDIX X: Node-RED flow of receiving and parsing LoRa messages | 215 |

List of tables

| | |
|--|-----|
| Table 1: PMS5003 pins and connections | 29 |
| Table 2: PMS5003 data calibration, inside, 'clean' air | 31 |
| Table 3: Plot of the PMS5003 test, inside, 'dirty' air | 32 |
| Table 4: Plot of the PMS5003 test, outside | 33 |
| Table 5: PPD42 tests | 38 |
| Table 6: AM2302 tests, five locations | 45 |
| Table 7: AM2302 calibration | 45 |
| Table 8: Comparison of Pycom LoPy, Arduino Uno, and Raspberry Pi microcontrollers | 46 |
| Table 9: KPN and The Things Network LoRa comparison | 48 |
| Table 10: components for the solar panels system | 55 |
| Table 11: casing requirements | 63 |
| Table 12: Database table with device addresses, measurements and timestamps | 67 |
| Table 13: Database table raw data, see Appendix L for more information | 67 |
| Table 14: Table with temperature values | 86 |
| Table 15: Table with humidity values | 87 |
| Table 16: Noise data for the three selected streets and during specific time slots | 89 |
| Table 17: Noise data of Voldersgracht during specific time slots | 90 |
| Table 18: Mean, maximum and minimum of PM10, PM2.5 and PM1 | 92 |
| Table 19: APPENDIX O - LoRa JSON abbreviations | 164 |

List of maps

| | |
|---|----|
| Map 1: The Things Network coverage in Delft (source: The Things Network, 2017 (edited)) | 49 |
| Map 2: The Things Network coverage test in the study area | 49 |
| Map 3: KPN LoRa connectivity test in the study area | 50 |
| Map 4: The study area in red, scale 1:2381 | 59 |

List of abbreviations

| | |
|---------------|---|
| ADR | Adaptive Data Rate |
| API | Application Programming Interface |
| AppSKey | Application Session Key |
| CF | Cubic Foot |
| COP | Common Operating Picture |
| CSS | Chirp Spread Spectrum |
| DevEUI | End Device Unique Identifier |
| DID | Deliverables Items Descriptions |
| DSL | Digital Subscriber Line |
| GIS | Geographic Information System |
| GNSS | Global Navigation Satellite System |
| GPS | Global Positioning System |
| HTTP | HyperText Transfer Protocol |
| ICT | Information and Communication Technology |
| IDE | Integrated Development Environment |
| IEEE | Institute of Electrical and Electronics Engineers |
| IOT | Internet Of Things / Internet Of Everything |
| ISM bands | Industrial, Scientific and Medical bands |
| JSON | JavaScript Object Notation |
| KPN | Koninklijke PTT Nederland |
| LED | Light-Emitting Diode |
| LoRa | Low Range (Low Power) |
| LoRaWAN | Long Range Wide Area Network |
| LPO | Lo Pulse Occupancy |
| MHz | Megahertz |
| MSc Geomatics | Master of Science Geomatics for the Built Environment |
| NwksKey | Network Session Key |
| OBS | Organizational Breakdown Structure |
| OGC | Open Geospatial Consortium |
| PM | Particulate Matter |
| POI | Point of Interest |
| QR-code | Quick Response code |
| REPL | Read-Evaluate-Print-Loop |
| RGB | Red Green Blue |
| SF | Spreading Factor |

| | |
|----------|---|
| SQL | Structured Query Language |
| SSH | Secure Shell |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| Telnet | TELEtype NETwork |
| TU Delft | Delft University of Technology |
| UART | Universal Asynchronous Receiver/Transmitter |
| WBS | Work Breakdown Structure |
| WPD | Work Package Description |
| MoSCoW | Must have, Should have, Could have, Will not have |
| UART | Universal Asynchronous Receiver/Transmitter |
| Wi-Fi | Wireless Fidelity |

Appendices

Appendix A: Organizational Breakdown Structure

In this research project several tasks and responsibilities are defined. These tasks and responsibilities are fitted into different roles. Every member of the team is assigned to one or more roles for which he or she is responsible or will assist. In this way, it is ensured that every member will be able to contribute equally to the project. The Organizational Breakdown Structure (OBS) in figure 44 shows the communication lines in this project. For every role a short description is provided in the next section.

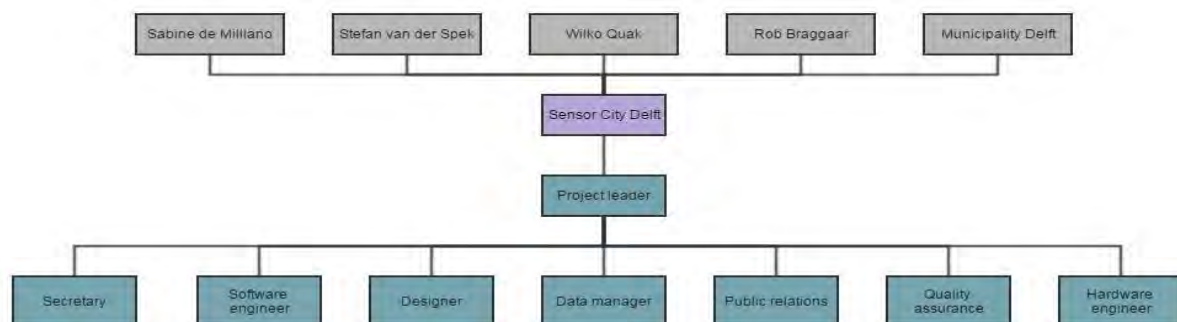


Figure 76: APPENDIX A - Organogram of the Organizational Breakdown Structure

Chairman (*Roeland Meulmeester*)

- Responsible for managing the overall process of Sensor City Delft
- Ensures the project stays on schedule
- Chair the meetings

Secretary (*Noortje Vaissier*)

- Keeps track of the agenda every week
- Sets the minutes during meetings

Software Engineer (*Nebras Salheb*)

- Responsible for making sure the code is working on all computers

Designer (*Gina Michailidou assisted by Noortje Vaissier*)

- Responsible for the design of the box containing the sensor

- Responsible for the logo and the uniform style of the lay-out of the report, presentation and posters

Data manager (*Cathelijne Kleijwegt assisted by Gina Michailidou*)

- Responsible for maintaining and cleaning the collected data

Public relations (*Cathelijne Kleijwegt*)

- Responsible for keeping social media up to date
- Arranges meetings with coaches and clients

Quality assurance (*Niek Bebelaar*)

- Ensures the quality of the written products

Hardware engineer (*Niek Bebelaar*)

- Responsible for the hardware of the sensors

Appendix B: Work Breakdown Structure

In order to provide a clear overview of the different tasks a Work Breakdown Structure (WBS) is constructed. In the WBS the different tasks are identified, this is shown in figure 45.

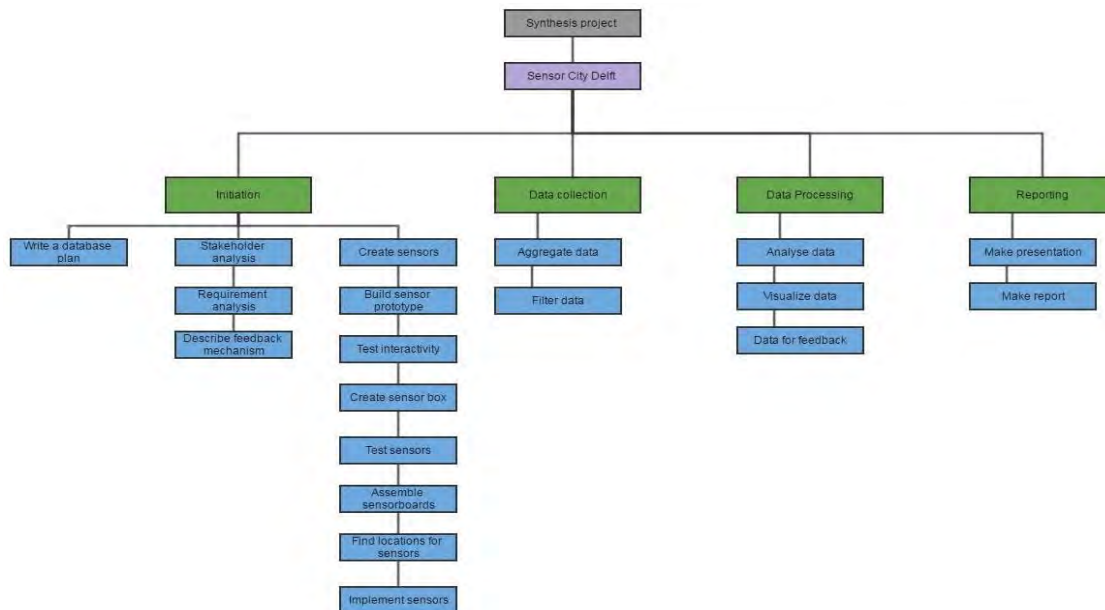


Figure 77: APPENDIX B - Work Breakdown Structure

In this structure the green boxes represent the general processes which include the initiation, data collection, data processing and reporting phases. The initiation phase marks the start of the process. In this phase the stakeholders are contacted and interviewed in order to gain more knowledge about their wishes and needs. A concept for a feedback system which interactively provides information is developed. Furthermore, information about sensors is collected and system requirements are researched. The location of the sensors is determined and the sensors are created and tested before their implementation.

In the data collection phase the data is collected and put into a database. Afterwards, the data will be filtered in order to gain clean data.

In the processing phase the filtered data is used to analyze and visualize it. The information generated from the processing can, then, be used for the feedback system.

The finalization phase includes the finishing of the report and the presentation. The presentation will show all the achievements and results from the project.

Appendix C: Work Package Description (WPD)

This section consist of the phases of the process as shown in the WBS each containing a short description.

Initiation

- Write a database plan
 - o Research into the requirements of the database that is going to be used.
 - o Detect the possibilities and limitations of the database.
- Stakeholder analysis
 - o A stakeholder meeting was organized in the first week of the project.
 - o The stakeholders included: municipality of Delft, citizens and shop owners.
- Requirement analysis
 - o Requirements of the project are described in chapter 4.4.
- Describe feedback mechanism
 - o Conduct research into feedback systems which provide interactive information.
 - o The requirement for a feedback system are determined.
- Create sensors
 - o Conduct research into sensor hardware and into software.
 - o Put the sensors together.
- Build prototype sensor
 - o Build one prototype sensor to test if it works.
- Test interactivity
 - o Test interactive feedback system.
- Create sensor box
 - o Determine requirements for the design of the boxes.
 - o Design boxes in which the sensors will be placed.
- Test sensors
 - o Make multiple sensors and test whether they are fully functional.
 - o Calibrate sensors based on existing sensors in the city.
- Assemble sensor boards

- o Place the sensors in the boxes.
- Find locations for sensors
 - o Conduct research into the places which are suitable for placing sensors.
- Implement sensors
 - o Place the casing with the sensors at the locations chosen.

Data collection

- Aggregate data
 - o Collecting the data in the database.
- Filter data
 - o Remove outliers from the datasets and clean up the data.

Processing

- Analyze data
 - o Conduct analysis on the filtered data.
 - o Generate relevant information from the data.
- Visualize data
 - o Make visualizations of the information to make it understandable for users.
- Data for feedback
 - o Use the generated data and visualized data for feedback.

Reporting

- Make presentation
 - o Compile results and achievements in the final presentation.
- Make report
 - o Finalize the report.

Appendix D: Schedule of the GSP (GANTT chart)

The exact project planning can be found in the GANTT chart.

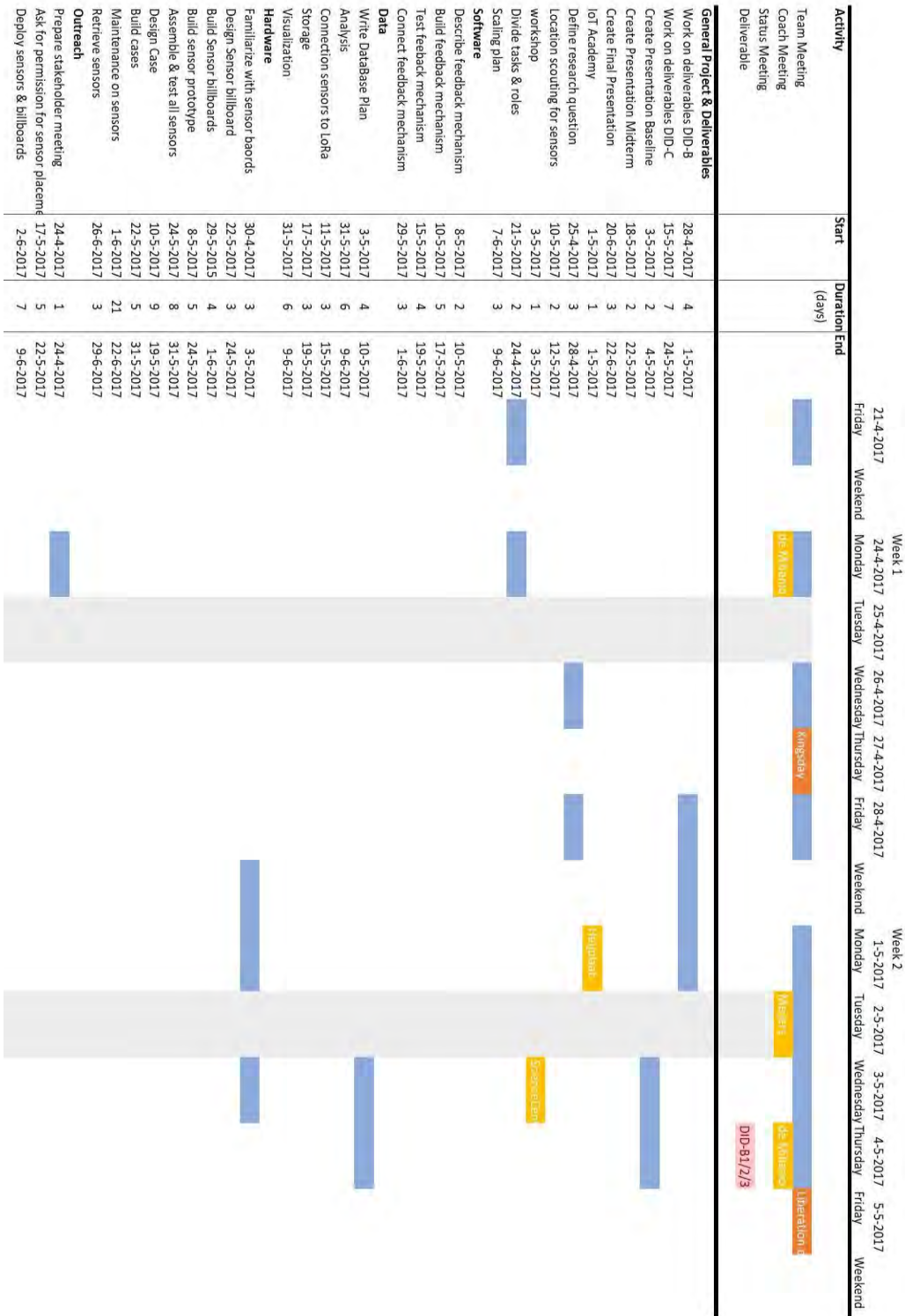
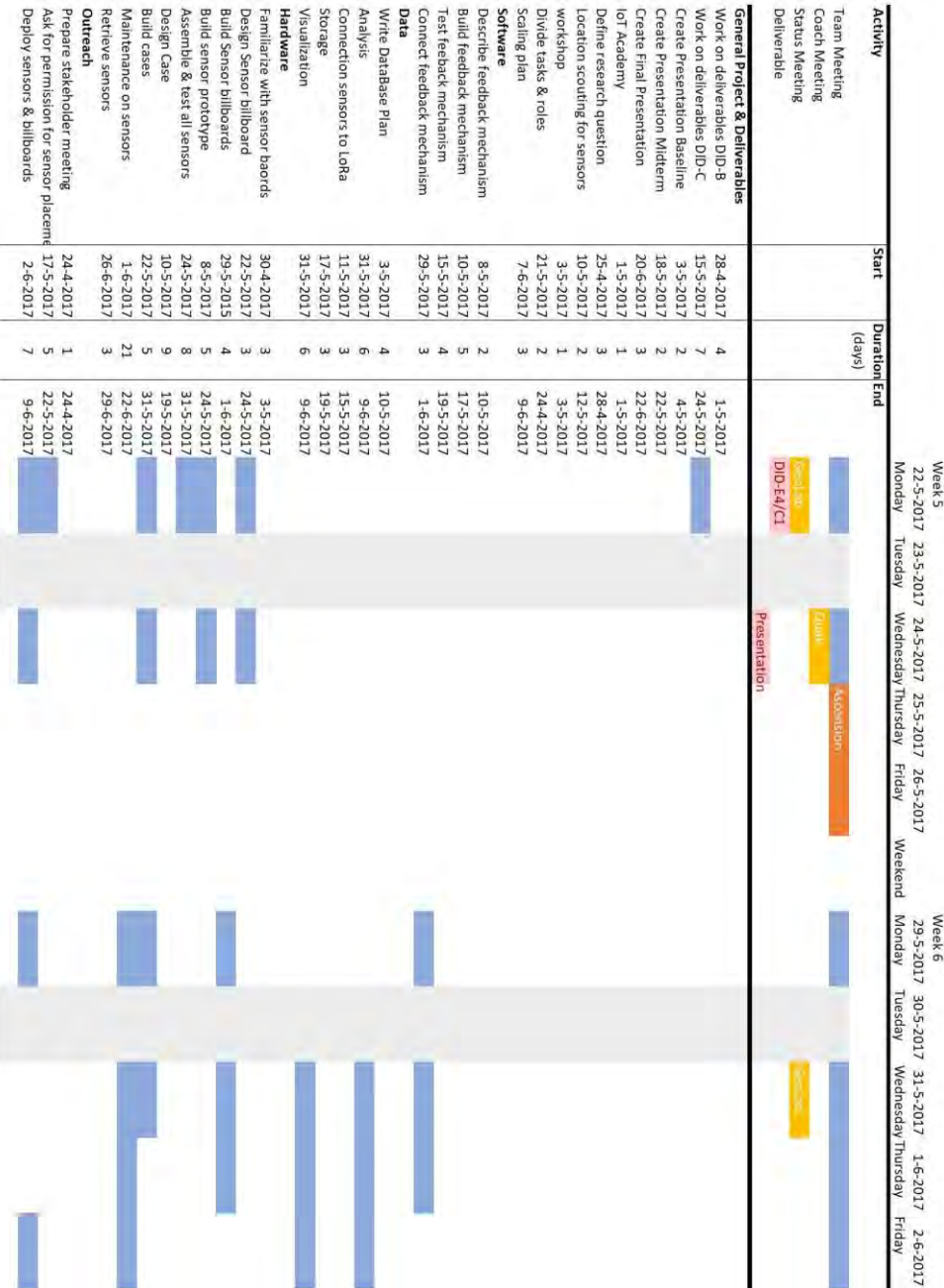


Figure 78: APPENDIX D - GANTT chart of the project

| Activity | Start | Duration (days) | End | Week 3 | | | | | Week 4 | | | | | |
|---|-----------|-----------------|-----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--------|
| | | | | 8-5-2017 | 9-5-2017 | 10-5-2017 | 11-5-2017 | 12-5-2017 | 15-5-2017 | 16-5-2017 | 17-5-2017 | 18-5-2017 | 19-5-2017 | |
| | | | | Monday | Tuesday | Wednesday | Thursday | Friday | Weekend | Monday | Tuesday | Wednesday | Thursday | Friday |
| General Project & Deliverables | | | | | | | | | | | | | | |
| Work on deliverables DID-B | 28-4-2017 | 4 | 1-5-2017 | | | | | | | | | | | |
| Work on deliverables DID-C | 15-5-2017 | 7 | 24-5-2017 | | | | | | | | | | | |
| Create Presentation Baseline | 3-5-2017 | 2 | 4-5-2017 | | | | | | | | | | | |
| Create Presentation Midterm | 18-5-2017 | 2 | 22-5-2017 | | | | | | | | | | | |
| Create Final Presentation | 20-6-2017 | 3 | 22-6-2017 | | | | | | | | | | | |
| IoT Academy | 1-5-2017 | 1 | 1-5-2017 | | | | | | | | | | | |
| Define research question | 25-4-2017 | 3 | 28-4-2017 | | | | | | | | | | | |
| Location scouting for sensors | 10-5-2017 | 2 | 12-5-2017 | | | | | | | | | | | |
| Workshop | 3-5-2017 | 1 | 3-5-2017 | | | | | | | | | | | |
| Divide tasks & roles | 21-5-2017 | 2 | 24-4-2017 | | | | | | | | | | | |
| Scaling plan | 7-6-2017 | 3 | 9-6-2017 | | | | | | | | | | | |
| Software | | | | | | | | | | | | | | |
| Describe feedback mechanism | 8-5-2017 | 2 | 10-5-2017 | | | | | | | | | | | |
| Build feedback mechanism | 10-5-2017 | 5 | 17-5-2017 | | | | | | | | | | | |
| Test feedback mechanism | 15-5-2017 | 4 | 19-5-2017 | | | | | | | | | | | |
| Connect feedback mechanism | 29-5-2017 | 3 | 1-6-2017 | | | | | | | | | | | |
| Data | | | | | | | | | | | | | | |
| Write DataBase Plan | 3-5-2017 | 4 | 10-5-2017 | | | | | | | | | | | |
| Analysis | 31-5-2017 | 6 | 9-6-2017 | | | | | | | | | | | |
| Connection sensors to LoRa | 11-5-2017 | 3 | 15-5-2017 | | | | | | | | | | | |
| Storage | 17-5-2017 | 3 | 19-5-2017 | | | | | | | | | | | |
| Visualization | 31-5-2017 | 6 | 9-6-2017 | | | | | | | | | | | |
| Hardware | | | | | | | | | | | | | | |
| Familiarize with sensor boards | 30-4-2017 | 3 | 3-5-2017 | | | | | | | | | | | |
| Design Sensor billboard | 22-5-2017 | 3 | 24-5-2017 | | | | | | | | | | | |
| Build Sensor billboards | 29-5-2015 | 4 | 1-6-2017 | | | | | | | | | | | |
| Build sensor prototype | 8-5-2017 | 5 | 24-5-2017 | | | | | | | | | | | |
| Assemble & test all sensors | 24-5-2017 | 8 | 31-5-2017 | | | | | | | | | | | |
| Design Case | 10-5-2017 | 9 | 19-5-2017 | | | | | | | | | | | |
| Build cases | 22-5-2017 | 5 | 31-5-2017 | | | | | | | | | | | |
| Maintenance on sensors | 1-6-2017 | 21 | 22-6-2017 | | | | | | | | | | | |
| Retrieve sensors | 26-6-2017 | 3 | 29-6-2017 | | | | | | | | | | | |
| Outreach | | | | | | | | | | | | | | |
| Prepare stakeholder meeting | 24-4-2017 | 1 | 24-4-2017 | | | | | | | | | | | |
| Ask for permission for sensor placement | 17-5-2017 | 5 | 22-5-2017 | | | | | | | | | | | |
| Deploy sensors & billboards | 2-6-2017 | 7 | 9-6-2017 | | | | | | | | | | | |



| Activity | Start | Duration (days) | End |
|---|-----------|-----------------|-----------|
| Team Meeting | | | |
| Coach Meeting | | | |
| Status Meeting | | | |
| Deliverable | | | |
| General Project & Deliverables | | | |
| Work on deliverables DID-B | 28-4-2017 | 4 | 1-5-2017 |
| Work on deliverables DID-C | 15-5-2017 | 7 | 24-5-2017 |
| Create Presentation Baseline | 3-5-2017 | 2 | 4-5-2017 |
| Create Presentation Midterm | 18-5-2017 | 2 | 22-5-2017 |
| Create Final Presentation | 20-6-2017 | 3 | 22-6-2017 |
| IoT Academy | 1-5-2017 | 1 | 1-5-2017 |
| Define research question | 25-4-2017 | 3 | 28-4-2017 |
| Location scouting for sensors | 10-5-2017 | 2 | 12-5-2017 |
| Workshop | 3-5-2017 | 1 | 3-5-2017 |
| Divide tasks & roles | 21-5-2017 | 2 | 24-4-2017 |
| Scaling plan | 7-6-2017 | 3 | 9-6-2017 |
| Software | | | |
| Describe feedback mechanism | 8-5-2017 | 2 | 10-5-2017 |
| Build feedback mechanism | 10-5-2017 | 5 | 17-5-2017 |
| Test feedback mechanism | 15-5-2017 | 4 | 19-5-2017 |
| Connect feedback mechanism | 29-5-2017 | 3 | 1-6-2017 |
| Data | | | |
| Write Database Plan | 3-5-2017 | 4 | 10-5-2017 |
| Analysis | 31-5-2017 | 6 | 9-6-2017 |
| Connection sensors to LoRa | 11-5-2017 | 3 | 15-5-2017 |
| Storage | 17-5-2017 | 3 | 19-5-2017 |
| Visualization | 31-5-2017 | 6 | 9-6-2017 |
| Hardware | | | |
| Familiarize with sensor boards | 30-4-2017 | 3 | 3-5-2017 |
| Design Sensor billboard | 22-5-2017 | 3 | 24-5-2017 |
| Build Sensor billboards | 29-5-2015 | 4 | 1-6-2017 |
| Build sensor prototype | 8-5-2017 | 5 | 24-5-2017 |
| Assemble & test all sensors | 24-5-2017 | 8 | 31-5-2017 |
| Design Case | 10-5-2017 | 9 | 19-5-2017 |
| Build cases | 22-5-2017 | 5 | 31-5-2017 |
| Maintenance on sensors | 1-6-2017 | 21 | 22-6-2017 |
| Retrieve sensors | 26-6-2017 | 3 | 29-6-2017 |
| Outreach | | | |
| Prepare stakeholder meeting | 24-4-2017 | 1 | 24-4-2017 |
| Ask for permission for sensor placement | 17-5-2017 | 5 | 22-5-2017 |
| Deploy sensors & billboards | 2-6-2017 | 7 | 9-6-2017 |

Week 7
5-6-2017 6-6-2017 7-6-2017 8-6-2017 9-6-2017
Monday Tuesday Wednesday Thursday Friday
Weekend

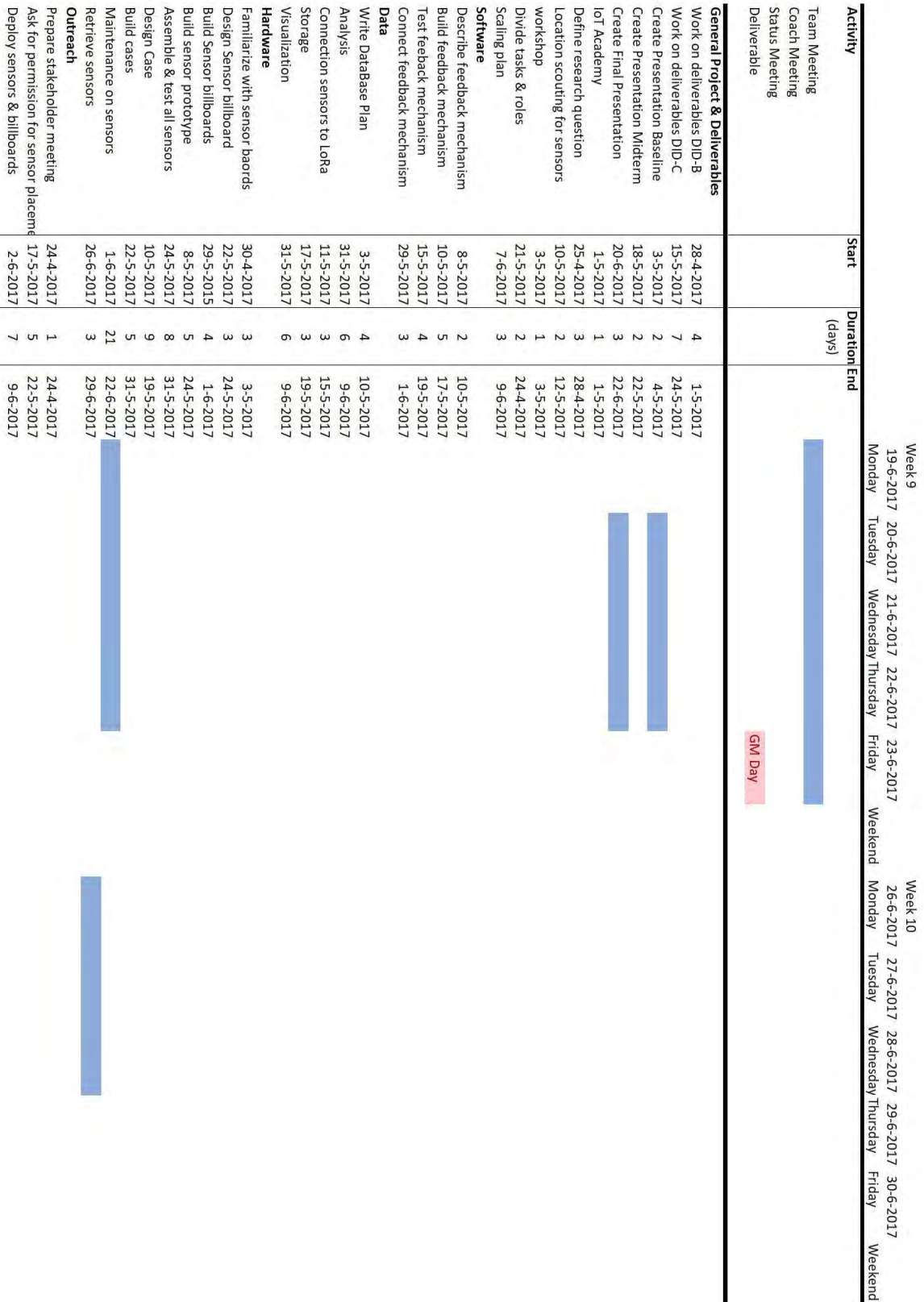
Week 8
12-6-2017 13-6-2017 14-6-2017 15-6-2017 16-6-2017
Monday Tuesday Wednesday Thursday Friday

W Monday

Geod. lab

Enrolab

2DID-EST



To show in a more simplified way the general workflows exist, an aggregated workflow is

shown in Figure 79 and described in a few paragraphs.

