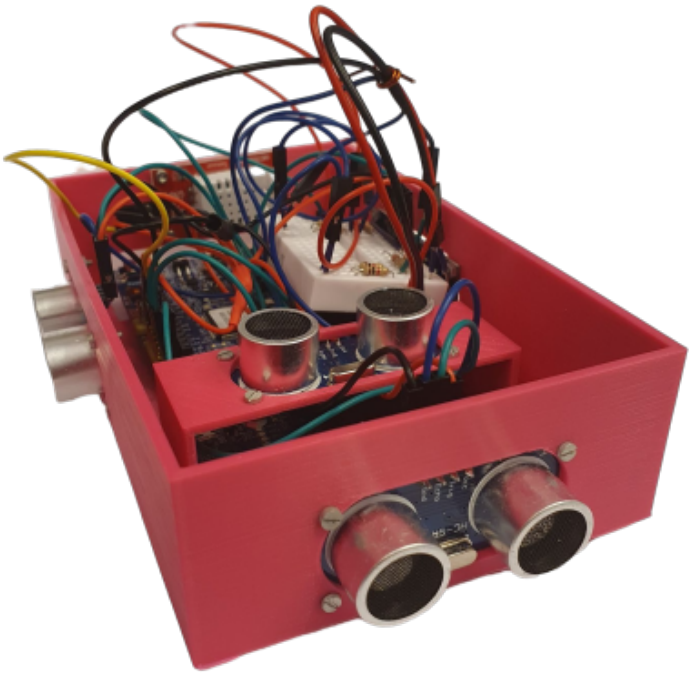
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Jonas Øygard Bjerke

Position Measurement for Quadcopter

Master's thesis in Cybernetics and Robotics
Supervisor: Tor Onshus
June 2022



NTNU
Norwegian University of
Science and Technology

Jonas Øygard Bjerke

Position Measurement for Quadcopter

Master's thesis in Cybernetics and Robotics
Supervisor: Tor Onshus
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This project concludes a master's degree in the 2-year master's program in Industrial Cybernetics at the Norwegian University of Science and Technology (NTNU). The last subject in this program is the 30 credit course TTK4900 - Engineering Cybernetics, Master's Thesis. The project lasted from January to June 2022.

The project is a continuation of an autonomous mapping system consisting of robots developed by earlier students. The idea is to include quadcopters as part of this mapping system. The work conducted in this thesis is mostly a continuation of a previous report [7]. This report's results had flaws and unanswered questions that required further examination. The available resources for this project were a workspace with a computer and software and hardware used in the previous report. The software consists of code developed for the ultrasonic sensor HC-SR04. The hardware consists of the ultrasonic sensor HC-SR04, nRF52840 SoC microcontroller, and nRF51 Dongle. Another significant codebase and reports written on related topics by students at NTNU were also available. As a contributor to this project, access to a Motion Capture Lab at NTNU room B333 was also available. The mechanical and electrical workshop at NTNU, in Electrical block D, is also open to students.

The beginning of the thesis was tough. With little to no knowledge of the used programming languages, a considerable amount of time was spent learning the foundation of the codebase while banging my head against the wall. However, as Dumbledore said: "Help will always be given at Hogwarts to those who deserve it.". The same applies to NTNU, and help is what I got. I would like to thank my supervisor Tor Onshus who has provided outstanding guidance when needed but, most importantly, shown a light-hearted point of view towards meaningless worries from my side. I would also like to thank the students who sit in the same room as me, working on similar projects. They have shown exceptional knowledge and saved me from hours of struggle. Lastly, a special thanks to the workshop employees for help with soldering and general guidance.

Jonas Bjerke

Trondheim, June 2022

Problem description

The problem at hand can be divided into two parts. The first half of the problem is to test whether ultrasonic sensors can estimate the position of a quadcopter. A single ultrasonic sensor's accuracy will first be tested. Afterward, several identical sensors will be connected to gather distances in several directions, providing a position estimate.

