Masteroppgåve

NTNU
Noregs teknisk-naturvitskaplege universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

Sivert Kittelsen

Development of an integrated environmental sensor for indoor air quality

Masteroppgåve i Kybernetikk og robotikk
Veileder: Geir Mathisen

Januar 2021

NTNU
Kunnskap for ei betre verd

Sivert Kittelsen

Development of an integrated environmental sensor for indoor air quality

Masteroppgåve i Kybernetikk og robotikk
Veileder: Geir Mathisen
Januar 2021

Noregs teknisk-naturvitskaplege universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

MASTER THESIS DESCRIPTION

Candidate:	**Sivert Kittelsen**
Course:	**TTK4900 Engineering Cybernetics**
Thesis title (Norwegian)	**Utvikling av en sensorplattform for måling av innendørs luftkvalitet.**
Thesis title (English):	**Development of an integrated environmental sensor for indoor air quality**

Thesis description: Good indoor air quality is of great importance to those who stay on the premises. Poor air quality has negative impact on people's health, performance and well-being. It is not always easy to see this cause-effect relationship, often due to:

- People have different sensitivities to less good indoor climate.
- The indoor climate can vary a lot within the same room.
- People are not aware that the climate parameters in different (work) places in the same room may be below the recommended value.

Thus, we want to develop a small, lightweight platform for measuring relevant parameters that are important for indoor climate, so that more platforms e.g. can be placed in a network. Size, weight, form of communication and energy consumption must be adapted so that in the future the platform may be transported by a small indoor drone.

The tasks will be:

1. Conduct a literary study of which parameters affect the indoor climate and sensors for measuring these. Also study available experience in use of drones for measuring indoor climate.
2. Propose a design for a sensor platform for measuring indoor climate. The platform shall have size, weight, form of communication and energy consumption adapted to be transported by small indoor drones.
3. As far as time permits, implement the proposed design from point 2.

Start date: 18th August, 2020
Due date: 25th January, 2021

Thesis performed at: Department of Engineering Cybernetics
Supervisor: Professor Geir Mathisen, Dept. of Eng. Cybernetics

Samandrag

Lufta vi pustar har innverknad på helsa vår, og dårleg luft kan gje både milde og alvorlege plager. Som eit bidrag til å forhindre ubehag og skade, eller å forsikre høg produktivitet på ein arbeidsplass, nyttar ein sensorar for å overvåke luftkvalitet. I dag blir dette gjort ved hjelp av fast monterte sensorar, som gir målingar som skal gje eit bilete av luftkvaliteten i eit rom eller ein bygning. Som eit alternativ til dette, ser ein på mogleheita for å gjere tilsvarande målingar frå flygande dronar.

Denne masteroppgåva tek føre seg denne problemstillinga, med eit fokus på sensorikk. Oppgåva skildrar utviklinga av ein prototype for ein sensornode som kan måle ymse parametrar knytt til luftkvalitet. Ønsket om at produktet skal vere lite nok til å bli bore av ein drone, har vore førande undervegs. Med dette som bakteppe blei komponentar testa og valt ut, og blei implementert i eit eigenutvikla kretskort. Dette kretskortet bestod av sensorar, mikrokontroller og forskjellige kommunikasjonsgrensesnitt med meir. Som del av denne prosessen blei det utvikla programvare til mikrokontrolleren, som gjorde samhandling med sensorar mogleg. Sensornoden hadde sensorar som målte parametrane temperatur, trykk, luftfuktigheit, karbondioksid (CO_2) og flyktig organisk materiale (VOC). Nøyaktigheita til desse sensorane blei testa ut gjennom ei rekke lengre testar, men ikkje i samhandling med dronar. Den resulterande sensornoden blei òg vurdert opp mot kriterium for vekt og storleik.

Resultata synte at nokre av parametrane klarte sensorane å gje att veldig presist. To parametrar, CO_2 og VOC, viste seg vanskelegare å måle med måleutstyr i så liten skala. Vidare blei nyttegraden av målingane dratt i tvil og diskutert, ettersom dei ikkje direkte kan overførast til luftforhold rundt dronar. Storleik og vekt på den resulterande prototypen endte opp med å bli større enn ønska, men moglege forbetringar og endringar som kan endre på dette blei foreslått.

Oppsummert dokumenterer oppgåva ei vellykka utvikling av maskinvara til sensornoden. Samtidig har problemområde blitt framheva, og forslag til forbetringar eller alternativ har blitt drøfta. Oppgåva har gitt god oversikt over kva som har fungert godt og dårleg. Arbeidet vil kunne utgjere eit grunnlag for vidare forskning på problemstillinga.

Abstract

The air we breathe affects our heath, and poorly ventilated air can cause both mild nuisances and serious harm. As a contribution to prevent discomfort and harm, or in order to insure high productivity in a working environment, sensors are used to monitor air quality. This is currently done by means of fixed sensors, which give measurements that offer a description of the air quality in a room or a building. As an alternative, the possibility of performing similar measurements from flying drones is proposed.

This master thesis takes on this problem, with an emphasis on sensory testing methods. The thesis describes the development of a prototype of a sensor node, which can measure various parameters linked to air quality. A preference for small size and low weight, to be compatible with a drone, has been an area of focus. With this in mind components were tested and chosen, and were implemented in a custom made printed circuit board (PCB). The PCB consisted of sensors, microcontroller, various communication interfaces, and more. Software for the microcontroller was developed as part of this process, which enabled interaction with sensors. The sensor node consisted of sensors measuring the parameters of temperature, pressure, humidity, carbon dioxide (CO_2) and volatile organic compound (VOC). The accuracy of the sensors was tested by means of several extended tests, but not whilst interacting with a drone. The resulting sensor node's weight and size was also evaluated.

The results showed very precise reproduction of some parameters. Two parameters, CO_2 and VOC, proved difficult to measure with equipment of this scale. The usefulness of the measurements was questioned, as they aren't directly comparable with air conditions around drones. Size and weight of the resulting prototype was somewhat bigger than anticipated, but possible improvements and changes were suggested.

In summary the thesis documents a successful development of the sensor node's hardware. Areas of concern, and suggestions for improvement have been discussed. The thesis offers a good overview of what proved to function well and what didn't. The work can serve as basis for further inquiry into the subject.

Preface

This text along with the work that is presented make up my master thesis at the department of Engineering Cybernetics at the Norwegian University of Science and Technology. The topic and background for the thesis was suggested by the department in collaboration with Kjeldsberg Eiendomsforvaltning AS, but the work is a result of my own individual efforts.

The work on the thesis was conducted during the fall and winter of 2020/2021. The conclusion of this thesis marks the end to my time as a student at NTNU. I am very grateful for all the enjoyable years in Trondheim, both at and outside of the university.

I would like to thank my supervisor Geir Mathisen for his advice and support throughout the work of this thesis. I would also like to thank him for offering an interesting, hands-on project within the field of embedded systems.

Sivert Kittelsen
January 2021

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
AQI	Air Quality Index
CO_2	Carbon Dioxide
GNSS	Global Navigation Satellite System
IAQ	Indoor Air Quality
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IOT	Internet of Things
LLC	Logic Level Converter
NDIR	Non-dispersive infrared
PCB	Printed Circuit Board
ppm	Parts per million
SLAM	Simultaneous Localization And Mapping
SPI	Serial Peripheral Interface
TVOC	Total Volatile Organic Compound
TWI	Two-Wire Interface
UAV	Unmanned Aerial Vehicle
VOC	Volatile Organic Compounds

Table 1: Abbreviations used in this thesis.

Chapter 1

Introduction

This chapter explains the background for the thesis, limitations that have put restraints on the execution of the thesis, and also what was achieved. At the end of the chapter, a disposition for the rest of the thesis is formulated.

1.1 Motivation

The quality of the air we breathe contributes to our health and well-being. In severe cases, illness and long-lasting harm can be caused, by for instance poisonous gases. In most cases though, poor air quality stems from poor ventilation, or nearby pollution from for example traffic. The effects on human health is less severe, but can manifest itself in irritation, headaches or fatigue. In a working environment, this can reduce productivity, and impact the well-being of everyone affected. To combat this, measurements of air quality are performed, to attain overview of the situation.

Today, measurements of indoor air quality is typically done using statically mounted sensors. Their measurements are collected, and used to give an idea of the current quality of the air. The current method has some drawbacks. Firstly, depending on the size of the room or building that is being monitored, multiple sensors may be required, scattered around the area they are monitoring. Another concern is that fixed sensors are less able to adapt to changes in the airflows, for instance when a room is refurnished. Suddenly, what used to be an optimal sensor placement may be sub-optimal.

As a replacement to having multiple mounted sensors, an unmanned aerial vehicle (UAV) could fly around and perform measurements from different locations in a room. This could either happen continuously as the UAV is flying, or the UAV can "jump" from location to location, where measurements are performed. The problems that follow fixed sensors can then be dealt with, by altering the route of the drone's flight to match with the building that is being measured.

1.2 Limitations

The one major restraint on this master thesis was that no UAV was used for testing. Instead, assumptions on a possible UAV design was made from the master thesis Eikeland [1]. That master thesis takes on the same problem as this one, but with a focus on designing and constructing a miniaturized UAV. The master thesis gave a theoretical framework for this one. Still, many aspects of a UAV's influence on sensors were not properly examined. Unfortunately, the UAV developed in Eikeland [1] was unavailable for us to use.

In addition to this, during the course of the semester, some unforeseen external events and factors put restraints on the execution of the project.

- During the hardware design, a potential component, namely a sensor named SGPC3, was unavailable for testing. It was sold out, and the expected delivery date was too late with regard to the deadline of the thesis. This is further elaborated in Section 5.3.1.

- One sensor that was tested during the hardware design, T6173, proved to be defect. This is elaborated in Section 5.6.1.

- I was put into quarantine twice whilst working on this thesis. One of those happened at a critical time towards the end of the project, which kept me from performing some tests regarding power consumption of the designed hardware. This is also mentioned as a footnote to Table 7.1, and in Section 11.2.3.

The mentioned limitations affected what could be tested and discussed, and thus negatively impacted the foundations on which decisive conclusions could be drawn.

1.3 Contribution

The spesialization project, on which this master thesis is partly based, offers a theoretical basis for the measurement of indoor air quality. In part, the health effects of poor air quality is explained, and current sensor technologies are elaborated. Additionally, limited testing of sensors was performed, which gave a starting point for the selection of components for this thesis.

This thesis describes the work of designing a prototype for a sensor node to measure indoor air quality, whilst having physical properties that allows it to be carried by a small UAV. The process from design through production, implementation and testing has been documented. This documentation offers insight to the performance of various air quality sensors, microcontrollers and other components that is required in a sensor node. The software developed and implemented in the prototype shows how an implementation with said components is functional. The work has also shed light on challenges to solving the presented problem. By testing of sensors, a better insight into obstacles in attaining a full picture of air quality has been established. A study of the current literature on the

field was also performed.

1.4 Structure

The thesis is structured in the following fashion:

- **Chapter 2** looks into the current literature on measurement of indoor air quality (IAQ), and how this is done in relation to UAVs.

- **Chapter 3** presents and explains various theoretical concepts that are used in this thesis. The concepts concern embedded systems, manufacturing tools for these, as well as sensor technologies.

- **Chapter 4** gives a brief summary of what was done in the spesialization project, and important lessons that can be drawn from it for this thesis.

- **Chapter 5** and **Chapter 6** documents the design process for hardware and software, respectively. Here, a sensor node is designed that tries to offer a solution to the problem presented at the start of the theis. Acceptance criteria, which are discussed in Chapter 11, are formulated.

- **Chapter 7** and **Chapter 8** explains the implementation of hardware and software, respectively. This is based on the design from the previous chapters.

- **Chapter 9** explains the setup of various tests, used to evaluate the sensor node's performance. The tests set the sensor node's performance up against other measurement equipment. **Chapter 10** contains the results from these tests.

- **Chapter 11** is a discussion of the resulting product, and to which degree the original goals of the thesis are fulfilled. It concerns both the results from Chapter 10, as well as the final hardware and software.

- **Chapter 12** and **Chapter 13** concludes the thesis, and lists up future challenges that need to be addressed.

Chapter 2

Literature Study

This chapter gives a summary of how air quality is being measured and monitored today, as described in contemporary literature. It gives an overview of what are the most common applications, and associated examples. Emphasis is put on the technologies being used with respect to instrumentation. The current trends in the field, along with some of their consequences are presented. Furthermore, it takes a closer look at the connection between UAVs and measurements of air quality, both indoor and outdoor.

2.1 Measurement of indoor air quality

Measuring indoor air quality is done through the measurement of different gases and particles in the air. Parameters that are measured are those affecting human health, productivity or comfort. VOC, CO_2, radon, humidity, temperature, along with different gases, are typical parameters that are measured when assessing the quality of the air.

In a study from the Techical University of Denmark, a group of experiment participants were exposed to different environmental conditions in a simulated office environment. The difference of environmental conditions was a result of varying ventilation rates, which again controlled the relative humidity and temperature in the room. The study concludes, that the ventilation rate cannot be set without regard to the indoor air temperature and humidity, in order to maintain a comfortable and healthy indoor environment[2].

Indoor air quality is not just limited to houses and offices, but also other constructions. In a paper from South Korea in 2010, the levels of CO_2 are measured in Seoul's subway system, as it is considered a good indication for the air quality[3]. This paper, like many other similar papers, puts it in the context of using the measurements to verify or improve the performance of a ventilation system.

Aircrafts are an other example of an environment typically monitored with respect to its indoor air. As a result of the air being to a large extent recycled, and the plane normally being quite crowded, the air is far from similar to that of the outdoors. Contaminant levels are on the other hand in general low, but CO_2 levels higher than most other indoor environments[4].

2.1.1 Instrumentation

Normally, the instruments used for testing air quality are large, scientific measuring kits. Kwon *et al.* [3] describes the use of a large testing kit, about the size of a suitcase, to measure the levels of CO_2 in Seoul's subway. The kit consisted of, amongst other technologies, a non-dispersive infrared sensor (NDIR) to measure the CO_2.

Lee *et al.* [5] presents an investigation of indoor air quality in Hong Kong residential homes. Here, a high-end data logger with a sensor probe (Q-Trak 8551) was used to measure and monitor CO_2, humidity and temperature. Another, similar instrument (Dusttrak 8520) was responisble for measuring particulate matter. Additionaly, other techniques were used for measurements of bacteria and other contaminants. What this shows, is that a lot of expensive and large equipment is being used for this kind of measurement.

Lately, a consumer-oriented market for measurements of indoor air quality has emerged. These products aim to be easy to install and maintain, and give the customer information on which better choices for keeping the indoor air fresh can be made. Examples of this are Foobot and Airthings[6, 7]. With the ability to connect to cloud-based storage and analysis services, these products regularly measure parameters like humidity, temperature, particles, VOC. Some Airthings products measure CO_2, and others also measure radon gas, which requires long-term employment in order to give a usable measurement. When compared to traditional, scientific instruments, these products are cheap. Yet still, they are being applied in scientific research of indoor air quality[8]. Specifically Foobot's sensors' performance has been evaluated and compared against high-end equipment, which concluded that their sensor outputs were in significant agreement[9].

In recent years, Internet of Things (IoT) applications have been booming, with an estimated increase from 0.9 billion to 26 billion IoT units during the years 2015 to 2020. Smaller monitoring and control systems, often used in what is called "smart homes", have been a part of this surge[10]. It is not just limited to stationary sensors, but also mobile and wearable applications, which previously have been left out, because of the lack of small sensory components[11, p. 291].

Abraham and Li [12] is an example of this trend. In this paper, low-cost sensors and network components are used to monitor indoor air. Micro gas sensors are used for the measurement, bundled together in a measurement node, with several of these measurement nodes collecting measurements from different locations simultaneously.

Caron *et al.* [13] tests out several such small sensors, with a focus on sensors measuring VOC, to

test their response in an environment with many indoor plants. The responses to changes in gas composition did vary across different sensors, and the study concludes that these sensors do not allow for an accurate interpretation of complex VOC mixtures. Some of the sensors did, on the other hand, offer a sufficient response to change in overall VOC concentration in the air.

2.2 IAQ measurements with UAVs

This section tries to present the current use of Unmanned Aerial Vehicles (UAVs), more commonly known as drones, in measurements of air quality. It also covers some key technologies surrounding the subject. In the literature you can find many projects and papers that describe this. The following sections will not cover everything, but give an overview and examples of the usage and implementation.

2.2.1 Outdoor UAVs

Using UAVs for sensing applications, like for instance air pollution, has limitations in long-term deployment. On the other hand, it appears to be of high interest within the scientific and public communities, rapidly progressing with new technologies for sensing and data processing[11, p. 292].

In an article from 2016, an overview of the applications of small UAVs in air quality measurements was presented [14]. After searching through different bibliographic databases, using over 60 search terms and different combination of these terms, a total of 60 papers were found. The article states "This relatively small number of papers implies that the field is still in its early stages of development."[14] Even though the article is four years old, and UAVs have become more mainstream, it can be assumed to still be true.

A frequent application of UAVs is monitoring of Air Quality Index (AQI). AQI is not a uniquely defined measure, but different ones are defined by governments to quantify levels of air pollution[15]. Which measures affect the index differs from country to country, but are in general decided by various air pollutants, like dangerous gases and particulate matter[16].

The deployment of UAVs have many motivating factors. A large number of static sensors have to be applied in order to achieve high-resolution picture over an area. It also limits measuring across different heights. In Yang *et al.* [15], the case of Beijing is presented, which had only 28 monitoring stations for AQI. The distance between neighbouring monitoring stations could be tens of kilometres, and only giving a new sample every 2 hours. In this study, a quad-copter carrying sensors was used to construct fine-grained AQI maps.

Many other studies use UAVs to measure selected air pollutants. Koval and Irigoyen [17] applies a quad-copter together with an embedded circuit to measure methane, hydrogen and liquefied petroleum gas. In Villa *et al.* [18] a hexacopter is used to measure pollutants from car exhaust, like carbon

dioxide and carbon monoxide. Many similar examples exist, often with a focus on monitoring the air around industrial sites or urban areas.

An example of this is elaborated in Wu *et al.* [19]. This paper describes using drone swarms to monitor traffic and roads. The drones carried both cameras and some air quality sensors. The input from the cameras was used in monitoring the traffic, estimating the number of cars. The other sensors measured pollution, namely carbon dioxide and particulate matter. Emphasis in the paper was not put on the air quality measurements. Instead, it elaborates on the drones relation to the rest of the swarm, the network and data transfer, and data analysis.

UAVs have been used to measure outdoor air quality, measurements which then have been used in controlling Indoor Air Quality (IAQ). This is for instance described in Zhi *et al.* [20]. The paper describes the effort to control the IAQ by controlling the ventilation system. Measurements from UAVs flying around the building, along with static indoor sensors, were used as input for the controller. Air pumped from the outdoors into a building will naturally alter the composition of the indoor air.

2.2.2 Indoor UAVs

The research on the use of UAVs indoors typically focuses on navigation and positioning. For this use, several different technologies are in use. LIDARs are for instance used together with gyroscopes mounted on UAVs to perform simultaneous localization and mapping (SLAM) for navigation[21]. Oh *et al.* [22] implements control of a UAV using cameras mounted statically. That is, the UAV doesn't have any cameras, but rather the walls around it have cameras to detect the UAV. The paper further describes how the images are processed, and a control signal is calculated and sent to the drone. Another way of positioning UAVs indoor that has been proposed, is an ultra-wideband positioning system[23].

Min *et al.* [24] describes the development of a small UAV to be used indoors. Emphasis is put on developing a small and lightweight UAV, and the resulting product does not use any complex system for navigation or positioning. Rather, it is controlled from a remote control. The paper does describe that the UAV could potentially be used for environmental monitoring

The use of UAVs indoors to measure air quality is limited. Neumann *et al.* [25] introduce a swarm of aerial robot drones to monitor indoor air quality. 4 drones were used in experiments mapping ethanol gas distribution in an area. It did not try to map the general indoor air quality, but more spare weight was available to the drone, so more sensors could have been deployed. Some concern regarding the drones' impact in the surrounding environment was addressed, as one of the experiments had to be altered because only one drone was originally able to measure any ethanol gas. Propellers of the drone closest to the gas source were thought to "blow away" most of the gas, so none of the other drones were able to sense it. To deal with this, the gas release rate was simply increased. Apart from this, the air conditions around the attached sensors and its representation of

the air further away from the drones is not further discussed. The study concluded that the swarm of aerial nano robots was able to address gas distribution mapping, which again would be used to improve health and safety of workplaces.