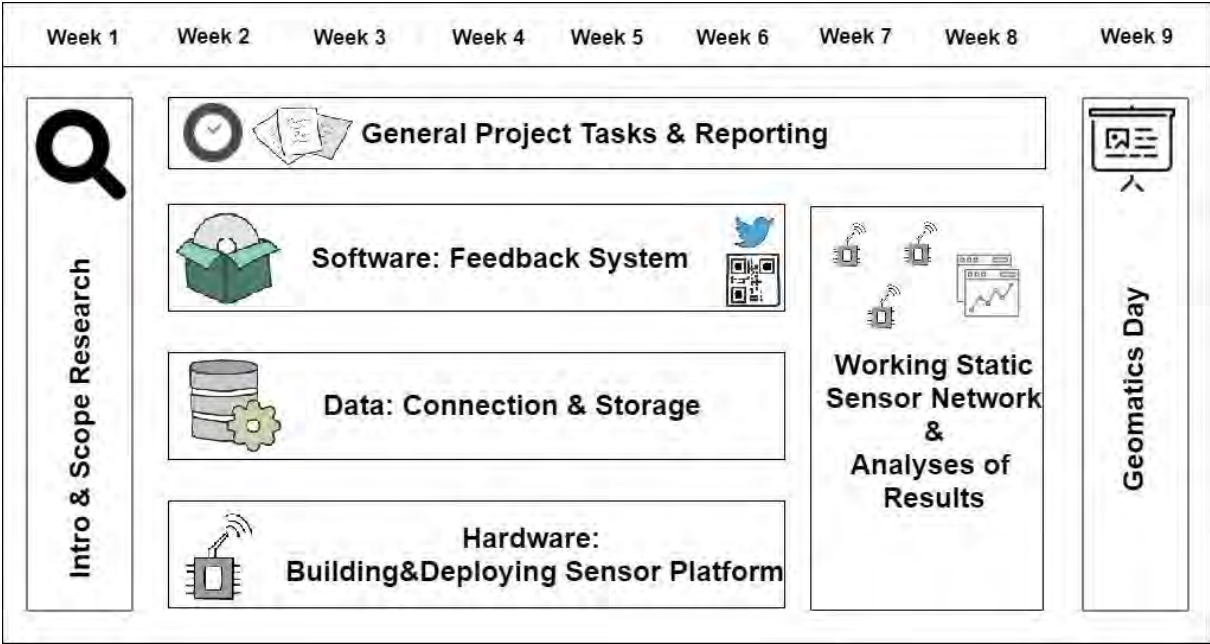


Figure 79: APPENDIX D - The aggregated workflow

The simplified workflow shows that for the majority of the project four parallel but highly interrelated workflows exist. During almost the entire project, general project tasks and reporting takes place. These general project tasks are for example meetings with coaches, workshops and other activities every group member takes part in. The Reporting part consists of both writing the deliverables, logging and other secretarial tasks.

The three (roughly) parallel processes that take place are the software development, the data connection and storage element and the hardware element. The majority of the software tasks are developing a feedback system, like for example a Twitter bot. Also a website and other social media need to be created and maintained. Another software task is writing code to read and send the sensor data: this code is placed on the microcontroller on each sensor platform.

The data connection element focuses on the connection and storage of the data the sensor platforms deliver. The connection therefore is both the connection over LoRa as well as the connection between the software and the data. This is one of the reasons these parallel processes are highly interrelated.

The hardware element covers the building and deploying of the sensor platforms. The casing

of the sensor platforms are also part of this. Since the sensor platforms are controlled by a microcontroller that sends over LoRa, this process is related to the 'Data' element in this project.

The three processes come together in the sense that the resulting product is a working sensor network that provides feedback to the user. In order to draw conclusions of both the construction of the network and two of its functions, at least one week of 'uptime' of the network is favorable.

The Geomatics Day is the great finale of this project: the entire process and results are presented on that day.

Appendix E: Media outreach strategy

Introduction

The main goal of the project Sensor City Delft is to raise environmental awareness of the citizens of Delft by interactively providing environmental data. To achieve this goal media outreach plays a major role. Media outreach is similarly important for bridging the gap between the TU Delft and the city itself by making the efforts of the students their research visible for the citizens. By communicating the project and its outcome clearly to the citizens the research gets a broader public and it will also increase the visibility of TU Delft's Geomatics program.

The audience

The main audience of the project's media outreach are the citizens and visitors of the city of Delft. Very often they do not have a clue what the students are doing at university whilst they are very much interested in this. Often the living area of the citizens is being researched and the citizens themselves can also benefit from these researches. Not only citizens can benefit from research from TU Delft but also community organizations such as homeowners associations or shopping owners associations are interested. Therefore the citizens, visitors and community organisations are the audience for this media outreach.

The second audience target group is professionals. Reaching out to professionals is a good way to profile the education of Geomatics in professional surroundings. They might be interested in the project or the master programme. Likewise it is a good way for the students to show what they are capable of to the professionals.

The message

It is aimed to encourage citizens and professionals to access the collected environmental data. The idea behind it is that by allowing the citizens to know more about the environment in their neighbourhood it will raise their environmental awareness. Eventually, this awareness can even lead to an improvement in the environment. Hence the message should include what the project is about and on how to access the data.

The media message also aims to inform the public about the fact that this is a pilot project with space for improvement and expansion. Therefore, it should be communicated as being a base where future student groups can build on.

Media outreach methods

Social Media

Facebook is used to keep the public updated with the latest news of the project. Both minor and important news are included in order to keep the followers updated on the latest news of the project. The Facebook page is shared with the IoT Dynamic group of the Synthesis project.

Twitter is merely used to provide the data to whomever asks for it using the right hashtag with the right text. In this stage it is not used to give other information because it is important to be focused on the functionality itself. The idea of interacting with the sensors using Twitter also encourages the user to retweet the incoming results or even take a picture of it (image xxx[N1]).



Figure 80: APPENDIX E - Using Twitter as a feedback mechanism also encourages citizens to tweet about the project

Billboards and flyers

Billboards are used to make the citizens aware of the project that is running in the street and

to intrigue the pedestrian to visit the website and the social media. It includes instructions on how to obtain the data via Twitter. It also has a QR Code that will take the user directly to the website.

In addition, flyers are placed in popular places such as cafes and shops. The flyers contain the same information as the billboards including; Twitter feedback instructions, QR code of the website URL, social media and website address. The flyers also include a short description of the projects and the location of the sensors (appendix Q).

Website

The website URL of the project is www.scdelft.nl. The main functionality is that it provides feedback to the users about the sensors data. It contains both the latest readings and a time series of the readings within a time frame. The website also provides an insight about the project itself in the form of different pages. Each page describes a specific aspect of the project including an interactive map of the locations of all the sensors. An option that could have been of good use but was only discovered in a later stage is the option to link the website to all social media, which publishes a post on the website directly on different social media including: Twitter Facebook, Instagram, Telegram and LinkedIn.

In addition, it contains a news section. The news published on the website is the same as the posts on Facebook which tells the public the latest development in the progress of the project. Likewise, the website aims to motivate the visitors to be engaged in the project. It encourages them to contact the project team and provides the tools to do so.

To create the website Wordpress.org is used which provides a simple tool to create websites. It was chosen because it provides different plugins with many functionalities. Such as, embedding live google maps, efficient contact us forms and the ability to embed an a webpage from other websites.

Interviews with local media

Interviews are made with two local newspapers. The first is AD Delft where an article describing the project was published on 14-06-2017 (see figure 49). It explains very well to the target group of citizens what the project is about and what the aim is of the project. I was also published at the website of AD on 17-06-2017 (<http://www.ad.nl/delft/tuktuk-en-rondvaartboot-meten-temperatuur-en-luchtkwaliteit~a3b8dc82/>).

► **TU-studenten doen met sensoren metingen in de binnenstad**

Tuktuk en rondvaartboot meten temperatuur en luchtkwaliteit

Met een Twitter-berichtje erachter komen hoe het gesteld is met de luchtkwaliteit. TU-studenten plaatsen sensoren in de stad.

Laura Kroet
Delft

Wie onlangs door de Choorstraat, Voldersgracht of over de Oude Langendijk liep, is het misschien al opgefallen. Op een aantal plaatsen in die drie straten zijn kastjes met sensoren te vinden. Ze zijn daar opgehangen door TU Delft-studenten die in het kader van een studieproject voor de masteropleiding Geomatics onderzoek doen. Met metingen willen ze in kaart brengen hoe het gesteld is met de temperatuur, de luchtvochtigheid, het geluidsniveau en de luchtkwaliteit in de stad.

„We hebben bewust voor die drie heel verschillende straten gekozen”, legt Cathelijne Kleijwegt uit. Ze is een van de zes studenten die het project op deze locaties uitvoert. „De straten hebben een heel ander profiel. Door de een mag je nog gewoon met een dieselauto rijden, terwijl de ander afgesloten is voor alle auto's, bussen en vrachtwagens. De diverse straten leveren ongetwijfeld heel andere gegevens op.”

Meetweek

Elk kwartier worden de verzamelde gegevens ververst. En, burgers kunnen de gemeten informatie zelf op elk moment van de dag opvragen. „Het was een van de voorwaarden van het project, dat we ook de verbinding met inwoners van de stad zoeken. Wij hebben bedacht om daar Twitter voor in te zetten”, legt Kleijwegt uit.

Door een berichtje via Twitter te sturen naar @SensorCityDelft kunnen inwoners de meetwaarden



▲ Cathelijne Kleijwegt en Panagiotis Karudakis maken deel uit van het projectteam van TU Delft-studenten. FOTO FRED LEEFLANG

inzien van de straten. Via de projectwebsite zijn de gegevens van de laatste vier uur ook terug te zien. Na afloop van de meetweek gaan de studenten al die gegevens analyseren. Ze hopen zo patronen te ontdekken in bepaalde dagen of een verschil tussen dag of nacht te vinden. Die informatie kan uiteindelijk weer gebruikt worden om de leefbaarheid in stad te verbeteren, legt Kleijwegt namens de projectgroep uit.

Door Twitter in te zetten, zoeken we de verbinding met inwoners van de stad

—Cathelijne Kleijwegt

ren, legt Kleijwegt namens de projectgroep uit.

Het zijn trouwens niet alleen de sensoren op de drie vaste locaties die gegevens verzamelen; er zijn ook 'dynamische sensoren' door heel de stad te vinden. Deze zijn geplaatst door zes andere TU Delft-studenten. De dynamische sensoren meten, zoals de naam al aangeeft, op steeds wisselende locaties.

Ze worden daarvoor op de tuktuks van Delft City Shuttle en de boten van Rondvaart Delft bevestigd. Zo is het mogelijk om elke paar minuten het geluidsniveau, de temperatuur en de luchtvochtigheid te meten, op steeds een ander punt in de stad. „Deze sensoren zijn heel interessant omdat je

zo gegevens kan verzamelen in een heel groot gebied”, zegt Kleijwegt. „De boten en de tuktuks zijn vrijwel continu in beweging en meten dus op veel verschillende punten.”

Groeien

Het project van de studenten bevindt zich nu nog in een testfase. Het doel is dat studenten in volgende jaren het project verder uitbouwen en door laten groeien.

„Voor ons is het echt het in de praktijk brengen van onze opgedane kennis”, zegt Kleijwegt. „Het is namelijk de afronding van ons eerste jaar van de masteropleiding Geomatics. We hadden wel een aantal voorwaarden en richtlijnen waar we aan moesten voldoen,

maar we mochten ook een deel zelf bepalen. Wat voor sensoren we bestelden bijvoorbeeld. Het verzoek om er een te plaatsen die de luchtkwaliteit kan meten, kwam vanuit de gemeente.”

Het hoort natuurlijk bij een testfase, maar de groep loopt hier en daar nog wel tegen wat aandachtspunten aan. „We weten nu hoe vaak de batterijen in die sensoren vervangen moeten worden bijvoorbeeld. Al dat soort kinderziektes zijn er na deze meetweek hopelijk uit”, zegt Kleijwegt.

Het project is ook online te volgen, via de facebookpagina van de studenten: Facebook.com/SensorCityDelft. En de meetgegevens kunnen worden bekeken via de website scdelft.nl.

Figure 81: APPENDIX E - news article about the Sensor City Delft project on AD.nl/delft

The second article was published in Delft op Zondag (figure 50).



Sensoren op de rondvaartboten meten het geluidsniveau, temperatuur en luchtvochtigheid.

Studenten TU Delft onderzoeken met sensoren de lucht en temperatuur in binnenstad

za 17 jun 2017, 15:01



DELFT - Studenten van de TU Delft onderzoeken hoe het in de Delftse binnenstad is gesteld met de temperatuur, luchtvochtigheid, luchtkwaliteit en het geluidsniveau. Dit meten ze met allerlei sensoren die met elkaar in verbinding staan en de gemeten data naar een computer versturen.

Daarbij worden twee soorten sensoren gebruikt: statische sensoren en dynamische sensoren. De statische sensoren zijn geplaatst in de Choorstraat, Voldersgracht en Oude Langendijk. Deze meten elk kwartier de temperatuur, luchtvochtigheid, geluidsniveau en luchtkwaliteit. Door een Twitter-bericht te sturen naar @SensorCityDelft kan iedereen de meetwaardes opvragen. Via de website is ook de data van de laatste vier uren terug te vinden. Na afloop van de meetweek zullen er analyses uitgevoerd worden om patronen te ontdekken tussen de dagen van de week of dag en nacht.

Door de hele stad zullen verder sensoren op de Tuk Tuks van Delft City Shuttle en rondvaartboten van Rondvaart Delft geplaatst worden. Deze platforms zullen elke paar minuten geluidsniveau, temperatuur en luchtvochtigheid meten. De metingen zijn een aanvulling op de metingen gedaan door de statische sensoren.

Beide sensorplatformen hebben in de week tussen 8 en 14 juni data verzameld waarna nu een analyseperiode volgt. Het project is een pilot waar in de toekomst een vervolg op zal komen. De projecten zijn te volgen op Facebook via SensorCityDelft en op www.scdelft.nl

Figure 82: APPENDIX E - news article about the Sensor City Delft project in Delft op Zondag

Nieuwsoverzicht

- Nieuwe strop en prijsverhoging dreigen bij Avalex 19:36
- Oud-atleet Joan van den Akker blikt terug 19:54
- 'Het was voor ons een perfecte locatie' 18:28
- Jabra GN Elite Sport: voor wie van sporten, muziek en gadgets houdt 15:42
- SUP-R: waar alle zintuigen worden geprikkeld 15:08

[Naar archief](#)

Meest gelezen

-  Mamabloggers bezoeken Delftse pareltjes 14/6
 -  DAS verhuist alsnog naar Spoorzone 13/6
 -  Gloednieuwe publiekshal van Stads kantoor is geopend 15/6
 -  Jeugd taptoe op de Markt 13/6
- [Meer in het archief >](#)

Fotoseries



Movie visiting professor Thomas H. Kolbe

The company OCVLVS FILM shoots a movie for the visiting professor of the faculty of Architecture this year Thomas H. Kolbe. During the project the two IoT groups of the synthesis project got approached to be in this movie too. Since this was in a late stage of the project the shootings will take place after the end presentation at Geomatics day. Thomas H. Kolbe will receive this movie as a thank-you present from the faculty and being in this movie will profile the project on an international level.

Future media outreach

After the project has finished the abstract together with the executive summary will be spread through four different media channels. Both documents will be shared with GIS magazine, a magazine that publishes developments in spatial information and Geo-ICT. The abstract will be posted on the Wordpress of Geomatics (<https://delftgeomatics.wordpress.com/>) and the blog of the Science Centre (<http://themakingof weblog.tudelft.nl/>). In a previous stage there was a reach out to B nieuws (<https://www.tudelft.nl/bk/actueel/magazines/bnieuws/>) the magazine of the faculty of Architecture. They did not seem interested but after the project has finished they will be contacted again with some results. This will be together with Delta (<http://www.delta.tudelft.nl/>) the online magazine of the TU Delft. To reach more professionals more magazines or even scientific journals can be reached out to, this has not come under discussion since professionals are not our main target group.

Recommendations

- Publish in scientific magazines to reach more specialized audience.
- Facilitating other social media methods is important such as; Instagram, Telegram and LinkedIn. This can be accomplished by linking the website to all social media as mentioned before.

Appendix F: Stakeholders and experts

Stakeholders

Municipality Delft: Provides financial support for the project. The Municipality wants to use the project to improve the livelihood in the city center of Delft and wants this project as a base for further development of similar projects in the city of Delft.

Science Centre TU Delft: Provides financial support for the project. The Science Centre also provides work spaces and equipment for the project development.

Residents of Delft: Next to the Municipality Delft the residence are the main user of the project where the sensors can interactively provide information to the citizens.

SpingSmart: SpingSmart is a technical consultancy firm which supports the team in both project management, software development and data analysis.

Experts

Stefan van der Spek: Project team main supervisor.

Wilko Quak: Project team supervisor, main contact regarding ICT such as the project database management.

Teun Verkerk: Project team supervisor, Developer Maker Education at the Science Centre Delft. Provides support regarding the project management and technical details.

Martijn Meijers: Provides technical support regarding ICT of the project.

Sabine de Milliano: CEO of Sping Smart. Provides insight about the project specifications.

Rob Braggaar: Provides insight and support about the physical component of the project such as; the sensors, LoPy and gateways.

Niels Stamhuis: A product designer working at the IoT Academy and the RDM Makerspace. Niels helps the group with expert knowledge of the (KPN) LoRa network.

Appendix G: MoSCoW priority management

'MoSCow' is a project management technique that aids in understanding and managing priorities (Verbree, 2017; Agile Business Consortium, 2017). 'MoSCoW' is an acronym for:

- Must Have
- Should Have
- Could Have
- Won't Have this time

Here MoSCoW is based on the fact that the project is a pilot project. Hence, the project is an initial prototype where minor and major glitches might not be addressed.

Must have

In order for the project to succeed it most interactively providing environmental information to the citizens using the sensor network. Hence the minimum usable subset (Must) of requirements which the project guarantees to deliver are:

- A LoRa network fixed in certain places in the city center of Delft. It should be able to collect data from sensors and store this data in a central database. For these network to be valid a minimum number of sensor platforms of 5 must be completed.
- The ability for the citizens to interactively and simply retrieve data from the network.
- One or more sensors fixed on each LoPy chip, that can collect data from the environment and transmit this data using LoRa.
- A proper way of designing the sensors, and fixing them around the city center.

Should have

There are requirements that are important for the project. Even though the solution is still viable without these requirements it is important for the project to include these requirements. These requirements can still require some workaround or completion by the end of the project time (Agile Business Consortium, 2017).

- The number of completed LoPy sensor platforms should be at least 15.
- The platforms are spread around three streets in the city center of Delft.

- Each LoPy chipset has a sensor connected to it that is able to collect information about air quality.
- Each LoPy chipset has a sensor connected to it that is able to measure the noise in the designated location.
- The ability for regular citizens to interact with the platform in two different ways; using QR code (directing to a website) and Twitter.
- The platforms should be powered in a sustainable way by using solar panels in addition to the built-in batteries.
- The boxing of the sensors should be designed in a way that is visually appealing, goes well with the surrounding environment and stimulates the curiosity of the citizens to interact with them.

Could have

These are the requirements that are less likely to be accomplished. Including these functionalities would be mainly dependent on the possibility to deliver within the deadline.

- The sensor platforms could have sensors that measure Temperature and humidity.
- The sensor platforms could have GNSS sensors that helps accurately determining the location of the sensors especially if the platforms are moved by stakeholders or simply they were taken by people.
- Using the platform for tracking could be possible if the privacy issue was carefully studied.
- All the possible 40 LoPy chips ideally could be used. Hence, the total number of completed sensor platforms would be 40.
- The sensors locations are placed in a way that produce an ideal results. Hence some additional streets might be included other than the ones that are already specified.
- Additional ways of the citizens to interaction with the sensor platforms could be added by using the built in Wi-Fi and Bluetooth Low Energy.
- The results of the sensors measurements could be made visual by using LED lights. For example, red light for low air quality and green light for good air quality.

Will not have this time

In order to focus the scope of the project and to manage the expectations the following

requirements will simply not make it into the deployed product. However, in the future these requirements may be added to the product.

- More sensors can be used other than the ones mentioned above (Noise, air quality, tracking people, Temperature, humidity), in order to provide a maximum use of the already established platforms. Examples of such sensors are: Light and vibration sensors.
- The information linkage would not extend to business, cars and public transportation.
- The project would not have a coverage for the whole city center of Delft.
- The project will not provide long time measurements. Because of the limited time frame, it is only possible to provide measurements of one or two weeks in best case scenario.

Appendix H: Boundary Conditions

Each project has its requirements and boundaries. Some are more obvious than others and some will only become visible in a later stage of the project. For starters these are the requirements and boundaries that are most obvious. Other requirements and boundaries will become clear during the project and should be added to the list for the other team members to make sure every team member is up to date to these requirements and boundaries.

The outcome of the project should be **interactive**, it should have good interaction with the citizens and visitors of the city of Delft and should be clearly visible. Even though it is an important part of the project the project should not fall or stand with this interactivity, in other words: the project should not be dependent on the amount of response it gets from the citizens and visitors.

One of the boundaries of this project is the **timespan** that is possible for the project. Because of this maximum of nine weeks that is available for the project a limited amount of time is available for taking the measurements in the city center. Together with the research, data analysis and reporting there are only one (or at most two) weeks left to do the data collection in the city center.

One other time limitation is the **time of hardware shipment**. For example ordering more sensors will take about two weeks to arrive. Hence any other hardware needed should be decided upon in early stages. Another boundary condition is the role of the municipality as one of the main stakeholders. They have expressed their interest in the acquisition of a certain type of data, particularly; noise levels, air quality, and on a lower level tracking of people. In addition, they expressed their interest in three particular streets which are: Choorstraat, Voldersgracht and Oude Langendijk. These requirements are not obligatory for the project but they do form a boundary that should be considered.

A different boundary of the project is the **amount of sensors** that are used for measuring. These sensors are already provided to the project and their amount is set to a maximum of 40 sensors. These sensors are placed in three streets of the city center of Delft that have a different profile and different characteristics. The maximum of three streets is set to the amount of sensors, this way 13 sensors will be installed in each street and one will be a spare

one. The initial intention is to build and test those 40 sensors in a week time and then to install them. Since this is a short timespan it is possible that the aim of 40 sensors will not be reached, this will end up in less sensors per street with a minimum of 5 sensors in each street. If it is not possible to place 5 sensors in each street one street has to be eliminated from the list or other actions should be taken.

There are many types of sensors that can measure different things. For this project only some environmental data are of **interest to collect**. Only this data that is of interest to collect is processed and used in the project and reported to the citizen that is interested in the outcomes of the sensors. If it is possible that more data than only the data of interest is collected (if there is enough space on the sensor platform) this will be collected and made available for the TU Delft and other interested parties as raw (unprocessed) data.

Appendix I: Functional, non-functional and killer requirements

The requirements are the things that are wanted or needed in the framework of this project. A functional requirement describes the behavior of the system, while a non-functional requirement describes the non-behavioral aspects of a system, capturing the properties and the constraints under which the system must operate (Chung & Do Prado Leite, 2009). So, functional requirements states what the product that is created by this project should be able to do, while a non-functional requirement describes a performance characteristic of the system. A general overview of the parts in the system is included in Appendix D.

Functional requirements

The Sensor City Delft project should deliver a product that can do the things as described in this section. In short, the end-product is a system consisting of a network of sensor platforms placed in the city center of Delft that measures environmental data, transmitting that data to a database, and the system must be able to respond to users who want to access the data via a web service. To accomplish this are the following functional requirements defined:

- Every sensor platform in the network should be able to measure air quality, current position, the local noise level, temperature and humidity on the place where the sensor platform is located.
- The data collected by each sensor platform must be transmitted to a database that runs on a server.
- The database must be accessible via a web service, where clients can sent requests to the database and get the information they asked for (this information is as 'real time' as possible).
- With regard to the user, for a further description of the implications of the received information, there must be a website where this context information is explained to the user.

Non-functional requirements

As described above, these kind of requirements can be considered as performance requirements of a specific characteristic of the system.

The sensor platforms themselves consist of the PyCom LoPy development board, an external LoRa antenna, and an expansion board for upgrading the firmware. For every environmental data type to investigate for this project there is a sensor attached to the board. For air quality this is the 'Plantower' PMS5003 sensor, connected via UART. For sound is a microphone array used, which is connected via UART, and for measuring global sound is the MAX9814 generic microphone used, connected via OneWire. Temperature and humidity is measured with the AM2302 device, connected to the sensor board via I2C. Further, the sensors are powered with on-board batteries and it is possible to attach a solar panel to the sensor board for extra energy. For operating and installing the sensor boards small electronic components such as RGB LEDs, resistors, transistors, buttons and switches will be used. Arduino Unos and Protoshields might be used when sensors from the Arduino collection are needed. The software for operating the sensors is written in the MicroPython programming language, that is used in the Atom development environment.

Further, the boxes in which these sensor boards are placed are 3D-printed boxes. For the best measurement results of the sensors, such a box must have openings on the correct locations. The boxes are placed in the city, so they must be suitable for use in an outdoor environment. Moreover, with regard to theft they must be placed relatively high in the air (4 meter or higher), and attached to a building or city object. It is preferred to place them on a window with suction cups. For the construction of the sensor board boxes are the 3D printer Ultimaker 2, 3D print materials and Rhino 3D software needed.

The sensor boxes are located in three streets in Delft: Oude Langendijk, Voldersgracht and the Choorstraat. These streets have their own characteristics regarding the type of use of the buildings, the infrastructures in the street, crowdedness, and length of the street. For each street is a typology made, based upon these characteristics. The number of sensor platforms for each street are chosen in such a way that the distance between the sensor platforms is around the same in all the streets.

Then, the local placement (height, distance to road, distance to building facades) of the sensor platforms in the built environment can have an effect on the reliability of the data. Therefore, the placement of the sensors must be done in a correct and justified manner for all locations, and there is chosen for a placement of about 4 meters above street level, so on the first floor

of a building. The distance to the road must be the same amount of meters for every sensor in a certain street. The distance to the building facades must be minimal, since the law on noise pollution in the Netherlands sets standards for noise levels on building facades (Law on Noise Pollution Article 82).

The data that is collected must be transmitted to the database in a way as efficient as possible, taking into account the limited bitrate of the LoRa communication system. Therefore, standards are needed for correct data transmission. It is a goal to offer the user who requests the data the information as fast as possible, so as 'real time' as possible. Based upon the maximum bitrate of LoRa is defined how 'real time' the information will be for the user. This all depends on how much data will be transmitted and how many sensors are attached to the sensor board.

From every LoRa sensor platform in the city is the data transmitted to the KPN LoRa service (KPN, 2017). This service sends the data via the internet to our database which runs on a server. Therefore, a free KPN LoRa developer account, a PostgreSQL database, and a server that can store large amounts of data are needed.

The web service is the feedback part of the system. This web service must be easy to use by the users, first of all by giving the user several possibilities to access the data, e.g. online via Twitter or by scanning a QR-code, or offline by a LED-display that shows the values after a request. Next to that, the information must be communicated effectively and efficiently to the user. Effectively, because the user must understand the message as clear as possible. Therefore, a web site is created, where additional information such as statistics about the environmental values and how the values must be interpreted can be found. The information must be communicated efficiently, because the information is limited to the 140 characters of Twitter. For this part of the project are access to the development part of a website, a Twitter account, and a QR code generator needed.

Killer requirements

Killer requirements can obstruct the feasibility of the project. They can be regarded as being a risk to a successful end result.

- With regard to the placement of the 'sensor boxes' in the city: what if we cannot find

enough locations to place them? Then other streets than the initial three must be investigated, maybe even streets of the group members themselves, or streets of fellow students.

- Messages sent via LoRa will be affected by the distance between buildings, but it is not yet sure how much. Maybe an amplifier is needed to transmit the data.
- The KPN LoRa service might be unavailable sometimes. That affects the quality of the feedback service.
- With regard to security: how to deal with the possibility that the sensor boxes are stolen? And how 'hackable' are the sensor boards?
- Will the QR code work during the night, when it is dark? When the QR code is unavailable, another online way of communication is possible, namely via Twitter.
- It is not allowed to place sensor boards on old buildings (which are often monuments), therefore suction cups are a good solution. These suction cups must be able to hold the total weight of a sensor box.