The second half of the problem is software modification. Previous students' existing software on the same project has to fit the quadcopter's hardware. This object means making the code able to communicate and send data to an external server.

Hence, the problem descriptions based on the above are as follows

- Investigate the use of ultrasonic sensors to provide an accurate position estimate for a hovering quadcopter in a three-dimensional space.
- Look into and modify existing software, making it suitable for collecting and transmitting data to a server regarding new hardware.

Summary and conclusion

This thesis investigates the possibility of using ultrasonic sensors to measure and communicate the position of a hovering quadcopter to an external server. The ultrasonic sensors had previously been tested for this specific intention but under imprecise conditions. For this thesis, the quality and accuracy of the testing were improved drastically. The specifications of the ultrasonic sensor were also compared with other types of distance sensors.

Getting the communication to work required altering a larger piece of software to fit the specific hardware used for this project. One such alteration was creating and implementing the software used to operate the ultrasonic sensors.

The sensor's ability to measure distance was tested by wiring them up to a microcontroller. A casing was designed, 3D-printed, and used to hold this wiring in place and make it easier to maneuver the sensors during the tests.

A test was conducted to see if the sensors could communicate and map measured distances. Another test investigated the sensor's ability to estimate the position of a quadcopter. This estimate was done using three sensors pointing in three different directions in a three-dimensional space. A motion capture system was used to analyze the results and get a picture of how accurate the estimated position provided by the sensors was.

The time spent increasing the quality of the test regarding the ultrasonic sensor was valuable. The uncertainties from the previous report were gone, and interpreting the results was uncomplicated.

Getting the communication working was challenging due to a lack of knowledge in the programming language. Specific arguable but necessary changes were made to get the code working. In the end, the server could receive measured distances from the microcontroller and create the virtual map. The results from the three-dimensional test demonstrated that by using three or more ultrasonic sensors, the position could be estimated with an error of $\pm$ 2 cm. This error is significant enough to be displayed in the virtually created map. The ultrasonic sensor is also disadvantaged due to its limited detection range. If the space to be mapped is wider than approximately eight meters in either direction, the HC-SR04 Ultrasonic Sensor will not be able to measure the needed distances.

A different type of sensor is required to eliminate the error and make a suitable positioning estimate for larger spaces. The best option would be to use a light-based distance sensor due to its faster measuring frequency and longer range.

Sammendrag og konklusjon

Denne oppgaven utforsker muligheten for å benytte ultralydsensorer til å måle og kommunisere posisjonen til et flyvende quadcopter til en ekstern server. Ultralydsensorene har tidligere blitt testet for denne spesifikke hensikten, men i upresise omgivelser. I denne oppgaven har kvaliteten og nøyaktigheten av testene blitt drastisk forbedret. Spesifikasjonene til ultralydsensorene er også sammenlignet med andre type avstandssensorer.

Å få kommunikasjonen til å fungere krevde endringer i en større kodebase for å passe til komponentene brukt i dette prosjektet. En av disse endringene var å lage og implementere koden som brukes for å operere ultralydsensorene.

Sensorens evne til å måle avstander ble testet ved å koble dem opp til en mikrokontroller. Et etui ble designet, 3D-printet og brukt for å holde utstyret på plass og samtidig gjøre det enklere å bevege sensorene rundt når de skulle testes.

En test ble gjennomført for å sjekke om sensorene kunne kommunisere og kartlegge måle avstander. En annen test utforsket sensorenes mulighet til å estimere posisjonen til et quadcopter. Denne estimeringen ble gjennomført ved bruk av tre sensorer pekende i tre forskjellige retninger i et tre-dimensjonalt rom. Et bevegelsesdeteksjons-system ble brukt for å analysere resultatene og danne et bilde av nøyaktigheten til den estimerte posisjonen.