2.2.3 Telemetry

Telemetry is the automatic transfer of measured variables, for instance scientific data, over greater distances using telecommunication. The term spans both wired and wireless communication[26].

Common use cases for large UAVs are military applications and precision agriculture. In these cases, video, sound, or data sampled from ground sensors can constitute telemetry. In comparison to control communication links, data links used for telemetry are usually more latency-tolerant, and have less strict security requirements. Global navigation satellite systems (GNSS), for example GPS, are often used. Different variations of signal relay and ground gateways can also be used to build communication links. These are often complex and expensive, and not aimed at short range, indoor use[27].

The drone swarm in Neumann *et al.* [25], as described in Section 2.2.2, used the Crazyflie 2.0 drone[28], which is a flying open-source development platform based on a quadcopter. The paper further states, that a 2.4 GHz data link is used for both commands and telemetry. From the drone's schematics[29] it can be seen that a nRF51822 chip[30] is used for the wireless communication.

Another paper mentioned in Section 2.2.2, Min *et al.* [24], uses a ZigBee module to communicate with a handheld controller. As this paper avoids using air quality sensors, the only signals sent over the ZigBee communication line are for steering the UAV. Even though it was not implemented, the paper describes the drones potential for environmental monitoring, signals which could be sent over the same link.

2.3 Summary of literature study

Measurements of indoor air quality consists of measuring different parameters, like gases, temperature, humidity and other matter hanging in the air. The concept of indoor air quality is not just limited to houses, but also industrial sites, aeroplanes and metros, for instance.

Traditionally, sensors used in assessing air quality have been expensive and large. In recent years, as a result of a trend of miniaturisation and the surge of IoT, small and affordable sensors have made its way into the field. As a consequence of this, a market for consumer-oriented measuring stations for IAQ has emerged.

The use of UAVs for air quality measurements has gained popularity in the recent years, but is still

considered to be a relatively fresh field. In many cases, the UAVs are used as a supplement for, or to complement static sensors in the outdoors measuring different pollutants. Most of the focus in current research lies on the air quality in urban areas, and areas surrounding industrial sites. Only to some extent are these measurements used to affect the indoor air.

For applications of UAVs indoors, research puts an emphasis on navigation and positioning. This is often a greater challenge compared to being able to move relatively freely, with few obstacles, outdoors. The use of UAVs indoors with regard to air quality measurements is limited, and concerns of the propellers impact on attached sensors' measurements is investigated to a small degree.

Wireless communication has, together with navigation and positioning, been given a lot of attention. In indoor uses of UAVs, the measurement data is normally sent over the same communication link as the control signals for the UAV.

Chapter 3

Theory

In this chapter, descriptions and explanations to theoretical and technological concepts used in this thesis are presented. Topics surrounding embedded communication interfaces, embedded hardware and sensor technologies are elaborated, and some are also illustrated.

3.1 Embedded communication interfaces

3.1.1 Inter-Integrated Circuit

Inter-Integrated Circuit, normally referred to as I^2C or IIC, is a cheap and effective communication bus used to inter-connect peripheral devices within small-scale embedded systems[31, p. 174]. It is a two-wire bus, with one clock line (SCL) and one data line (SDA). It used to be a trademarked term, so in order to avoid trademark infringements, some vendors used the term Two-Wire Interface (TWI) for the same technology. Both terminologies I^2C and TWI are in use today.

An example schematic of an I^2C-bus setup can be seen in Figure 3.1. Both data and clock lines have a pull-up resistor to a higher voltage line, which represents the logical high. As data is being transported on the bus, the communicating component pulls the bus lines to a logical low or keeps it high, all according to the data being transferred. The components are interfaced with bus addresses, which should be unique for each component on the same bus.

3.1.2 UART

UART is an abbreviation of Universal Asynchronous Receiver Transmitter. It is the simplest form of serial data interface. It does not work as a bus, so only two nodes can be connected using UART. It consists of two data lines, each one carrying the data flow in one direction. The interface is asynchronous, meaning there is no common clock line. Both nodes on the interface have to be adjusted to read and write at the same frequency. For UART, this frequency is normally called "baud rate".

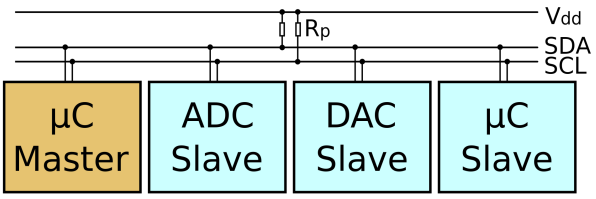


Figure 3.1: An example setup of an I2C-bus and some components[32]. V_{dd} is the high voltage supply, R_p are pull-up resistor to V_{dd}.

Synchronization between the two clocks is done by looking at special bits at the start of a transmitted message.

A UART's functionality is quite similar to that of a shift register. When data is transmitted, the UART reads data from a register, and shifts each bit onto the serial transmission line. On the receiving end, each bit is shifted into a register, which can be read and processed by a microcontroller[31, pp. 180 – 182].

3.1.3 SPI

Serial Peripheral Interface (SPI) is an interface between microcontrollers and peripheral chips. It is sometimes called a "four-wire interface", named after the wiring of the interface. The interface has a master/slave communication model, and is supported by a large amount of embedded components.

The interface requires four signals to function: Master Out Slave In (MOSI), Master In Slave Out (MISO), Serial Clock (SCLK) and Slave Select (SS). MOSI and MISO are data lines, transfering data from master to slave, or slave to master, respectively. SCLK is a common clock for all chips on the bus, and is controlled by the master. Lastly, SS is used for the master to select which of the slaves to communicate with. An arbitrary amount of slaves can be added to the bus, as long as the master can ensure a way of uniquely selecting each slave[31, p. 160] An example of an SPI setup can be seen in Figure 3.2.

3.1.4 ICSP

In-Circuit Serial Programming (ICSP), also known as In-System Programming (ISP), describes a process where a microcontroller has its program memory programmed in-circuit. If this feature is not present, the microcontroller has to be programmed before it is soldered to the embedded application. Another feature this brings with it is that the microcontroller can be re-programmed in circuit, making software development easier. Software can be made to fit custom use cases, and calibration doesn't have to be done during manufacturing[31, p. 18][34, p. 1-1].

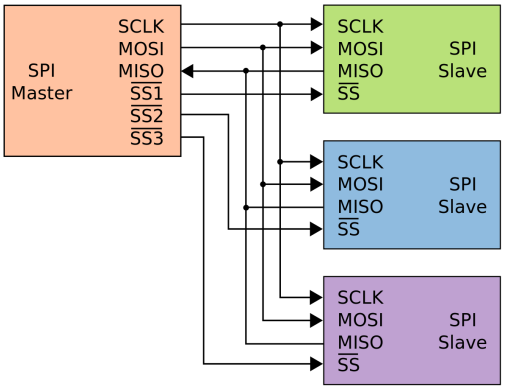


Figure 3.2: An example setup of one SPI master connected to three slaves[33]. The arrows indicate the direction of the signal. If more slaves need to be added, additional "slave select" signals are necessary in the master.

3.2 Embedded hardware

3.2.1 Printed Circuit Board

A printed circuit board, commonly referred to as a PCB, are fibreglass sheets plated with copper. Copper, being a conductive material, is etched away in patterns, so that it works as a substitute for electrical wires, and connects the electrical components on the PCB. Both active and passive components are normally soldered to the PCB after it is manufactured, securing the electrical connections to the copper of the PCB. Active components denote components like microcontrollers, op-amps and transistors, which relies on external power in order to operate. Passive components don't need external power to function, and include resistors, capacitors and diodes [31, p. 140]. Figure 3.3 shows an example of a PCB, equipped with many components on top.

3.2.2 Reflow soldering

Reflow soldering is a widely used soldering technique for PCBs. A blend of flux and solder powder, called solder paste, is applied to the PCB where components are to be jointed. Before the soldering is complete, this solder paste will work as a temporary glue, and will hold components in place. Then, when all solder paste and components are placed on the PCB, it is heated to a temperature where the solder paste melts, thus soldering the components to the PCB. This can be achieved using a reflow oven[36, Chapter 1.1.4.2].

3.2.3 Arduino

Arduino is an open-source software and hardware platform. It is intended to be easy to use, and offers a development environment which often makes embedded software development easier and quicker. On the software side, it offers its own simple cross-platform Integrated Development Environment (IDE). The software development environment offers many open-source C++ extension libraries.

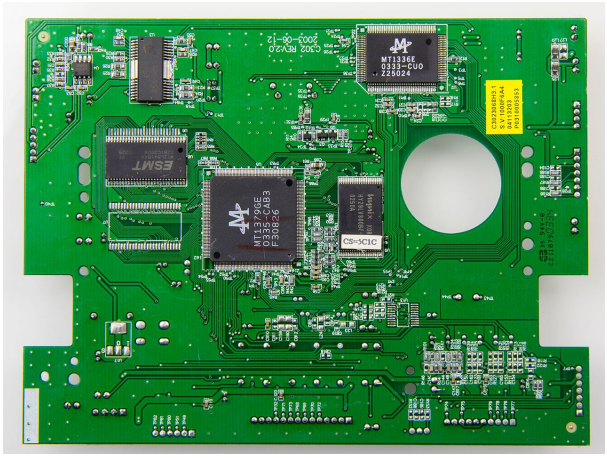


Figure 3.3: Example of a printed circuit board in a DVD player[35]. Various components are placed on the PCB, easily recognisable in dark grey. The wiring between the components can be seen in the darker shades of green on the board.

An arduino board consists of a microcontroller, power controls, buttons, LEDs and necessary passive components. It can function on its own, needing only a USB cable for power[37].

3.2.4 Decoupling capacitor

Most electrical components require a steady and reliable power supply. Power lines in electrical circuits are on the other hand susceptible to noise. To cope with this, decoupling capacitors are placed close to the power pin of electrical components, between the power line and ground. The capacitor thereby decouples the noise from the power pin, and gives the component a clean, smooth voltage source. This helps with maintaining stability in circuit performance[31, pp. 121–122].

3.2.5 PCB through-hole connectors

A common feature of PCBs are through-hole connectors. In contrast to surface-mounted components, which are soldered "on top" of a layer, many components have pins which are intended to go through several copper layers of the PCB. For this, holes are drilled through the PCB, and both sides of the hole are wired together. For a typical two-layered PCB, a components pins are soldered on the opposite side of where the component is placed[31, p. 142].

These connectors do not need to be used for components which are mounted to the PCB, but can be used as interfaces to external devices. 2.54 mm pin headers are very common. The pin headers are electrical connectors with its leading metal exposed. Opposite "female" variants are normally called header sockets. These pin headers, as well as sockets, can be connected to easily using jumper wires.

3.2.6 Breakout board

A breakout board is a specific type of PCB, namely a minimal one to be used for prototyping. Normally, it only contains the main component, like for example a sensor, and other minor components like necessary resistors and capacitors, so it can be implemented and tested quickly[38].

In some cases, the breakout boards can be used directly, and be implemented in a project. In other cases, where size and weight are important factors, having several breakout boards can be inefficient in these terms. Redundant components and inefficient physical interfaces are unalterable. In those cases, it could be beneficial to design a new custom PCB, rather than to connect several breakout boards.

3.2.7 KiCad

KiCad is an open-source, PCB designing software, available on Windows, macOS and Linux. The "Cad"-part of the name is an abbreviation of computer-aided design. In electronics, software of this kind is often referred to as EDA, or electronic design automation. It is used to draw electrical schematics, which are used to design a PCB layout. From this layout, "gerber files" can be generated. These gerber files are files that are necessary to produce a circuit board, containing information about copper layers, wiring, drilling and more[39].

3.2.8 Logic Level Converter

A logic level converter(LLC), also known as a level shifter, is a device or collection of components that translates a logical signal to another, with shifted high and low voltages. The logical value will be kept, but the voltage will be different. An example can be a UART link, where one device operates at 25 V logic, and another at 3.3 V. For these to communicate, and to avoid damaging any components, the UART signals need to be stepped up and down between 3.3 V and 25 V[40]. An example of serial communication passing through a logic level converter can be seen in Figure 3.4.

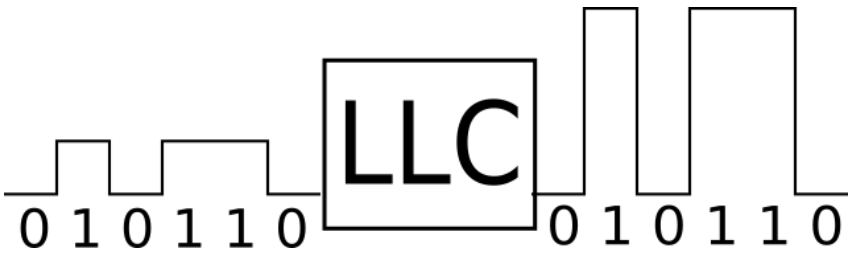


Figure 3.4: Illustration of the use of a LLC on a serial communication-link. The illustration shows how a signal is amplified or weakened, depending on which direction the signal is travelling. The binary signal on both sides will be similar.

3.3 Sensor technologies

3.3.1 Non-dispersive infrared spectrometry

Non-dispersive infrared (NDIR) spectrometry is a technology used for various gas measurements. Infrared light is beamed at a sensor measuring the intensity of the beam. The technology utilizes how various gases absorb or resonate with different wavelengths of electromagnetic radiation. The measured intensity for certain wavelengths is affected by how much of a certain gas is present between light source and sensor. From this, the amount of a given gas in the air can be estimated[41, p. 51.15].

3.3.2 Baseline correction

To compensate between physical differences in sensors, some sort of calibration routine is often performed, often during or straight after production. But even after a correct calibration, a sensor's output value, given the same measuring conditions, can shift during operation. This is called a baseline shift[42, p. 5], and is exemplified in Figure 3.5. The correction of this shift is called baseline correction. Depending on the implementation and nature of the sensor, different methods are used to correct the baseline. Sometimes, a constant multiplication of the output signal is sufficient. In other cases, more advanced signal processing is applied[43].

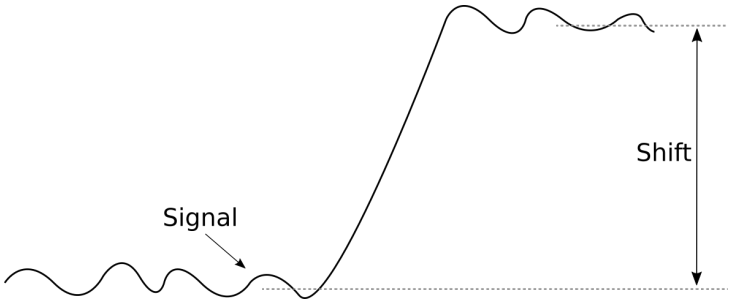


Figure 3.5: Illustration of a generic baseline shift. The nature of the signal doesn't change, but its constant offset is shifted. A baseline shift can also be applied by multiplying the signal with a constant factor alone, or in combination with an offset.

Chapter 4

Summary of Specialization Project

This master thesis expands on a specialization project from autumn 2019[44]. The overall description of the task was similar to that of this master thesis, but was limited to preliminary work and sensor testing, and not the construction of a sensor node. The following chapter summarizes the work that was done in the specialization project.

4.1 Overall summary

The literature study is split into two main parts. First, the health effects on humans with regard to indoor air quality was examined. The most important parameters of air were presented, and their inpact on human health was further elaborated. These parameters were carbon dioxide, temperature, humidity, Volatile Organic Compounds (VOC), dust and radon. The second part of the literature study focused on the measurement of said parameters. The currently available sensor technologies were presented, and explained. The chemical and physical processes used in the sensors, as well as accuracy and time scopes were presented.

Based on the parameters from the literature study, a market study was conducted. This market study looked at which sensors were available for said parameters. Emphasis was put on sensors that could easily be implemented in an embedded solution.

A handful of sensors from the market study were selected for purchase. The selection was based on their claimed performance, price and size. Small sensors were favoured, as weight and size plays a vital role for UAVs. The sensors that finally were selected for testing are listed in Table 4.1[44, p. 16]. In addition to the contents of Table 4.1, a recommandation for a temperature and humidity sensor was given. This sensor, Si7021, was not tested, as there was less uncertainty surrounding these types of sensors' accuracy.

Model	Dealer	Measures	Dimensions (mm)	Weight (g)	Communication
CCS811	Sparkfun	VOC, eCO2			I2C
SGP30	Adafruit	VOC, eCO2	18x18x3	1.1	I2C
T6713	Farnell	CO2	30x15.6x8.6		I2C, UART
GP2Y1010 AU0F	Farnell	Dust	46x30x17.6	16	Analog
SPS30	Farnell	Dust	41x41x12	26	I2C, UART

Table 4.1: Sensors purchased for testing in specialization project[44, p. 16]. Empty cells indicate that no information was stated in the data sheets.

These sensors were tested, in order to get an impression of their performance, and if they would be suitable for the intended application. To interface and communicate with the sensors, they were connected to an Arduino[45] using an I^2C bus (see Section 3.1.1). Because of problems with a shipment, which caused a some necessary cables to arrive late, the two dust sensors, namely SPS30 and GP2Y1010AU0F, were excluded from testing. There was not enough time left to implement testing of these. For the other sensors, a total of three tests were performed, all in an office environment, where the sensors outputs were logged.

4.2 Findings and remaining work

In summary, the findings from the sensor testing can be described as being inconclusive. The main problem was the lack of trusted references. As a result of this, the comparison of sensor performances was unable to offer decisive conclusions. The logged data showed that both CO_2-sensors and VOC-sensors gave different values when compared. And since there was no trusted reference, the cause could not be concluded. This is further described in Section 9.1.

The final chapter of the specialization project gives recommendations to what should be done to complete the work[44, chap. 9]. Because of the reasons mentioned above, more testing of the purchased sensors is required. They should be tested against trusted references, and more thoroughly than has already been done. The power consumption of the different sensors should also be further analyzed. Finally, sensors should all be combined on a PCB as a prototype, that can be tested with a UAV.

A simpler summary of what work needs to be done:

- Proper testing of sensors against trusted reference. This includes sensors for all parameters.
- Further analysis of power consumption of sensors.
- Final selection of sensors.
- Design and create a PCB with the selected sensors.
- Use the PCB to test the sensors' performance with a UAV.

Chapter 5

Hardware Design

In this chapter, a design for a sensor node is proposed. Firstly, criteria with which the result will be compared are defined. From this basis, a set of hardware components are chosen. Each component is presented with a justification for why it was suitable to solve the problem at hand. For some components, testing was required before a final choice could be made. Results from these tests, as well as reasoning for the choosing are also presented.

5.1 Hardware specification

5.1.1 Functionality

The main motivation for this thesis is to develop a sensor node that can measure parameters of indoor air quality, and also have physical properties that enables it to be carried by a UAV. The UAV design used as a reference is described in Eikeland [1]. Such a small UAV could run into difficulties when carrying a too heavy payload, because of this the sensor node has to be lightweight and small. The UAV has set aside a compartment of $36 \times 36 \times 21\text{mm}^3$ for sensors. A threshold for weight isn't defined, but the thesis makes simple assumptions of weight based on which sensors were used early on in the specializaton project. The summed weight adds up to 7.61 g. The sensor node should keep its weight within the same order of magnitude. The application of the sensor node also puts limitations on power consumption, as the UAV doesn't have unlimited battery capacity. Power usage needs to be addressed, and be kept as low as possible.

The focus on small size and weight puts limitations on which air quality sensors are applicable. The sensors selected and tested in the specialization project are used as a starting point, listed in Table 4.1. A microcontroller is needed in order to access sensor readings, as well as controlling communication to the UAV.

In summary, the sensor node should have the following functionality:

- Sensors measuring indoor air quality.
- Communication interface to UAV.
- Microcontroller to manage sensor readings and commnunication.
- Power controls.
- Low power consumption.
- Small physical size and low weight.

5.1.2 Additional functionality

In addition to the desired functionality mentioned in the previous section, some further desired functionality was discussed and formulated. Apart from the original implementation to being used in a UAV, it was desirable to make the node function on its own, being placed as a stationary measuring station. Specifically, it needed to be able to be powered through the 5 V interface of a micro-USB cable. This would enable it to be powered from computers and mobile power-banks.