Appendix J: MicroPython script main.py

```
import time
from machine import Pin, Timer, UART, WDT
from network import LoRa
import socket
import binascii
import struct
import config
import machine
from dth import DTH
from math import log
import utime
from struct import unpack
import array

wdt = WDT(timeout=1920000)

#LoRa
lora = LoRa(mode=LoRa.LORAWAN)

# create an ABP authentication params
dev_addr = struct.unpack(">1", binascii.unhexlify(config.DEV_ADDR))[0]
nwk_swkey = binascii.unhexlify(config.NWKS_KEY)
app_swkey = binascii.unhexlify(config.APPS_KEY)

# join a network using ABP (Activation By Personalization)
lora.join(activation=LoRa.ABP, auth=(dev_addr, nwk_swkey, app_swkey))

# create a LoRa socket
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)

# set the LoRaWAN data rate
s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)

SET_PORT = 'P11' # LoPy SET port, communicates with the PMS SET port (P3)
RESET_PORT = 'P8' # LoPy RESET port, communicates with the PMS RESET port(P6)

STARTUP_TIME = 30 # 30 seconds of startup time (needed for ventilator). 5 is better for
debugging.

set_pin = Pin(SET_PORT, mode=Pin.OUT)
reset_pin = Pin(RESET_PORT, mode=Pin.OUT)
data_uart2 = UART(1, baudrate=9600, parity=None, stop=1, pins=('P9', 'P10'))

# setup pin
th = DTH(Pin('P4', mode=Pin.OPEN_DRAIN), 1)

SAMPLE_WINDOW = 50 # Sample window width in ms (50 ms = 20Hz)
sample = 0
adc = machine.ADC(bits=10) # create an ADC object
apin = adc.channel(pin='P16') # create an analog pin on P16=G3

# timing object for measuring low pulse
chrono = Timer.Chrono()

def fan_on():
    #print('Turning the PMS5003 fan on') # the fan is ON by default.
    set_pin.value(1)

def fan_off():
    #print('Turning the PMS5003 fan off')
    set_pin.value(0)

def reset():
    #print('Resetting the PMS5003 sensor')
    reset_pin.value(1)

def set_sensor():
    #print('Preparing the fan...')
    fan_on()
    #print('Wait 30 seconds...')
    time.sleep(STARTUP_TIME) # wait 30 seconds
```

```

# print('Sensor is set')

while True:
    # temperature and humidity
    result = th.read()
    tempbeforecommal = int((str(result.temperature/1.0).split("."))[0])
    tempaftercommal = int((str(result.temperature/1.0).split("."))[1])
    humbeforecommal = int((str(result.humidity/1.0).split("."))[0])
    humaftercommal = int((str(result.humidity/1.0).split("."))[1])

    # noise
    # Measuring noise values. Measures only one moment.
    start_time = chrono.read_ms()
    peak_to_peak = 0 # initialize peak-to-peak level
    signal_max = 0 # maximum signal
    signal_min = 1024 # minimal signal
    chrono.start() # start the stopwatch
    while (chrono.read_ms() - start_time) < SAMPLE_WINDOW: # takes 50 ms
        sample = apin()
        if sample > signal_max:
            signal_max = sample
        elif sample < signal_min:
            signal_min = sample
    peak_to_peak = signal_max - signal_min
    if (peak_to_peak > 0):
        log_value = ((peak_to_peak/1024)+1)
        dB = 2 * int(round(100*(log(log_value, 10))))
    else:
        dB = 0
    chrono.stop() # stop the stopwatch
    chrono.reset() # reset the stopwatch

    set_sensor()
    d = data_uart2
    raw = d.read()
    unpacking = unpack('>16H', raw)
    pm1_u = int(unpacking[2])
    pm2_u = int(unpacking[3])
    pm10_u = int(unpacking[4])
    pm1_atm_u = int(unpacking[5])
    pm2_atm_u = int(unpacking[6])
    pm10_atm_u = int(unpacking[7])
    fan_off()

    hum = int(hex(humbeforecommal))
    hum1 = int(hex(humaftercommal))
    aq3 = int(hex(0))
    aq4 = int(hex(0))
    temp = int(hex(tempbeforecommal))
    temp1 = int(hex(tempaftercommal))
    noi = int(hex(dB))
    pm1 = int(hex(pm1_u))
    pm2 = int(hex(pm2_u))
    pm10 = int(hex(pm10_u))
    pm1_atm = int(hex(pm1_atm_u))
    pm2_atm = int(hex(pm2_atm_u))
    pm10_atm = int(hex(pm10_atm_u))

    s.setblocking(True)

    # send over lora
    s.send(bytes([aq3, aq4, temp, temp1, hum, hum1, noi, pm1, pm2, pm10, pm1_atm,
    pm2_atm, pm10_atm]))
    s.setblocking(False)
    wdt.feed()
    time.sleep(780)
    machine.deepsleep(90000)

```

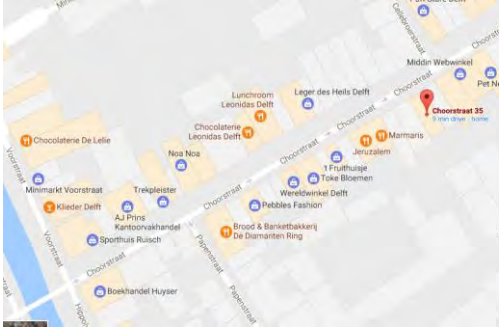

Appendix K: Sensor locations – facades

In order to hang the sensor boxes to a façade, the residents and shop owners have to be informed and asked for permission for hanging the sensor box. Because of uncertainty whether or not they will agree with placing the sensors, two scenarios have been researched.

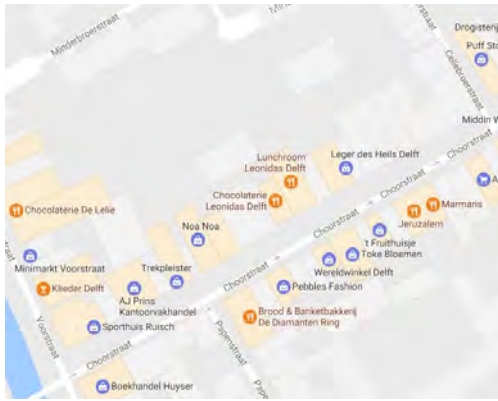
In the first scenario the residents and shop owners agree with putting up the sensor at their facade. In this case it is necessary to determine at which place on the facade the sensors can be hanged or attached. Hereby it is important to design the casing in such a manner that holes in the facade are not needed. An option for implementing the sensor boxes without causing damage to the facade is with using plastic tie-rips. Tie-rips are available in several sizes and thicknesses and make it possible to tie the sensor boxes around objects. Another option is using steel tie-rips.

The other scenario is that the residents and shop owners do not agree to attach the sensor their building. In this case it is necessary to examine trees and lanterns. These objects provide the possibility to hang the sensors at a sufficient height and have parts sticking out to attach the sensor to.

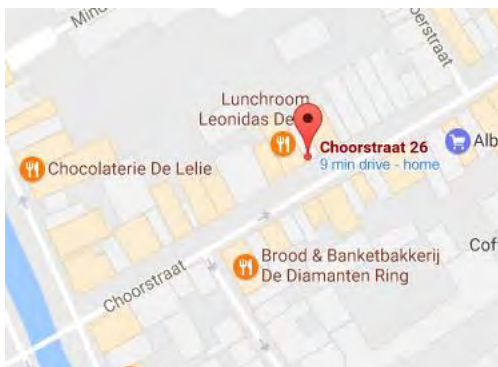
In the table down below an overview is shown with the investigated facades with possibilities to mount sensors and billboards.

| Choorstraat | Facade |
|--|--|
| <p data-bbox="188 1496 472 1532">Albert Heijn, number 35</p>  |  <ul data-bbox="754 1809 1142 1839" style="list-style-type: none">• Suitable for interactive board |

Restaurant Redjeki, number 52 to 56



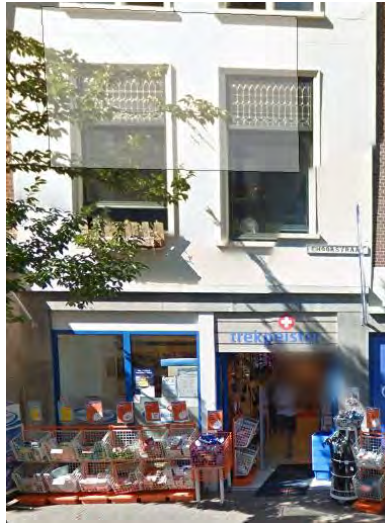
Leonidas, number 3



Diamant Schaar, number 11

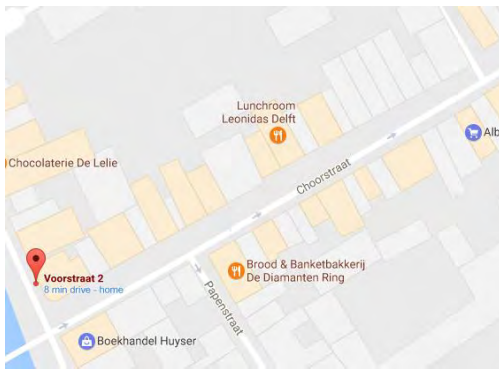


Trepleister, number 10



- Suitable for interactive board

Ruisch Sports, number 2



Voldersgracht

Facade

Restaurant puro, number 28



Knotten (Shop), number 23

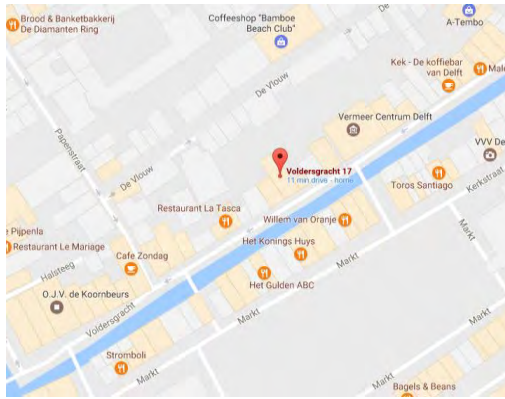


Museum Vermeer, number 21



- Suitable for interactive board

Hostel, number 17



- Suitable for interactive board

Restaurant La Tasca, number 13 & 14



O.J.V. de Koornbeurs, number 5



- Suitable for interactive board

Oude Langendijk

Facade

Kientz kunst, number 35



Jamin and Ziengs, number 15-16-17



- Suitable for interactive board

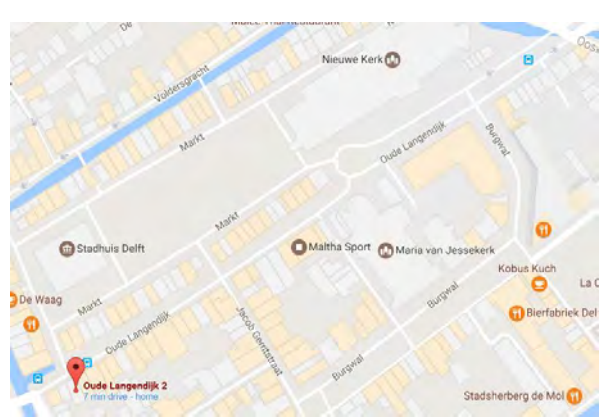
Resident building, number 23



Long John bodywear, number 4



Shop, Number 2



Appendix L: Sensor locations – trees and lanterns

Overview location scouting trees and street lanterns

Choorstraat

Trees



Figure 83: APPENDIX L - Possible locations on trees

Lanterns

| Street | Lantern |
|----------------|--|
| Choorstraat 31 |  |
| Choorstraat 16 |  |

Choorstraat 2



Voldersgracht

Trees

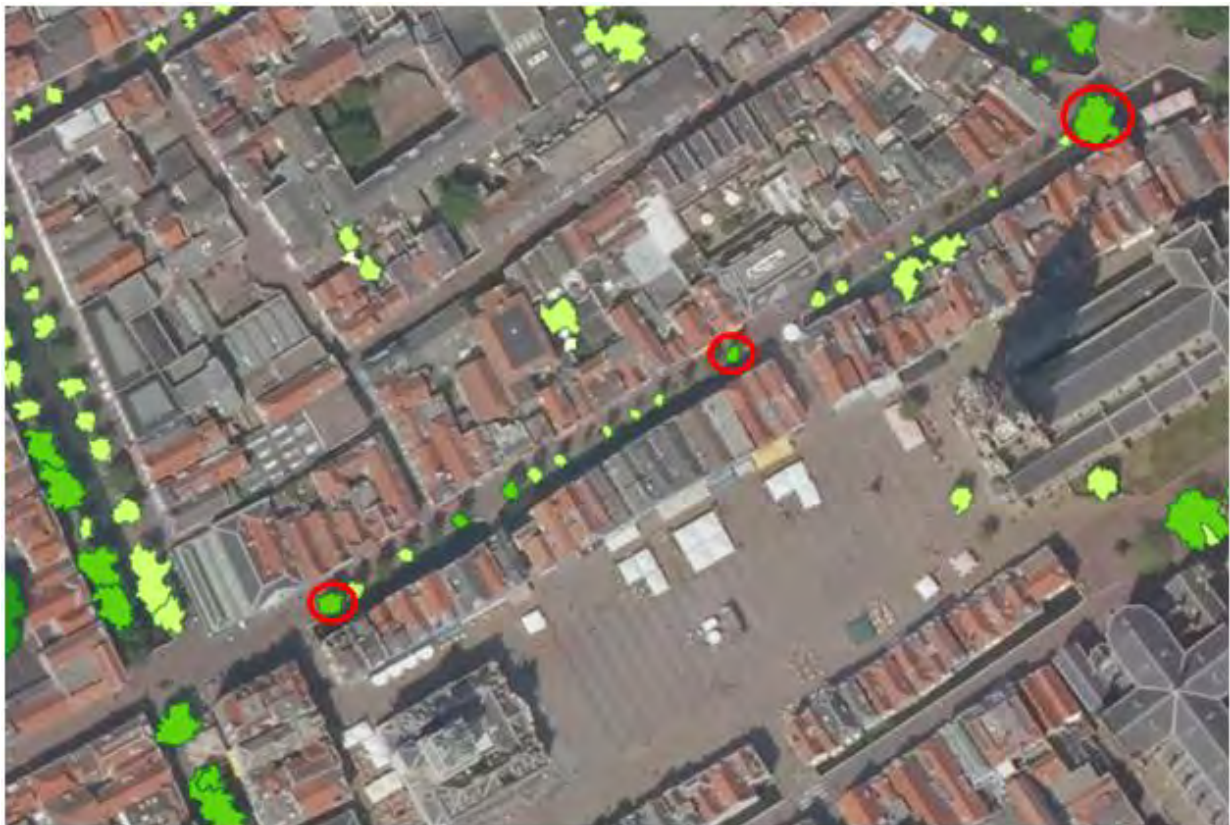
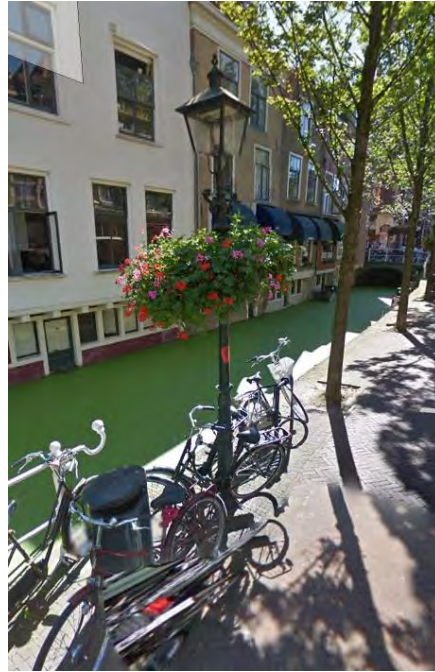


Figure 84: APPENDIX L - Possible locations in Voldersgracht

Lanterns

| Street | Lantern |
|------------------|--|
| Voldersgracht 30 |  |
| Voldersgracht 23 |  |

Voldersgracht 6



Oude Langendijk

Trees



Figure 85: APPENDIX L - Possible locations in Oude Langendijk

Lanterns

| Street | Lantern |
|-------------------|--|
| Oude Langendijk 3 |  A street-level photograph of Oude Langendijk 3. In the foreground, a black, ornate lantern post stands on the sidewalk. The background shows a row of multi-story buildings with large windows and doors. A few people are visible walking on the sidewalk. The scene is captured during the day. |

Oude Langendijk 18



Oude Langendijk 36



Appendix M: Casing

The 2d view of the end product with the dimensions and a brief explanation of its components included.

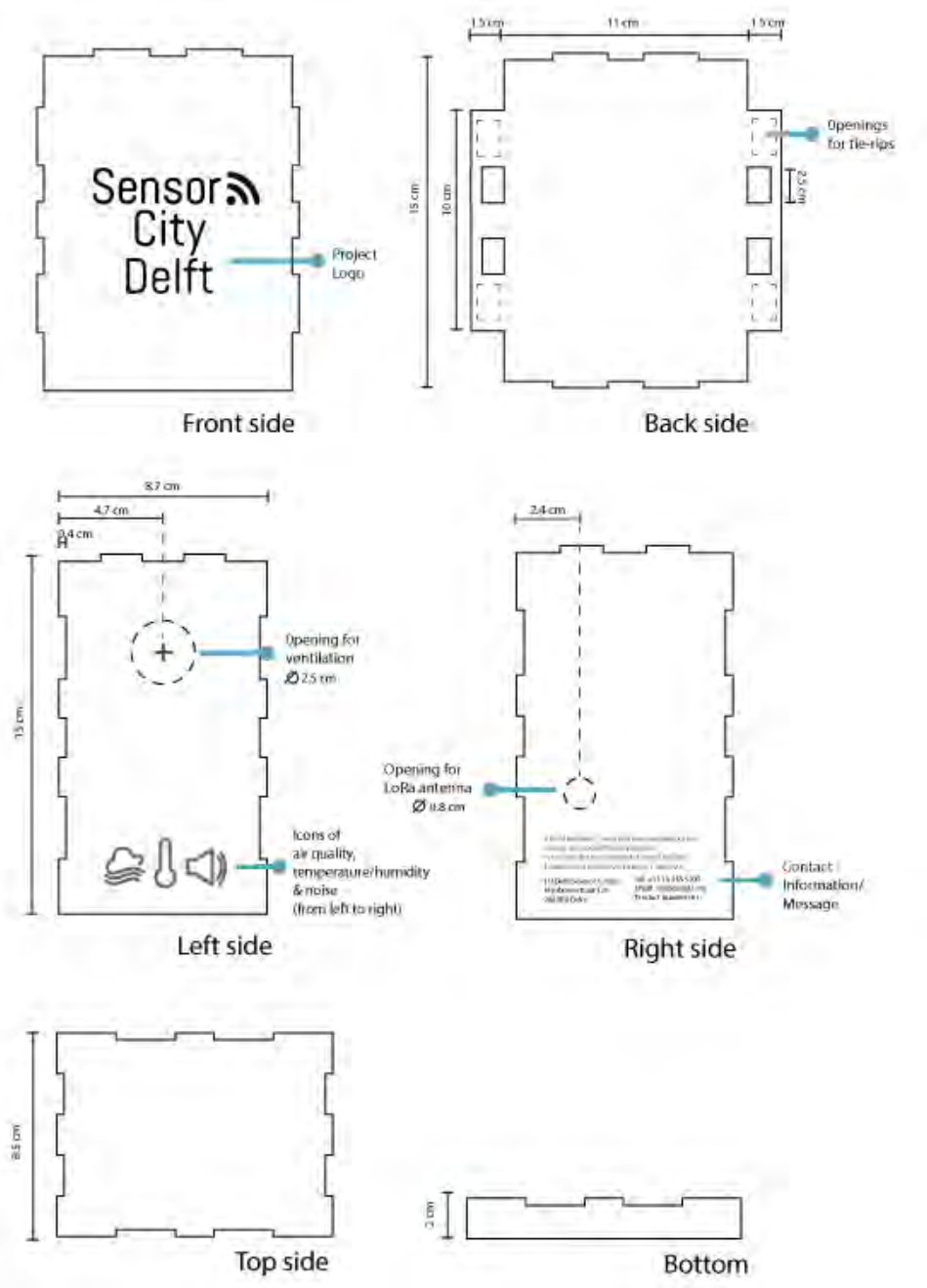


Figure 86: APPENDIX M – Casing

Appendix N: System overview

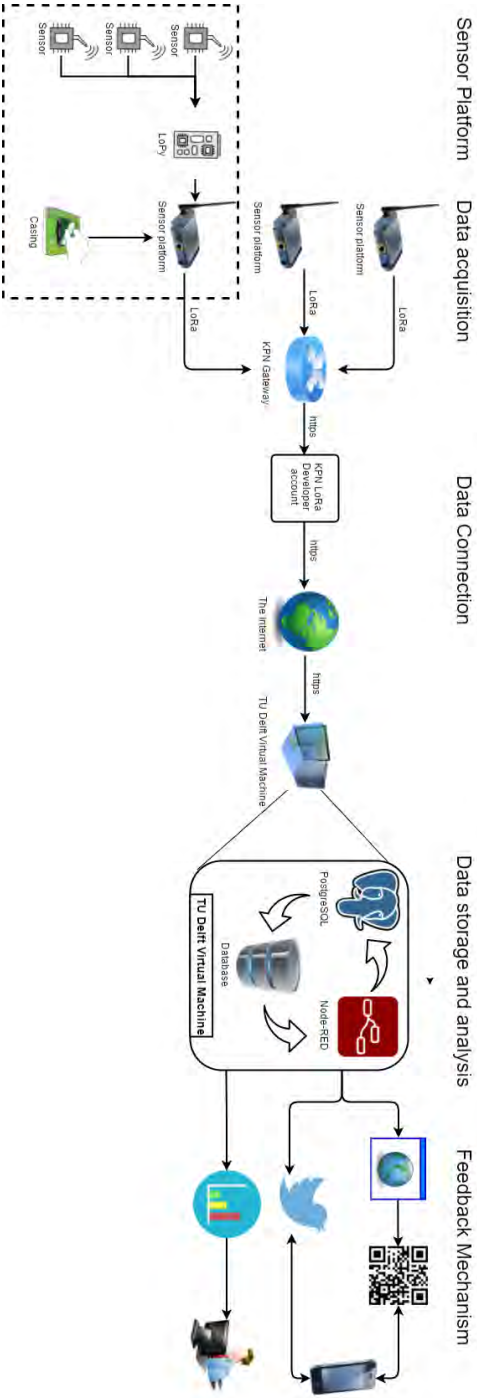


Figure 87: APPENDIX N - Sensor City Delft system overview

Appendix O: LoRa JSON message

LoRa messages are send as JSON objects. An exemplary JSON object is shown below, on the next page a table with abbreviations and explanations can be found. The measurements taken from the Lopy are send in the payload_hex attribute. This is a hexadecimal message and is encrypted by KPN. See Appendix S for more information about this encryption.

```
{
  "DevEUI_uplink":
  {
    "Time": "2017-06-14T10:44:30.369+02:00"
    "DevEUI": "0059AC00001807B8"
    "FPort": "2"
    "FCntUp": "1"
    "MType": "2"
    "FCntDn": "138",
    "payload_hex": "304840d1b9fa8d97623c1201bd"
    "mic_hex": "94454654"
    "Lrcid": "0059AC02"
    "LrrRSSI": "-110.000000"
    "LrrSNR": "4.000000"
    "SpFact": "7"
    "SubBand": "G1"
    "Channel": "LC3"
    "DevLrrCnt": "1"
    "Lrrid": "FF010226"
    "Late": "0"
    "LrrLAT": "52.019737"
    "LrrLON": "4.361882"
    "Lrrs":
    {
      "Lrr":
      {
        "Lrrid": "FF010226"
        "Chain": "0"
        "LrrRSSI": "-110.000000"
        "LrrSNR": "4.000000"
        "LrrESP": "-111.455406"
      }
    }
    "CustomerID": "100006356"
    "CustomerData":
    {
      "alr":
      {
        "pro": "SMTC/LoRaMote"
        "ver": "1"
      }
    }
    "ModelCfg": "0"
    "InstantPER": "0.000000"
    "MeanPER": "0.000000"
    "DevAddr": "142037B9"
  }
}
```

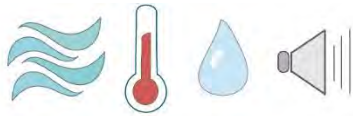
| JSON Attribute | Explanation |
|---------------------|--|
| LrrSNR | SNR (Signal Noise Ratio) measured by the best LRR. |
| Lrrid | ID of the LRR (Gateway) that processed the packet. |
| SpFact | SF (Spreading Factor) used by device |
| SubBand | SUB BAND used by the device. |
| CustomerData | ASCII customer data set by provisioning. |
| FPort | LoRaWAN FPort used by the device for this packet. |
| Channel | Radio Channel (LC) used by the device. |
| FCntUp | The uplink counter for this packet. |
| Time | LRR (Gateway) Timestamp for the packet. |

| | |
|--------------------|--|
| DevEUI | Device DevEUI. |
| payload_hex | LoRaWAN payload in hexadecimal ASCII format. |
| CustomerID | Customer ID associated to the Device Manager account. |
| LrrRSSI | RSSI (Received signal strength indication) measured by the best LRR. |
| ADRbit | ADRBit (Adaptive Data Rate on or off) set by the device. |
| ModelCfg | ASCII ThingPark Cloud data set by provisioning. |
| mic_hex | MIC in hexadecimal ASCII format. |
| LrrLON | Longitude of the best LRR. |
| LrrLAT | Latitude of the best LRR. |
| FCntDn | The last downlink counter to the device. |
| Lrcid | ID of the LRC (Core) that processed the packet. |
| DevLrrCnt | Number of LRRs which received this packet. |

Table 19: APPENDIX O - LoRa JSON abbreviations

Appendix P: Sensor billboard

An example of a billboard that is placed close to a sensor.



Sensor
City
Delft

De TU Delft is hier metingen aan het doen op gebied van milieu.

U kunt hier zien wat er gemeten wordt!

Tweet
#OudeLangendijk

@SensorCityDelft

of

Scan deze QR code



www. scdelft.nl



Like us on facebook
Sensor City Delft

Science
Centre Delft

TU Delft

SpingSmart
sense | connect | observe

Figure 88: APPENDIX P - Sensor billboard

Appendix Q: Sensor flyer

Sensor City Delft

Door Geomatics studenten van de TU Delft worden metingen uitgevoerd in Delft met sensoren. Deze metingen worden gedaan in 3 straten: Choorstraat, Voldersgracht en Oude Langendijk. De sensoren zijn geplaatst in speciale boxen met ieder een eigen nummer.



#Choorstraat
#Voldersgracht
#OudeLangendijk

*Verander naar #EN voor Engels

Dit zullen metingen zijn op 4 onderwerpen:

-  Geluid
-  Temperatuur
-  Luchtvochtigheid
-  Luchtkwaliteit

Op deze manier willen wij u lokaal over deze 4 onderwerpen van informatie verschaffen wanneer u dit opvraagt via Twitter.

Geïnteresseerd?



-  Scan de QR code
-  Twitter straat met onderwerp naar sensorcitydelft
-  www.facebook.com/
-  www.twitter.com/
-  www.geo1101.nl/



Figure 89: APPENDIX Q - Flyers for the Sensor City Delft project

Appendix R: Twitterbot code

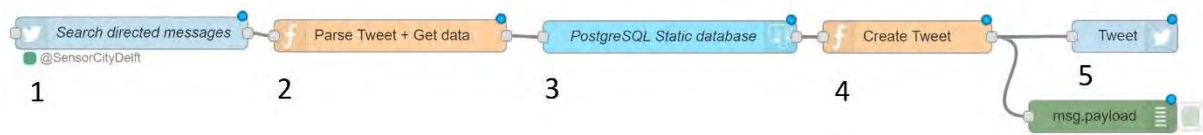


Figure 90: APPENDIX R - Node-RED flow of twitterbot

1: Search Tweets directed to @SensorCityDelft

2: Parse Tweet and Get Data

```
msg.queryParameters = {};
```

```
var O1 = "'142037B9'";  
var V2 = "'14204140'";  
var V3 = "'14203981'";
```

```
var C1 = "'14204073'";  
var C2 = "'1420366D'";  
var C3 = "'14203210'";
```

```
var V1 = "'14204772'";  
var O2 = "'142042F7'";  
var O3 = "'14204FD8'";
```

```
if ((msg.payload.includes("Temperature")) || (msg.payload.includes("temperature")) ||  
(msg.payload.includes("temperatuur")) || (msg.payload.includes("Temperatuur"))){  
  if (msg.payload.includes("#Voldersgracht")){  
    msg.payload = "Select temperature from public.data3 where (sid = "+V1+" and sensedtime =  
(select max(sensedtime) from public.data3 where sid="+V1+" ))";}  
    else if (msg.payload.includes("#Choorstraat")){  
    msg.payload = "Select temperature from public.data3 where (sid = "+C1+" and sensedtime =  
(select max(sensedtime) from public.data3 where sid="+C1+" ))";}  
    else if ((msg.payload.includes("#OudeLangendijk")) || (msg.payload.includes("#Oudelangendijk"))){  
    msg.payload = "Select temperature from public.data3 where (sid = "+O1+" and sensedtime =  
(select max(sensedtime) from public.data3 where sid="+O1+" ))";}  
    else if (msg.payload.includes("#Voldersgracht2")){  
    msg.payload = "Select temperature from public.data3 where (sid = "+V2+" and sensedtime =  
(select max(sensedtime) from public.data3 where sid="+V2+" ))";}  
    else if (msg.payload.includes("#Choorstraat2")){  
    msg.payload = "Select temperature from public.data3 where (sid = "+C2+" and sensedtime =  
(select max(sensedtime) from public.data3 where sid="+C2+" ))";}  
    else if ((msg.payload.includes("#OudeLangendijk2")) || (msg.payload.includes("#Oudelangendijk2"))){  
    msg.payload = "Select temperature from public.data3 where (sid = "+O2+" and sensedtime =  
(select max(sensedtime) from public.data3 where sid="+O2+" ))";}}  
  else if ((msg.payload.includes("Humidity")) || (msg.payload.includes("humidity")) ||  
(msg.payload.includes("vochtigheid")) || (msg.payload.includes("Vochtigheid"))){  
    if (msg.payload.includes("#Voldersgracht")){  
    msg.payload = "Select humidity from public.data3 where (sid = "+V1+" and sensedtime = (select  
max(sensedtime) from public.data3 where sid="+V1+" ))";}  
    else if (msg.payload.includes("#Choorstraat")){  
    msg.payload = "Select humidity from public.data3 where (sid = "+C1+" and sensedtime = (select  
max(sensedtime) from public.data3 where sid="+C1+" ))";}  
    else if ((msg.payload.includes("#OudeLangendijk")) || (msg.payload.includes("#Oudelangendijk"))){  
    msg.payload = "Select humidity from public.data3 where (sid = "+O1+" and sensedtime = (select  
max(sensedtime) from public.data3 where sid="+O1+" ))";}  
    else if (msg.payload.includes("#Voldersgracht2")){  
    msg.payload = "Select humidity from public.data3 where (sid = "+V2+" and sensedtime = (select  
max(sensedtime) from public.data3 where sid="+V2+" ))";}  
    else if (msg.payload.includes("#Choorstraat2")){  
    msg.payload = "Select humidity from public.data3 where (sid = "+C2+" and sensedtime = (select  
max(sensedtime) from public.data3 where sid="+C2+" ))";}  
    else if ((msg.payload.includes("#OudeLangendijk2")) ||
```



```

(msg.payload.includes("#Oudelangendijk2")){
    msg.payload = "Select humidity from public.data3 where (sid = "+02+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+02+" ));};}
else if ((msg.payload.includes("Noise"))||(msg.payload.includes("noise")) ||
(msg.payload.includes("Geluid"))||(msg.payload.includes("geluid"))){
    if (msg.payload.includes("#Voldersgracht")){
        msg.payload = "Select noise from public.data3 where (sid = "+V1+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+V1+"));";}
    else if (msg.payload.includes("#Choorstraat")){
        msg.payload = "Select noise from public.data3 where (sid = "+C1+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+C1+"));";}
    else if ((msg.payload.includes("#OudeLangendijk"))||(msg.payload.includes("#Oudelangendijk"))){
        msg.payload = "Select noise from public.data3 where (sid = "+01+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+01+"));";}
    else if (msg.payload.includes("#Voldersgracht2")){
        msg.payload = "Select noise from public.data3 where (sid = "+V2+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+V2+"));";}
    else if (msg.payload.includes("#Choorstraat2")){
        msg.payload = "Select noise from public.data3 where (sid = "+C2+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+C2+"));";}
    else if ((msg.payload.includes("#OudeLangendijk2"))||(msg.payload.includes("#Oudelangendijk1"))){
        msg.payload = "Select noise from public.data3 where (sid = "+02+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+02+"));";}
    else if ((msg.payload.includes("Air Quality"))||(msg.payload.includes("Air quality"))||
(msg.payload.includes("air quality")) || (msg.payload.includes("Lucht"))||
(msg.payload.includes("lucht"))){
        if (msg.payload.includes("#Voldersgracht")){
            msg.payload = "Select airquality from public.data3 where (sid = "+V1+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+V1+"));";}
        else if (msg.payload.includes("#Choorstraat1")){
            msg.payload = "Select airquality from public.data3 where (sid = "+C1+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+C1+"));";}
        else if ((msg.payload.includes("#OudeLangendijk"))|| (msg.payload.includes("#Oudelangendijk"))){
            msg.payload = "Select airquality from public.data3 where (sid = "+01+"and sensedtime = (select
max(sensedtime) from public.data3 where sid="+01+"));";}
        else if (msg.payload.includes("#Voldersgracht2")){
            msg.payload = "Select airquality from public.data3 where (sid = "+V2+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+V2+"));";}
        else if (msg.payload.includes("#Choorstraat2")){
            msg.payload = "Select airquality from public.data3 where (sid = "+C2+" and sensedtime = (select
max(sensedtime) from public.data3 where sid="+C2+"));";}
        else if ((msg.payload.includes("#OudeLangendijk2"))|| (msg.payload.includes("#Oudelangendijk2"))){
            msg.payload = "Select airquality from public.data3 where (sid = "+02+"and sensedtime = (select
max(sensedtime) from public.data3 where sid="+02+"));";}
        else {
            msg.payload = "Select temperature from public.data3 where sid = '00000000'";
        }
    }
}

return msg;