Furthermore, a wireless communication interface was desired. If the sensor node is used outside a UAV, and therefore is unable to transfer data over a wired link to a drone controller, a wireless communication interface would make the sensor node more versatile. The additional functionality can be summarized in two main points:

- Micro-USB power
- Wireless communication interface

5.1.3 Acceptance criteria

Based on the specification described in the previous sections 5.1.1 and 5.1.2, the following acceptance criteria for the hardware of the drone were formulated:

- Temperature, humidity, pressure, CO_2 and VOC measurement capabilities.
- Smaller than $36 \times 36 \times 21\text{mm}^3$.
- Can be powered from both USB and battery.
- Sensor measurements of IAQ can be send both wirelessly and over wire.
- An upper weight limit of 10 g for UAV applications.

5.2 Exclusion of dust sensors

Sensors measuring prevalence of dust were excluded completely, both for testing and the final implementation. After consideration of the intended use case, a small UAV carrying the sensor node, the dust sensors were regarded as unfit. Physical size and weight was an important factor, and can be seen in Table 4.1.

With regard to weight, a better overview can be seen in Table 5.1, which shows that both dust sensors, SPS30 and GP2Y, have a much higher weight. UAVs for indoor applications are small, and

weight reduction is critical for them to function properly. In reality, the weight difference is even greater than the tables show. The weight of the dust sensors is the weight of the sensor only. This is also the case for the CO_2 sensor T6713. In the case of the others, CCA811 and SGP30, the weight is the weight of the breakout board, which is explained in Section 3.2.6. Therefore, the real contribution of weight on a PCB would be smaller.

Sensor	Weight(g)
CCS811	2
SGP30	1.1
T6713	4
GP2Y1010AU0F	16
SPS30	26
SUM	49.1

Table 5.1: Summed weight of all the purchased sensors or breakout boards from the spesialization project[44, p. 26].

In the case of size, the picture is similar to that of weight. The dimensions of CCS811, the lacking information in Table 4.1, was not stated in the datasheet, but is very similar to that of SGP30. Here too, because of the inefficiency of using multiple breakout boards, these sizes can become smaller. The dust sensors cannot be made smaller, as the dimensions represent the actual size of the sensors, and not a breakout board to which they are connected.

And lastly, even if the problems of weight and size were be dealt with in some way, dust sensors are not a good match with UAVs. The sensor measurements can in that case not be considered to be representative of the dust in the air, simply because the propellers of the UAV would stir up the air.

5.3 VOC sensors

The starting point for the discussion on VOC sensors is where the spesialization project left of, where the sensors CCS811 and SGP30 were concerned. The following sections pick up on this for the design on the sensor node.

5.3.1 Considerations concerning VOC sensors

CCS811 vs SPG30

As can be seen in Table 4.1, two sensors for VOC measurement were already aquired for testing. Both sensors claim in their respective data sheets to be well suited for IoT applications[46, 47].

During the course of the master thesis, a meeting was held with an employee from Airthings, a company briefly mentioned in Section 2.1.1. This is a company specializing in consumer-oriented

air quality measurements. We were told by the Airthings representative, who had been involved in testing of both CCS811 and SGP30, that he, based on his experience, recommended we pursue SGP30 over CCS811. Specifically the baseline calibration (Section 3.3.2) of SGP30 was praised as more precise in comparison with that of CCS811.

SPG30 vs SPGC3

SPG30's manufacturer, Sensirion, also offers a low-power version called SGPC3[48]. These two sensors' shapes and sizes are exactly the same. Because of this, a PCB designed to fit with one would fit the other, though a change in software would be needed. Even though the two sensors are similar, only the more power-hungry SGP30 is offered on a breakout board, and thereby easier to prototype and implement.

Regarding the lower power usage of SGPC3, it does come at a cost, namely that it doesn't offer a CO_2 estimate. This is because it is not sensitive to hydrogen gas, which is used in SPG30 to estimate the CO_2 concentration. This can be seen when comparing their respective datasheets[47, 49]. All of these discussed pros and cons are summarized in Table 5.2.

SGP30	SGPC3
- High power consumption	+ Low power consumption
+ Breakout board	- No breakout board
+ eCO_2	- no eCO_2

Table 5.2: Comparison of Sensirion sensors SGP30 and SGPC3.

Considering the poor performance of CCS811, as explained above, and the marked study performed in the specialization project, the remaining viable VOC sensors would be one of the sensirion sensors SGP30 or SGPC3. Substituting the power-hungry SGP30 with the low-power SGPC3 thus requires another way of measuring CO_2.This would give us two main alternatives: using the SGP30, or using the SGPC3 together with another CO_2-sensor, namely T6713. A comparison of these two alternatives is shown in Table 5.3.

	SGP30	SGPC3 + T6713
Avg. power	48 mA	26 mA
Weight	< 1 g	> 4 g
Size (mm)	$2.45 \times 2.45 \times 0.9$	$2.45 \times 2.45 \times 0.9$ $+30 \times 15.6 \times 8.6$

Table 5.3: Comparison of two alternatives of VOC and CO_2 measurement.

In short, these two alternatives each have their advantages: SGP30 would use more power, but be smaller and less heavy; SGPC3 together with T6713 would use less power, but be a heavier altern-

ative, and use more space.

In the end, SGP30 was chosen. It would help give answers to one of the main issues of this thesis, namely to see if indoor air quality can be measured with miniaturized sensory equipment. For this, the possibility of measuring CO_2 with a small sensor like SGP30 instead of the bigger T6713 needed to be examined. In addition to this, due to the sensor being newly introduced to the market, there were no SGPC3 sensor available at the time. By waiting for the sensor to be in stock again, the completion of the sensor would have been delayed to such an extent, that it would have caused a shortage of time for testing the completed sensor node.

5.4 Temperature and humidity sensors

5.4.1 Considerations concerning temperature/humidity sensors

In the specialization project, although it was not acquired for testing, a recommendation for a sensor measuring temperature and humidity was given. This was the sensor Si7021, and was based on parameters like size, stated accuracy, communication interface, and power consumption [44, p. 15].

In addition to this, another sensor mentioned in the specialization project, BME280, was also taken into consideration. It is listed several times in the appendixes of the specialization project[44, A1 & A2]. It was not necessary to purchase it, as we already had it in store ready for testing. Therefore, because it had originally been up for consideration, and was easily available, it was tested alongside Si7021.

	BME280		Si7021	
	min	*max*	*min*	*max*
operating voltage	1.71 V	3.6 V	1.9 V	3.6 V
standby current	0.2 µA	0.5 µA	0.06 µA	0.62 µA
temperature range	−40 °C	85 °C	−10 °C	85 °C
humidity range	0%	100 %	0%	100%
pressure range	300 hPa	1100 hPa	-	-
Accuracy				
humidity	± 3%		± 3%	
temperature	±1 °C		±0.4 °C	
pressure	±1 hPa		-	

Table 5.4: Comparison of BME280 and Si7021. Si7021 offers no pressure measurements.

The information in Table 5.4 is collected from the two sensors' data sheets[50, 51]. What the table shows, is that both temperature sensors offer a measuring range suitable for indoor usage, and well beyond. They are both interfaced over an I_2C bus. Their average power usage was not stated in the data sheets, as it depends heavily on measurement frequency. It is however very much lower

than that of CO_2 or VOC sensors discussed in Section 5.3.1. Both have low standby currents, and depending on which parameter is being measured (temperature, humidity or pressure), BME280 uses currents in the order of approximately 350 µA, Si7021 in the order of 150 µA. The exception to this is pressure measurement. This is not offered by Si7021, and has a stated current consumption of 714 µA.

5.4.2 Testing of temperature/humidity sensors BME280 and Si7031

As discussed in Section 5.4.1, the two sensors shown in Table 5.4 were up for consideration to be used in the final sensor node. In order to try out the performance of these two sensors, some tests were performed. Two tests were performed, designed to compare the two sensors against each other.

The tests were performed in my office, where temperature and relative humidity were measured every 5 seconds. Both tests lasted for approximately 24 hours. Two sensor breakout boards, one for each sensor, were connected over an I^2C bus to an Arduino. The Arduino was connected to a computer over USB, over which the measurements were sent. On the computer, a script was running that ensured the logging of the measurements. A summary of the two tests is shown in Table 5.5.

	Start time	End time	Duration	Interval	# Measurements
1	12. oct 11:22	13. oct 11:22	24 h	5 s	16748
2	13. oct 13:09	14. oct 12:14	23 h 5 min	5 s	16627

Table 5.5: Summary of tests of temperature/humidity sensors.

Results of tests comparing BME280 and Si7031

Figure 5.1 and Figure 5.2 show graphs of how the temperature and humidity readings of the two sensors developed during the tests. The difference between the measurements is also shown in separate graphs.

5.4.3 Discussion and conclusion of temperature/humidity tests

The variation of temperature in Figure 5.1 follows what you would expect from an office. During working hours, the temperature is to a larger degree varying, as people come and go, windows are opened and closed, but the temperature stays somewhere above 20, which is a comfortable indoor temperature. Outside working hours, the temperature is much more stable. From the graphs it can even be seen that at 05:00, the temperature drops both days, probably stemming from the ventilation system being switched on. This drop can also be seen in the measurements of relative humidity in Figure 5.2. Around 08:00 - 09:00, the temperature starts to rise as people start their work day.

All the figures show that both sensors give out basically the same measurements, but with an almost constant offset. Even though it is not quite clear which of the two sensors give the closest representation of the true values, both change in the same manner when the conditions change. Because

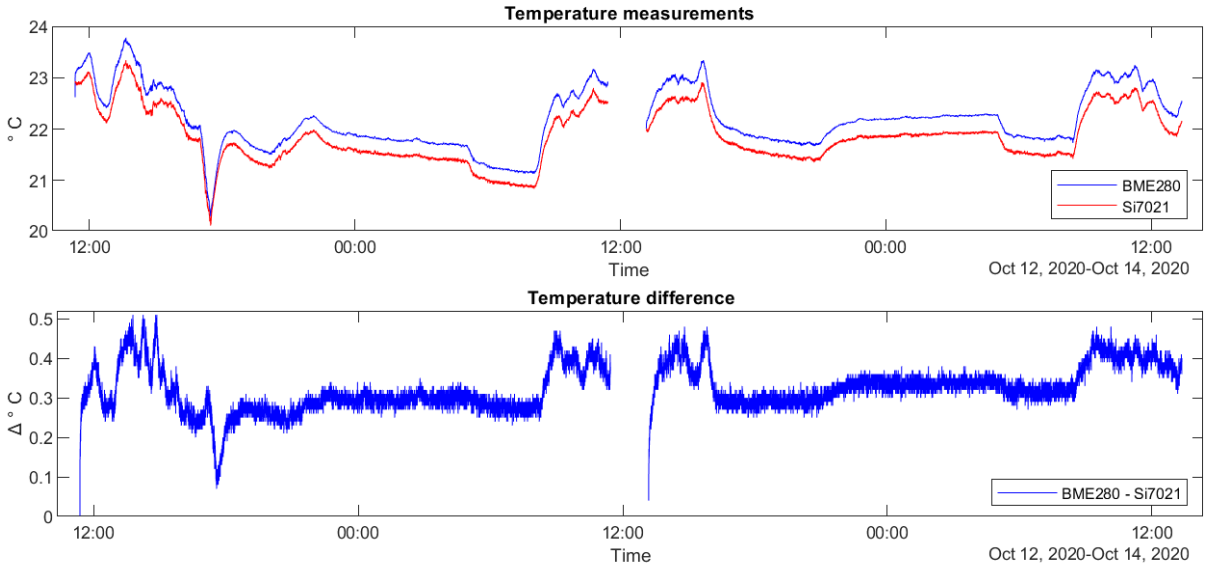


Figure 5.1: Comparison of temperature response in BME280 and Si7021.

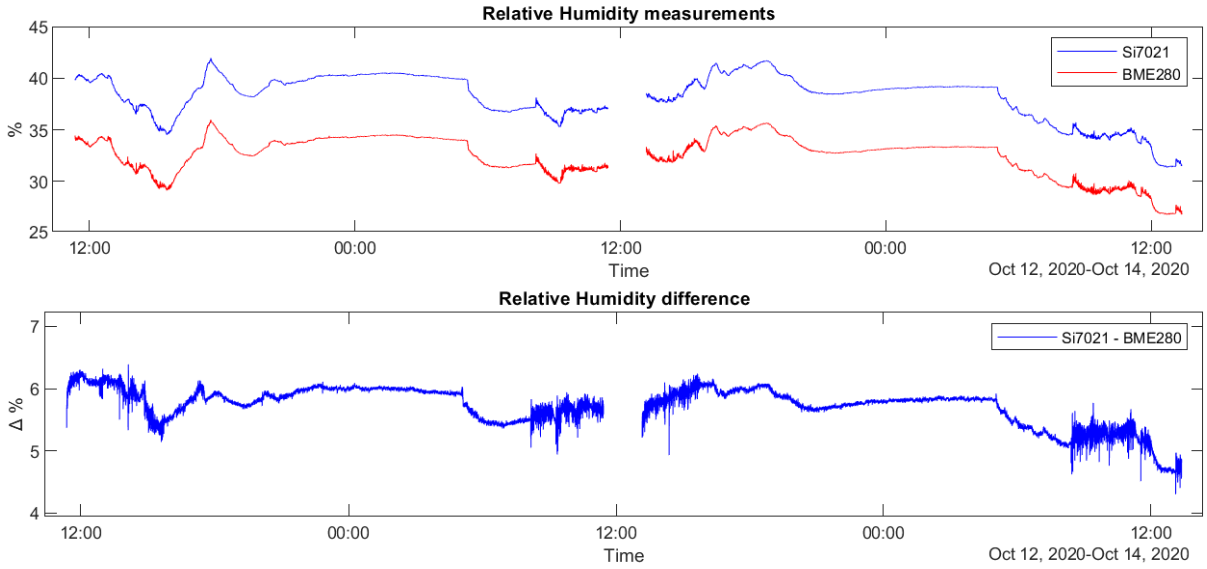


Figure 5.2: Comparison of humidity response in BME280 and Si7021.

of this, the constant offset can be taken into account and removed in software. This calibration can easily be done by comparing to a trusted measurement, and adding or removing an offset, making the measurements match.

Based on the considerations done in Section 5.4.1, and the results from the testing, the sensor

BME280 was chosen for the final implementation. Both sensors could be used for precise measurements, use little power and are lightweight and small. BME280 was chosen because it, unlike Si7021, offers measurements of pressure. Pressure sensors are further elaborated in Section 5.5.

5.4.4 Validation of temperature/humidity sensor BME280

As described in Section 5.4.3, the sensor BME280 was chosen for the final implementation. It was then further tested, by being validated against a trusted reference sensor. The test environment was the same as for the tests described in Section 5.4.2, in an office. The duration of the test was on the other hand considerably shorter. The results from this test can be seen in Figure 5.3.

The sensor used as a reference sensor was "Rotronic CP11", CP11 for short[52]. This is a handheld sensor and data logger, capable of measuring CO_2, relative humidity and temperature. The data logger was borrowed by courtesy of the department of Energy and Process Engineering at NTNU.

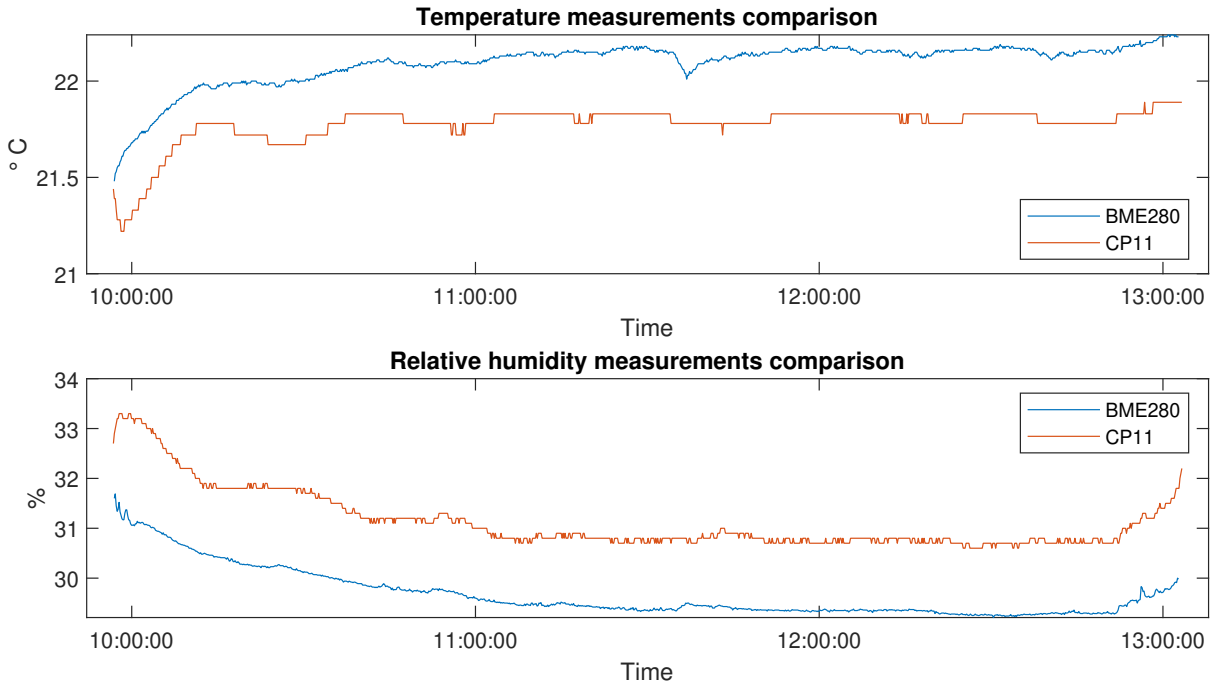


Figure 5.3: Validation of BME280 temperature and humidity response against trusted reference sensor CP11.

Figure 5.3 shows that the test results from the validation, is quite similar as when comparing the two sensor candidates against each other. The main picture is the same: the sensors' responses are similar, only with a close to constant offset. Much of the reasoning used in Section 5.4.3 holds in the case of validation as well. The constant offset can be compensated for in software, and thereby makes the BME280 sensor a fitting sensor.

5.5 Pressure sensor

Based on the work done in the spezialication project, a recommendation for the sensor MPL115A2 was given to measure pressure. This was given in a context where the sensor Si7021 was to give temperature and humidity measurements. A table showing the selection can be seen in Table 5.6.

Model	Measures	Weight(g)	Accuracy
Si7021	Temperature, humidity	1	0.4 °C 3%RH
MPL115A2	Pressure	0.61	10hPa

Table 5.6: Recommended sensor from spesialization project on pressure, temperature and humidity to use in the sensor node[44, p. 16].

Based on the conclusion in Section 5.4.3, a sensor was chosen to measure relative humidity and temperature, that wasn't the originally recommended one. An other advantage of this sensor, is that it provides pressure measurements. Because of this, a dedicated sensor for pressure, as originally recommended in the specialization project, is no longer necessary. An extra pressure sensor could have offered redundancy in pressure measurement, but as size and weight is of such importance for the final product, it was decided to be excluded.

5.6 Expansion connectors

This following sections describe components that can be added to the sensor node, but not directly through the copper wiring on the sensor node PCB. These components will not be soldered as part of the sensor node PCB in the soldering process, but can be added by being connected later. They thereby act as possible expansions. The connections are 2.54 mm through-hole connectors, as explained in Section 3.2.5. Components could thereby also be connected through pins or headers.

The connection to the components are set up in this way partly due to their own construction, being in reality breakout boards. In addition to this, using such connectors enables the sensor node to be fitted with components according to its intended use. As the sensor node should be applicable both on a UAV and independently, this will help reduce number of unused components for each implementation.

5.6.1 T6713

T6713 is a CO_2-sensor that uses NDIR technology, a technology which is discussed in Section 3.3.1. This was one of the sensors purchased for testing during the specialization project, and is listed in Table 4.1. Based on the experience from using the sensor previously, it had shown promising results. It would be in accordance with the ability to measure CO_2, as listed in the specification in Section 5.1.

At the same time as the validation of temperature and humidity sensor BME280 was performed, as described in Section 5.4.4, a similar validation of T6713 was also performed. The reference sensor CP11's CO_2 logging capabilities were used to assess T6713's sensing accuracy. The results can be seen in Figure 5.4.

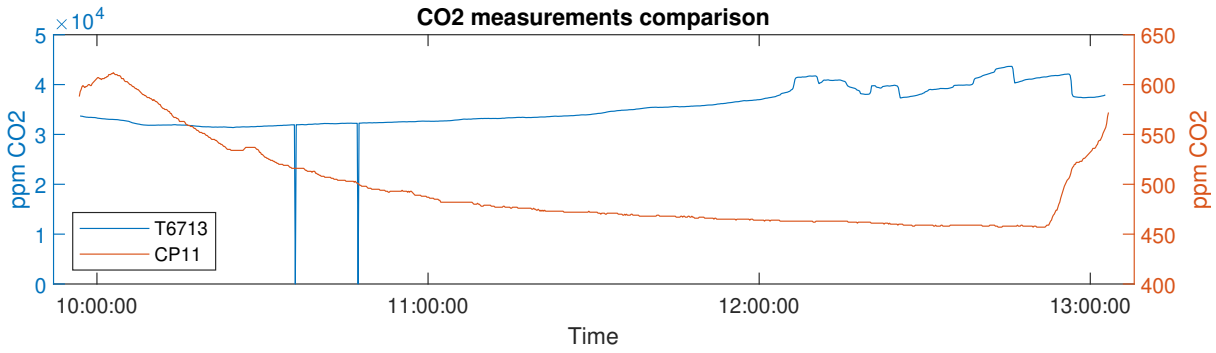


Figure 5.4: Validation of T6173 against trusted reference sensor CP11. The highly elevated values of T6713, along with the poor response are clear signs of a defective sensor.

Figure 5.4 shows very poor results, with no obvious relation between T6713's and CP11's measurements. One thing that is important to note, is that the two graph of the figure are on completely different scales. CP11's values are in the area between 400 and 600 parts per million (ppm), which is realistic. Ambient concentration lies at around 400 ppm, and indoor air would be somewhat higher as a result of respiration. The values that were read out from T6713 were up to 100 times higher, between 3 - 5%. CO_2 at these high concentrations would have a suffocating effect, and could result in humans becoming unconscious[53].

The most logical conclusion is that the T6713 CO_2 sensor was faulty. During the preparation of the testing, this was already under suspicion. The response from the sensor was not similar to that the response during testing of the same sensor a year earlier, in the specialization project. An example of this is shown in Figure 5.5. Here, the readout from T6713 is plausible.

Even though the sensor was faulty and didn't offer valuable measurements, it was decided to be included in the final design as a possible expansion. It has previously shown promising results, and is one of few available low-cost NDIR-sensors for embedded systems. By setting up PCB through-hole connectors to which it can be connected via headers, a functioning T6713 can be connected to the sensor node at will.

5.6.2 Wireless communication interface

As described in Section 5.1.2, the ability for the sensor node to communicate wirelessly was desired. For this task, an nRF24L01 was chosen. To avoid confusion, some explanation is required, as nRF24L01 often us used to describe two separate components.

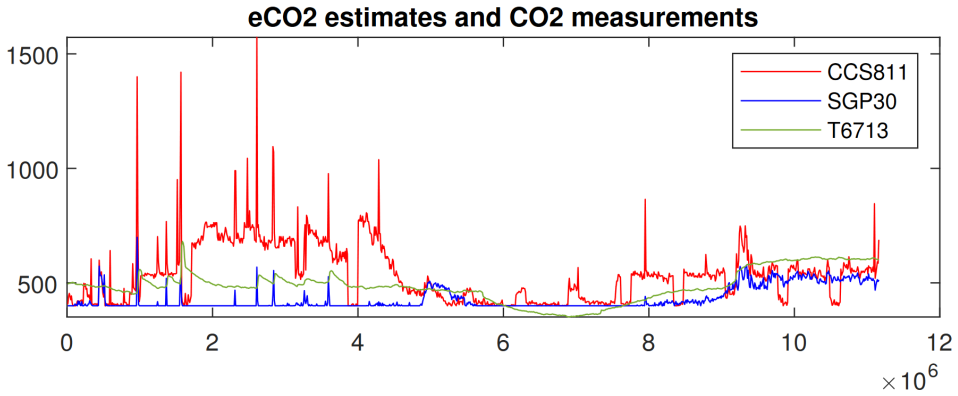


Figure 5.5: Test of CO_2 sensor and estimates from the spesialization project[44, p. 21]. Values along Y-axis are ppm, and X-axis in milliseconds.

Originally, nRF24L01 denotes an embedded transceiver chip designed for low power wireless applications. In order to receive and transmit messages, it must be interfaced with using SPI. It operates at the ISM frequency band from 2.4 GHz to 2.4835 GHz, which can be used without licensing. This chip uses a proprietary communication protocol called "Enhanced ShockBurst", which simplifies the usage of the chip, and helps dealing with noise and interference on the frequency band[54].

nRF24L01 is also used to describe a breakout board with the same chip on it. This breakout board is based on a design given in the official data sheet of the chip, and gives an example of which passive components to place and connect to the transceiver chip. It also has in integrated antenna. One of the PCB design figures form the official data sheet can be seen in Figure 5.6. From this breakout a 4x2 grid of 2.54 mm header pins can be connected, allowing an SPI connection to be set up, and also for powering the chip.

The wireless transmission does not require an existing network, like a WiFi network, or in order to function. It does require a similar component to receive the data, as it implements its own custom transmission protocol. The data sheet doesn't mention anything regarding range of transmission. This is understandable, as the data sheet describes only the transceiver chip, and range would be dependent on how an antenna is implemented. The range for the breakout board design is also not mentioned. Information about its range can on the other hand be found in less reliable sources, but must be taken with a grain of salt. Various blogs claim ranges from 50 m to 100 m. These claimed ranges are in outdoors, open conditions[55–57].

Wireless communication isn't critical for the operation of the sensor node, and may only be used in cases other than the mainly intended case, namely when used with a UAV. Because of this, the most fitting implementation would be one where it can be added on demand, but not be physically included if there is no need. This would also cut down on weight, size and power consumption. Such an implementation can be done using the 4x2 grid of header pins, making connections on the

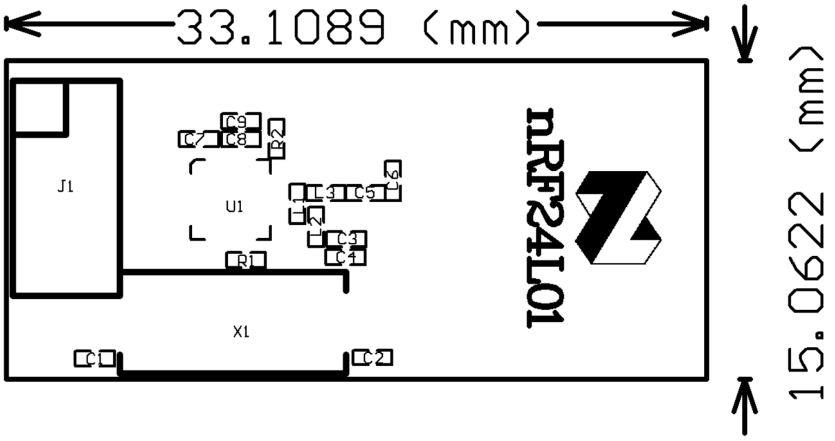


Figure 5.6: Top overlay of suggested nRF24L01 PCB layout. "U1" on the figure depicts the placement of the nRF24L01 chip. "J1" denotes the placement of the header pins. From Nordic Semiconductor [54], figure 31.

sensor node so that these pins can be connected on demand.

Testing of nRF24L01

In order to verify the range of the breakout board, some simple tests were performed. An emphasis was put on testing its performance indoors, as that was most relevant for this thesis.

Two identical circuits were wired up, both with similar components: An Arduino, an LED connected through a resistor, and the nRF24L01 breakout board. The functionality of these two circuits were simple, with one acting as a transmitter, transmitting some data repeatedly with a given frequency, whilst the other simply received the data, and did nothing with it. The nRF24L01's build in communication protocol automatically returns an acknowledgement when a packet is received, but this wasn't necessary to explicitly implement. The LED was used as a physical indicator of a successful transmission. As a measure of signal quality, the transmitter counted the number of failed attempts between each successful transmission. The number of fails between transmissions was logged using a USB link to a computer.

The transmitter was set up to send a 12 byte message to the receiver at a frequency of 1 Hz. All the tests were performed indoors. The test environment was not one designed for testing, but rather a normal office environment at the university. Because of this, unpredictable factors like interference were not controlled. It was on the other hand closer to the environments the sensor node was designed to be used in.

When the transmitter didn't receive an acknowledgement of a successful transmission from the receiver, it was assumed not to have been received properly. But sometimes it could be seen from the LEDs, that the receiver indicated having received a message, but the transmitter did not indicate a

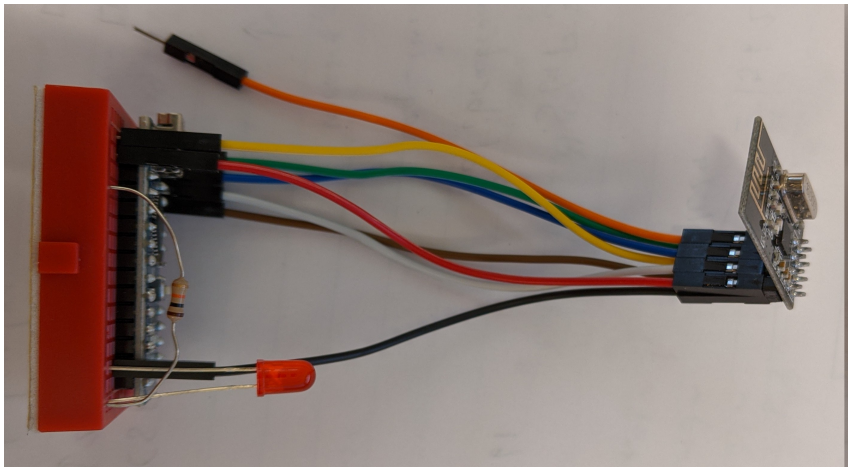


Figure 5.7: Picture of the transceiver circuit used for testing nRF24L01. On the breadboard to the left, an Arduino nano is connected with an LED for debugging purposes. To the right is the nRF24L01. The unconnected orange jumper wire is an unused interrupt signal.

successful transmission. This indicates a situation where the message was indeed received, but the acknowledgement from receiver to transmitter was lost. These cases were not logged in our testing. Table 5.7 sums up the result of the various tests, counting only cases where the transmitter received an acknowledgement.

Results from testing nRF24L01

#	Duration	Physical distance	Notes	Avg. Retries
1	1 h	1.5 m	Open line of sight	0.1489
2	20 min	6 m	Separated by thick wooden wall	1.0128
3	20 min	23 m	Separated by 4 thick wooden walls	∞
4	20 min	20 m	Open line of sight	0
5	20 min	40 m	Separated by thin wooden door	0.0192

Table 5.7: Results from testing of nRF24L01 wireless transceiver. The average retries column denotes the number of times a packet had to be re-transmitted, divided by the total number of packets that were transmitted. The ∞-symbol indicates that no packets were successfully transmitted at all.

From Table 5.7 some observations can be concluded. Firstly, the transceivers works best with a clear line of sight to each other. The tests with the most obstruction between receiver and transmitter, tests 2 and 3, are also the ones with the worst result. Test number 2 needed on average a re-transmit on every single transmission, and then some. In the case of test number 3, not a single message was successfully transmitted with acknowledgement. Another observation is that the increasing distances tested did not make a big difference for the result. This probably shows that the distances over which signal isn't strong enough to carry a signal wasn't reached. That means, that the two transceivers need to be separated by more than 40 m with a clear line of sight before distance becomes an issue.

When transmitting data over 40 m, fewer retries were needed than when data was transmitted over 1.5 m. This could be hints at signal interference. Another element that may have affected the signal is the shape of the rooms where the tests were performed. The long distance tests of 20 m and 40 m took place in long hallways, and the tunnel-like shape may actually have helped with the signal transmission.

From this a conclusion can be drawn, that in relatively open indoor environments, the transceiver module works as intended, and is able to transfer data with relative ease. The implementation of a wireless communication interface directly on the sensor node might be unnecessary, if the node is communicating over a wired interface to a drone, which has its own wireless interface. In addition to weight and size gain, an extra antenna could interfere with the control signals of the UAV. Because of this, through-hole connectors should be included, to allow for implementation of the nRF24L01 if needed.

5.7 UAV interface

The complete sensor node needs to be able to interface with the miniaturized drone described in Eikeland [1]. The flight controller used in the drone offers 5 UART connections, all of which operate at 5 V. Only one of these connections are in use by the flight controller, as a connection between the flight controller and a wireless transceiver, leaving 4 unused. Using one of the UART connections would be suitable for our sensor node to communicate with the UAV[1, p. 38].

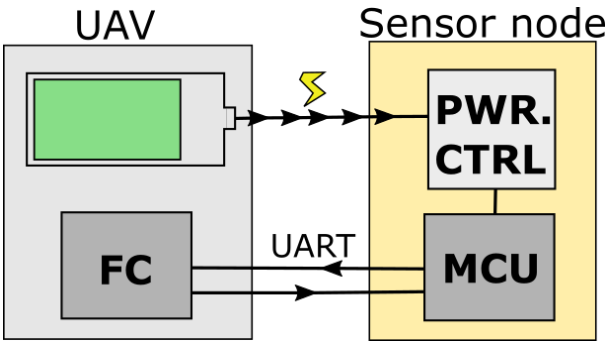


Figure 5.8: Simplified figure of sensor node - drone interface. The UAV offers power from its battery. The sensor node contains its own power control unit, regulating the voltage input for its components. The UAV flight controller communicates with the sensor node microcontroller over a UART connection.

The UAV interface consists of both power and data transfer. The nature of UART is explained in Section 3.1.2, and will deal with data transfer. There is no standard UART connector, so anything connecting the RX and TX pins correctly, as well as ensuring a common electrical ground level and baud rate, will work. For the sensor node, a simple implementation of two 2.54 mm pin connectors was chosen, one for RX and one for TX. This is similar to what was done for expansion connectors

in Section 5.6. Likewise, two similar connectors for input power from the UAV battery were chosen, one for high voltage and one for electrical ground. As the power input from the UAV battery can be unpredictable, a power management system is needed for the sensor node. This is further elaborated in Section 5.9.1.

5.8 Microcontroller

A suitable microcontroller for the sensor node would have to be able to communicate with both the chosen sensors and the UAV, or over a wireless link, as described in Section 5.1. All the nuances of timing and sequence regarding communication with the embedded sensors should be handled by the microcontroller. Depending on the sensor node's use, the microcontroller would be programmed differently, so a way of re-programming the MCU is desired. A summary of requirements for the MCU is listed in Table 5.8.

Requirement	Description/Details
UART	Interface to communicate with UAV
I^2C	Interface to digital sensors
SPI	Interface to wireless module
Re-programming	Adjust functionality according to use case
Program memory	16 kB
Low power	Low power consumption

Table 5.8: Summary of requirements for the microcontroller of the sensor node.

Based on the requirements in Table 5.8, the microcontroller ATmega168p was chosen. All communication interfaces UART, SPI and I^2C are supported. It offers an ICSP interface, making in re-programmable. It features 16 kB flash program memory, and is the low-power version of ATmega168, thus offers low-power sleep modes[58]. The requirement of 16 kB program memory was set based on the software size during testing of components. During testing, the program took up bout 11 kB, thus leaving some headroom for improvement of the software.

To clock the MCU, a 16 MHz external crystal was chosen. This is a very common clock frequency for these MCUs, as it is the standard for Arduino boards, which use similar MCUs. This clock enables asynchonous UART connections [58, p. 213], and lets the MCU be programmed with Arduino software. The MCU can run at input voltages ranging from 2.7 V to 5.5 V. Despite a lower current consumption at low voltage levels, a 5 V input should be used. This lets the MCU interface with the T6713 CO_2-sensor, which uses 5 V logic. In addition to this, it lets the MCU interface with the UAV flight controller's 5 V UART without the need for logic level converters. This gives an estimated active current of the ATmega168p MCU of 12 mA[58, p. 387].

The physical ICSP connection connection is normally implemented using 2×3 2.54 mm header pins, and was chosen for the sensor node. Other necessary components are two capacitors of 20 pF for the clock crystal, decoupling capacitor at the power supply, and a resistor at the reset pin [58, p. 38][59].

Additionally, an LED in series with a resistor was included. It could be connected to one of the MCUs digital output pins, to be used as a heartbeat indicator.

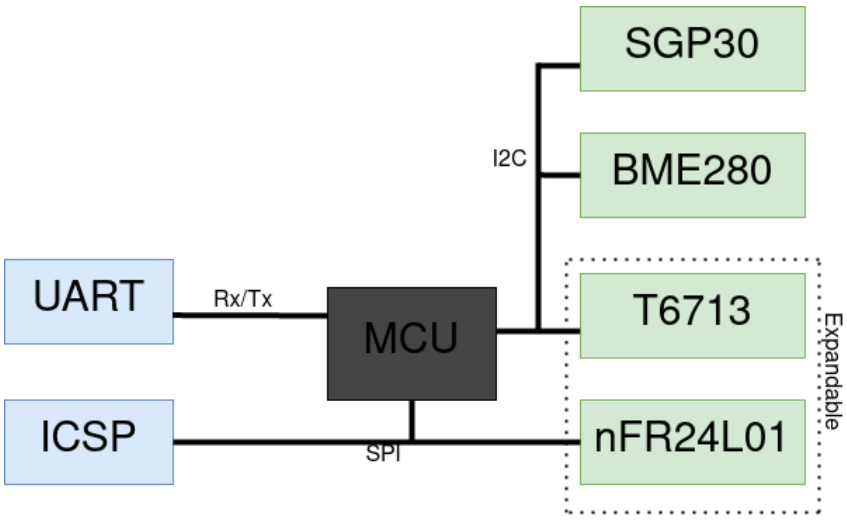


Figure 5.9: A draft of the main sensor node components and their connections to the microcontroller. In this figure, no consideration is made with regard to power and voltage controls.

5.9 Power and voltage controls

5.9.1 Voltage regulator

Proposed components for the sensor node operate at different voltage levels, and require stable supply of power. To deal with this, voltage regulators were introduced in the design.

Regulated voltage	Component	Avg. current	Peak current
5 V	T6713	20 mA	155 mA
	ATmega168p	12 mA	-
3.3 V	nRF24L01	-	13.5 mA
	BME280	3.6 µA	714 µA
1.8 V	SGP30	48 mA	-

Table 5.9: Current usage of sensor node components. "-" indicates that it isn't mentioned in data sheet, or dependent on software implementation.

The highest required stable voltage supply is 5 V. From the specification in Section 5.1, the sensor node should have two different ways of being powered: via a USB interface, or by drawing power from a drone's battery. The USB interface offers a 5 V power line[31, p. 208]. The drone battery pack offers a nominal voltage of 11.1 V[1, p. 59]. This means that there may or may not be an external 5 V supply available, if not, the sensor node needs to be able to provide its own regulated 5 V supply. As Table 5.9 shows, this 5 V needs to be further stepped down to 3.3 V and 1.8 V as well, to meet

the requirements of the various components. This setup is depicted in Figure 5.10

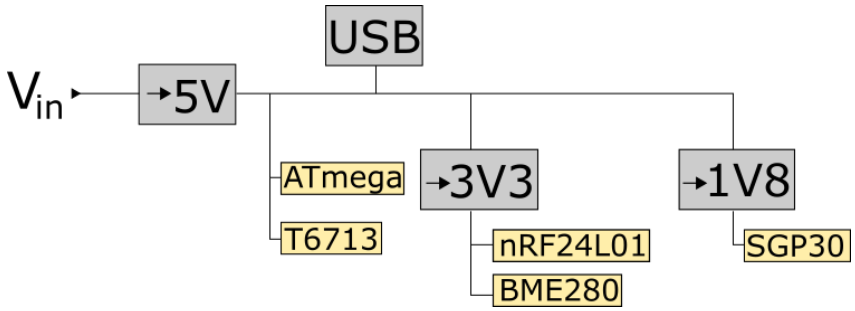


Figure 5.10: Configuration of voltage regulators, and voltage-dependent components. Three voltage regulators are necessary for the three different voltage levels.

Regulator	Max output current	Input range	Output voltage
TLV1117-50	800 mA	6.4 V - 15 V	5 V
LP2985-3.3	150 mA	2.2 V - 16 V	3.3 V
MIC5504-1.8	300 mA	2.5 V - 5.5 V	1.8 V

Table 5.10: Voltage and current ratings of the chosen voltage regulators[60–62].