```

3: Database connection

PostgreSQL connection node

4: Create Tweet

```

msg.queryParameters = {};
data = msg.payload[0];
msg.test = msg.tweet.text;

if (msg.test.includes("#EN")){
    if ((msg.test.includes("Temperature")) || (msg.test.includes("temperature"))){
        if (msg.test.includes("#Voldersgracht1")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' The temperature at the
Voldersgracht 1 is '+ data.temperature+'°C'; }
        else if (msg.test.includes("#Choorstraat1")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' The temperature at the
Choorstraat 1 is '+ data.temperature+'°C'; }
        else if (msg.test.includes
("#OudeLangendijk1"))|| (msg.test.includes("Oudelangendijk1"))){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' The temperature at the Oude
Langendijk 1 is '+ data.temperature+'°C'; }
        else if (msg.test.includes("#Voldersgracht2")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' The temperature at the
Voldersgracht 2 is '+ data.temperature+'°C'; }
        else if (msg.test.includes("#Choorstraat2")){

```

```

        msg.payload = '@'+ msg.tweet.user.screen_name + ' The temperature at the
Choorstraat 2 is '+ data.temperature+'°C'; }
        else if ((msg.test.includes
("#OudeLangendijk2")) || (msg.test.includes("Oudelangendijk2"))){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' The temperature at the Oude
Langendijk 2 is '+ data.temperature+'°C'; }
            else if ((msg.test.includes ("Humidity")) || (msg.test.includes("humidity"))){
                if (msg.test.includes ("#Voldersgracht1")){
                    msg.payload = '@'+ msg.tweet.user.screen_name + ' The humidity at the
Voldersgracht 1 is '+ data.humidity; }
                    else if (msg.test.includes ("#Choorstraat1")){
                        msg.payload = '@'+ msg.tweet.user.screen_name + ' The humidity at the Choorstraat
1 is '+ data.humidity; }
                    else if ((msg.test.includes
("#OudeLangendijk1")) || (msg.test.includes("#Oudelangendijk1")) ){
                        msg.payload = '@'+ msg.tweet.user.screen_name + ' The humidity at the Oude
Langendijk 1 is '+ data.humidity; }
                        else if (msg.test.includes ("#Voldersgracht2")){
                            msg.payload = '@'+ msg.tweet.user.screen_name + ' The humidity at the
Voldersgracht 2 is '+ data.humidity; }
                            else if (msg.test.includes ("#Choorstraat2")){
                                msg.payload = '@'+ msg.tweet.user.screen_name + ' The humidity at the Choorstraat
2 is '+ data.humidity; }
                                else if ((msg.test.includes
("#OudeLangendijk2")) || (msg.test.includes("#Oudelangendijk2")) ){
                                    msg.payload = '@'+ msg.tweet.user.screen_name + ' The humidity at the Oude
Langendijk 2 is '+ data.humidity; }
                                    else if ((msg.test.includes ("Noise")) || (msg.test.includes("noise"))){
                                        if (msg.test.includes ("#Voldersgracht1")){
                                            msg.payload = '@'+ msg.tweet.user.screen_name + ' The noise level at the
Voldersgracht 1 is '+ data.noise+' dB'; }
                                            else if (msg.test.includes ("#Choorstraat1")){
                                                msg.payload = '@'+ msg.tweet.user.screen_name + ' The noise level at the
Choorstraat 1 is '+ data.noise+' dB'; }
                                                else if ((msg.test.includes ("#OudeLangendijk1")) ||
(msg.test.includes("#Oudelangendijk1")) ){
                                                    msg.payload = '@'+ msg.tweet.user.screen_name + ' The noise level at the Oude
Langendijk 1 is '+ data.noise+' dB'; }
                                                    else if (msg.test.includes ("#Voldersgracht2")){
                                                        msg.payload = '@'+ msg.tweet.user.screen_name + ' The noise level at the
Voldersgracht 2 is '+ data.noise+' dB'; }
                                                        else if (msg.test.includes ("#Choorstraat2")){
                                                            msg.payload = '@'+ msg.tweet.user.screen_name + ' The noise level at the
Choorstraat 2 is '+ data.noise+' dB'; }
                                                            else if ((msg.test.includes ("#OudeLangendijk2")) ||
(msg.test.includes("#Oudelangendijk2")) ){
                                                                msg.payload = '@'+ msg.tweet.user.screen_name + ' The noise level at the Oude
Langendijk 2 is '+ data.noise+' dB'; }
                                                                else if ((msg.test.includes ("Air Quality")) || (msg.test.includes("Air
quality")) || (msg.test.includes("air quality"))){
                                                                    if (msg.test.includes ("#Voldersgracht1")){
                                                                        msg.payload = '@'+ msg.tweet.user.screen_name + ' The air quality at the
Voldersgracht 1 is '+ data.airquality; }
                                                                        else if (msg.test.includes ("#Choorstraat1")){
                                                                            msg.payload = '@'+ msg.tweet.user.screen_name + ' The air quality at the
Choorstraat 1 is '+ data.airquality; }
                                                                            else if ((msg.test.includes
("#OudeLangendijk1")) || (msg.test.includes("#Oudelangendijk1")) ){
                                                                                msg.payload = '@'+ msg.tweet.user.screen_name + ' The air quality at the Oude
Langendijk 1 is '+ data.airquality; }
                                                                                else if (msg.test.includes ("#Voldersgracht2")){
                                                                                    msg.payload = '@'+ msg.tweet.user.screen_name + ' The air quality at the
Voldersgracht 2 is '+ data.airquality; }
                                                                                    else if (msg.test.includes ("#Choorstraat2")){
                                                                                        msg.payload = '@'+ msg.tweet.user.screen_name + ' The air quality at the
Choorstraat 2 is '+ data.airquality; }
                                                                                        else if ((msg.test.includes
("#OudeLangendijk2")) || (msg.test.includes("#Oudelangendijk2")) ){
                                                                                            msg.payload = '@'+ msg.tweet.user.screen_name + ' The air quality at the Oude
Langendijk 2 is '+ data.airquality; }
                                                                                            else {
                                                                                                msg.payload = '@' + msg.tweet.user.screen_name + ' There has been a mistake, please
try a different request ';;}
                                                                                                    }
                                                                                                    else {
                                                                                                        if ((msg.test.includes ("Temperatuur")) || (msg.test.includes("temperatuur"))){
                                                                                                            if (msg.test.includes ("#Voldersgracht")){

```

```

        msg.payload = '@'+ msg.tweet.user.screen_name + ' De temperatuur op de
Voldersgracht is '+ data.temperature+'°C'; }
    else if (msg.test.includes("#Choorstraat")){
        msg.payload = '@'+ msg.tweet.user.screen_name + ' De temperatuur op de Choorstraat
is '+ data.temperature+'°C'; }
    else if ((msg.test.includes
("#OudeLangendijk"))||(msg.test.includes("Oudelangendijk"))){
        msg.payload = '@'+ msg.tweet.user.screen_name + ' De temperatuur op de Oude
Langendijk is '+ data.temperature+'°C'; }
    else if (msg.test.includes("#Voldersgracht2")){
        msg.payload = '@'+ msg.tweet.user.screen_name + ' De temperatuur op de
Voldersgracht 2 is '+ data.temperature+'°C'; }
    else if (msg.test.includes("#Choorstraat2")){
        msg.payload = '@'+ msg.tweet.user.screen_name + ' De temperatuur op de Choorstraat
2 is '+ data.temperature+'°C'; }
    else if ((msg.test.includes
("#OudeLangendijk2"))||(msg.test.includes("Oudelangendijk2"))){
        msg.payload = '@'+ msg.tweet.user.screen_name + ' De temperatuur op de Oude
Langendijk 2 is '+ data.temperature+'°C'; }}
    else if ((msg.test.includes ("Vochtigheid"))||(msg.test.includes("vochtigheid"))){
        if (msg.test.includes ("#Voldersgracht")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De vochtigheid op de
Voldersgracht is '+ data.humidity; }
        else if (msg.test.includes ("#Choorstraat")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De vochtigheid op de Choorstraat
is '+ data.humidity; }
        else if ((msg.test.includes
("#OudeLangendijk"))||(msg.test.includes("#Oudelangendijk")) ){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De vochtigheid op de Oude
Langendijk 1 is '+ data.humidity; }
        else if (msg.test.includes ("#Voldersgracht2")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De vochtigheid op de
Voldersgracht 2 is '+ data.humidity; }
        else if (msg.test.includes ("#Choorstraat2")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De vochtigheid op de Choorstraat
2 is '+ data.humidity; }
        else if ((msg.test.includes
("#OudeLangendijk2"))||(msg.test.includes("#Oudelangendijk2")) ){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De vochtigheid op de Oude
Langendijk 2 is '+ data.humidity; }}
    else if ((msg.test.includes ("Geluid"))|| (msg.test.includes("geluid"))){
        if (msg.test.includes ("#Voldersgracht")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' Het geluidsniveau op de
Voldersgracht is '+ data.noise+' dB'; }
        else if (msg.test.includes ("#Choorstraat")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' Het geluidsniveau op de
Choorstraat is '+ data.noise+' dB'; }
        else if ((msg.test.includes ("#OudeLangendijk")) ||
(msg.test.includes("#Oudelangendijk")) ){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' Het geluidsniveau op de Oude
Langendijk is '+ data.noise+' dB'; }
        else if (msg.test.includes ("#Voldersgracht2")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' Het geluidsniveau op de
Voldersgracht 2 is '+ data.noise+' dB'; }
        else if (msg.test.includes ("#Choorstraat2")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' Het geluidsniveau op de
Choorstraat 2 is '+ data.noise+' dB'; }
        else if ((msg.test.includes ("#OudeLangendijk2")) ||
(msg.test.includes("#Oudelangendijk2")) ){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' Het geluidsniveau op de Oude
Langendijk 2 is '+ data.noise+' dB'; }}
    else if ((msg.test.includes ("Lucht")) || (msg.test.includes("lucht"))){
        if (msg.test.includes ("#Voldersgracht")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De luchtkwaliteit op de
Voldersgracht is '+ data.airquality; }
        else if (msg.test.includes ("#Choorstraat")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De luchtkwaliteit op de
Choorstraat is '+ data.airquality; }
        else if ((msg.test.includes
("#OudeLangendijk"))||(msg.test.includes("#Oudelangendijk")) ){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De luchtkwaliteit op de Oude
Langendijk is '+ data.airquality; }
        else if (msg.test.includes ("#Voldersgracht2")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De luchtkwaliteit op de
Voldersgracht 2 is '+ data.airquality; }
        else if (msg.test.includes ("#Choorstraat2")){
            msg.payload = '@'+ msg.tweet.user.screen_name + ' De luchtkwaliteit op de

```

```
Choorstraat 2 is '+ data.airquality; }
    else if ((msg.test.includes
("#OudeLangendijk2")) || (msg.test.includes("#Oudelangendijk2"))) ){
        msg.payload = '@'+ msg.tweet.user.screen_name + ' De luchtkwaliteit op de Oude
Langendijk 2 is '+ data.airquality; }}
    else {
        msg.payload = '@' + msg.tweet.user.screen_name + ' Er ging iets mis, probeer het
opnieuw';}}

return msg;
```

5: Tweet block

The Dashboard flow in Node-RED visualizes the data in graphs. It takes the last 2 hours of measurements, coordinated per street. PostgreSQL fetches this data from the database. In the function blocks the device addresses are linked to the right streets. Node-RED provides an intuitive user interface to plot the data (figure 91). Function blocks have the code to fetch the right data.

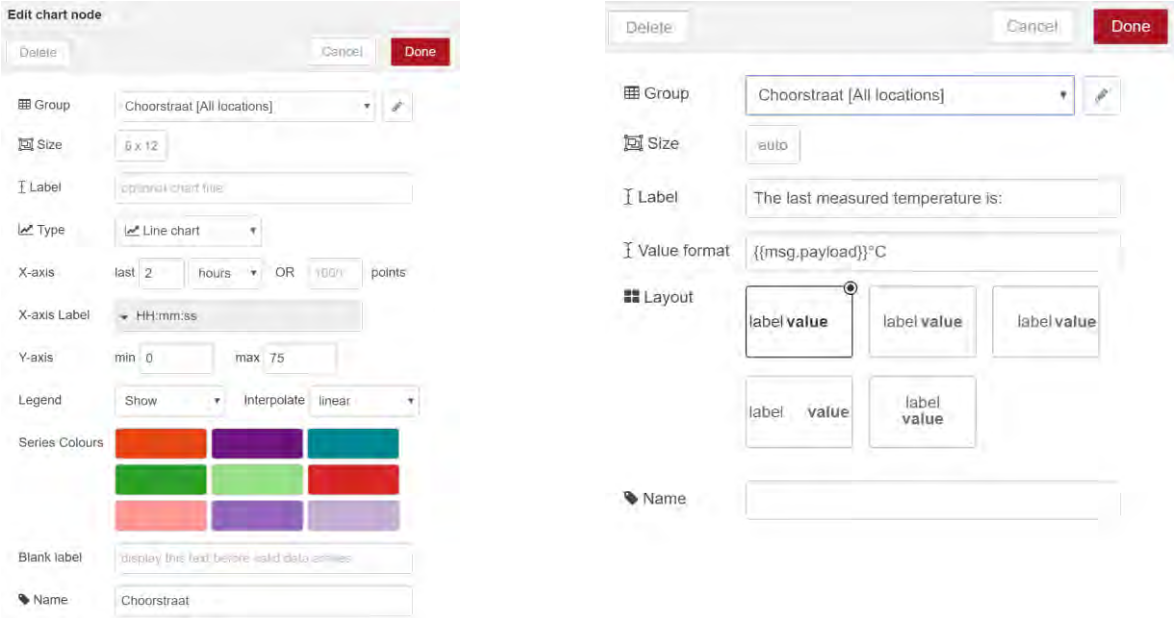


Figure 91: APPENDIX O - Data plotting interface in Node-RED

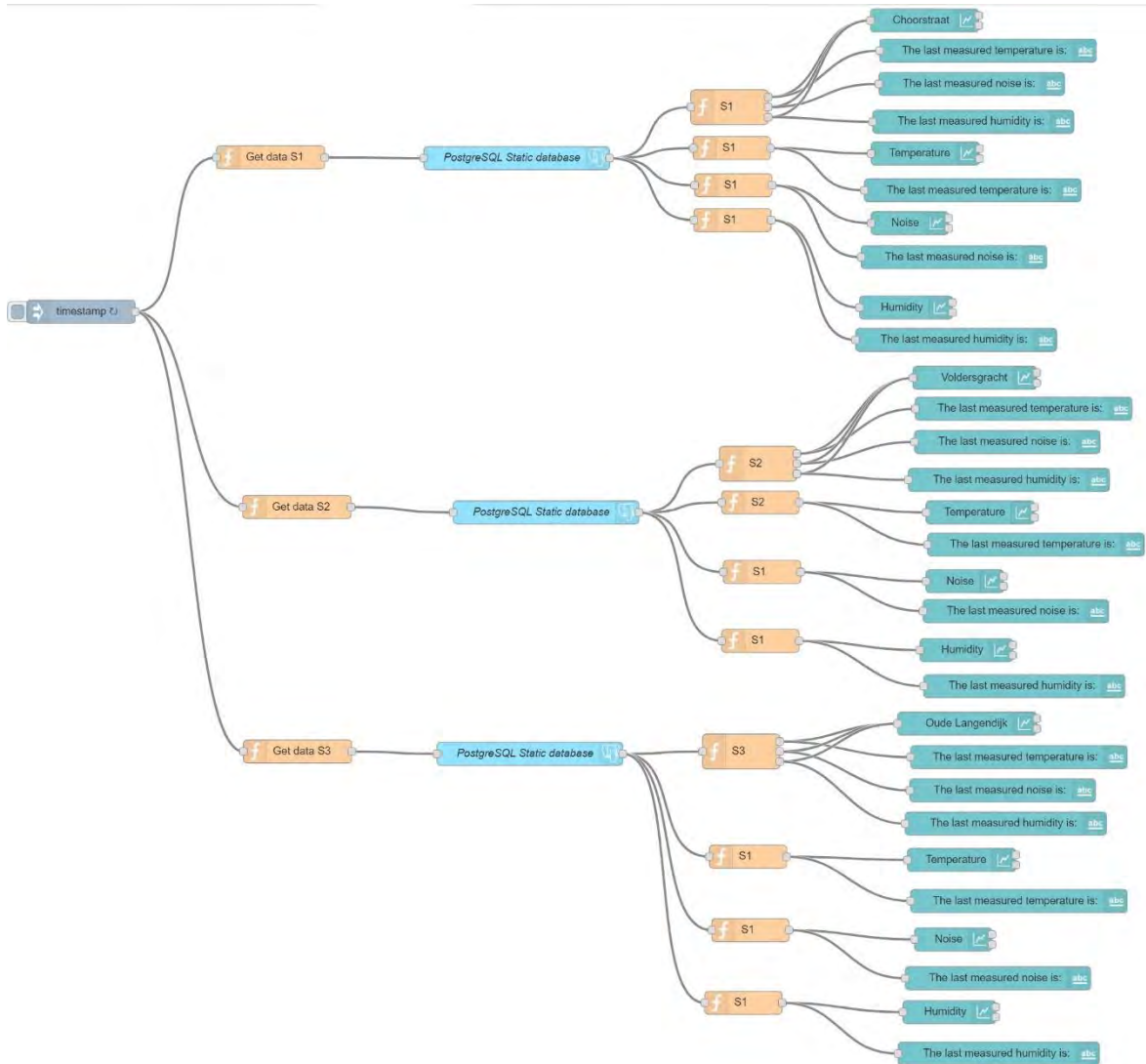


Figure 92: APPENDIX R - Dashboard flow in Node-RED

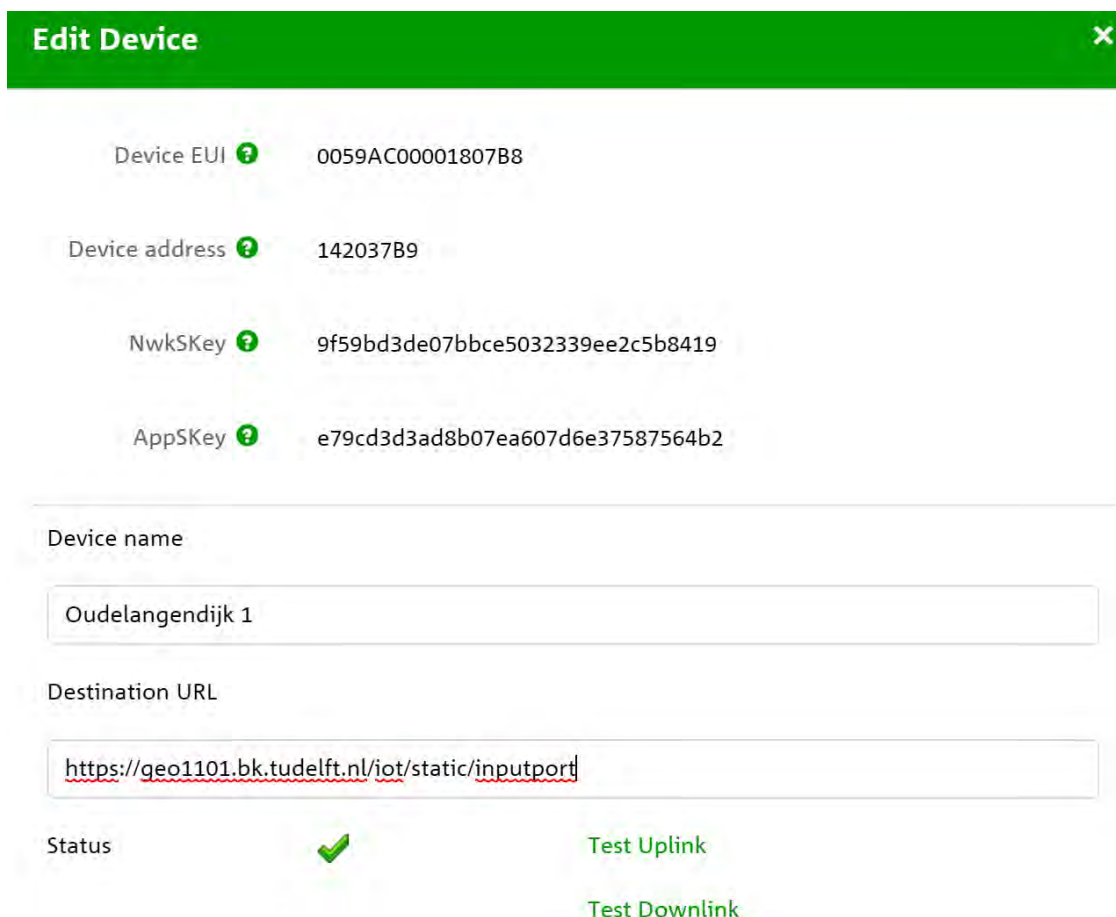
Appendix S: LoRa encryption

On the online forum of KPN (KPN Zakelijk) the explanation of the LoRa encryption is explained; see the next two sections:

“KPN sends the payload of LoRa messages in an encrypted way. The LoRa payload is by default encrypted with AES-128 bits encryption, based on the generic algorithm described in IEEE 802.15.4/2006 Annex B [IEEE802154] as described in the LoRaWAN specifications” (Jol, 2016a).

“Payload information is encrypted with an Application Session Key (AppSKey). This key needs to be shared between device and customer Application Server. The AppSKey does not have to be shared with the network operator, but developers can choose to share the AppSKey with KPN to decrypt the payload and sent the decrypted payload over a secure https connection.” (Jol, 2016b).

In the Node-RED environment a public library is installed (node-red-contrib-loradecrypt) to take care of this decryption. The information that should be provided is the AppsKey and the Device address. Both of which are provided by KPN (figure 93). An example of how the encryption works is shown in figure 94.



The screenshot shows the 'Edit Device' interface in the KPN Developer Portal. It features a green header bar with the title 'Edit Device' and a close button. Below the header, there are four rows of device information, each with a label, a help icon, and a value:

- Device EUI: 0059AC00001807B8
- Device address: 142037B9
- NwkSKey: 9f59bd3de07bbce5032339ee2c5b8419
- AppSKey: e79cd3d3ad8b07ea607d6e37587564b2

Below this information, there are two input fields:

- 'Device name' with the value 'Oudelangendijk 1'.
- 'Destination URL' with the value 'https://geo1101.bk.tudelft.nl/iot/static/inputport'.

At the bottom, there is a 'Status' section showing a green checkmark icon, and two buttons: 'Test Uplink' and 'Test Downlink'.

Figure 93: APPENDIX S - Device information KPN Developer Portal (screenshot from KPN website)

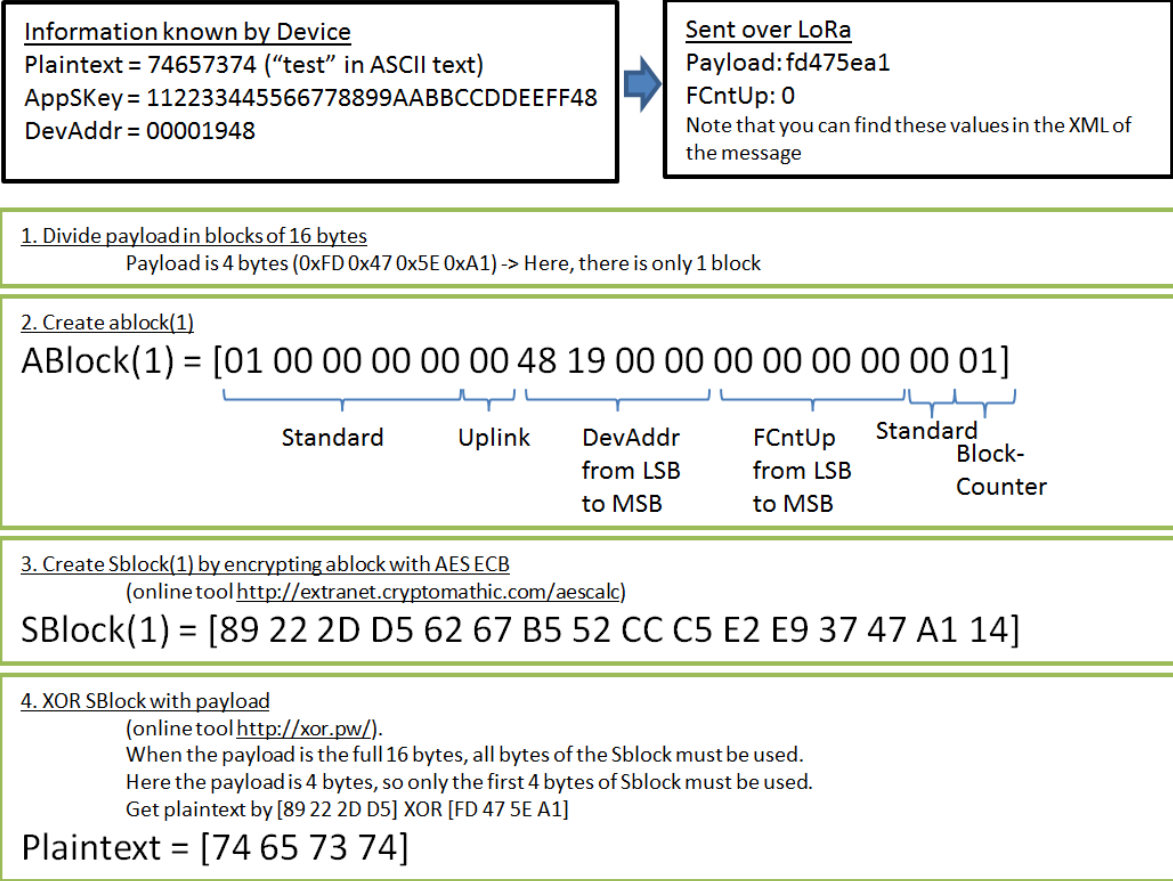


Figure 94: APPENDIX S - Decryption scheme example KPN LoRa message (Source: Jol, 2016a)

Appendix T: Test script MAX9814

```
import machine
from math import log
import time
import pycom

SAMPLE_WINDOW = 50 # Sample window width in ms (50 ms = 20Hz)
sample = 0
adc = machine.ADC(bits=10) # create an ADC object
apin = adc.channel(pin='P16') # create an analog pin on P16=G3

# timing object for measuring low pulse
chrono = machine.Timer.Chrono()

def median(data):
    sorteddata = sorted(data)
    dataLen = len(data)
    index = (dataLen - 1) // 2

    if (dataLen % 2):
        return(sorteddata[index])
    else:
        return(sorteddata[index] + sorteddata[index + 1])/2.0

def capriati():
    db_values = []
    while True:
        chrono.start()
        start_time = chrono.read_ms()

        peak_to_peak = 0 # peak-to-peak level
        signal_max = 0 # max signal
        signal_min = 1024 # min signals

        while (chrono.read_ms() - start_time) < SAMPLE_WINDOW: # 50 ms
            sample = apin() # read an analog value
            if sample > signal_max:
                signal_max = sample
            elif sample < signal_min:
                signal_min = sample
            peak_to_peak = signal_max - signal_min

        if (peak_to_peak > 0):
            log_value=((peak_to_peak/1024)+1)
            dB= 2 * int(round(100*(log(log_value,10)))) #calculate sound level
        else:
            dB = 0

        chrono.stop()
        chrono.reset()
        print(dB)
        if dB < 35: # lights based on sound levels (not necessary for test script)
            pycom.rgbled(0xff0000)
        elif 35 < dB < 40:
            pycom.rgbled(0x0000ff)
        elif dB > 50:
            pycom.rgbled(0x00ff00)
        db_values.append(dB)
        time.sleep(1)
        if len(db_values) == 10:
            avg = sum(db_values)/len(db_values)
            print('Average: ', avg)
            mediandB = median(db_values)
            print('Median: ', mediandB)
            break

def run():
    for x in range(0, 10):
        capriati()
```