The selected voltage regulators, with their current and voltage ratings, are listed in Table 5.10. These were selected to correspond with the power need of the components at each voltage level, listed in Table 5.9. The 5 V regulator needed to have an output current higher than the sum of all components at all power levels, as the lower voltage regulators draw their power from the 5 V line.

5.9.2 Logic level converters

The three of the sensors intended to be incorporated in the sensor node, offer an I^2C communication interface. An I^2C bus was intended to be used as communication bus, to enable the microcontroller to read sensor data. Table 5.11 shows a summary of the relevant components, and their voltage requirements.

Component	Function	I^2C logic high level
ATmega168p	Microcontroller	5 V
T6713	CO_2-sensor	5 V
BME280	Temperature/humidity/pressure-sensor	3.3 V
SGP30	VOC/eCO_2-sensor	1.8 V

Table 5.11: I^2C logic level for various components of the sensor node.

Because of the different voltage requirements, logic level converters were required on the I^2C bus. The concept of logic level converters is explained in Section 3.2.8. Based on out components, two

shifts were necessary, from 5 V to 3.3 V, and then another from 3.3 V to 1.8 V.

Figure 5.11 shows how a single LLC is implemented. An n-channel MOSFET tansistor connected with two resistors like the figure shows, does so that both sides of the LLC can pull the signal low. When this happens, the logical level on the other side will follow, and also go low. Because of this, the signal will travel and propagate throughout the entire bus, independently of voltage level.

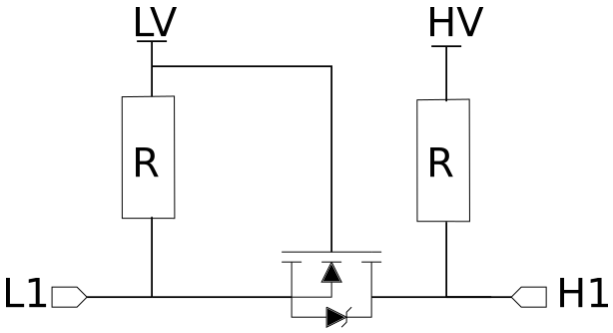


Figure 5.11: Implementation of an LLC. L1 and H1 will have the same logic level, with H1's logical high as HV, L1's logical high as LV.

5.10 Hardware design summary

This chapter has discussed and stated the reasons for choosing the sensor node's components. Table 5.12 lists all these, as well as passive components necessary for them to function. Necessary passive components like decoupling capacitors and resistors were stated in the active components' data sheets. The table sorts components by "subsystem", that is which function they serve in the sensor node. Moreover, the chapters where the respective components' implementation is discussed is also listed.

Subsystem	Chapter	Component	Value	Amount
Voltage regulators	5.9.1	TLV1115	5 V	1
		LP2985	3.3 V	1
		MIC5504	1.8 V	1
		1N4001 diode		1
		Capacitor	100 µF	1
			47 µF	1
			2.2 µF	1
			1 µF	3
			100 nF	1
	5.7	Pin header	1 × 2	1
LLC	5.9.2	BSS138		4
		Resistor	10 kΩ	6
Sensors	5.3	SGP30		1
	5.4	BME280		1
		Capacitor	100 nF	3
Interface pin headers	5.8	ICSP	2 × 3	1
	5.7	UART	1 × 2	1
	5.9.1	USB micro-B		1
Microcontroller	5.8	ATmega168p		1
		Crystal	16 MHz	1
		Capacitor	100 nF	1
			20 pF	2
		Resistor	10 kΩ	1
			4.7 kΩ	1
		LED	Red	1
Expansion sockets	5.6.2	nRF24L01	2 × 3	1
	5.6.1	T6713	1 × 6	1

Table 5.12: Summary of all components of the sensor node. "Value" column is context-dependant. Depending on the component, it denotes regulated output voltage, passive capacitance, pin layout, electrical resistance, colour or oscillator frequency.

Chapter 6

Software Design

Based on the components chosen in the hardware design, this chapter sets specifications and criteria for the software that have to be respected with the given hardware. Functionality that need to be implemented in software for the sensor node is then presented.

6.1 Software specification

The software of the sensor node should enable it to function with a UAV, and report its measurements to the UAV. The functionality is thereby the same as was described for the hardware design, in Section 5.1. For the sensor node to operate as intended, the software needs to take the hardware choices listed in Section 5.10 into consideration. The hardware specification also mentions the possibility of the sensor node to function as a measurement station transferring its measurement wirelessly. This was thought as capability that could be implemented if needed, and not a necessity. Because of this, it is not part of the software specifications.

The parts chosen for the sensor node puts certain requirements and limitations on software. I^2C bus and UART communication has to be enabled. The power consumption should also be as low as possible. All of this has to happen within the size limitations that the program memory of ATmega168p sets. In summary, the software should enable the following:

- Interface sensors over I^2C.
- UART communication to drone.
- LED Heartbeat.
- Minimize power usage.

6.1.1 Acceptance criteria

In order to be able to judge whether the implemented software is sufficient for the sensor node, some acceptance criteria were formulated. These are based on the specification in Section 6.1.

- Reliable, evenly spaced intervals of LED blink.
- UART receive and transmit without errors.
- Be able to read sensor values from all connected sensors.
- Software size of less than 16 kB.

6.2 UART interface

In the UAV design of Eikeland [1], the flight controller offers UART connections through which it can be interfaced. This is also described in Section 5.7, and is the justification of why it was implemented in the hardware of the sensor node. What the UAV design doesn't mention, is how the software of the flight controller handles UART connections. Because of this, some assumptions have to be made about how the software of the sensor node should be designed in order to work with the UAV.

For the setup of the UART communication, a baud rate needs to be specified. Here, a commonly used baud rate like 9600 bits per second can be implemented. Another assumption needs to be made regarding the direction of communication between the UAV and its peripherals functions. In order to assure flexibility in data sample rate, a way of polling for data over UART should be implemented. For this solution to work with the UAV, its flight controller software would have to poll the sensor node over UART when sensor readings should be reported. The assumptions of baud rate and polling instead of automatic periodic reporting, can be changed if needed. A slight reprogramming of the sensor node software would then be necessary.

6.3 Sensors' software requirements

All sensors included in the hardware design, SGP30, BME280, and T6173 are interfaced over an I^2C bus. The sensors have unique bus addresses, so extracting data from the sensors should not be problematic.

SGP30, unlike the other two sensors, requires some extra software considerations. In order for its internal calibration algorithm to function properly, which calculates a sensor baseline, a measurement request has to be sent every second[47, p. 9]. Additionally, the sensor baseline should be stored periodically in some non-volatile memory, so the sensor can be initialized with the stored baseline value. This backup of the baseline is recommended to happen approximately once every hour[63, p. 10].

6.4 MCU sleep

Instead of letting the microcontroller be active all the time, and thereby having a high power consumption, power saving modes of the MCU should be utilized. The chosen MCU ATmega168p offers one sleep mode where the UART remains active, and that is the "idle" state. The UART needs to remain active, so that the sensor node is able to receive polling calls over UART at any time. A timer should be set up, so that the MCU can be woken up from the idle state reliably. The intervals at

which SGP30 has to be measured, and a low delay for a UART response has to be taken into account for the sleep time of the MCU.

6.5 Exclusion of wireless interface driver

The sensor node has the possibility of having a wireless interface added to it, as described in Section 5.6.2. This can be used where the sensor node does not have a UART connection where the sensor data is sent, like for example to a UAV flight controller. Since this wireless interface is not a necessary part of the sensor node's main use case, it does not need to be implemented. A change of software in the sensor node MCU would therefore be necessary if a switch from UART to wireless interface were to be performed.

6.6 Software design summary

The software needs to be implemented with the hardware choices made in Chapter 5 in mind. To be able to communicate with the UAV, UART communication is necessary. Due to lack of functional specification of the software of the UAV, some assumptions regarding the UART communication have to be made. A way to request measurements over UART should be implemented.

The MCU on the sensor node has to be able to read measurement values from the sensors over an I^2C bus. Some extra requirements on the I^2C bus communications are demanded by the SGP30 sensor, which have to be respected. Timing sensitive processes of the sensor forces a timer to be implemented in the MCU. The MCU should utilize its power saving sleep functions in order to use less power.

Chapter 7

Hardware Implementation

In this chapter, the path from hardware design to an actual functioning sensor node prototype is described. It covers computer-aided design, manufacturing and validation of hardware components.

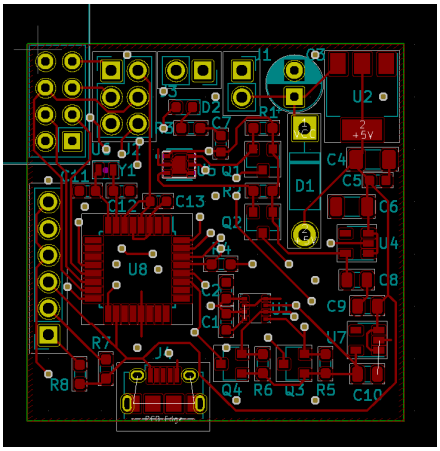
7.1 Printed circuit board design

Based on the design in Chapter 5, a PCB schematic and design was implemented using KiCad. This software is further elaborated in Section 3.2.7. For most passive components, that is resistors and capacitors, parts of the form factor "0603" were chosen. This is an abbreviation of 0.06 inch×0.03 inch, which is a common PCB surface-mounted component size. It is small, thereby helping creating a smaller PCB, yet large enough to be easily handled without precise machinery. Some components, like the high-capacitance capacitors, were not available in such a small form factor, and were therefore of a somewhat bigger form factor.

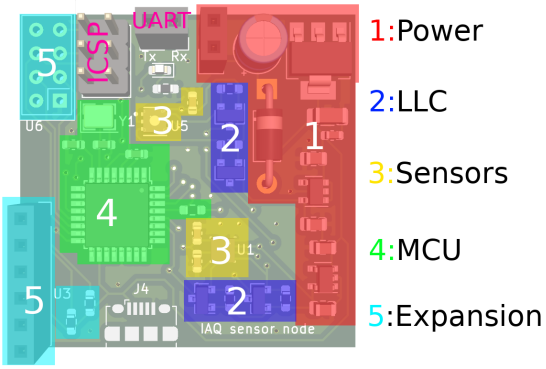
After all components were added, connected and placed correctly, the resulting design schematics were rendered and exported. The design consists of both electrical schematics, and a component placement layout. KiCad also offers an additional way of previewing the design, and that by rendering a 3D-model. Most components have their own 3D model, which is rendered together with all other on the correct placement on the PCB. Some models are quite exact presentations of the component, whilst other use more generic presentations. The final design can be seen in Figure 7.1, with the wiring and component layout in Figure 7.1a and a rough figure of component placement in Figure 7.1b. The complete electrical schematics can be found in Appendix A.

PCB manufacturing

To manufacture the PCB, a Chinese PCB manufacturer called PCBWay was used[64]. Gerber files generated from KiCad are everything that was required for the manufacturer to create physical PCBs



(a) PCB layout



(b) Rough breakdown of the placements of the sensor node components.

Figure 7.1: Figures of the final PCB design.

of the design. After the necessary files were uploaded and payment was complete, a total of 7 days elapsed before the PCB was manufactured and transported all the way to Norway by mail.

7.2 PCB component soldering

Table 5.12 gives a complete list of all components that was to be added to the PCB. To solder all components to the PCB board, the reflow soldering technique was utilized. Reflow soldering is briefly explained in Section 3.2.2. This was performed at Omega Verksted, a student-run workshop at NTNU, that offers help and equipment for hobby electronics projects[65]. There, all necessary equipment for reflow soldering was available. After soldering paste and most components listed in Table 5.12 had been applied to the PCB, it was run through a reflow oven. The reflow oven at the workshop was a model SEF 548.04G[66] Extension headers and header pins were excluded from the reflow soldering, as the plastic could have melted. These were instead soldered on by hand later.

7.3 Hardware result

7.3.1 Sensor node hardware summary

When all soldering was complete and all components were connected, the sensor node construction was finished. The final result can be viewed in Figure 7.2. Figure 7.2a shows the sensor node only, whilst Figure 7.2b also includes an nRF24L01 wireless transceiver as well as a T6713 NDIR CO_2-sensor connected through expansion sockets.

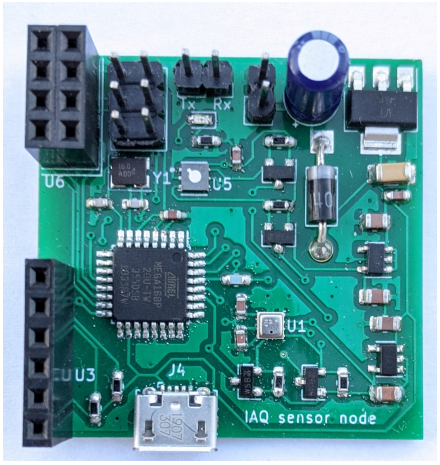
Table 7.1 summarises some physical quialities of the finished sensor node. Its weight in grams has a margin of error of 0.2 g. The size, particularly when expansion connectors are used, is easier to

[1] These entries are empty as the tests were not performed. I was put in quarantine at an unfortunate time due to the COVID-19 pandemic, and was unable to complete them.

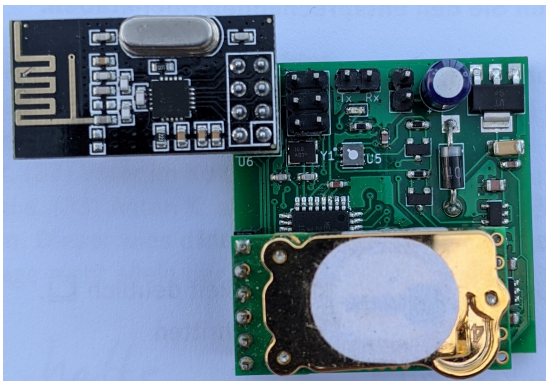
Hardware	Weight	Size	Current (11.1 V)
Node	8 g	$36 \times 36 \times 15\text{mm}^3$	61 mA
Node + T6713	12 g	$36 \times 36 \times 22\text{mm}^3$	[1]
Node + T6713 + nRF24L01	13.4 g	$36 \times 36 \times 22\text{mm}^3 + 33 \times 15 \times 21\text{mm}^3$	[1]

Table 7.1: Summary of sensor node/expansion connector combinations. The current was measured at 11.1 V input voltage, which is similar to that of the battery voltage in the UAV in Eikeland [1].

understand when seen in relation to Figure 7.2b. The green PCB has an area of $36 \times 36\text{mm}^2$. The total size when the wireless transceiver is included makes the total area not be rectangular, and is therefore denoted as an addition of size in the table. The height increase, from 15 mm to 22 mm results from the fact that the CO_2-sensor T6713's highest point sits higher than the higher point of the bare, green PCB's components. The height of the sensor node with wireless transceiver and CO_2-sensor could have been smaller, by soldering these directly rather than via sockets. Sockets were preferred over a smaller physical size, as it enabled on-demand removal and addition of these.



(a) Bare sensor node with unused expansion headers.



(b) Sensor node with wireless and CO_2-sensor expansions added.

Figure 7.2: Photos of the completed physical sensor node.

7.3.2 Hardware validation

Some simple testing and validation was performed to ensure no mistakes were made during assembly and soldering. Various voltages were applied to power inputs, and the voltage level outputs from the various voltage regulators of the sensor node were measured using a multimeter. All power lines held the specified voltage levels, no matter which input voltage was applied to the sensor node. The same is true when it was powered using a USB cable. Later, a simple program making the LED blink was programmed on to the MCU, and the same variation of input voltages was performed. This also showed that the ICSP implementation was correct, as the MCU could be programmed. No errors were detected, and the sensor node was considered to be finished.

Chapter 8

Software Implementation

The functionality described in Chapter 6 offered an overview of necessary functionality. In this chapter, the implementation of this design, and choices made underway, are documented.

8.1 Software architecture

Based on the software design criteria presented in Chapter 6, it was recognised that a very complex setup for the software would be unnecessary. Summed up very roughly, the software of the sensor node should contain three aspects: Keep track of time, read data from sensors, and transmit this data to a receiver. For this, an infinite loop software structure was deemed suitable. This denotes a sequence of instructions that will run over and over again, and not stop until it is externally interrupted.

To implement the software, C++ was chosen as preferred programming language. The microcontroller family which our ATMega168p is a part of, AVR, features a well-established toolchain for C and C++ programming called avr-gcc[67]. C and C++ are very common in use for development of these microcontrollers, and thereby offers a large existing code-base from which help can be drawn, and makes it easier to find answers to difficulties during programming.

8.2 Sensor drivers

The software interface between the MCU and the sensors BME280 and SGP30 was implemented using pre-developed drivers [68, 69]. For the T6713 sensor, a self-implemented driver was applied. This was the same driver as was developed for T6713 during the specialization project. They are written in C++, and define classes and methods for handling I^2C bus communication with their respecitve sensors.

Our T6713 sensor performed very poorly during testing, as described in Section 5.6.1. From this it

was concluded that the sensor likely was broken. Because of this, reading from the sensor was not included in the final software of the sensor node, rendering the use of the T6713 driver unnecessary.

8.3 MCU sleep settings

In the data sheet's description of timer interrupts, a general equation for the frequency of timer interrupts is given. This is shown in Equation (8.1)[58, p. 161]. $f_{clk\_I/O}$ is the clock frequency of the timer, in our case the same as the external crystal oscillator of 16 MHz. The N of the equation is a tuneable prescaler, whose largest possible value is 1024. Lastly, $OCRnx$ is a register value from 0 to 255. The lowest possible interrupt frequency attainable, with no fractional component, is the combination shown in Equation (8.2). Because the SGP30 sensor needs to be accessed every second, it is important that the interrupt frequency doesn't have a fractural component. In this way, by counting as many interrupts as the frequency value in hertz, one second can reliably be measured.

$$f_{INT} = \frac{f_{clk\_I/O}}{N \cdot (1 + OCRnx)} \tag{8.1}$$

$$f_{INT} = \frac{16\,\text{MHz}}{1024 \cdot (1 + 124)} = 125\,\text{Hz} \tag{8.2}$$

8.4 Program flow

At start, the software initializes its UART communication, its timers and interrupts, as well as starting up all sensors. After this, an eternal loop is entered, which does the following, based on certain conditions:

- Read SGP30 sensor values.
- Read BME280 sensor values.
- Store SGP30 baseline values.
- Check UART if sensor values have been polled, and return a print of sensor values.

After these actions, the main loop enters a sleep mode. Outside the main loop, a timed interrupt controls when the conditions for the listed actions are set to true, as well as waking the main loop from its sleep. As a result of this, the reading of sensors is performed at regular intervals. Additionally, the interrupt routine controls a regular blinking of the LED, acting as a heartbeat. A graphic representation of the program flow is depicted in Figure 8.1.

8.5 Software result

The files resulting from the implementation of the software can be seen in Appendix B. There, the file placement of the functions that are in use is also denoted. This includes the developed driver for T6713, even though it was unused due to reasons explained earlier. The sensor node MCU was programmed using the tools listed in Table 8.1. For this, the ICSP interface was utilized, as elaborated in Section 5.8.

	Employed tool	Reference
Code-editor	Visual Studio Code	[70]
Code compiler	avr-gcc	[67]
Programming software	Avrdude	[71]
ISCP programmer	USBasp	[72]

Table 8.1: Employed tools and technologies for programming the microcontroller of the sensor node.

During the software implementation, it was noticed that SGP30's measurement would "get stuck" at either very low or very high values, and not change, if it was initialized with a baseline. The setting of the baseline was supposed to remove the need for a long burn-in period, during which a calibration routine is performed by the sensor. Because of this error, the advantages of the baseline calibration could not be properly utilized. Whether the error stemmed from hardware or software was not determined.

Instead of initializing from a stored baseline, the sensor instrad needs to perform a 12 hour long burn-in and calibration. This is described in SGP30's data sheet, and additionally requires the sensor to be read every second, and the baseline read every hour, to ensure proper execution of the calibration. This is no different from how the sensor node software was implemented anyway, so by just having the sensor node run normally for 12 hours before use, the calibration routine will be complete, and the sensor node ready for use.

The final software ended up requiring 11 726 B, thereby keeping it well below the maximum size of 16 kB. This free storage space is also a reflection of the fact, that this excludes code for T6713 and nRF24L01, the CO_2-sensor and wireless transceiver, respectively.

Figure 8.1: Flowchart of sensor node software. The main loop to the left stops at "sleep_mode()", and resumes after the interrupt routine "ISR(TIMER_COMPA_vect)" finishes. The main loop alternates between "sleep_mode()" and executing commands will not stop until the sensor node is powered off.

Chapter 9

Sensor Node Testing

After the completion of both a functioning hardware and software, the performance of the sensor node prototype had to be examined more closely. This chapter describes the reasoning behind, and how three tests were set up to test the sensor node's ability to give an accurate description of indoor air quality.

9.1 On the use of reference sensors

When sensors are to be implemented for a certain application, their accuracy and trustworthiness must be considered. Both can be assessed when a sensor's measurements are compared with a trusted reference. The reference is then considered to be a "true reading", with which other readings can be compared.

If sensor readings are compared without a trusted reference, disparities are not easy to explain. First of all, if there are no disparities, it does not mean the measurements are accurate and true. They might all be wrong, just coincidentally in the same manner. If there is a disparity, it can be hard to deduct which of the sensors is accurate and which is not.

Disparities can result from many different causes. A sensor might be precise enough for the intended application, but have a constant offset, so it always produces wrong readings. This can easily be noticed when compared with true readings, and can then be compensated for, either by re-calibration of the sensor or mathematically in software. The same is true for scaling errors, where a sensors response to changes in the measured parameter might give a too great or small response. Errors like this can occur across the whole range of the measurement spectrum, or only a certain area. Another common error is noise in measurements. With a trusted reference, analytical tools can be used to filter out the noise to approximate the true value, for instance using different averaging techniques.

9.1.1 Reference sensors in this thesis

In the case of this thesis, there is some leeway with regard to precision of the sensors. As discussed in the specialization project, the accuracy of the sensors need only be so good, that it gives meaningful information about the indoor air quality[44, p. 12]. For example, a difference in 0.01 °C does not have a noticeable effect in indoor air quality. But even though the overall accuracy of some sensors doesn't need to be very precise, the sensors used in this projects should be tested against a known reference, considering all the reasons mentioned in Section 9.1.

As mentioned in Section 4.2, the lack of trusted measurements to compare with was problematic in the specialization project. Both measurements of CO_2 and VOC had differing results across different sensors. Whether this was caused by one or more sensors being imprecise, faulty, or correct but scaled, could not be concluded. To avoid the same ambiguity, each sensor of the sensor node was tested either against a trusted reference sensor, or by provoking a change in a measured variable.

9.2 Description of tests

In the following sections, descriptions of the tests of the sensor node are presented. The measurement capabilities of the sensor node was tested over a total of three tests, each directed at different parameters. The tests' respective results are prestented in Chapter 10.

9.2.1 Temperature, humidity and CO_2

All three parameters of temperature, relative humidity and CO_2 were measured simultaneously. On the sensor node, BME280 gave temperature and humidity measurements, and SGP30 an estimate of CO_2. The CO_2-sensor T6713 was not included, as our copy was faulty. This is further described in Figure 5.4. As reference to the measurements, the sensor and data logger "Rotronic CP11" did measurements of the same parameters alongside the sensor node. CP11 is the same sensor that was used for validating BME280's temperature and humidity measurements in Section 5.4.4, as well as T6713's CO_2 measurements in Figure 5.4. These previous tests of the sensors only tested the breakout boards, and not the complete implementation of the sensor node.

The sensor node and reference sensor were placed on top of an unused desk in a small, but actively used office. The reasoning behind this was to do the measurements in an environment where the instruments wouldn't be disturbed by people sitting too close by, but still experience changing air conditions. The conditions could be changing due to ventilation kicking in, people entering and working there, or windows being opened and closed.

Because of the errors detected in the setting of SGP30's baseline values, as described in Section 8.5, the sensor node was started earlier, so it had enough time to perform its 12 h long burn-in calibration routine. Because of limited storage capabilites in CP11, and the long duration of the test, a sampling interval of 30 seconds was used. As limited storage was no issue for the other sensors, an

interval of 10 seconds was used. Furthermore, because of a too low battery level in CP11, its CO_2 measurements began after a short while to drift. The sensors screen gave notice of this, and the batteries were changed early next morning. Because of this, the CO_2 measurements are only valid after this point.

Sensors	Sampling interval	Start date	Start time	End date	End time
SGP30, BME280	10 s	01.12-2020	15:54	04.12-2020	12:46
CP11	30 s	02.12-2020	12:21	04.12-2020	12:46
CP11 (CO_2)	30 s	03.12-2020	08:20	04.12-2020	12:46

Table 9.1: Summary of final sensor node testing.

9.2.2 VOC

During the thesis, we were not able to find a good reference for VOC values in the air. A reliable reference sensor could not be found, and a reliable way of creating conditions with known VOC values was not feasible. What instead was done, was to see if SGP30 would reliably result in higher measured values, when exposed to VOC sources. For this, a simple test setup with ethanol was used. Ethanol, which is the common alcohol in alcoholic beverages, is a VOC, and the sensor should therefore show a spike when measuring close to a strong spirit.

A small amount of 40 % vodka was poured in a small cup. With the sensor node powered up, it was carried over the cup, so that it was positioned directly above its opening, held there for a short time, and then removed. This was repeated two times. A sampling interval of 10 seconds was used.

9.2.3 Pressure

Apart from testing BME280's measurement of temperature and relative humidity, which is described in Section 9.2.1, BME280 also features pressure measurements. This had to be tested separately from temperature and relative humidity, as the reference sensor used in that test, CP11, didn't offer pressure measurements.

This test was based on meteorological measurement. The sensor node was placed outdoors at a balcony, being powered over a micro-USB cable. Wireless communication was set up, with a receiver similar to that which was used in Section 5.6.2. An Arduino connected to a nRF24L01 which receives measurements from the sensor node, and transfers the data over a USB serial line to a computer, which logs the data and timestamps it. The pressure measurements were sent from the sensor node to the receiver with 5 minute intervals.

To validate the sensor node's measurements, meteorological measurements from the Norwegian Meteorological Institute were used. By using a publicly available API, meteorological measurements for the sensor nodes geographical placement were fetched every hour[73]. Here, the interval of one hour was used, because this corresponded to the update frequency of the API's data. Because of this

long interval between updated pressure measurements, the test was performed over a long period of time, 7 days in total.

Chapter 10

Results

This chapter presents the results from all tests described in Section 9.2. The sensor measurements are grouped together according to which sensors were tested simultaneously. Discussion and further analysis of the result is further elaborated in Chapter 11.

10.1 Test results

10.1.1 Temperature, humidity and CO_2

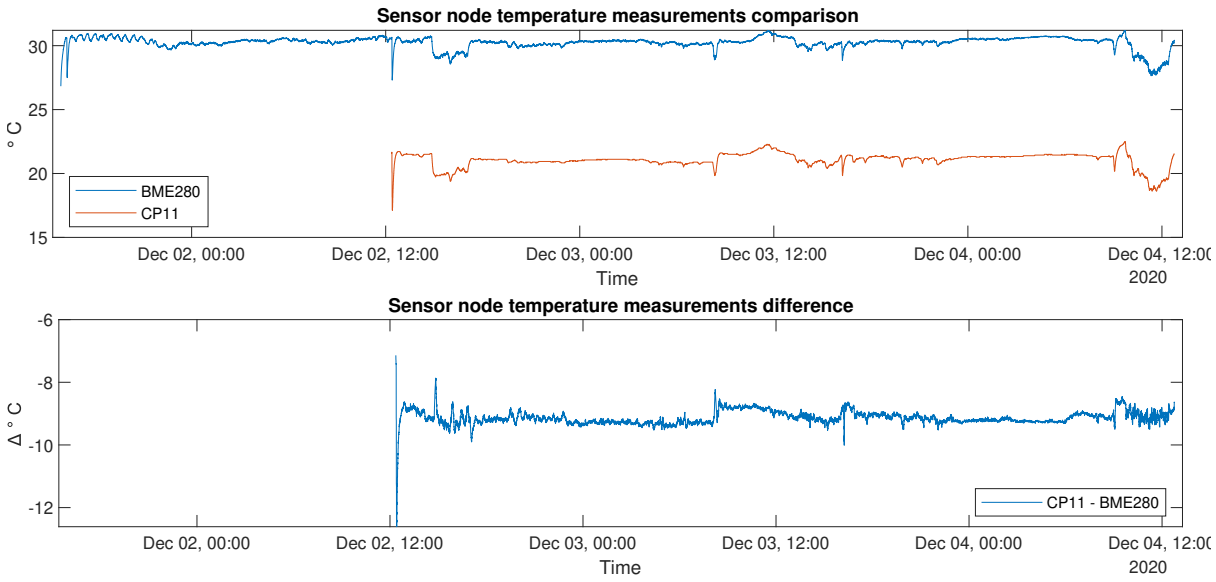


Figure 10.1: Comparison and difference of the sensor node's BME280 and CP11 datalogger. The topmost graph shows variation of measured temperature in degrees celsius over time, whilst the bottommost graph shows the difference of the top where both measurements exist.

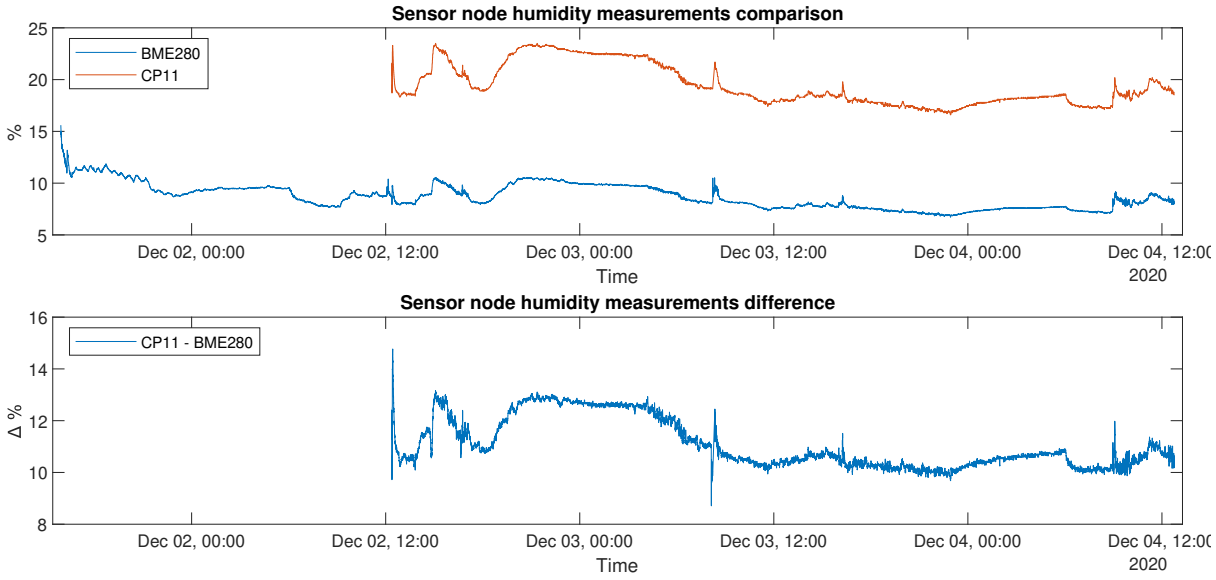


Figure 10.2: Comparison and difference of the sensor node's BME280 and CP11 datalogger. The topmost graph shows variation of measured relative humidity in percentage over time, whilst the bottommost graphs shows the difference of the top where both measurements exist.

In the graphs in Figure 10.1 and Figure 10.2, measurements from BME280 start earlier than those of CP11. This is stems from the fact that both BME280 and SGP30 were tested simultaneously. SG30 required a 12 hour calibration routine before it was ready. Later, after this calibration was finised, CP11 was set up to start logging.

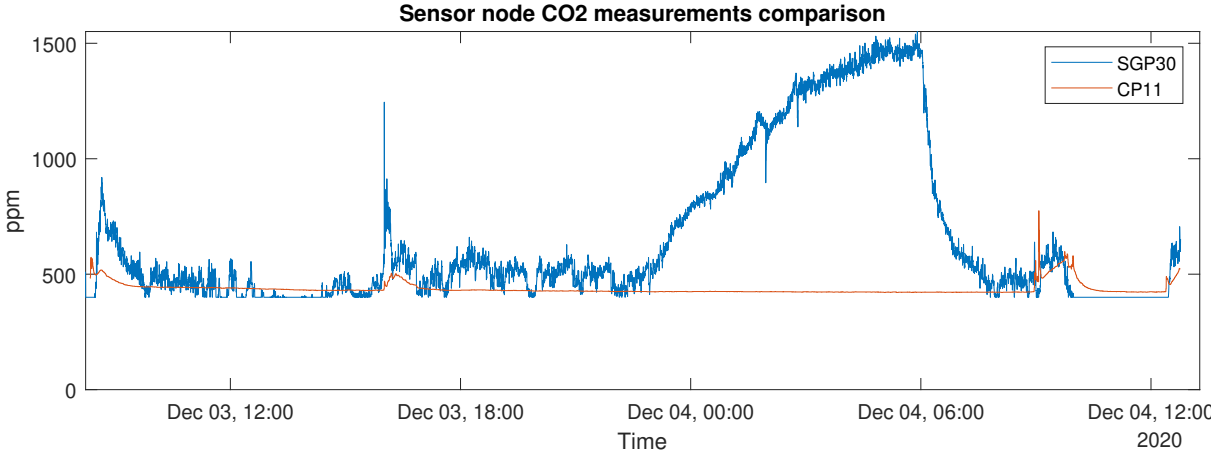


Figure 10.3: Comparison of the sensor node's SGP30 and CP11 datalogger. The graph shows SGP30's CO_2 estimate and CP11's measured CO_2 values, indicated in parts per million (ppm) over time.

Unlike the graphs for temperature and humidity, the graph showing CO_2 values in Figure 10.3 doesn't cover the same time span. It is considerably shorter, and stems from a malfunction in CP11's CO_2 measurements, as mentioned in Section 9.2.1.

10.1.2 VOC

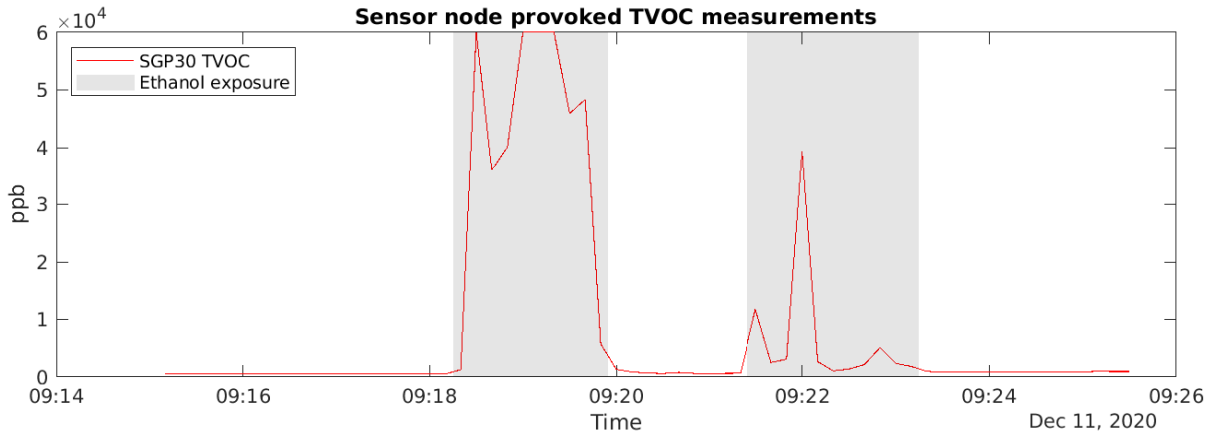


Figure 10.4: Input to and response from the sensor node's SGP30 sensor. The VOC values are measured in parts per billion (ppb) over time. The greyed out area shows time of sensor input.

The greyed out area of Figure 10.4, marked as "ethanol exposure" and sensor input simply indicates when the sensor node was held above the ethanol source.

10.1.3 Pressure

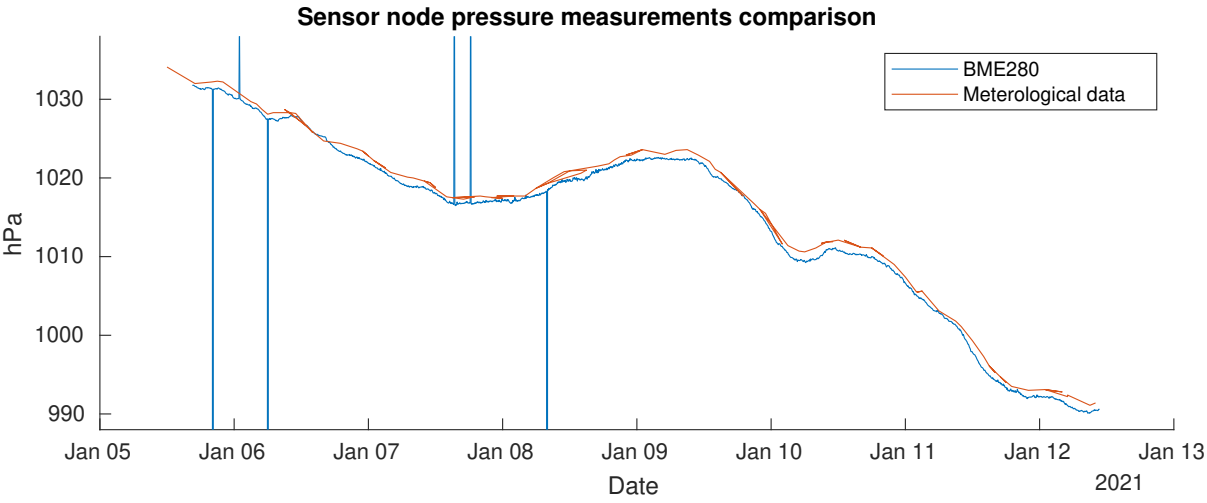


Figure 10.5: Comparison of the sensor node's BME280 and meteorological data from the Norwegian Meteorological Institute. The graph shows pressure measurements measured in hectopascal over time.

Chapter 11

Discussion

This chapter is split into three main parts, that all discuss the resulting sensor node that was developed for this thesis, and to which degree it fulfils the goals that originally were set out. Firstly, the test results from Chapter 9 are discussed, with a focus on the sensors' performance. Then, the sensor node hardware and software is examined and elaborated. Lastly, based on the argumentation from the two earlier sections, the acceptance criteria from Section 5.1.3 and Section 6.1.1 are listed and rated.

11.1 Test results

Temperature, Humidity and Pressure

Figure 10.1, Figure 10.2 and Figure 10.5 show the results from testing temperature, humidity and pressure, respectively. From these, a clear picture is painted: The sensor measurements follow changes in the measured parameters well and responsively, but with a relatively constant offset. This is very similar to the observations made when sensors for the node were selected during the design process. One major difference between those tests and the ones of the finished sensor node is that previously, in the case of temperature and humidity, two uncalibrated sensors were tested against each other. Whereas the final testing used a trusted data logger as reference. Because of this, the sensors' response and accuracy can be verified and accepted with an even larger degree of certainty. The sensors' performance is stable, and no errors were detected during testing.

The pressure measurements execution stands out, as it was the only one outdoors. With regard to pressure, unlike temperature and humidity, there normally isn't substantial difference in pressure outside compared to inside. Also, some outliers can be seen in the pressure measurements. The number of pressure measurements is very large, having sampled every fifth minute for a week, and with this in consideration, a handful of outliers doesn't keep it from being qualified as stable. From this, it can be concluded that they offer a reliable and precise measurement of these parameters.

CO_2

An unfavourable property of SG30's CO_2 estimate is its "spikiness". From the test's graph in Figure 10.3 it can be seen, that its estimate value signal is noisy, and has big changes in value over short time spans. Because of its noisiness relative to the smooth reference signal, a plot of the signals' difference was excluded, as it would mostly just portray SGP30's noise. The noisy nature of the signal prevents us from being able to use the raw signals from the sensor node to get a snapshot of the CO_2 conditions in the air.

If the noise of the signal is to be dealt with, more complexity needs to be introduced to the sensor node software. For instance, a moving mean could continuously be calculated, possibly resulting in better estimates. This would increase the size of the software. Also, calculating means and averages on microcontrollers are not always trivial, as the few bits used to represent numbers could behave poorly in divisions and representations of non-integer numbers.