Appendix U: Test script AM2302

```
"""
Test script for the Temperature and Humidity sensor AM2302
The DTH class that needs to be imported from the dth module is available at
https://github.com/JurassicPork/DHT\_PyCom (the dth project is licensed under the terms of the
MIT license).
"""

from dth import DTH

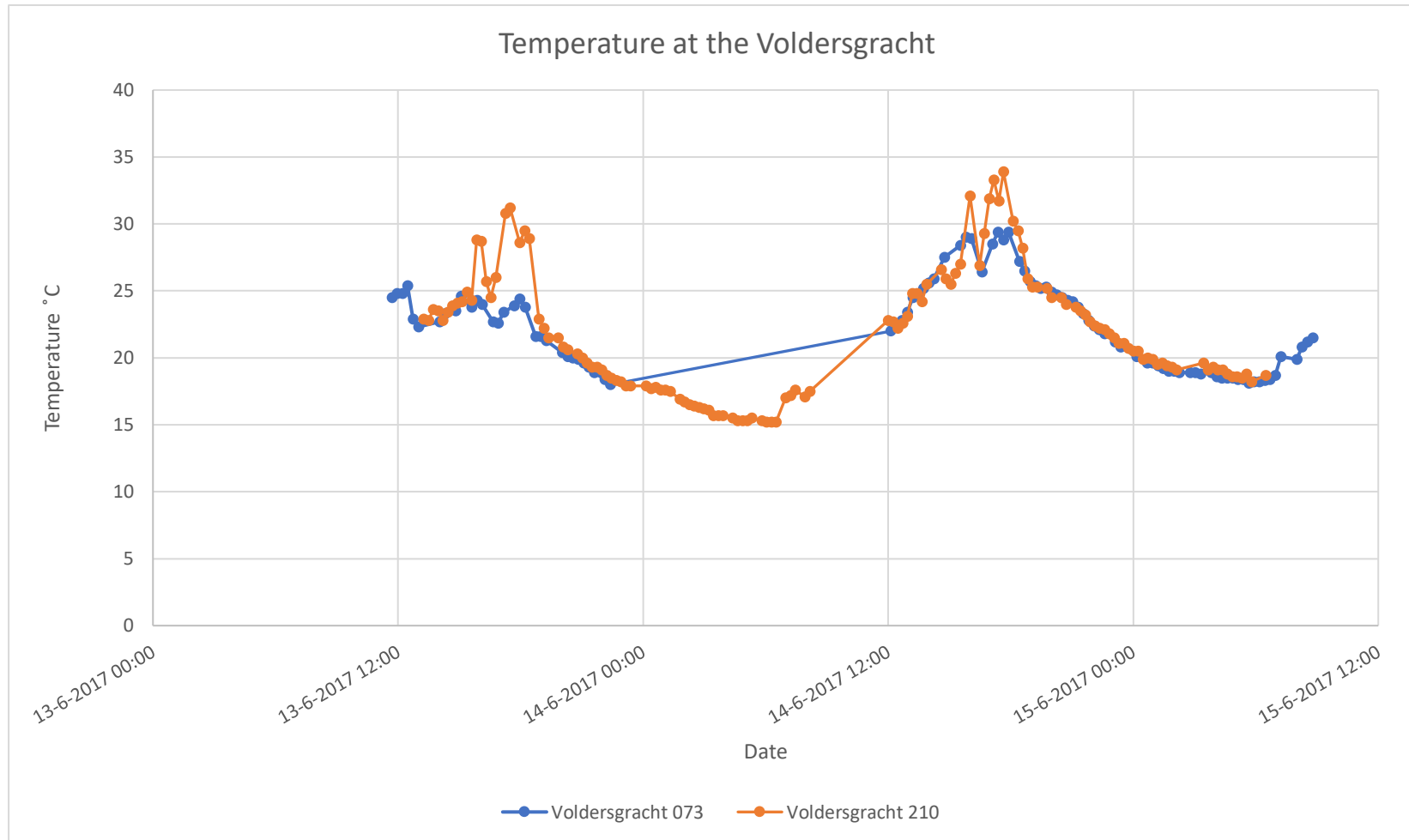
th = DTH(Pin('P4', mode=Pin.OPEN_DRAIN), 1)

def temp_hum():
    temperatures = []
    humidities = []
    while True:
        result = th.read()
        print(result, type(result))
        print(result.is_valid())
        # if result.is_valid():
        print("#", len(temperatures)+1)
        print('Temperature: {:.2f}'.format(result.temperature / 1.0))
        print('Humidity: {:.2f}'.format(result.humidity / 1.0))
        temperatures.append(result.temperature)
        humidities.append(result.humidity)
        if len(temperatures) == 10:
            sum_temp = sum(temperatures)
            avg_temp = sum_temp / 10.0
            print(" ")
            print("Summary about the last 20 seconds:")
            print(" ")
            print('The average temperature was: {:.2f}'.format(avg_temp))
            print('The maximum temperature was: {:.2f}'.format(max(temperatures)))
            print('The minimum temperature was: {:.2f}'.format(min(temperatures)))
            temperatures = []
        if len(humidities) == 10:
            sum_hum = sum(humidities)
            avg_hum = sum_hum / 10.0
            print(" ")
            print('The average humidity was: {:.2f}'.format(avg_hum))
            print('The maximum humidity was: {:.2f}'.format(max(humidities)))
            print('The minimum humidity was: {:.2f}'.format(min(humidities)))
            humidities = []
        time.sleep(5)

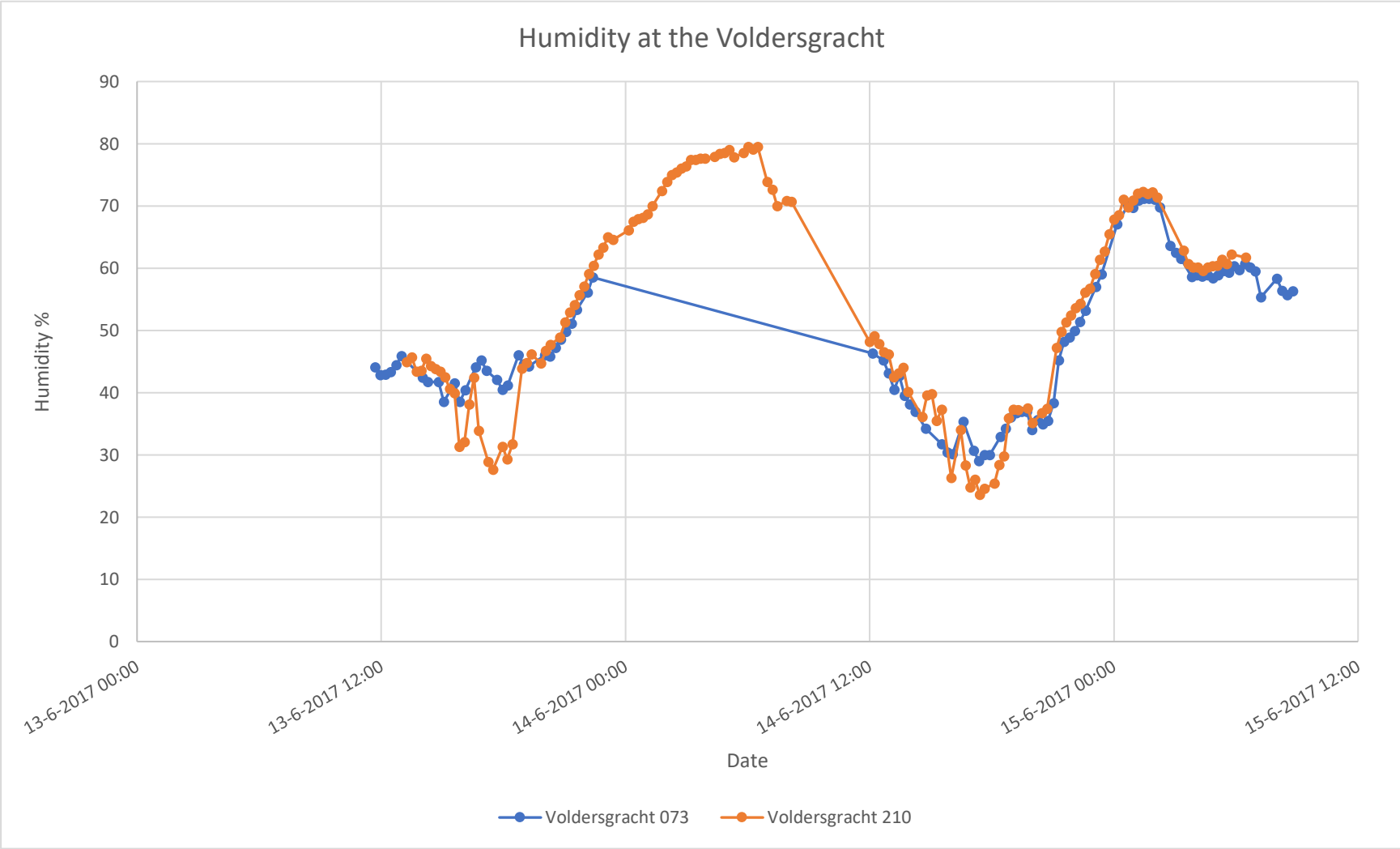
for x in range(0, 10):
    temp_hum()
```

Appendix V: Graphs and tables for AM2302 measurements

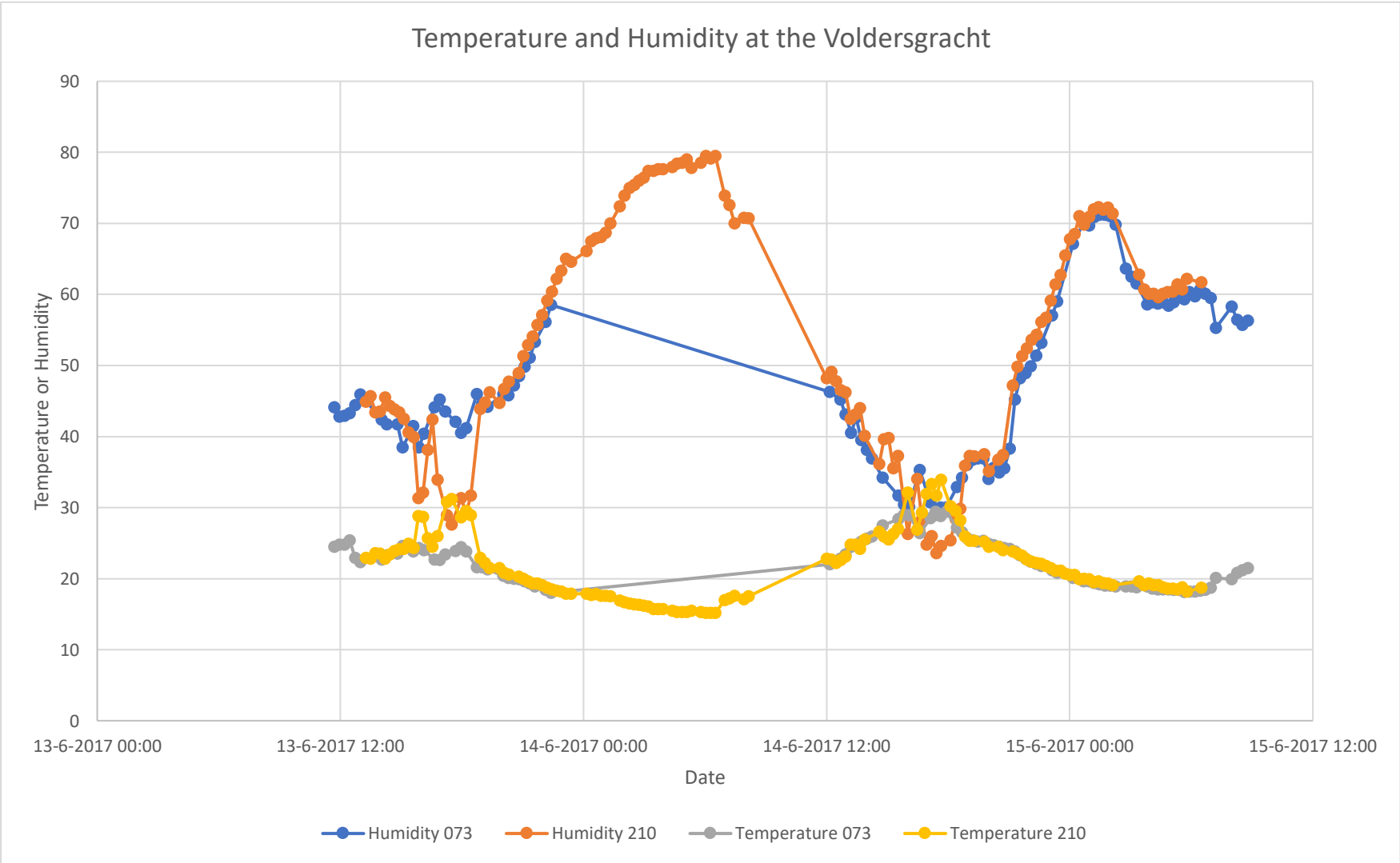
Temperature at the Voldersgracht



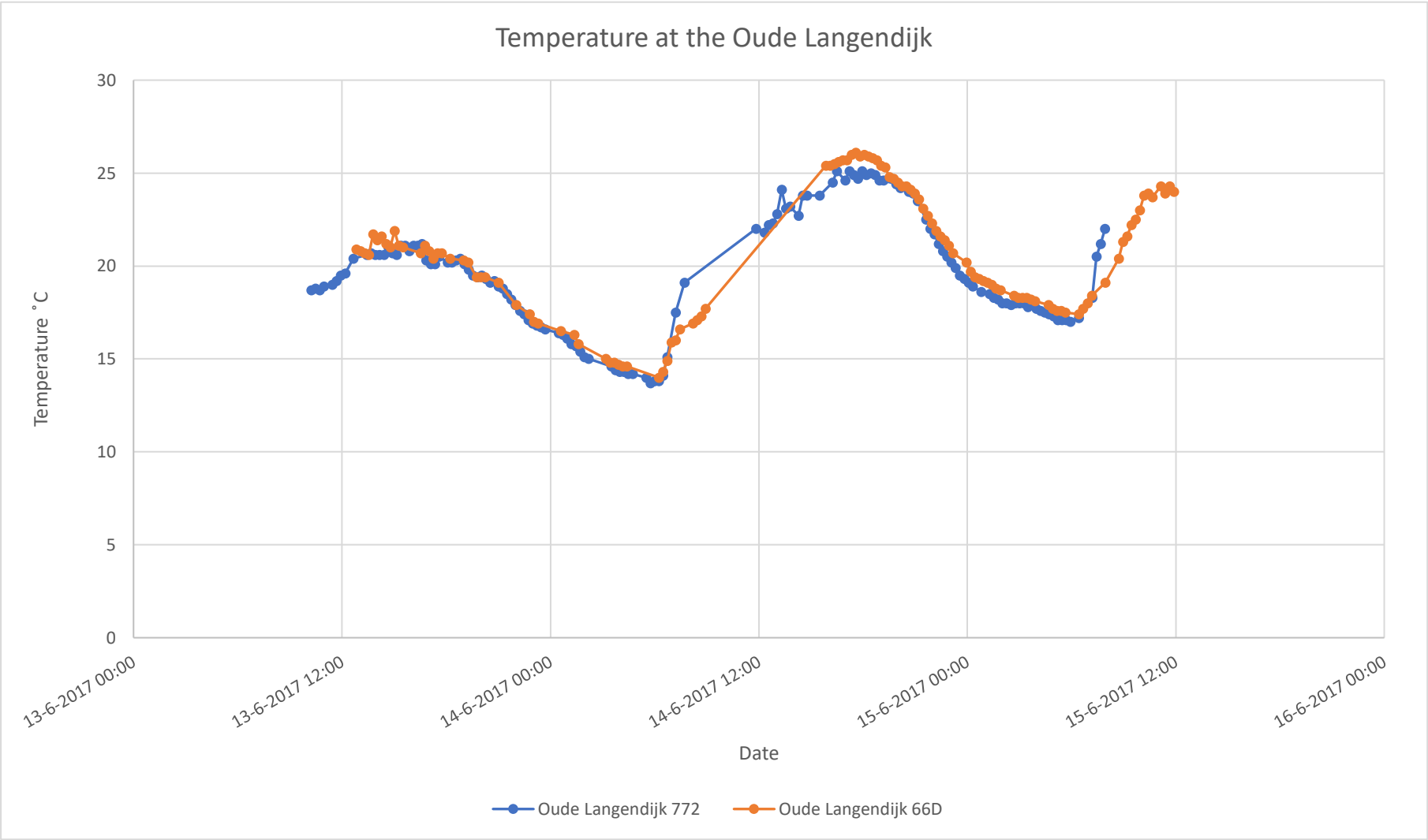
Humidity at the Voldersgracht



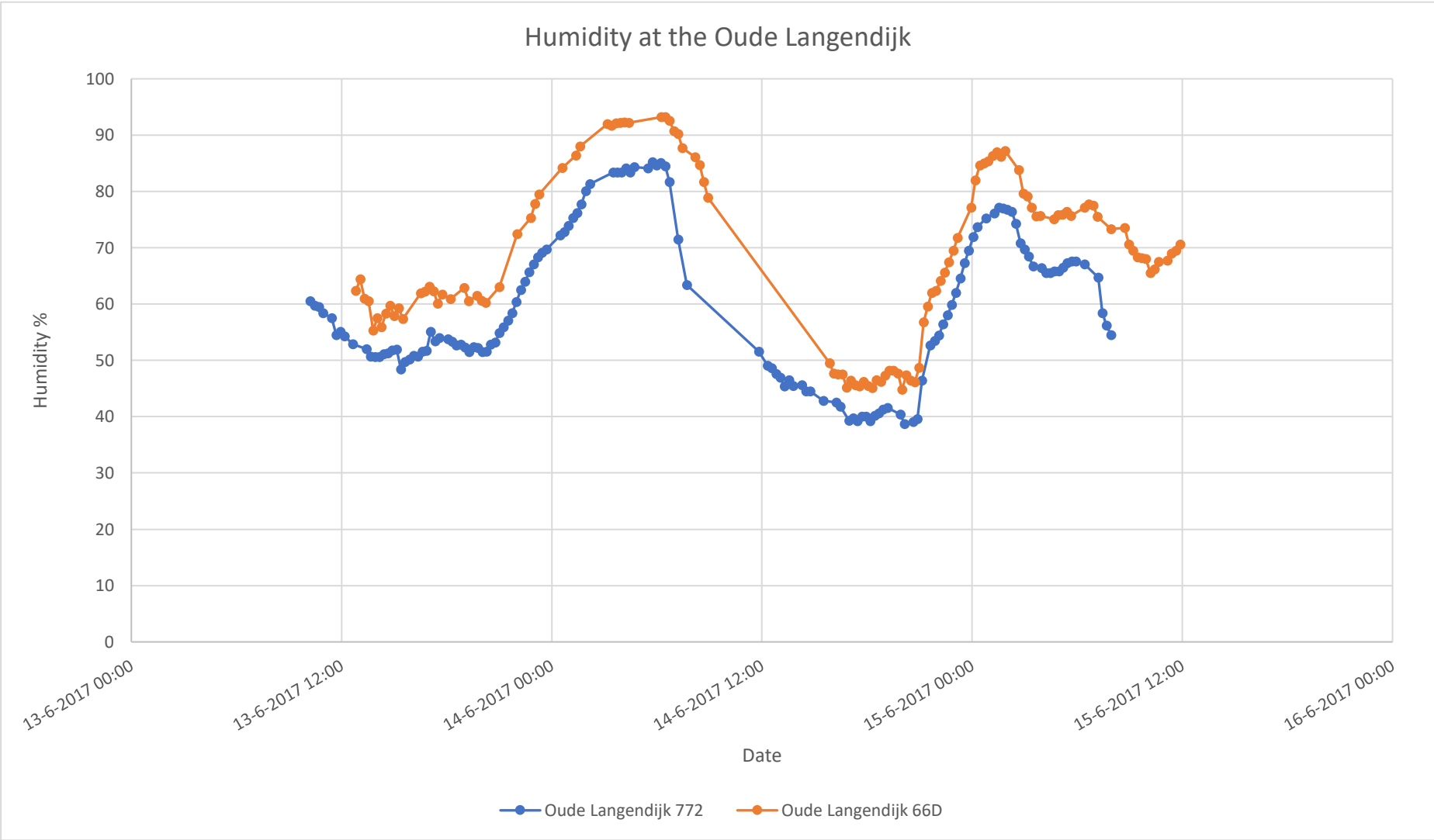
Temperature and humidity at the Voldersgracht



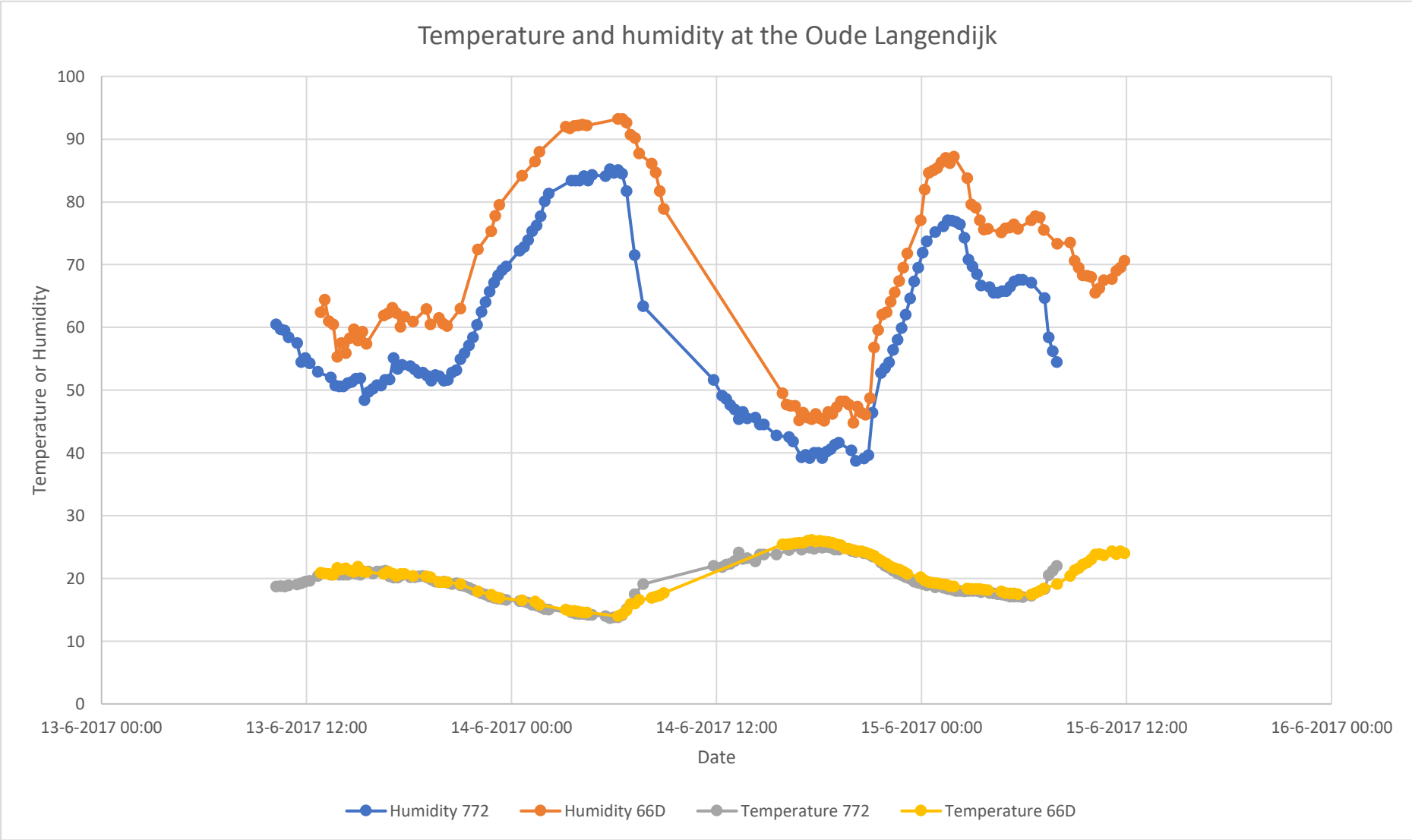
Temperature at the Oude Langendijk



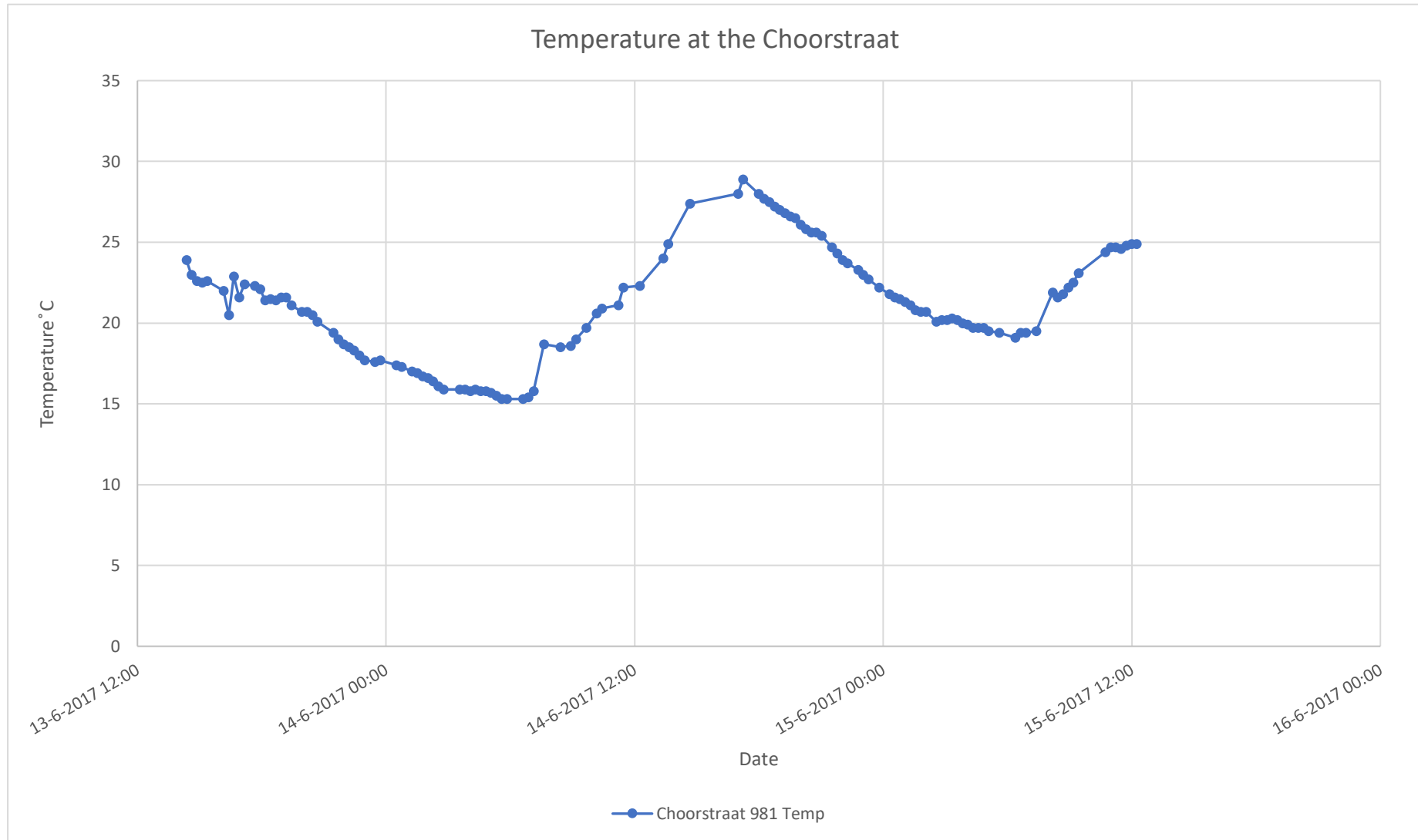
Humidity at the Oude Langendijk



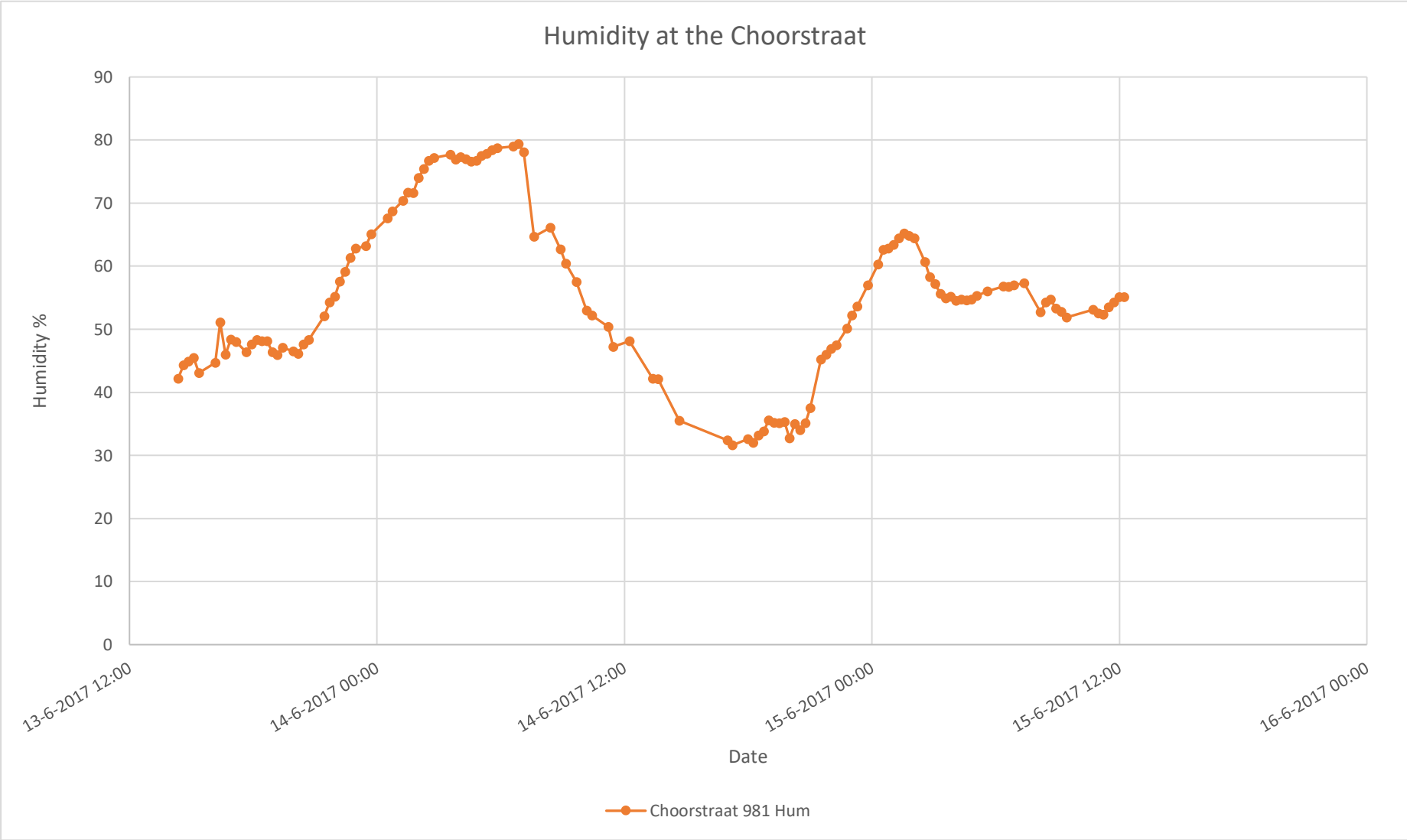
Temperature and humidity at the Oude Langendijk