But still, if SGP30's estimate signal were smoothed, it deviates at times vastly from the actual CO_2 value. Partly, overly aggressive responses to slight increases in CO_2 can be observed. Sometimes just as spikes, but also long-lasting. This could be an indication of a too sensible response, which could be adjusted with a constant factor. Other times, the deviation seems completely unprovoked. This is not just temporarily with some outliers which easily can be discarded, but continuously over hours. The sensor tries to estimate the CO_2 value from other gases in the air, which might change without change in CO_2, and clearly doesn't provide a stable and accurate representation of the CO_2 conditions in the air.

VOC

This test tried to stimulate SGP30's VOC sensing capabilities, and provoke a response. Because the test was set up in such a simple way, the answers drawn from it could only give elementary insight to the sensor's performance. The results in Figure 10.4 do on the other hand give one clear answer: when the sensor node is exposed to very high concentrations of VOC, it does show on its measurements. Some measurement values even reach the sensor's maximum value of 60 000 ppm. It must be said, that the test environment was not a representation of typical conditions in air, as VOC concentrations in air are normally not anywhere near that of the VOC concentrations in the immediate proximity of ethanol. In a real-world usage of the sensor node, having such an intense VOC source close to the sensors would alter the measurements extensively. That is, if the objective is to measure VOC concentration in a room. A similar case would be a thermometer used for measuring the temperature in a room, but having it placed directly above a radiator. The measurement would not be representative for the entire room.

One observation worth noticing is that the second response in Figure 10.4 is smaller than the first. It could indicate that there in fact was less ethanol in the air at that time. That seems somewhat unlikely, as there are less than two minutes between the exposures. Another possibility is that the on-chip calibration, which is continuously performed by the SGP30 sensor, has kicked in and re-

strained the response at the second exposure. And lastly, due to the nontechnical execution of the test, inaccuracies may have arised simply from the sensor node being held differently above the ethanol source. A decisive conclusion regarding the differences in response can't be drawn.

During use of the sensor node at other times, variations in VOC measurements were observed, and not to extreme values like those measured during this test. But these observations bring little concluding evidence, as there was nothing to compare with. From these tests, conclusions regarding the precision of the sensor measurements can't be drawn. Apart from the little information that could be acquired from the test, VOC is a difficult parameter to measure. A total image of all VOCs in air is hard to attain, because there are so many different types, and sensors respond differently depending on which VOCs are present, and not just their overall concentration.

11.2 Sensor node implementation

The following discussion puts focus on the resulting sensor node's hardware and software, and to which degree it matches with the goals that were set out in the start of the project.

11.2.1 Hardware

With regard to the quality of the hardware, good results were observed. Apart from one exception, all components behaved as expected, and thus enabled the sensor node to be soldered, programmed and used, and in the end produce air quality measurements. The one component of concern was the CO_2 sensor T6713. Problems with this sensor was noticed already before the completion of the sensor node, during the selection process of in the sensor node design. This was, as mentioned in Section 5.6.1, thought to be a problem of the single sensor unit, and not a general problem of the model, based on previous successful uses of the sensor in the specialization project. But even though almost all the components worked as intended, some potential flaws and improvements were recognized.

Power

Even though no specific threshold or acceptance criteria was set, a stated overall goal for the project was that the sensor node would use small amounts of electric power. Some choices that were made serves as counter to this. The MCU was chosen to run at 5 V and 16 MHz with an external crystal oscillator. According to the MCU's data sheet, lowering one or both of these parameters would result in a lesser power requirement[58, p. 387]. For the other selected components, three power buses of 5 V, 3.3 V and 1.8 V would be needed either way, as T6713, nRF24L01 and SGP30 need to be supplied at these voltages, respectively. The problem of decreasing the voltage level of the MCU is that the logic level of its digital communication interfaces would be altered accordingly. For the I^2C bus, that wouldn't be a problem, as two logic level converters already are in use, converting between these logical voltage levels. But the UART, which needs to be able to interface with a UAV operating at 5 V, would need a logic level converter if the MCU did not use 5 V logic. In the end, a lower voltage level for the MCU would be in direct opposition to a simpler construction with few components.

Decreasing the MCU frequency, on the other hand, wouldn't affect other components on the sensor node. One of the arguments for choosing 16 MHz was to ensure that the MCU could be programmed using Arduino software. In this case, a potential decrease in power usage by lower MCU clock could have been achieved, at the expense of making the implementation of software more difficult. It is not straightforward to estimate how much this affects the total power consumption of the node. The software utilizes sleep modes to a large extent, during which the current of the MCU is much lower, and not as dependant on frequency.

A successful result from the implementation is the versatility with regard to power source. The sensor node is able to respect all the above mentioned requirements of voltage levels and power consumption of components because of its voltage regulators, and can be powered externally in several ways. The first alternative is to be powered over a USB cable, through the sensor node's USB micro port. Alternatively, as can be seen in Table 5.10, any DC input source that one physically can connect to the input terminals will do. This is, if the source is able to supply enough current, and has a relatively stable voltage between 6.4 V to 15 V. As a result of this, the sensor node doesn't work any different if the power source is a battery on a UAV, a USB port on a PC or something completely different, and the requirement for voltage is met.

Size

The UAV from Eikeland [1] set physical limitations on the size we wanted the sensor node to have. Its sensor chamber, which was set aside for a sensor node like the one proposed in this thesis, left a space of $36 \times 36 \times 21\text{mm}^3$ to be used. The completed sensor node from this thesis, as is stated in Table 7.1, ended up using the same ground area of $36 \times 36\text{mm}^2$. Its height, depending on whether CO_2 sensor T6713 is included, amounts to 15 mm or 22 mm. If the node were to fit, it would be just barely, and would rule out the possibility of including the CO_2 sensor. Using too much space in the sensor chamber might also prevent air flow, again affecting and possibly impairing measurements.

The current size is right at the borderline of how small it should be, so small improvements could be enough to make it feasible. Some general improvements might be implementable, better use of PCB surface area from component placement, using both sides of the PCB for components, and using smaller components. Some components' sizes were chosen with the fact that it had to be done by hand in mind. Smaller components could have been used, but could, on the other hand, have increased the risk of errors like soldering short circuits. The node's height when using expanded hardware, in our case T6713, could be significantly lower if it were soldered directly to the PCB, and not connected through pin headers. By doing this, the flexibility of being able to add or remove the component at will is removed.

A smaller PCB would also make the node more lightweight, another area where the sensor node is at the borderline of what was regarded as acceptable. A limit of 10 g was set as acceptance criterion, and the sensor node weighs 8 g or 12 g, depending on whether T6713 is included. A breach of this criterion isn't as severe as that of size, because Eikeland [1] doesn't state an absolute weight limit like it does size, and the weight is still within the same order of magnitude.

Other remarks

Measurement data from the sensor node has two ways of being transferred, either by wire over UART or wirelessly. When connected using UART, the data could still be transferred wirelessly, on the same communication link for instance a UAV would use to communicate with its ground controller. But data could also be transferred wirelessly using its own wireless functionality. One implication of the current design is the need for a dedicated receiver module, in this case. The wireless transceiver module, nRF24L01, can only communicate with other similar modules. This means that it cannot be built on top of a pre-existing network, like WiFi or Bluetooth. The lack of a design proposal for a receiver module increases the cost if implementation, as a receiver would have to be designed and set up. In cases with very specific implementations, a custom receiver would probably be designed anyway, but an extra barrier is still present, preventing early testing.

If the sensor node is carried by a UAV, there are two alternatives for measuring air quality. The first is that it is done literally on the fly, simultaneously as the UAV is flying. The other possibility is that the UAV flies around, landing at places where it wants to perform measurements. In this case, the UAV would be static, and propellers of, thus offering more stable aerial conditions around the UAV. The results that have been produced in this project is more representative of the latter, as there was no evaluation of how measurements were affected or impacted if there were active propellers nearby. As mentioned in the literature study in Section 2.2.2, this has also hardly been discussed in other studies where UAVs are used to measure air quality.

11.2.2 Software

One major drawback of the developed software that was discovered was regarding sensor initialization based on previous baselines. As described in Section 8.5, this simply didn't work, and it seemed to make SGP30's measurements freeze or "get stuck" at seemingly random values. The desired functionality was that the baseline initialization would have us skip the long burn-in period of the sensor, but this bug made that impossible. Instead, for that specific sensor to reach a ready state, a 12 h burn-in routine had to be performed. The obvious downside is that the affected measurements this sensor attributed to the sensor node, namely VOC and eCO_2, could now not be trusted for a long time after the sensor node's start-up. If it were to be used with a UAV, this would rule the measurements completely out, as it is very unlikely to be used by a drone which has a flight time of over 12 hours. Also, if the sensor node were used as a stationary measuring device, an unexpected restart, like for instance from a power outage, it would restart, and run into the same problem.

The software for the drone was supposed to deal with this, by storing sensor baselines periodically to non-volatile memory. This periodic backup functioned properly, but as the initialization from the stored backup didn't work, so in reality it only added software complexity without adding utility. The error could potentially stem from various sources, and an answer wasn't found in the course of the project. Because of this, a hardware error cannot be ruled out. Another possible source is the imported open-source sensor driver[69]. When parts of the code aren't developed from scratch, overview could get lost. The driver could contain errors in its methods, or be incompatible with the rest of the software, resulting in unwanted behaviour. With the discussion of the test results in mind,

the loss of proper eCO_2 values isn't severe, as they probably aren't too reliable anyway. The fact that VOC measurement values from the same sensor could be rendered useless is more critical, but even in that case the sensor's reliability is not fully confirmed.

The subject of using imported libraries also brings up some more general considerations. Apart from the risk of importing errors, imported code is often written in a more general way, or originally intended for another implementations. As a result of this, the resulting software could end up being less optimized for its use than it would if all code were written from scratch. This could impact software speed, power efficiency and software size. An example of an unoptimal limitation in this project already discussed with regard to power. A 16 MHz external crystal was required on the MCU to run the drivers, which again lead to higher power consumption. Using the code does on the other hand decrease the work required for software implementation. Cost of implementation has to be weighted against its advantages and disadvantages in this case. Since a major bug in the code could be a result of this, the use of imported drivers can in this case be considered a misjudgement.

Another possible misjudgement is the upper limit of 16 kB set on software size. This limit was respected, and was no cause for complications for the sensor node software. But still, the limit might have been unnecessary. The sensor node was constructed in a way that some components could be added and removed at will, to make it more flexible to changes in use cases. In this respect, the rigid upper bound on software size decreases flexibility. An important factor to have in mind is that the chosen MCU, ATmega168p, also exists in a 32 kB variant, called ATmega328p. There are no other differences, and thereby no other downsides software-wise, in using that instead. The only cost is a purely monetary, with the latter normally being a couple of cents more expensive. For large-scale production it would be important, but not in a more experimental use like in this project.

11.2.3 Project execution

Based on the motivation and end goals of the thesis, most of what was originally planned has been performed. The hardware in the shape of the sensor node was completed without any major problems. With the time frame in mind, the sensor node hardware was completed early enough so there was enough time for software development and testing. A functioning software was implemented, further showing the usability of the sensor node. In addition, the tests that were performed of the implemented sensors were quite informative, and opened up for interpretation of their respective performances. Most of the short-comings cannot be blamed on external events and limitations, but rather as a result of less optimal design choices.

Some hiccups were run into during the course of the semester. The evaluation of the node's power consumption was not completed, because of an unforeseen imposed quarantine due to the COVID-19 pandemic. This happened towards the end of the semester, and enough time was not left after the quarantine to have the remaining testing performed. Overall, it had small implications, but made the analysis of the sensor node's power consumption less thorough. Another example of unfortunate unforeseen drawback was the defect T6713 CO_2 sensor, which also prevented its performance from being validated.

11.3 Acceptance criteria

This section summarizes the chapter by listing all the acceptance criteria presented in Section 5.1.3 and Section 6.1.1. They are listed and rated based on whether the criterion is considered to be met, justified in the discussion in this chapter.

	Acceptance criterion	Result
Hardware	Temperature, humidity, pressure, CO_2 and VOC measurement capabilities.	✓
	Smaller than $36 \times 36 \times 21\text{mm}^3$.	✗
	Can be powered from both USB and battery.	✓
	Sensor measurements of IAQ can be send both wirelessly and over wire.	✓
	An upper weight limit of 10 g for UAV applications.	✗
Software	Reliable, evenly spaced intervals of LED blink.	✓
	UART receive and transmit without errors.	✓
	Be able to read sensor values from all connected sensors.	✓
	Software size of less than 16 kB.	✓

Table 11.1: List of all the acceptance criteria for both hardware and software of the sensor node. The rightmost column lists the criterion's result: fulfilled or unfulfilled.

Chapter 12

Conclusion

In this thesis, a prototype of a sensor node for measuring indoor air quality was successfully developed and tested. It was developed with the goal of being small and lightweight, which would enable it to be carried as payload by a small UAV. The sensor node consists of an ATmega microcontroller, voltage regulators, various passive components as well as several sensors. The sensors measure various parameters in the air that give a picture of the air quality, such as humidity, temperature, pressure and CO_2. Additionally, the sensor node's design enables it to be expanded by adding an extra CO_2 sensor and a wireless transceiver for measurement data transmission.

It was shown that the sensor node was easily programmable and reprogrammable, by use of its ICSP interface. This is an important feature, which enables the sensor node to be customized to various use cases. The final hardware exceeded the limitations of weight and size which were defined at the start of the thesis. These limitations were built on assumptions of what a miniaturized UAV could carry. Several ways of reducing both weight and size have been proposed, which would deal with these issues.

Testing showed a mixed result of the implemented sensors' accuracy. Measurements of humidity, temperature and pressure were precise, whilst the measurement of more complex parameters of CO_2 and VOC were lacking. SGP30's CO_2 measurements were found to be wrong, and its VOC measuring accuracy couldn't be properly validated due to a lack of reference sensors. The sensor was shown to respond to VOC stimulation.

To conclude the thesis, even though it has been shown that there is much room for improvement, the thesis has shown that we were able to design and implement a small and lightweight sensor node measuring parameters concerning indoor air quality. This doesn't include critical parameters of CO_2 and VOC. To which degree a similar sensor node would produce precise measurements whilst in-flight has not been explored, and would require further investigation.

Chapter 13

Future Work

Based on what was attained from this thesis, both shortcomings and answers, this chapter offers some pointers to what should be in focus, if the work is to be picked up and continued. The following list sums up suggestions for further work:

- Test the sensor nodes performance with a UAV, or in conditions simulating those of being in-air aboard a UAV. In particular, the sensors' readings should be in focus, as they may for example be affected by propeller wind.

- Investigate alternatives to SGP30 for CO_2 measurements. Possible alternatives should be small and lightweight, to fit with the rest of the sensor node profile. If good alternatives are found, a future revision of the sensor node should update its design in a possible new sensor node.

- Investigate further the power consumption of the sensor node. Required power will be affected by how the sensor node is implemented, and vary according to which sensors are in use, and if data is transferred wirelessly or not.

- Set up and test a full system for measuring, transferring and storing data. This can be done either using the sensor node's own wireless transceiver, or in connection with a UAV. Depending on implementation, some form of receiver or base station should be developed.

Bibliography

[1] S. S. Eikeland, 'Development of a miniaturized drone for indoor climate monitoring', TTK4900 - Engineering Cybernetics, Master's thesis, NTNU, Trondheim, 2020.

[2] L. Fang, D. P. Wyon, G. Clausen and P. O. Fanger, 'Impact of indoor air temperature and humidity in an office on perceived air quality, sbs symptoms and performance.', *Indoor air*, vol. 14, pp. 74–81, 2004.

[3] S.-B. Kwon, D. Park, Y. Cho and E.-Y. Park, 'Measurement of natural ventilation rate in seoul metropolitan subway cabin', *Indoor and Built Environment*, vol. 19, no. 3, pp. 366–374, 2010.

[4] M. Waters, T. Bloom, B. Grajewski and J. Deddens, 'Measurements of indoor air quality on commercial transport aircraft', *Proceedings of indoor air*, pp. 782–787, 2002.

[5] S. C. Lee, W.-M. Li and C.-H. Ao, 'Investigation of indoor air quality at residential homes in hong kong—case study', *Atmospheric Environment*, vol. 36, no. 2, pp. 225–237, 2002.

[6] Foobot. (2020). Foobot Website, [Online]. Available: `https://foobot.io/features/` (visited on 20/10/2020).

[7] Airthings. (2020). Airthings Website, [Online]. Available: `https://www.airthings.com/` (visited on 20/10/2020).

[8] U. Jaimini, T. Banerjee, W. Romine, K. Thirunarayan, A. Sheth and M. Kalra, 'Investigation of an indoor air quality sensor for asthma management in children', *IEEE Sensors Letters*, vol. 1, no. 2, pp. 1–4, 2017.

[9] A. Moreno-Rangel, T. Sharpe, F. Musau and G. McGill, 'Field evaluation of a low-cost indoor air quality monitor to quantify exposure to pollutants in residential environments', *Journal of Sensors and Sensor Systems*, vol. 7, pp. 373–388, 2018.

[10] I. Lee and K. Lee, 'The internet of things (iot): Applications, investments, and challenges for enterprises', *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[11] L. Morawska, P. K. Thai, X. Liu, A. Asumadu-Sakyi, G. Ayoko, A. Bartonova, A. Bedini, F. Chai, B. Christensen, M. Dunbabin, J. Gao, G. S. Hagler, R. Jayaratne, P. Kumar, A. K. Lau, P. K. Louie, M. Mazaheri, Z. Ning, N. Motta, B. Mullins, M. M. Rahman, Z. Ristovski, M. Shafiei, D. Tjondronegoro, D. Westerdahl and R. Williams, 'Applications of low-cost sensing technologies for air quality monitoring and exposure assessment: How far have they gone?', *Environment International*, vol. 116, pp. 286–299, 2018, ISSN: 0160-4120. DOI: https://doi.org/10.1016/j.envint.2018.04.018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0160412018302460.

[12] S. Abraham and X. Li, 'A cost-effective wireless sensor network system for indoor air quality monitoring applications.', in *FNC/MobiSPC*, 2014, pp. 165–171.

[13] A. Caron, N. Redon, F. Thevenet, B. Hanoune and P. Coddeville, 'Performances and limitations of electronic gas sensors to investigate an indoor air quality event', *Building and Environment*, vol. 107, pp. 19–28, 2016, ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2016.07.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360132316302530.

[14] T. Villa, F. Gonzalez, B. Miljievic, Z. Ristovski and L. Morawska, 'An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives', *Sensors*, vol. 16, no. 7, p. 1072, Jul. 2016, ISSN: 1424-8220. DOI: 10.3390/s16071072. [Online]. Available: http://dx.doi.org/10.3390/s16071072.

[15] Y. Yang, Z. Zheng, K. Bian, L. Song and Z. Han, 'Real-time profiling of fine-grained air quality index distribution using uav sensing', *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 186–198, 2018.

[16] K. Dimitriou, A. Paschalidou and P. Kassomenos, 'Assessing air quality with regards to its effect on human health in the european union through air quality indices', *Ecological Indicators*, vol. 27, pp. 108–115, 2013, ISSN: 1470-160X. DOI: https://doi.org/10.1016/j.ecolind.2012.11.023. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1470160X12004141.

[17] A. Koval and E. Irigoyen, 'Mobile wireless system for outdoor air quality monitoring', in *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16*, M. Graña, J. M. López-Guede, O. Etxaniz, Á. Herrero, H. Quintián and E. Corchado, Eds., Cham: Springer International Publishing, 2017, pp. 345–354, ISBN: 978-3-319-47364-2.

[18] T. Villa, F. Salimi, K. Morton, L. Morawska and F. Gonzalez, 'Development and validation of a uav based system for air pollution measurements', *Sensors*, vol. 16, no. 12, p. 2202, Dec. 2016, ISSN: 1424-8220. DOI: 10.3390/s16122202. [Online]. Available: http://dx.doi.org/10.3390/s16122202.

[19] D. Wu, D. I. Arkhipov, M. Kim, C. L. Talcott, A. C. Regan, J. A. McCann and N. Venkatasubramanian, 'Addsen: Adaptive data processing and dissemination for drone swarms in urban sensing', *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 183–198, 2017.

[20] S. Zhi, Y. Wei, Z. Cao and C. Hou, 'Intelligent controlling of indoor air quality based on remote monitoring platform by considering building environment', in *2017 4th International Conference on Systems and Informatics (ICSAI)*, 2017, pp. 627–631.

[21] R. Li, J. Liu, L. Zhang and Y. Hang, 'Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments', in *2014 DGON Inertial Sensors and Systems (ISS)*, 2014, pp. 1–15.

[22] H. Oh, D.-Y. Won, S.-S. Huh, D. H. Shim, M.-J. Tahk and A. Tsourdos, 'Indoor uav control using multi-camera visual feedback', *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 57–84, Jan. 2011, ISSN: 1573-0409. DOI: 10.1007/s10846-010-9506-8. [Online]. Available: https://doi.org/10.1007/s10846-010-9506-8.

[23] J. Tiemann, F. Schweikowski and C. Wietfeld, 'Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments', in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2015, pp. 1–7.

[24] B. Min, C. Cho, K. Choi and D. Kim, 'Development of a micro quad-rotor uav for monitoring an indoor environment', in *Advances in Robotics. FIRA 2009*, vol. 5744, 2009, pp. 262–271, ISBN: 978-3-642-03982-9. DOI: https://doi.org/10.1007/978-3-642-03983-6_30.

[25] P. P. Neumann, P. Hirschberger, Z. Baurzhan, C. Tiebe, M. Hofmann, D. Hüllmann and M. Bartholmai, 'Indoor air quality monitoring using flying nanobots: Design and experimental study', in *2019 IEEE International Symposium on Olfaction and Electronic Nose (ISOEN)*, 2019, pp. 1–3.

[26] F. Lied. (2020). Telemetri. i store norske leksikon, [Online]. Available: https://snl.no/telemetri (visited on 10/09/2020).

[27] Y. Zeng, R. Zhang and T. J. Lim, 'Wireless communications with unmanned aerial vehicles: Opportunities and challenges', *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[28] Bitcraze AB. (2020). Crazyflie 2.1, [Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1/ (visited on 10/09/2020).

[29] Bitcraze AB. (2014). Crazyflie 2.0 schematic, [Online]. Available: https://www.bitcraze.io/documentation/hardware/crazyflie_2_0/crazyflie_2.0_rev.c_schematics.pdf (visited on 10/09/2020).

[30] Nordic Semiconductor. (2020). Nrf51822, [Online]. Available: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF51822 (visited on 10/09/2020).

[31] J. Catsoulis, *Designing Embedded Hardware*. O'Reilly Media, Inc., 2005, 2nd Edition, ISBN: 978-0-596-00755-3.

[32] *An example schematic with one master (a microcontroller), three slave nodes (an ADC, a DAC, and a microcontroller), and pull-up resistors Rp*. [Online]. Available: https://en.wikipedia.org/wiki/I%C2%B2C#/media/File:I2C.svg (visited on 24/10/2020).

[33] C. M. L. Burnett, *SPI three slaves*. [Online]. Available: https://en.wikipedia.org/wiki/File:SPI_three_slaves.svg (visited on 05/12/2020).

[34] Microchip. (2003). In-Circuit Serial Programming Guide, [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/30277d.pdf (visited on 24/11/2020).

[35] R. Spekking, *SEG DVD 430 - Printed circuit board*. [Online]. Available: https://commons.wikimedia.org/wiki/File:SEG_DVD_430_-_Printed_circuit_board-4276.jpg (visited on 24/10/2020).

[36] N.-C. Lee, *Reflow soldering processes*. Elsevier, 2002.

[37] Arduino. (2020). What is arduino?, [Online]. Available: https://www.arduino.cc/en/guide/introduction (visited on 24/11/2020).

[38] (2020). Breakout Boards, [Online]. Available: https://www.adafruit.com/category/42 (visited on 24/10/2020).

[39] (2020). About KiCad, [Online]. Available: https://kicad-pcb.org/about/kicad/ (visited on 24/10/2020).

[40] B. Horan, *Practical Raspberry Pi*. Apress, 2013, pp. 167–168.

[41] J. D. Spengler, J. F. McCarthy and J. M. Samet, *Indoor air quality handbook*. McGraw Hill Professional, 2000.

[42] P. Addison, 'Respiratory effort from the photoplethysmogram', *Medical Engineering & Physics*, vol. 41, Jan. 2017. DOI: 10.1016/j.medengphy.2016.12.010.

[43] Z.-M. Zhang, S. Chen and Y.-Z. Liang, 'Baseline correction using adaptive iteratively reweighted penalized least squares', *Analyst*, vol. 135, no. 5, pp. 1138–1146, 2010.

[44] S. Kittelsen, *Development of environmental sensor for indoor air quality*, TTK4550 Specialization Project, NTNU, 2019.

[45] Arduino. (2020). Arduino homepage, [Online]. Available: https://www.arduino.cc/ (visited on 16/09/2020).

[46] AMS. (2019). CCS811 - Data sheet, [Online]. Available: https://www.sciosense.com/wp-content/uploads/2020/01/CCS811-Datasheet.pdf (visited on 19/10/2020).

[47] Sensirion. (2020). Datasheet SGP30, [Online]. Available: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9_Gas_Sensors/Datasheets/Sensirion_Gas_Sensors_SGP30_Datasheet.pdf (visited on 09/10/2020).

[48] Sensirion AG Switzerland. (2020). VOC Sensor SGP30 / SGPC3, [Online]. Available: https://www.sensirion.com/en/environmental-sensors/gas-sensors/sgp30/ (visited on 09/10/2020).

[49] Sensirion. (2020). Datasheet SGPC3, [Online]. Available: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9_Gas_Sensors/Datasheets/Sensirion_Gas_Sensors_SGPC3_Datasheet.pdf (visited on 09/10/2020).

[50] Bosch Sensortec. (2018). BME280 - Data sheet, [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (visited on 13/10/2020).

[51] Silicon Laboroties. (2016). Datasheet Si7021-A20, [Online]. Available: https://www.silabs.com/documents/public/data-sheets/Si7021-A20.pdf (visited on 13/10/2020).

[52] ROTRONIC AG. (2020). CP11 - Handheld Instrument for CO2, Humidity and Temperature, [Online]. Available: https://www.rotronic.com/en/cp11.html (visited on 01/11/2020).

[53] H. Haraldsen and B. Pedersen. (2019). Karbondioksid. i store norske leksikon, [Online]. Available: https://snl.no/karbondioksid (visited on 07/01/2021).

[54] Nordic Semiconductor. (2020). nRF24L01 Product Specification, [Online]. Available: https://www.mouser.com/datasheet/2/297/nRF24L01_Product_Specification_v2_0-9199.pdf (visited on 03/11/2020).

[55] C. Hallard. (2013). NRF24L01 real life range test, [Online]. Available: https://hallard.me/nrf24l01-real-life-range-test/ (visited on 04/11/2020).

[56] Dejan, HowToMechatronics.com. (2019). How To Use the NRF24L01 with Arduino - Complete Guide, [Online]. Available: https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/ (visited on 04/11/2020).

[57] LastMinuteEngineers.com. (2020). How nRF24L01+ Wireless Module Works & Interface with Arduino, [Online]. Available: https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/ (visited on 04/11/2020).

[58] Microchip. (2018). ATmega48P/V/88P/V/168P/V Data Sheet, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48P_88P_168P-DS40002065A.pdf (visited on 19/11/2020).

[59] Microchip. (2018). AVR Microcontroller Hardware Design Consierations, [Online]. Available: http://ww1.microchip.com/downloads/en/Appnotes/AN2519-AVR-Microcontroller-Hardware-Design-Considerations-00002519B.pdf (visited on 20/11/2020).

[60] Texas Instruments. (2014). TLV1117 Adjustable and Fixed Low-Dropout Voltage Regulator, [Online]. Available: https://www.ti.com/lit/gpn/tlv1117 (visited on 13/11/2020).

[61] Texas Instruments. (2015). LP2985 150-mA Low-noise Low-dropout Regulator With Shutdown, [Online]. Available: https://www.ti.com/lit/gpn/lp2985 (visited on 13/11/2020).

[62] Microchip. (2019). MIC5501/2/3/4 - 300 mA Single Output LDO in Small Packages, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/MIC5501-02-03-04-300mA-Single-Output-LDO-in-Small-Packages-DS20006006B.pdf (visited on 13/11/2020).

[63] Adafruit Industries. (2020). Adafruit SGP30 TVOC/eCO2 Gas Sensor, [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-sgp30-gas-tvoc-eco2-mox-sensor.pdf (visited on 30/11/2020).

[64] PCBWay. (2019). PCBWay website, [Online]. Available: https://www.pcbway.com/ (visited on 03/12/2020).

[65] Omega Verksted. (2020). Omega Verksted FAQ, [Online]. Available: https://confluence.omegav.no/display/OV/FAQ (visited on 04/12/2020).

[66] Systec Entwicklung & Fertigung. (2013). Reflow-Soldering Systems Overview, [Online]. Available: http://www.sef.de/wordpress_/wp-content/uploads/2013/06/SMD-Programm_EN.pdf (visited on 04/12/2020).

[67] G.-J. Lay. (2020). Gcc wiki - avr-gcc, [Online]. Available: https://gcc.gnu.org/wiki/avr-gcc (visited on 14/12/2020).

[68] Ladyada for Adafruit Industries. (2020). GitHub: Adafruit BME280 Library, [Online]. Available: https://github.com/adafruit/Adafruit_BME280_Library (visited on 07/12/2020).

[69] Limor Fried for Adafruit Industries. (2020). GitHub: Adafruit SGP30 Gas / Air Quality I2C sensor, [Online]. Available: https://github.com/adafruit/Adafruit_SGP30 (visited on 07/12/2020).

[70] Microsoft. (2020). Visual studio code - homepage, [Online]. Available: https://code.visualstudio.com/ (visited on 14/12/2020).

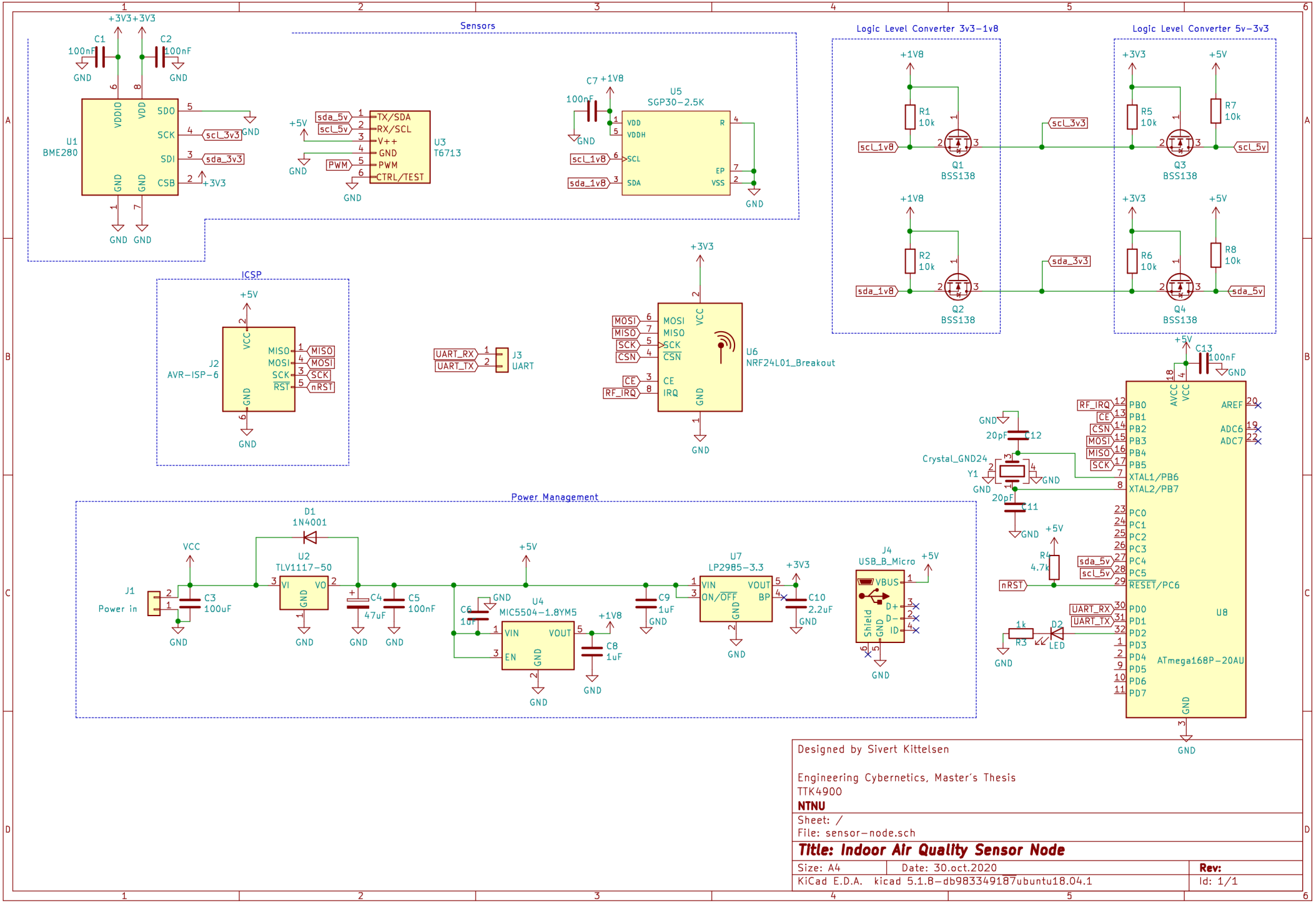
[71] (2020). Avr downloader/uploader - summary, [Online]. Available: http://savannah.nongnu.org/projects/avrdude (visited on 14/12/2020).

[72] T. Fischl. (2020). Usbasp - usb programmer for atmel avr controllers, [Online]. Available: https://www.fischl.de/usbasp/ (visited on 14/12/2020).

[73] M. Institutt. (2020). Nowcast, [Online]. Available: https://api.met.no/weatherapi/nowcast/2.0/documentation (visited on 04/01/2021).

Appendix A

Sensor Node Schematics

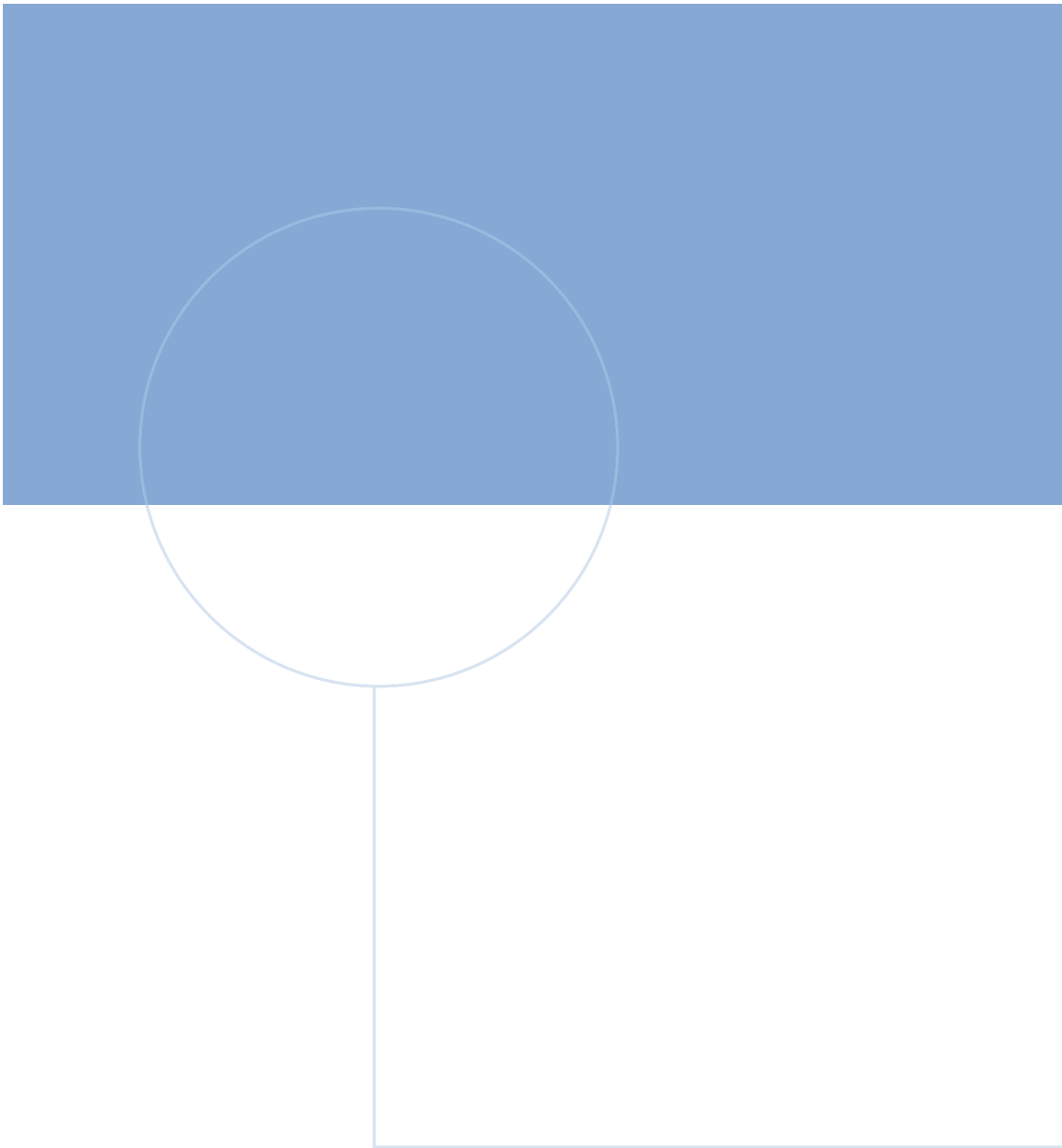

Sensors
Logic Level Converter 3v3–1v8
Logic Level Converter 5v–3v3
ICSP
Power Management
U1
BME280
U2
TLV1117–50
U3
T6713
U4
MIC5504–1.8YM5
U5
SGP30–2.5K
U6
NRF24L01_Breakout
U7
LP2985–3.3
U8
ATmega168P–20AU
J1
Power in
J2
AVR–ISP–6
J3
UART
J4
USB_B_Micro
Q1
BSS138
Q2
BSS138
Q3
BSS138
Q4
BSS138
D1
1N4001
D2
LED
Y1
Crystal_GND24
Designed by Sivert Kittelsen
Engineering Cybernetics, Master's Thesis
TTK4900
NTNU
Sheet: /
File: sensor–node.sch
Title: Indoor Air Quality Sensor Node
Size: A4
Date: 30.oct.2020
Rev:
KiCad E.D.A. kicad 5.1.8–db983349187ubuntu18.04.1
Id: 1/1

Appendix B

Files and Procedures

```
/
├── README.md
├── sensor-node-readings.ino
│   ├── void setup()
│   ├── void loop()
│   ├── ISR(TIMER2_COMPA_vect)
│   ├── void print_sensors()
│   ├── void store_baselines()
│   └── void read_eeprom_baselines(uint16_t * eco2, uint16_t * tvoc)
├── Timer.h
├── Timer.c
│   └── void Timer_init()
├── T6713Driver.h
├── T6713Driver.cpp
│   ├── bool T6713::begin()
│   ├── uint8_t T6713::readCO2value()
│   ├── uint16_t T6713::getCO2value()
│   ├── uint8_t T6713::reset()
│   ├── uint16_t T6713::getStatus()
│   ├── uint8_t T6713::readStatus()
│   ├── bool T6713::isInWarmupMode()
│   ├── uint16_t T6713::getAddress()
│   └── bool T6713::isMatchingAddress()
└── libraries
    ├── Adafruit_BME280.h
    └── Adafruit_BME280.cpp
        ├── bool Adafruit_BME280::begin(uint8_t addr, TwoWire *theWire)
        ├── float Adafruit_BME280::readTemperature(void)
        └── float Adafruit_BME280::readPressure(void)
```

```
    │   └── float Adafruit_BME280::readHumidity(void)
    ├── Adafruit_SGP30.h
    └── Adafruit_SGP30.cpp
        ├── boolean Adafruit_SGP30::begin(TwoWire *theWire)
        ├── boolean Adafruit_SGP30::IAQmeasure(void)
        └── boolean Adafruit_SGP30::getIAQBaseline(uint16_t *eco2_base, uint16_t *tvoc_base)
```



NTNU

Kunnskap for ei betre verd