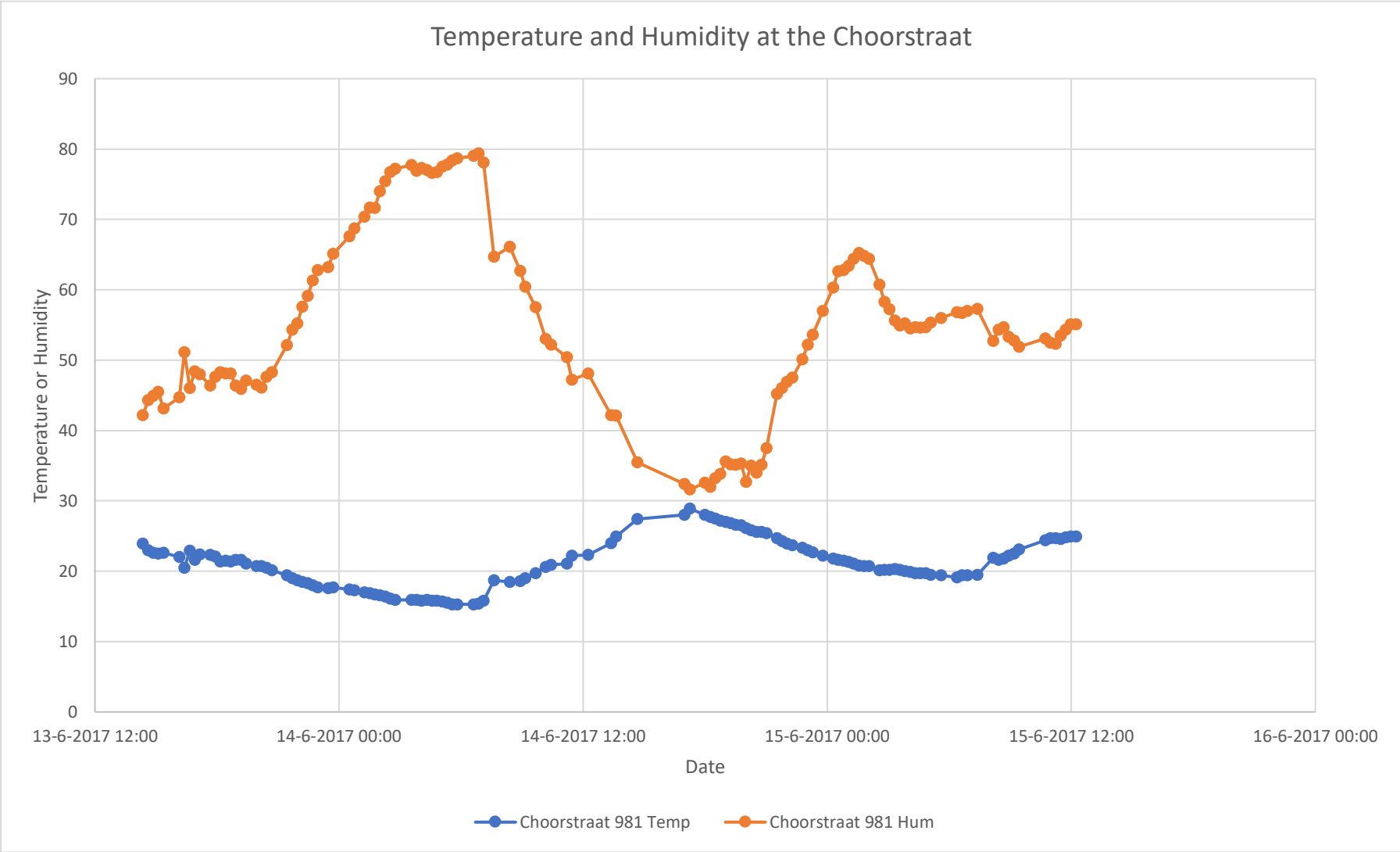
Temperature at the Choorstraat



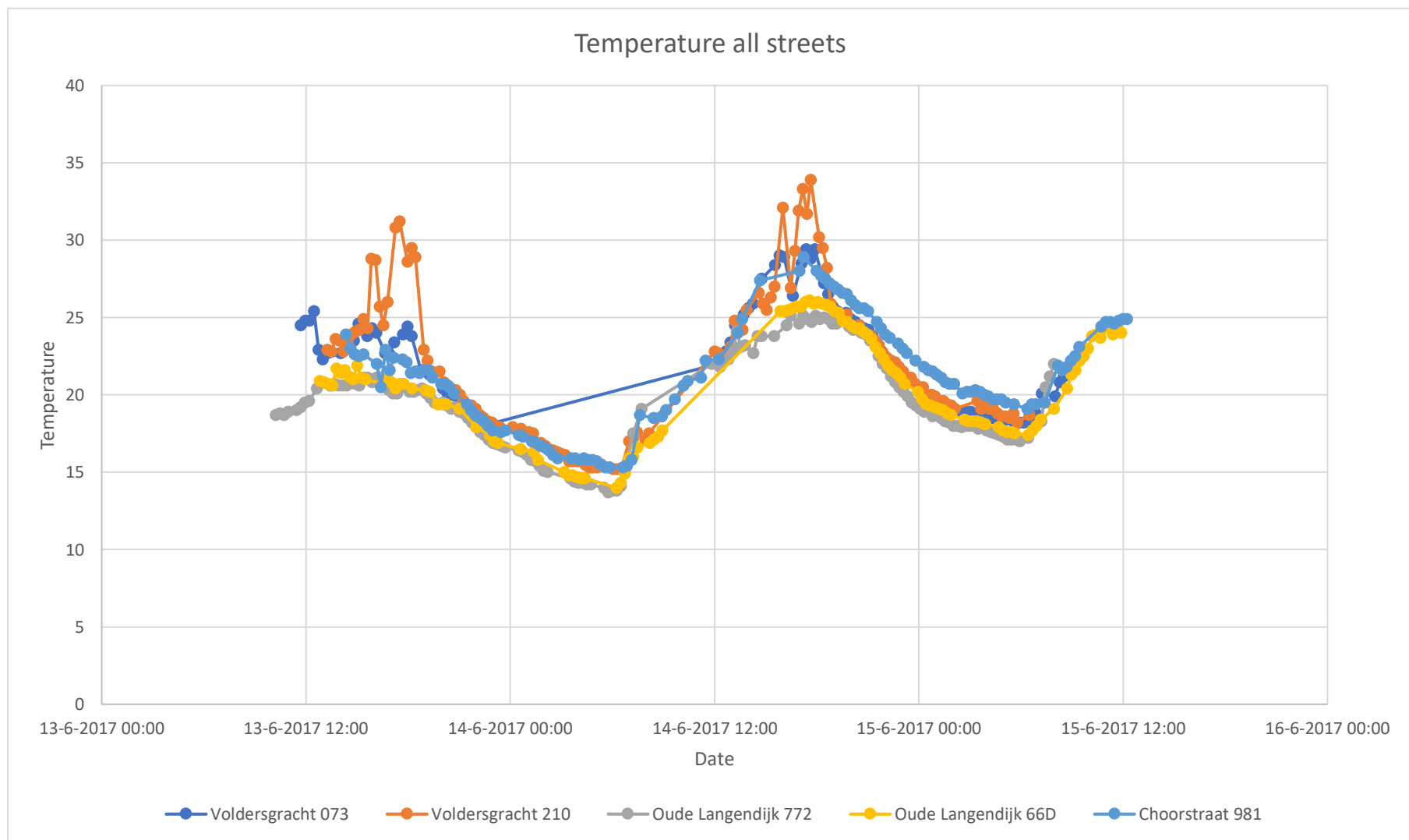
Humidity at the Choorstraat



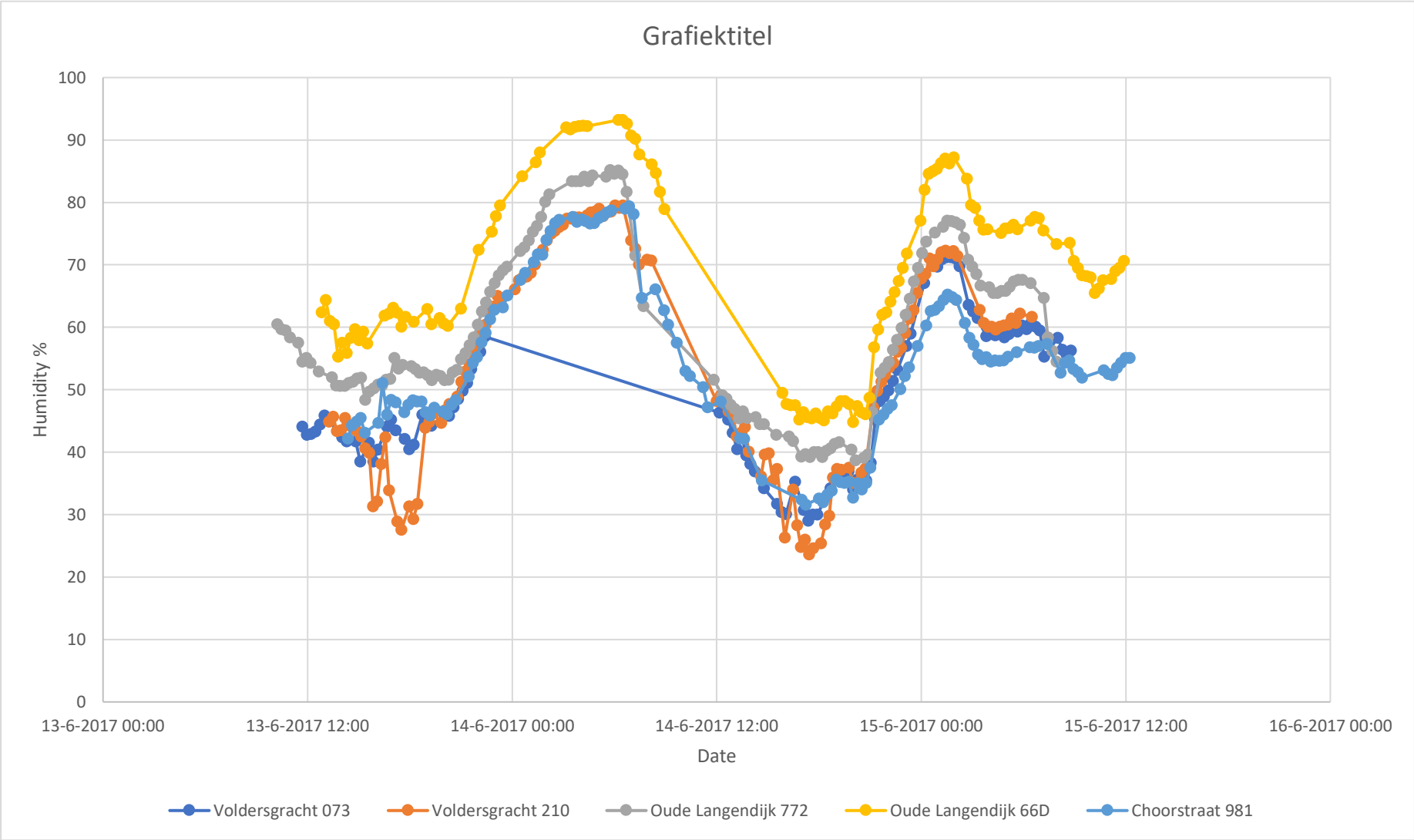
Temperature and humidity at the Choorstraat



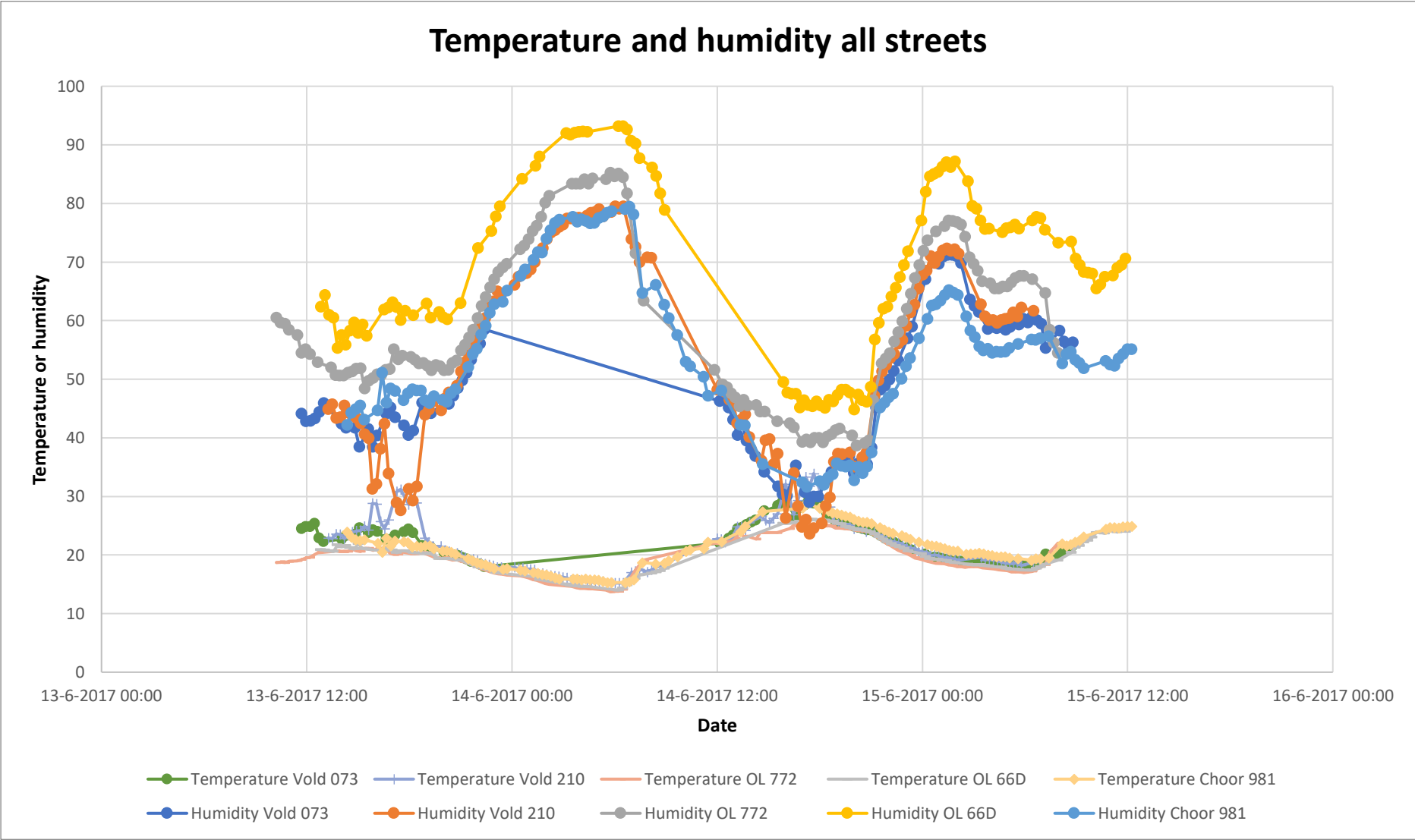
Temperature all streets



Humidity all streets



Temperature and humidity all streets



Metadata of sensor 981 Choorstraat

| Temperature | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 1 | 32 | 32 | 32 | 0 |
| 14:01 - 18:00 | 12 | 22,367 | 23,9 | 20,5 | 0,822781682 |
| 18:01 - 22:00 | 13 | 20,592 | 21,6 | 18,7 | 1,009506099 |
| 22:01 - 02:00 | 11 | 17,555 | 18,5 | 16,7 | 0,569848465 |
| 02:01 - 06:00 | 14 | 15,85 | 16,6 | 15,3 | 0,361088099 |
| 06:00 - 10:00 | 8 | 17,625 | 19,7 | 15,3 | 1,802973734 |
| 10:00 - 14:00 | 7 | 22,28571429 | 24,9 | 20,6 | 1,630366596 |
| 14:00 - 18:00 | 5 | 28,9 | 32,2 | 27,4 | 1,920937271 |
| 18:00 - 22:00 | 14 | 26,2 | 27,7 | 24,3 | 1,022064276 |
| 22:00 - 2:00 | 13 | 22,12307692 | 23,9 | 20,7 | 1,095562143 |
| 2:00 - 6:00 | 13 | 19,96923077 | 20,7 | 19,4 | 0,363741244 |
| 6:00 - 10:00 | 10 | 21,05 | 23,1 | 19,1 | 1,522607413 |
| 10:00 - 14:00 | 7 | 24,71428571 | 24,9 | 24,4 | 0,177281052 |
| Total: | 128 | 21,26640625 | | | |

| Humidity | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|--------|------|------|-------------|
| 10:01 - 14:00 | 1 | 28,6 | 28,6 | 28,6 | 0 |
| 14:01 - 18:00 | 12 | 48,769 | 55,2 | 45,9 | 3,090950397 |
| 18:01 - 22:00 | 13 | 65,373 | 71,7 | 57,6 | 4,978773124 |
| 22:01 - 02:00 | 11 | 76,993 | 78,7 | 74 | 1,185466014 |
| 02:01 - 06:00 | 14 | 70,057 | 79,4 | 60,4 | 8,403344006 |

| | | | | | |
|----------------------|-----|-------------|------|------|-------------|
| 06:00 - 10:00 | 8 | 68,4875 | 79,4 | 57,5 | 8,957588563 |
| 10:00 - 14:00 | 7 | 47,88571429 | 53 | 42,1 | 4,423583981 |
| 14:00 - 15:00 | 5 | 31,9 | 35,5 | 27,4 | 2,917190429 |
| 18:00 - 22:00 | 14 | 36,12142857 | 46 | 32 | 4,24412328 |
| 22:00 - 2:00 | 13 | 57,75384615 | 65,2 | 46,9 | 6,872483222 |
| 2:00 - 6:00 | 13 | 56,62307692 | 64,4 | 54,5 | 2,949902215 |
| 6:00 - 10:00 | 10 | 54,75 | 57,3 | 51,9 | 2,054939848 |
| 10:00 - 14:00 | 7 | 53,7 | 55,1 | 52,3 | 1,160459679 |
| Total: | 128 | 54,45625 | | | |

Metadata of sensor 073 Voldersgracht

| Temperature | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 6 | 24,11666667 | 25,4 | 22,3 | 1,225425096 |
| 14:01 - 18:00 | 12 | 23,6 | 24,6 | 22,6 | 0,686228162 |
| 18:01 - 22:00 | 11 | 20,59090909 | 23,8 | 18,9 | 1,394599975 |
| 22:01 - 02:00 | 2 | 18,2 | 18,4 | 18 | 0,282842712 |
| 02:01 - 06:00 | -- | -- | -- | -- | -- |
| 06:00 - 10:00 | -- | -- | -- | -- | -- |
| 10:00 - 14:00 | 7 | 24,02857143 | 25,6 | 22 | 1,325033692 |
| 14:00 - 18:00 | 10 | 28,22 | 29,4 | 25,9 | 1,225470431 |
| 18:00 - 22:00 | 14 | 24,84285714 | 27,2 | 22,8 | 1,182379794 |
| 22:00 - 2:00 | 12 | 20,35 | 22,4 | 19 | 1,25589519 |
| 2:00 - 6:00 | 13 | 18,58461538 | 18,9 | 18,1 | 0,276424052 |
| 6:00 - 10:00 | 9 | 19,67777778 | 21,5 | 18,2 | 1,31318104 |
| Total: | 96 | 22,415625 | | | |

| Humidity | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 6 | 43,9 | 45,9 | 42,8 | 1,171324037 |
| 14:01 - 18:00 | 12 | 41,675 | 45,2 | 38,5 | 2,034754843 |
| 18:01 - 22:00 | 11 | 47,06363636 | 53,3 | 41,2 | 3,424404394 |
| 22:01 - 02:00 | 2 | 57,3 | 58,5 | 56,1 | 1,697056275 |
| 02:01 - 06:00 | -- | -- | -- | -- | -- |
| 06:00 - 10:00 | -- | -- | -- | -- | -- |

| | | | | | |
|----------------------|----|-------------|------|------|-------------|
| 10:00 - 14:00 | 7 | 42,2 | 46,3 | 38,1 | 2,996108587 |
| 14:00 - 18:00 | 10 | 31,83 | 36,9 | 29 | 2,675007788 |
| 18:00 - 22:00 | 14 | 38,15714286 | 48,9 | 32,9 | 5,266835904 |
| 22:00 - 2:00 | 12 | 63,475 | 71,2 | 49,9 | 8,657642446 |
| 2:00 - 6:00 | 13 | 60,69230769 | 69,8 | 58,4 | 3,186550261 |
| 6:00 - 10:00 | 9 | 58 | 60,7 | 55,3 | 2,090454496 |
| Total: | 96 | 48,0875 | | | |

Metadata of sensor 210 Voldersgracht

| Temperature | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 4 | 23,2 | 23,6 | 22,8 | 0,40824829 |
| 14:01 - 18:00 | 15 | 26,12666667 | 31,2 | 22,8 | 2,757189736 |
| 18:01 - 22:00 | 14 | 21,82142857 | 29,5 | 19,1 | 3,325930424 |
| 22:01 - 02:00 | 13 | 17,88461538 | 18,7 | 16,9 | 0,468768162 |
| 02:01 - 06:00 | 15 | 15,83333333 | 16,7 | 15,3 | 0,489411697 |
| 06:00 - 10:00 | 8 | 16,5 | 17,6 | 15,2 | 1,094140237 |
| 10:00 - 14:00 | 9 | 23,63333333 | 25,5 | 22,2 | 1,198957881 |
| 14:00 - 18:00 | 12 | 29,2 | 33,9 | 25,5 | 3,172896583 |
| 18:00 - 22:00 | 14 | 25,41428571 | 30,2 | 22,7 | 2,318440173 |
| 22:00 - 2:00 | 17 | 20,67647059 | 22,4 | 19,3 | 1,043149924 |
| 2:00 - 6:00 | 12 | 18,9 | 19,6 | 18,2 | 0,388470193 |
| Total: | 134 | 21,70223881 | | | |

| Humidity | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 4 | 44,375 | 45,7 | 43,4 | 1,117661249 |
| 14:01 - 18:00 | 15 | 37,70666667 | 45,5 | 27,6 | 6,2094935 |
| 18:01 - 22:00 | 14 | 46,78571429 | 57,1 | 29,3 | 8,098989219 |
| 22:01 - 02:00 | 13 | 65,79230769 | 72,4 | 59,1 | 3,84739339 |
| 02:01 - 06:00 | 15 | 77,12 | 79 | 73,9 | 1,470276942 |
| 06:00 - 10:00 | 8 | 74,5125 | 79,5 | 70 | 4,200488917 |
| 10:00 - 14:00 | 9 | 45,27777778 | 49,1 | 40,1 | 3,015699661 |

| | | | | | |
|----------------------|-----|-------------|------|------|-------------|
| 14:00 - 18:00 | 12 | 31,325 | 39,8 | 23,6 | 6,27855296 |
| 18:00 - 22:00 | 14 | 38,67142857 | 52,4 | 25,4 | 8,486317365 |
| 22:00 - 2:00 | 17 | 65,04705882 | 72,3 | 53,6 | 6,898833022 |
| 2:00 - 6:00 | 12 | 61,65 | 71,4 | 59,6 | 3,210069385 |
| Total: | 134 | 53,55746269 | | | |

Metadata sensor 771 Oude Langendijk

| Temperature | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 12 | 19,55833333 | 20,7 | 18,7 | 0,803920319 |
| 14:01 - 18:00 | 15 | 20,72666667 | 21,2 | 20,1 | 0,371227052 |
| 18:01 - 22:00 | 17 | 19,37058824 | 20,4 | 17,9 | 0,749803896 |
| 22:01 - 02:00 | 14 | 16,42142857 | 17,6 | 15,1 | 0,73920807 |
| 02:01 - 06:00 | 10 | 14,25 | 15 | 13,7 | 0,377859468 |
| 06:00 - 10:00 | 5 | 15,92 | 19,1 | 13,8 | 2,296083622 |
| 10:00 - 14:00 | 8 | 22,6875 | 24,1 | 21,8 | 0,764269025 |
| 14:00 - 18:00 | 11 | 24,37272727 | 25,1 | 22,7 | 0,7630084 |
| 18:00 - 22:00 | 12 | 24,04166667 | 25 | 22 | 0,954852043 |
| 22:00 - 2:00 | 14 | 19,62142857 | 21,7 | 18,2 | 1,121934162 |
| 2:00 - 6:00 | 16 | 17,59375 | 18 | 17 | 0,382044936 |
| 6:00 - 10:00 | 5 | 19,84 | 22 | 17,2 | 2,018167486 |
| Total: | 139 | 19,57122302 | | | |

| Humidity | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 12 | 55,475 | 60,5 | 50,6 | 3,563737823 |
| 14:01 - 18:00 | 15 | 51,48666667 | 55,1 | 48,4 | 1,686020957 |
| 18:01 - 22:00 | 17 | 53,92941176 | 60,4 | 51,5 | 2,585479817 |
| 22:01 - 02:00 | 14 | 71,04285714 | 80,1 | 62,5 | 5,287389102 |
| 02:01 - 06:00 | 10 | 83,72 | 85,2 | 81,3 | 1,046475566 |
| 06:00 - 10:00 | 5 | 77,24 | 85,1 | 63,4 | 9,471958615 |

| | | | | | |
|----------------------|-----|-------------|------|------|-------------|
| 10:00 - 14:00 | 8 | 47,65 | 51,6 | 45,4 | 2,07639798 |
| 14:00 - 18:00 | 11 | 41,80909091 | 45,6 | 39,2 | 2,332575635 |
| 18:00 - 22:00 | 12 | 42,775 | 53,5 | 38,7 | 5,224091918 |
| 22:00 - 2:00 | 14 | 67,36428571 | 77,1 | 54,4 | 8,127503841 |
| 2:00 - 6:00 | 16 | 68,825 | 76,8 | 65,5 | 3,805872655 |
| 6:00 - 10:00 | 5 | 60,18 | 67,1 | 54,5 | 5,46781492 |
| Total: | 139 | 59,51366906 | | | |

Metadata of sensor 66D Oude Langendijk

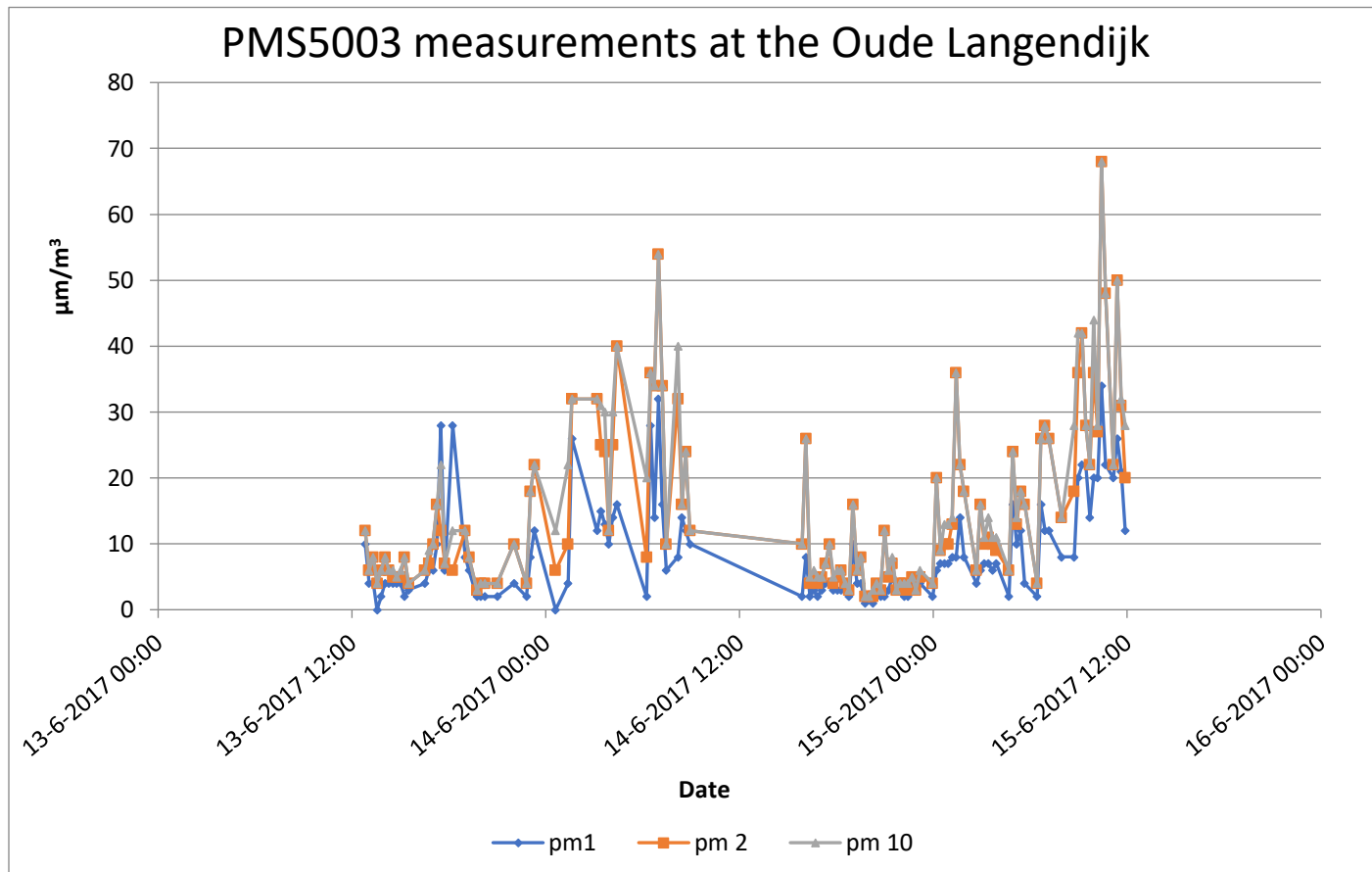
| Temperature | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 5 | 20,94 | 21,7 | 20,6 | 0,439317653 |
| 14:01 - 18:00 | 13 | 21,04615385 | 21,9 | 20,4 | 0,411532471 |
| 18:01 - 22:00 | 7 | 19,74285714 | 20,4 | 19,1 | 0,534967734 |
| 22:01 - 02:00 | 7 | 16,82857143 | 17,9 | 15,8 | 0,701698619 |
| 02:01 - 06:00 | 6 | 14,75 | 15 | 14,6 | 0,151657509 |
| 06:00 - 10:00 | 10 | 16,07 | 17,7 | 14 | 1,291897829 |
| 10:00 - 14:00 | -- | | | | |
| 14:00 - 18:00 | 9 | 25,7 | 26,1 | 25,4 | 0,254950976 |
| 18:00 - 22:00 | 17 | 24,49411765 | 26 | 22,3 | 1,128312118 |
| 22:00 - 2:00 | 14 | 20,00714286 | 21,9 | 18,7 | 1,122815339 |
| 2:00 - 6:00 | 11 | 17,99090909 | 18,4 | 17,5 | 0,338982435 |
| 6:00 - 10:00 | 11 | 20,14545455 | 23 | 17,4 | 2,09015006 |
| 10:00 - 14:00 | 7 | 23,98571429 | 24,3 | 23,7 | 0,234012617 |
| Total: | 117 | 20,5017094 | | | |

| Humidity | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|------|------|-------------|
| 10:01 - 14:00 | 5 | 60,72 | 64,4 | 55,3 | 3,386295911 |
| 14:01 - 18:00 | 13 | 59,79230769 | 63,1 | 55,9 | 2,298355042 |
| 18:01 - 22:00 | 7 | 61,37142857 | 63 | 60,2 | 1,151396667 |
| 22:01 - 02:00 | 7 | 80,51428571 | 88 | 72,4 | 5,855034464 |
| 02:01 - 06:00 | 6 | 92,08333333 | 92,3 | 91,7 | 0,213697606 |

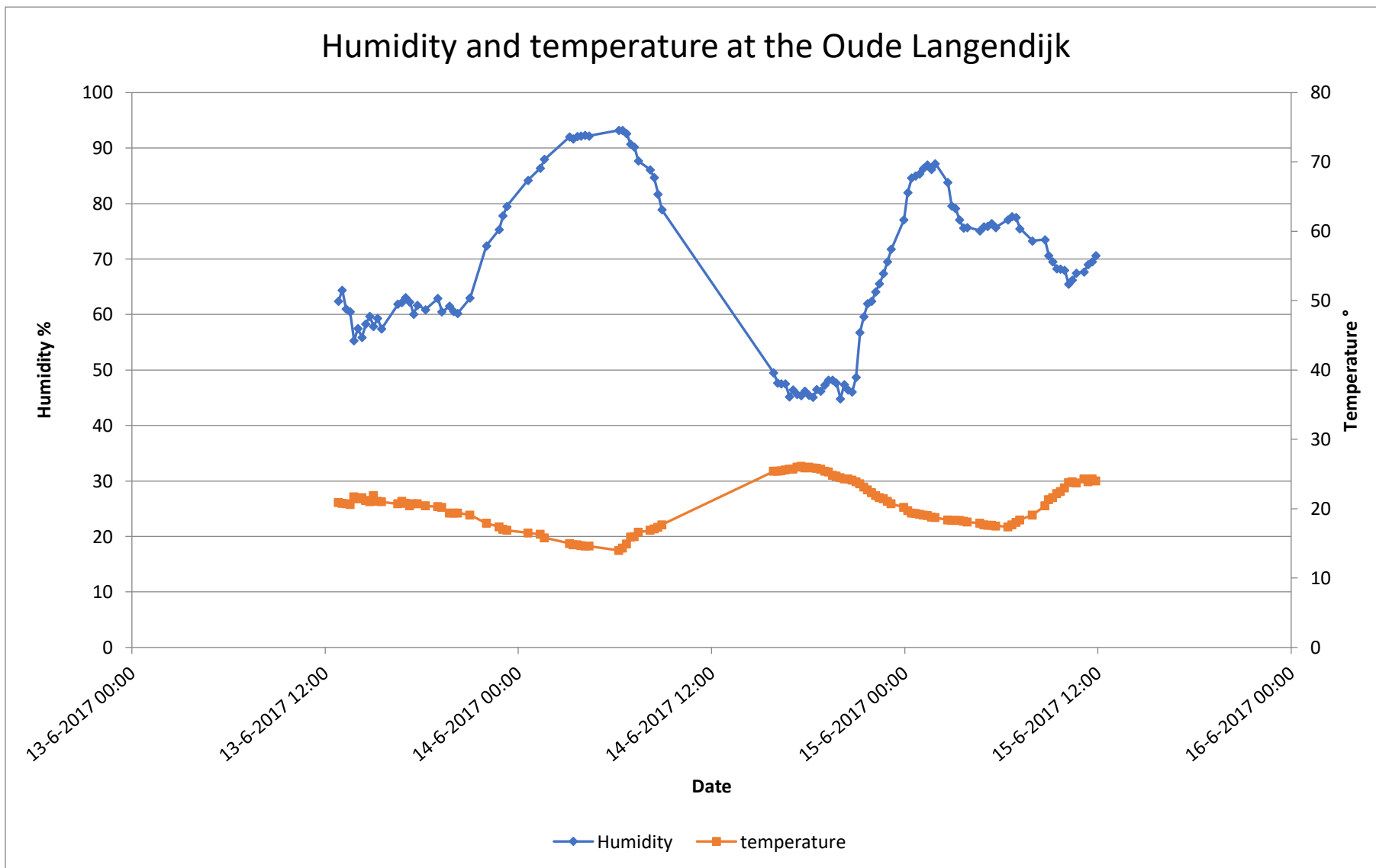
| | | | | | |
|----------------------|-----|-------------|------|------|-------------|
| 06:00 - 10:00 | 10 | 87,9 | 93,2 | 78,9 | 4,992883825 |
| 10:00 - 14:00 | | | | | |
| 14:00 - 15:00 | 9 | 46,77777778 | 49,5 | 45,2 | 1,394433378 |
| 18:00 - 22:00 | 17 | 49,93529412 | 62,4 | 44,8 | 6,071237639 |
| 22:00 - 2:00 | 14 | 78,51428571 | 87,2 | 64,1 | 8,910741758 |
| 2:00 - 6:00 | 11 | 77,25454545 | 83,8 | 75,1 | 2,620444098 |
| 6:00 - 10:00 | 11 | 72,65454545 | 77,7 | 68 | 3,905986082 |
| 10:00 - 14:00 | 7 | 68 | 70,6 | 65,5 | 1,818424226 |
| Total: | 117 | 68,37350427 | | | |

Appendix W: Graphs and tables for air quality measurements

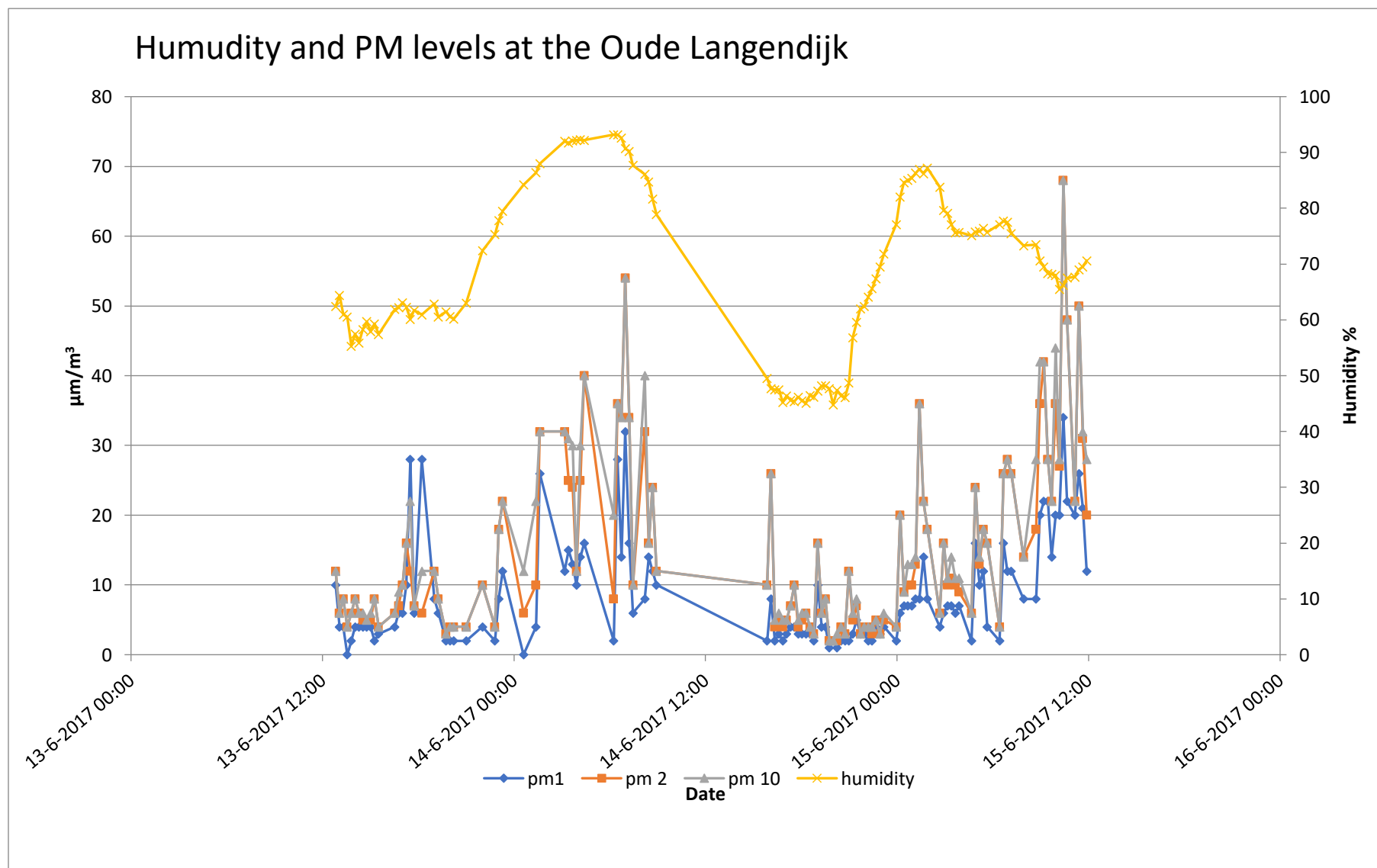
PMS measurements at the Oude Langendijk



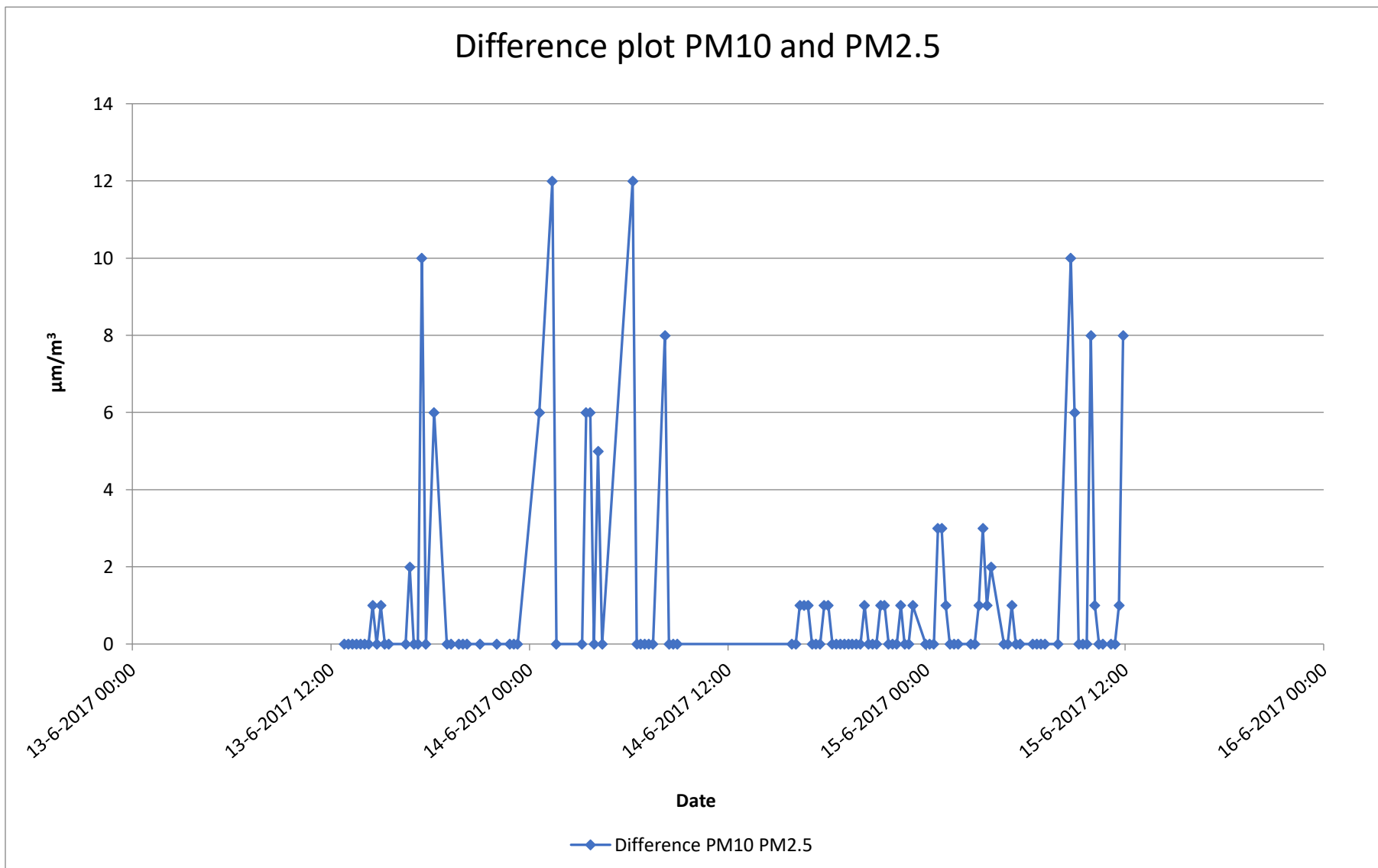
Humidity and temperature at the Oude Langendijk



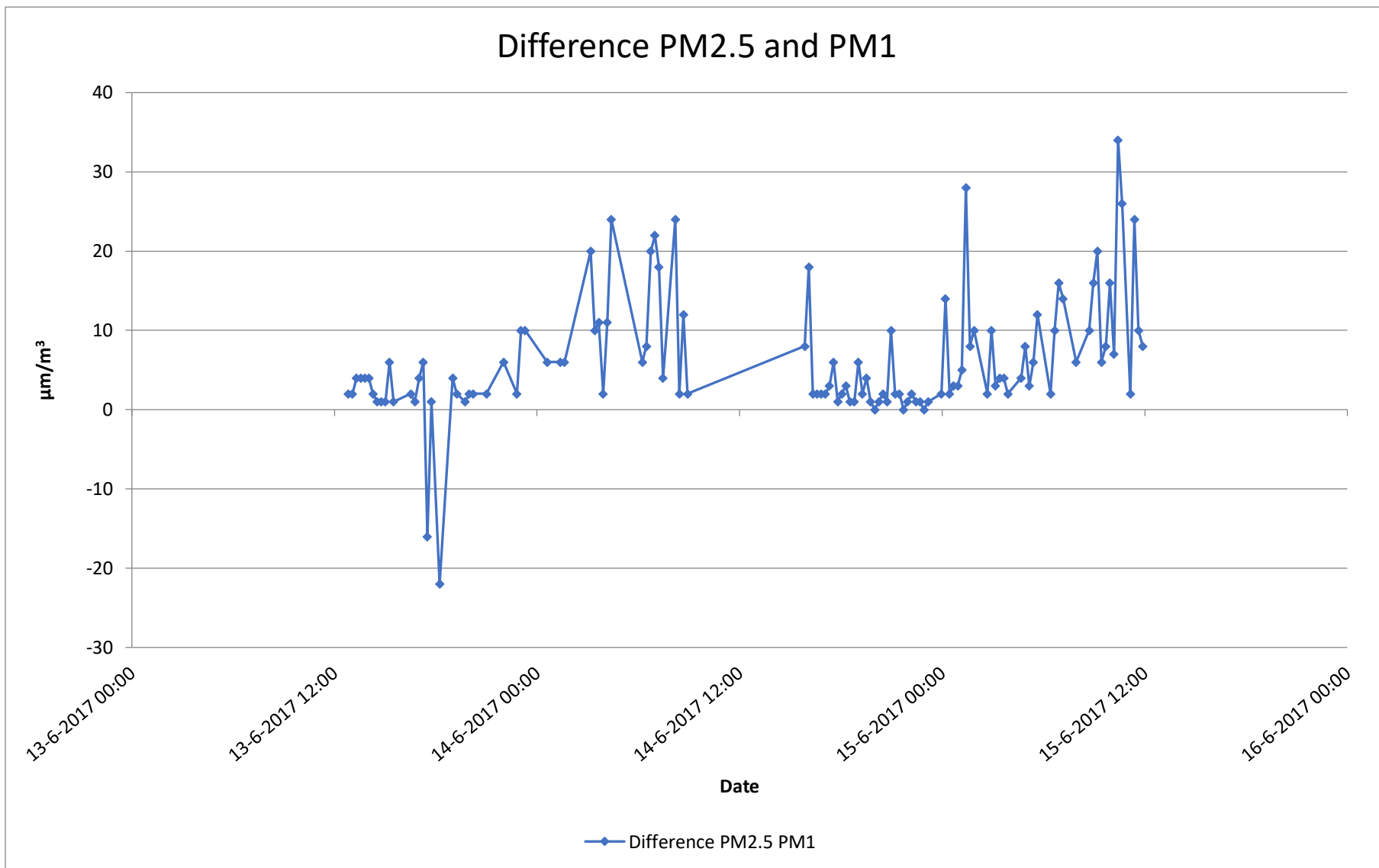
Humidity and PM levels at the Oude Langendijk



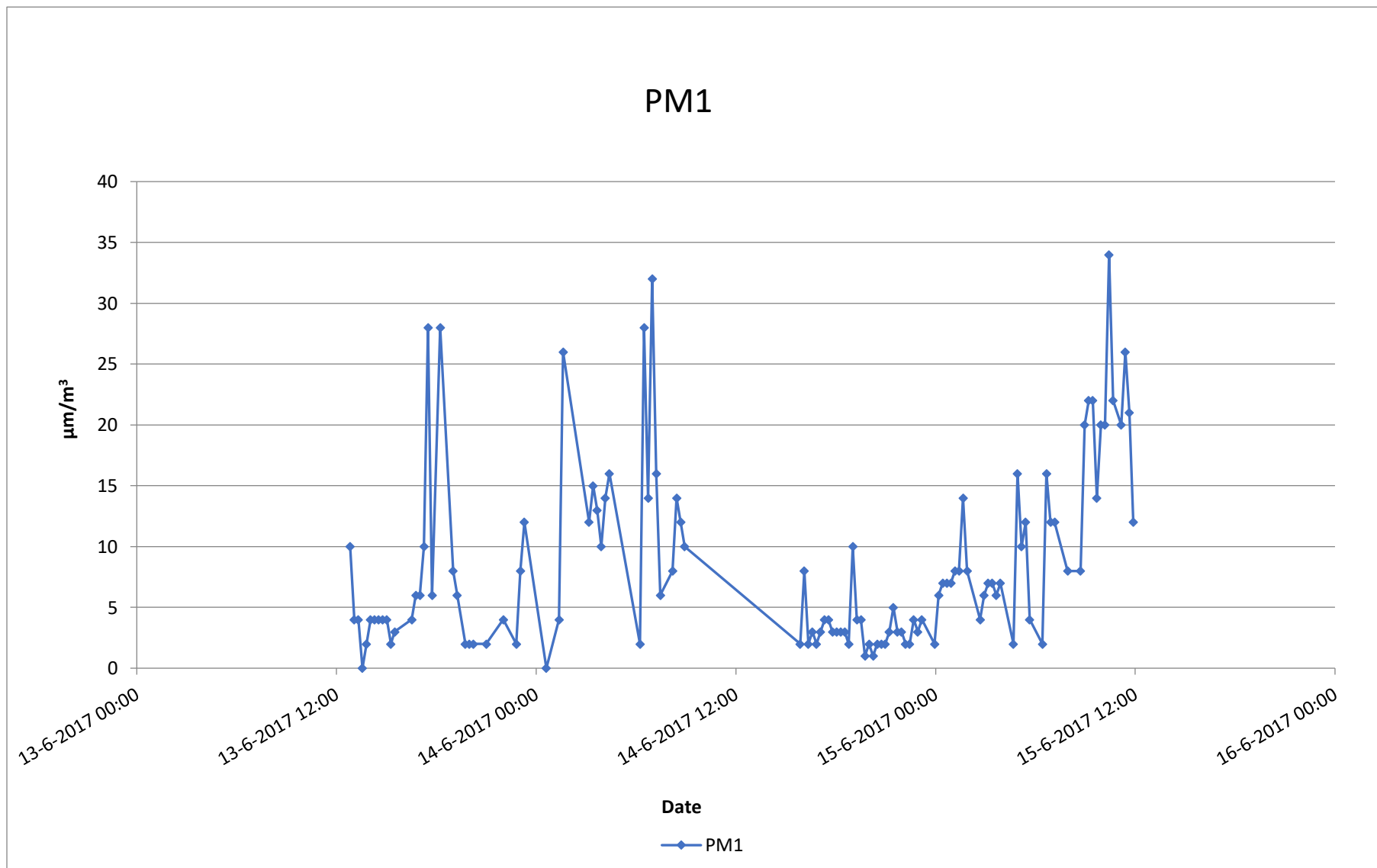
Difference plot PM10 and PM2.5 at the Oude Langendijk



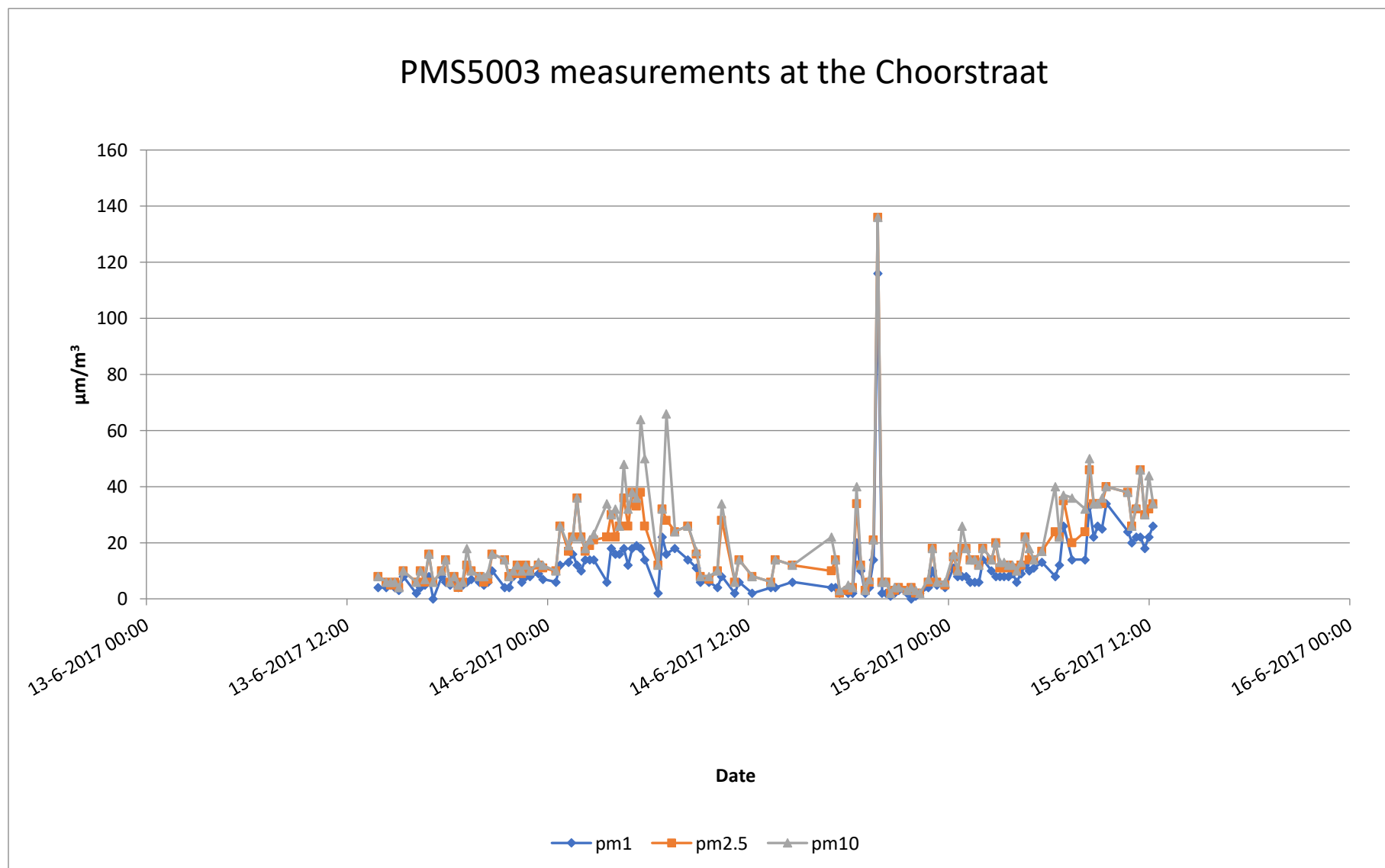
Difference plot PM2.5 and PM1 at the Oude Langendijk



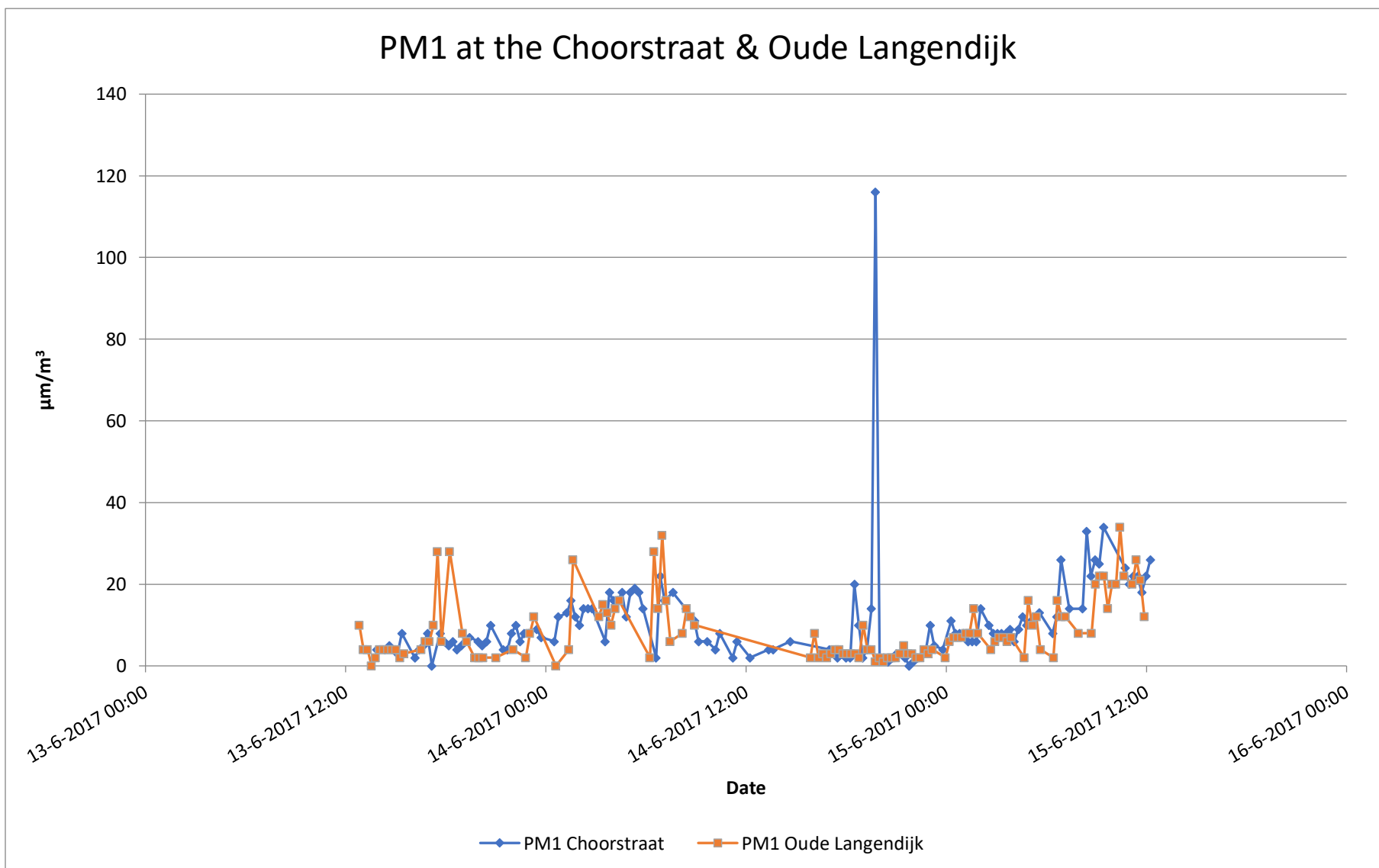
PM1 measurements at the Oude Langendijk



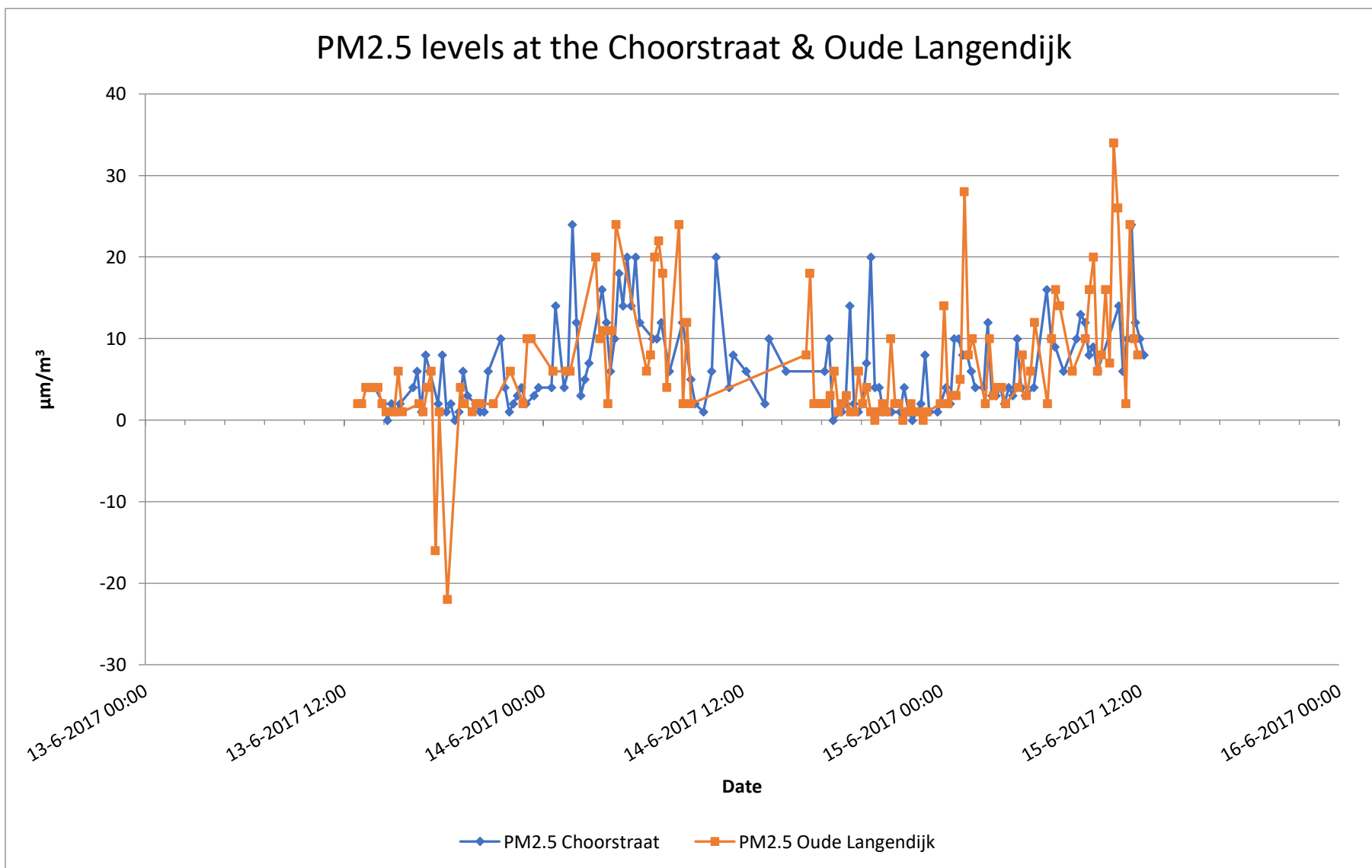
PMS5003 measurements at the Choorstraat



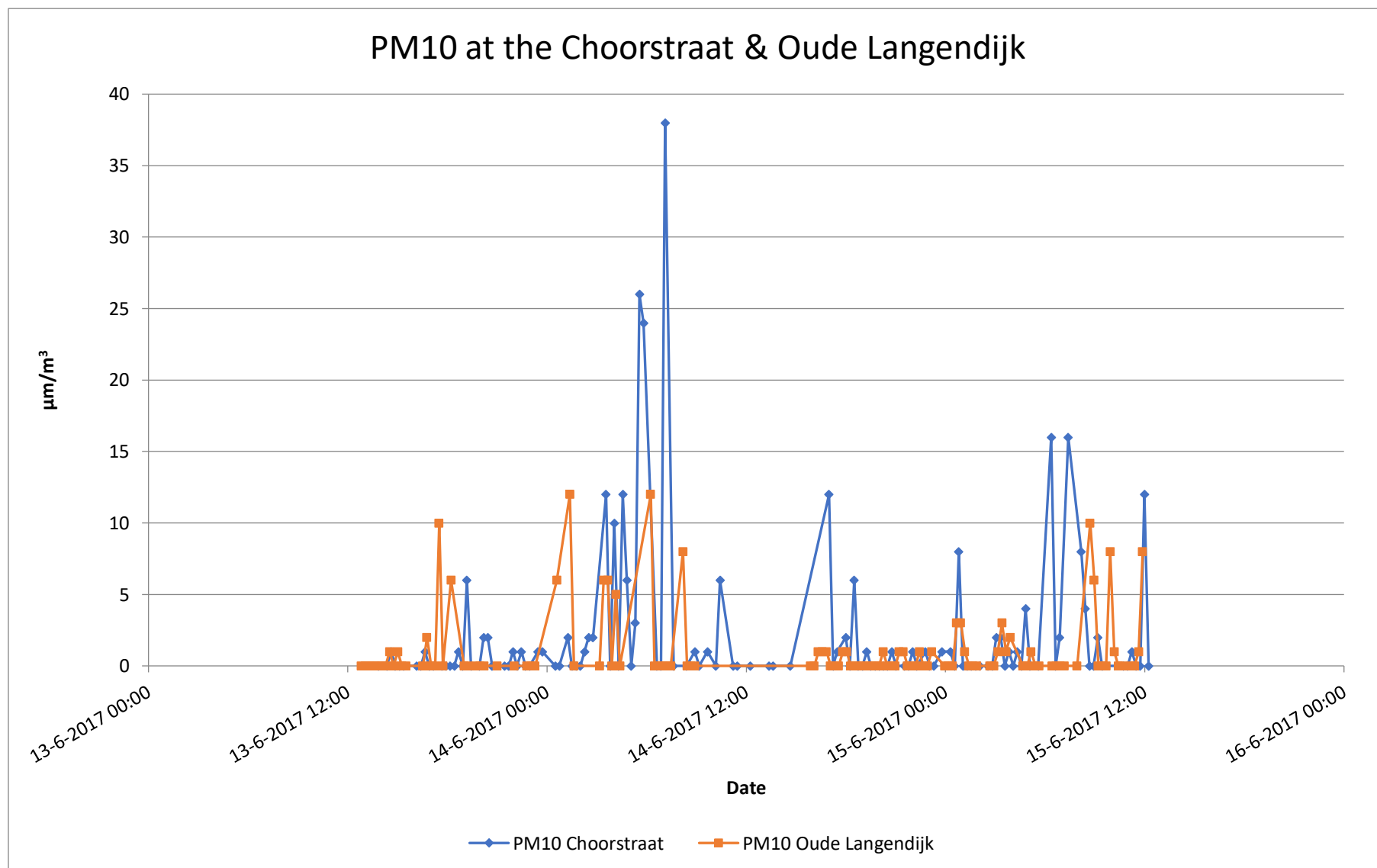
PM1 measurements at the Choorstraat and Oude Langendijk



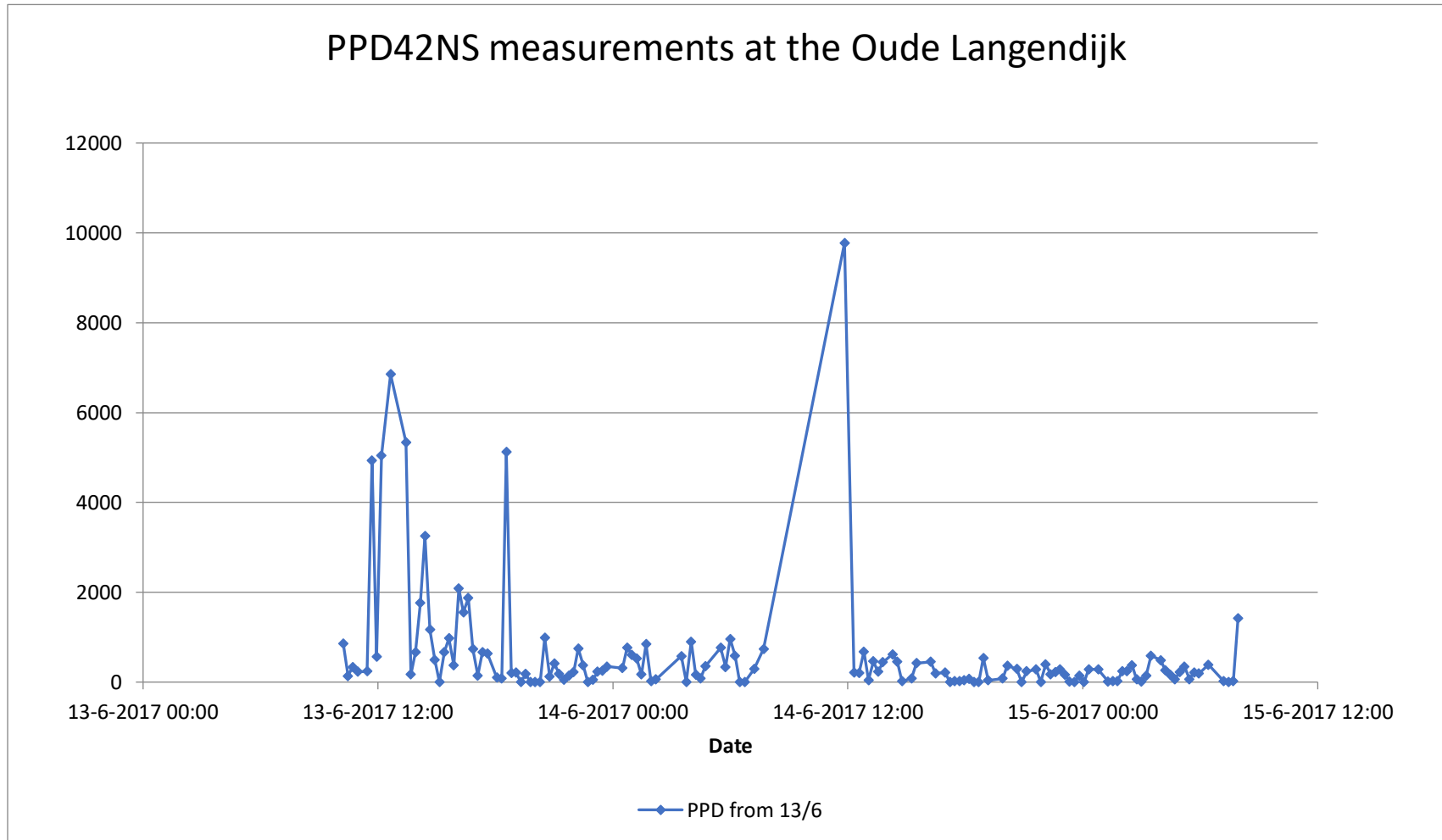
PM2.5 measurements Choorstraat and Oude Langendijk



PM10 measurements at the Choorstraat and Oude Langendijk



PPD42NS measurements at the Oude Langendijk



Sensor 1420366D Oude Langendijk

| | Category | N | Mean | Max | Min | Standard deviation |
|--------------|----------|----|-------------|-----|-----|--------------------|
| Total | PM10 | 48 | 1.145299145 | 12 | 0 | 2.600613181 |
| | PM2.5 | 47 | 14.5 | 34 | 0 | 6.966061346 |
| | PM1 | 48 | 8.47008547 | 34 | 0 | 7.568734064 |

| PM10 | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|-----|-----|-------------|
| 10:01 - 14:00 | 5 | 0 | 0 | 0 | 0 |
| 14:01 - 18:00 | 13 | 1.076923077 | 10 | 0 | 2.752621128 |
| 18:01 - 22:00 | 7 | 0.857142857 | 6 | 0 | 2.267786838 |
| 22:01 - 02:00 | 7 | 2.571428571 | 10 | 2 | 4.720774755 |
| 02:01 - 06:00 | 6 | 2.833333333 | 6 | 0 | 3.125166662 |
| 06:00 - 10:00 | 10 | 2 | 12 | 0 | 4.320494 |
| 10:00 - 14:00 | - | - | - | - | - |
| 14:00 - 18:00 | 9 | 0.444444444 | 1 | 0 | 0.527046277 |
| 18:00 - 22:00 | 16 | 0.235294118 | 1 | 0 | 0.437237316 |
| 22:00 - 2:00 | 14 | 0.642857143 | 3 | 0 | 1.081817762 |
| 2:00 - 6:00 | 11 | 0.727272727 | 3 | 0 | 1.009049958 |
| 6:00 - 10:00 | 11 | 2.181818182 | 10 | 0 | 3.842347767 |
| 10:00 - 14:00 | 7 | 1.428571429 | 8 | 0 | 2.935821456 |
| Total: | 116 | | | | |

| PM2.5 | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|-----|-----|-------------|
| 10:01 - 14:00 | 5 | 3.2 | 4 | 0 | 1.095445115 |
| 14:01 - 18:00 | 13 | 2.5 | 6 | 1 | 1.977142106 |
| 18:01 - 22:00 | 7 | 2.166666667 | 2 | 1 | 0.98319208 |
| 22:01 - 02:00 | 7 | 6.571429 | 10 | 2 | 2.760262 |
| 02:01 - 06:00 | 6 | 13 | 24 | 2 | 7.848567 |
| 06:00 - 10:00 | 10 | 11.8 | 24 | 2 | 8.560893 |
| 10:00 - 14:00 | - | - | - | - | - |
| 14:00 - 18:00 | 9 | 4.888888889 | 18 | 1 | 5.418589402 |
| 18:00 - 22:00 | 16 | 2.294117647 | 10 | 0 | 2.46892451 |
| 22:00 - 2:00 | 14 | 5.714286 | 28 | 0 | 7.569459 |
| 2:00 - 6:00 | 11 | 5.272727 | 10 | 2 | 3.981777 |
| 6:00 - 10:00 | 11 | 11.27273 | 20 | 2 | 5.53337 |
| 10:00 - 14:00 | 7 | 15.85714 | 34 | 26 | 12.00595 |
| Total: | 114 | 6.582608696 | | | |

| PM1 | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|----------|-----|-----|-------------|
| 10:01 - 14:00 | 5 | 4 | 10 | 0 | 3.741657387 |
| 14:01 - 18:00 | 13 | 6.538462 | 28 | 2 | 6.740616 |

| | | | | | |
|----------------------|-----|-------------|----|----|----------|
| 18:01 - 22:00 | 7 | 7.142857 | 28 | 2 | 9.511897 |
| 22:01 - 02:00 | 7 | 8 | 26 | 0 | 8.869423 |
| 02:01 - 06:00 | 6 | 13.333333 | 16 | 10 | 2.160247 |
| 06:00 - 10:00 | 10 | 14.2 | 32 | 2 | 9.354737 |
| 10:00 - 14:00 | - | | | | |
| 14:00 - 18:00 | 9 | 3.444444 | 8 | 2 | 1.878238 |
| 18:00 - 22:00 | 16 | 3.117647 | 10 | 1 | 2.057983 |
| 22:00 - 2:00 | 14 | 5.857143 | 14 | 2 | 3.324898 |
| 2:00 - 6:00 | 11 | 7.363636 | 16 | 2 | 3.981777 |
| 6:00 - 10:00 | 11 | 14.18182 | 22 | 2 | 6.539391 |
| 10:00 - 14:00 | 7 | 22.14286 | 34 | 12 | 6.693992 |
| Total: | 116 | 6.582608696 | | | |

Sensor 14203981 Choorstraat

| | Category | N | Mean | Max | Min | Standard deviation |
|--------------|----------|----|------------|-----|-----|--------------------|
| Total | PM10 | 43 | 2.1015625 | 38 | 0 | 5.391506654 |
| | PM2.5 | 40 | 6.46875 | 24 | 0 | 5.350601556 |
| | PM1 | 43 | 10.5234375 | 116 | 0 | 11.74798682 |

| PM10 | Amount | Mean | Max | Min | St Dev |
|----------------------|--------|-------------|-----|-----|-------------|
| 10:01 - 14:00 | - | - | - | - | - |
| 14:01 - 18:00 | 12 | 0.166666667 | 1 | 0 | 0.389249472 |
| 18:01 - 22:00 | 13 | 0.923076923 | 6 | 0 | 1.705947364 |
| 22:01 - 02:00 | 12 | 0.416666667 | 2 | 0 | 0.668557923 |
| 02:01 - 06:00 | 13 | 7.538461538 | 26 | 0 | 8.922026906 |
| 06:00 - 10:00 | 8 | 5 | 38 | 0 | 13.34166406 |
| 10:00 - 14:00 | 7 | 0.857142857 | 6 | 0 | 2.267786838 |
| 14:00 - 18:00 | 8 | 2.625 | 12 | 0 | 4.307385684 |
| 18:00 - 22:00 | 11 | 0.181818182 | 20 | 1 | 0.404519917 |
| 22:00 - 2:00 | 13 | 0.923076923 | 8 | 0 | 2.177978362 |
| 2:00 - 6:00 | 13 | 0.769230769 | 4 | 0 | 1.235168 |
| 6:00 - 10:00 | 10 | 4.8 | 16 | 0 | 6.408327915 |
| 10:00 - 14:00 | 7 | 1.857142857 | 12 | 0 | 4.488079 |

| | | | | | |
|---------------|-----|-----------|--|--|-------------|
| Total: | 126 | 2.1015625 | | | 5.391506654 |
|---------------|-----|-----------|--|--|-------------|

| PM2.5 | Amount | Mean | Max | Min | St Dev |
|----------------|--------|----------|-----|-----|-------------|
| 10:01 14:00 | - | - | - | - | - |
| 14:01 18:00 | 12 | 3.5 | 8 | 0 | 2.812311 |
| 18:01 22:00 | 13 | 2.923077 | 10 | 0 | 2.871165 |
| 22:01 02:00 | 12 | 6.833333 | 24 | 2 | 6.617241 |
| 02:01 06:00 | 13 | 12.07692 | 20 | 3 | 5.648916 |
| 06:00 10:00 | 8 | 7.25 | 12 | 1 | 4.367085 |
| 10:00 14:00 | 7 | 8 | 20 | 2 | 5.887841 |
| 14:00 18:00 | 8 | 5.125 | 14 | 0 | 4.882549 |
| 18:00 22:00 | 11 | 4.181818 | 20 | 1 | 5.600325 |
| 22:00 - 2:00 | 13 | 4.692308 | 10 | 0 | 3.750214 |
| 2:00 - 6:00 | 13 | 4.615385 | 12 | 2 | 2.930826 |
| 6:00 - 10:00 | 10 | 9.9 | 16 | 6 | 3.107339 |
| 10:00 14:00 | 7 | 12 | 24 | 6 | 5.887841 |
| Total: | 126 | 6.46875 | | | 5.350601556 |

| PM1 | Amount | Mean | Max | Min | St Dev |
|----------------|--------|------|-----|-----|--------|
| 10:01 14:00 | - | - | - | - | - |

| | | | | | | |
|---------------------|---|-----|------------|-----|----|-------------|
| 14:01 | - | | | | | |
| 18:00 | - | 12 | 4.75 | 8 | 0 | 2.490893 |
| 18:01 | - | | | | | |
| 22:00 | - | 13 | 5.846154 | 10 | 4 | 1.724633 |
| 22:01 | - | | | | | |
| 02:00 | - | 12 | 9.75 | 16 | 6 | 3.048845 |
| 02:01 | - | | | | | |
| 06:00 | - | 13 | 15.15385 | 19 | 6 | 3.508232 |
| 06:00 | - | | | | | |
| 10:00 | - | 8 | 11.875 | 22 | 2 | 6.854352 |
| 10:00 | - | | | | | |
| 14:00 | - | 7 | 4.285714 | 8 | 2 | 2.13809 |
| 14:00 | - | | | | | |
| 18:00 | - | 8 | 6.25 | 20 | 2 | 6.18177 |
| 18:00 | - | | | | | |
| 22:00 | - | 11 | 13.45455 | 116 | 0 | 34.2151 |
| 22:00 - 2:00 | - | | | | | |
| | | 13 | 6.076923 | 11 | 1 | 2.928638 |
| 2:00 - 6:00 | - | | | | | |
| | | 13 | 9.692308 | 14 | 6 | 2.287087 |
| 6:00 - 10:00 | - | | | | | |
| | | 10 | 21.4 | 34 | 8 | 8.983936 |
| 10:00 | - | | | | | |
| 14:00 | - | 7 | 22 | 26 | 18 | 2.581989 |
| Total: | | 126 | 10.5234375 | | | 11.74798682 |

Appendix X: Node-RED node codes

This appendix shows the code that is used in every node in the NodeRED environment.

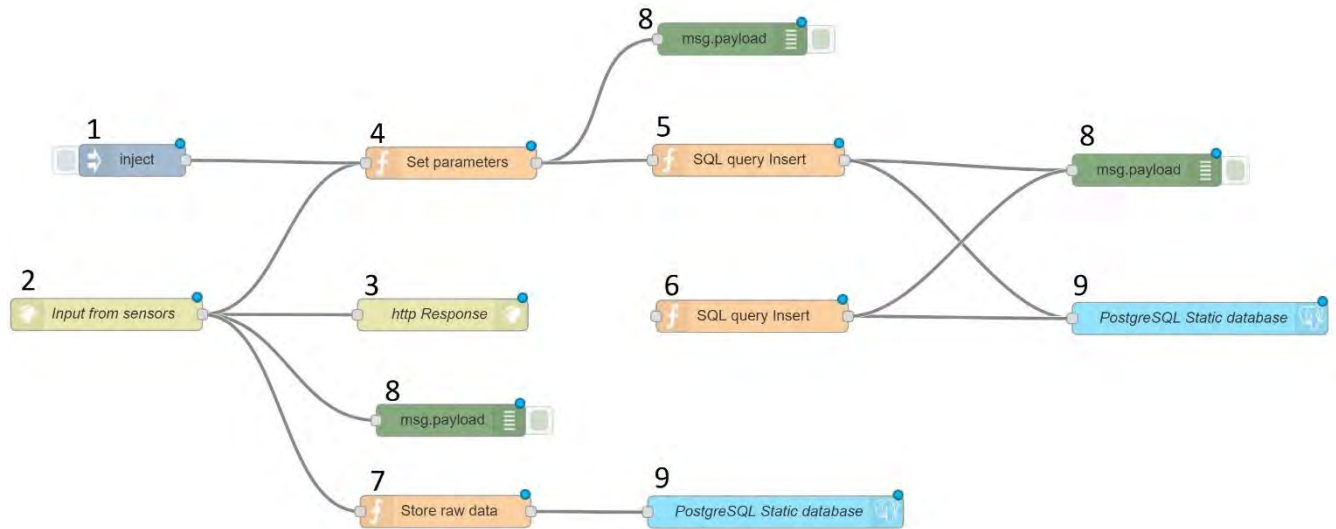


Figure 95: APPENDIX X: Node-RED flow of receiving and parsing LoRa messages

1: Inject

This node inserts a dummy LoRa message for testing purposes.

2: Input from sensors

This node has the endpoint for the LoRa messages to be send to (<https://geo1101.bk.tudelft.nl/iot/static/inputport>)

3: http response

Informs the gateway that the message was sent correctly

4: Set parameters

Parsing the JSON object to make the values ready to be saved in the database

```
msg.queryParameters = {};  
  
var lora_decrypt = global.get('loraModule')['lora_decrypt'];  
payload_hex = msg.payload.DevEUI_uplink.payload_hex;  
sequence_counter = msg.payload.DevEUI_uplink.FCntUp;  
addr = msg.payload.DevEUI_uplink.DevAddr;  
  
if (addr == "14203D86"){  
    key = '6653a47d8eadf590a7cdb2e130ef0280';  
    value = lora_decrypt(payload_hex, sequence_counter, key, addr);  
}  
else if (addr == "14203981"){  
    key = '0063dfb6d720c5ca0842ac29e86e9690';  
    value = lora_decrypt(payload_hex, sequence_counter, key, addr);  
}  
else if (addr == "142037B9"){ //V1  
    key = 'e79cd3d3ad8b07ea607d6e37587564b2';  
    value = lora_decrypt(payload_hex, sequence_counter, key, addr);  
}  
else if (addr == "14204140"){ //V2  
    key = '8104d36a39d73b4a50875c0114952460';
```



```

        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "14203981"){ //V3
        key = '0063dfb6d720c5ca0842ac29e86e9690';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "14204073"){ //C1
        key = 'dd7bb1f82c74c3cac204c5bf7ed3f195';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "1420366D"){ //C2
        key = 'd3d8bdea795215b1b5344f4837923a6f';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "14203210"){ //C3
        key = '2650252ad0464f9d46464edc54d16bd1';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "14204772"){ //O1
        key = 'c489fc45ccd1962b3ff368ca488669a8';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "142042F7"){ //O2
        key = '7ce920520cf54a4017f1fd803c0b4602';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "14204FD8"){ //O3
        key = '6e6b7a73f99eed8cc9252e06d0fd8b3c';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "14203077"){ //Spare
        key = '91fala2ec3489aed063f35a220e36498';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else if (addr == "14203937"){ //Spare
        key = 'b4bd4ddabe32dc5bb76d45af9cefe688';
        value = lora_decrypt(payload_hex, sequence_counter, key, addr);
    }
    else{
        value = 'not working';
    }

    test = value;

    aq1 = value[0];
    airq1 = aq1.toString();
    aq2 = value[1];
    airq2 = aq2.toString();
    aq = parseInt(airq1+airq2);
    temp = value[2];
    temp1 = value[3];
    hum = value[4];
    hum1 = value[5];
    noi = value[6];
    pm1 = value[7];
    pm2 = value[8];
    pm10 = value[9];
    pm1_atm = value[10];
    pm2_atm = value[11];
    pm10_atm = value[12];

    msg.payload.SensorID = addr;
    msg.payload.Temperature = temp + '.' + temp1;
    msg.payload.Humidity = hum + '.' + hum1;
    msg.payload.Noise = noi;
    msg.payload.AirQuality = aq;
    msg.payload.Pm1 = pm1;
    msg.payload.Pm2 = pm2;
    msg.payload.Pm10 = pm10;
    msg.payload.Pm1atm = pm1_atm;
    msg.payload.Pm2atm = pm2_atm;
    msg.payload.Pm10atm = pm10_atm;

    val = msg.payload.DevEUI_uplink.Time;

```

```

timesarray = val.split("T");
part1 = timesarray[0];
timepart = timesarray[1];
timearray = timepart.split(".");
time = timearray[0];
timestamp = part1 + " " + time;
msg.payload.Timestamp = timestamp;

return msg;

```

5: SQL query insert

```

msg.payload = "INSERT INTO public.data5
(sid, temperature, humidity, noise, airquality, pm1, pm2, pm10, pm1_atm, pm2_atm, pm10_atm,
sensedtime)
VALUES ('"+msg.payload.SensorID+"',
 '"+msg.payload.Temperature+"',
 '"+msg.payload.Humidity+"',
 '"+msg.payload.Noise+"',
 '"+msg.payload.AirQuality+"',
 '"+msg.payload.Pm1+"',
 '"+msg.payload.Pm2+"',
 '"+msg.payload.Pm10+"',
 '"+msg.payload.Pm1atm+"',
 '"+msg.payload.Pm2atm+"',
 '"+msg.payload.Pm10atm+"',
 '"+msg.payload.Timestamp+"');";

return msg;

```

6: SQL create table statement

Not used, tables are created in PgAdmin 4.

7: SQL raw data insert

```

test = msg.payload;
data = JSON.stringify(test);

msg.payload = "INSERT INTO public.rawdata (rawdata) VALUES ('"+data+"');";

return msg;

```

8: Debug Nodes

Show information on output of nodes.

9: PostgreSQL connection

Connects to the PostgreSQL database.

Appendix Y: Test script PMS5003

```
import time
from machine import Pin, Timer, UART
from network import LoRa
import socket
import binascii
import struct
import config
import machine
from dth import DTH
from math import log
import utime
from struct import unpack
import array

SET_PORT = 'P11'    # LoPy SET port, communicates with the PMS SET port (P3)
RESET_PORT = 'P8'  # LoPy RESET port, communicates with the PMS RESET port(P6)

STARTUP_TIME = 30  # 30 seconds of startup time (needed for ventilator). 5 is better for
debugging.

set_pin = Pin(SET_PORT, mode=Pin.OUT)
reset_pin = Pin(RESET_PORT, mode=Pin.OUT)
data_uart2 = UART(1, baudrate=9600, parity=None, stop=1, pins=('P9', 'P10'))

def fan_on():
    print('Turning the PMS5003 fan on') # the fan is ON by default.
    set_pin.value(1)

def fan_off():
    print('Turning the PMS5003 fan off')
    set_pin.value(0)

def reset():
    print('Resetting the PMS5003 sensor')
    reset_pin.value(1)

def set_sensor():
    print('Preparing the fan...')
    fan_on()
    print('Wait 30 seconds...')
    time.sleep(STARTUP_TIME)          # wait 30 seconds
    print('Sensor is set')

while True:
    set_sensor()
    d = data_uart2
    raw = d.read()
    unpacking = unpack('>16H', raw)
    pm1_u = int(unpacking[2])
    pm2_u = int(unpacking[3])
    pm10_u = int(unpacking[4])
    pm1_atm_u = int(unpacking[5])
    pm2_atm_u = int(unpacking[6])
    pm10_atm_u = int(unpacking[7])
    print('')
    print('big starting number: ', unpacking[0])
    print('frame length (2*13+2): ', unpacking[1])
    print('PM1.0 concentration, standard particle: ', unpacking[2])
    print('PM2.5 concentration, standard particle: ', unpacking[3])
    print('PM10 concentration, standard particle: ', unpacking[4])
    print('PM1.0 concentration, atmospheric env: ', unpacking[5])
    print('PM2.5 concentration, atmospheric env: ', unpacking[6])
```

```
print('PM10 concentration, atmospheric env: ', unpacking[7])
print('Data 7: ', unpacking[8])
print('Data 8: ', unpacking[9])
print('Data 9: ', unpacking[10])
print('Data 10: ', unpacking[11])
print('Data 11: ', unpacking[12])
print('Data 12: ', unpacking[13])
print('Data 13 (reserved):', unpacking[14])
print('Check code: ', unpacking[15])
print('')
print('Finished!')
fan_off()
```