Tiden som ble brukt for å forbedre kvaliteten av testene for ultralydsensorene var verdifull. Usikkerhetsmomentene fra den forrige rapporten ble borte og tolking av resultater ble mindre komplisert.

Å få kommunikasjonen til å fungere var utfordrende grunnet mangel på kunnskap innenfor de benyttede programmeringsspråkene. Visse argumenterbare, men nødvendige endringer ble utført for å få koden til å fungere. Resultatet ble at serveren mottok målte avstander fra mikrokontrolleren og konstruerte virtuelle kart. Resultatet fra den tre-dimensjonale testen demonstrerte at bruk av tre eller flere ultralydsensorer kan estimere posisjonen med en feilmargin på $\pm$ 2 cm. Denne feilen er stor nok til å vises i det virtuelle kartet. Ultralydsensorene har også en svakhet grunnet begrenset avstandsmåling. Hvis området som skal kartlegges er større en omtrent åtte meter i en retning, vil HC-SR04 ultralydsensoren ikke klare å måle avstanden.

En annen type sensorer kreves for å fjerne feilen i målingene og gjøre posisjonsestimeringen mulig for større avstander. Det beste alternativet ville vært å benytte en lys-basert avstandsmåler grunnet bedre målefrekvens og muligheten for å måle lengre avstander.

Table of Contents

Preface **I**

Problem description **II**

Summary and conclusion **III**

Sammendrag og konklusjon **IV**

Abbreviations **VIII**

1 Introduction **1**

1.1 The Robot Project 1

2 System Description **2**

2.1 Hardware 2

2.1.1 nRF52840 SoC 2

2.1.2 nRF51 Dongle 2

2.1.3 HC-SR04 2

2.1.4 AM2302 DHT22 4

2.1.5 Motion Capture System 4

2.1.6 Quadcopter 5

2.2 Software 6

2.2.1 Robot application 6

2.2.2 Server application 6

2.2.3 Segger Embedded Studio 6

2.2.4 Visual Studio Code 7

2.2.5 J-Link RTT Viewer 7

2.2.6 nRF connect for Desktop 7

2.2.7 MatLab 7

2.2.8 Inventor 7

2.2.9 Prusa Slicer 8

3 Theory and background **9**

3.1 Distance sensors 9

3.1.1 ToF principle 9

3.1.2 Required and satisfactory conditions 10

3.1.3 Comparison 11

4 Improving the accuracy of the test environment **13**

4.1 Data processing and System monitoring 13

4.2 Improving the time aspect 14

4.3 The speed of sound 16

4.4 Correcting OptiTrack´s view 17

5 Altering software **19**

5.1 main.c 19

5.2 HC_SR04.c and HC_SR04.h 20

5.3 SensorTowerTask.c 20

5.4 robot_config.h 20

5.5 Javaserver 20

6 Case creation and assembly **22**

6.1 Design and weight 22

6.2 Mounting hardware and cables 24

7 Preparing the test environment **25**

7.1 One-dimensional test setup 25

7.2 Three-dimensional test setup 27

8 Results **29**

8.1 Results from the one-dimensional test 29
8.2 Server testing 30
8.3 Results from the three-dimensional test 31
8.4 Stationary measurements 34

9 Discussion **35**

10 Further Work **37**

10.1 New sensors 37
10.2 Software 37
10.3 Casing 38
10.4 IMU 38

List of Figures **39**

Bibliography **40**

A Appendix **43**

A.1 Number of measurements 43
A.2 Smallest noticeable change 43
A.3 Speed of sound 43
A.4 Voltage divider 44
A.5 Battery 45

Abbreviations

SLAM = Simultaneous Localization and Mapping

SoC = System on Chip

BLE = Bluetooth Low Energy

DK = Development Kit

ANT = Advanced and Adaptive Network Technology

DC = Direct Current

RH = Relative Humidity

FPS = Frames Per Seconds

RTOS = Real-Time Operating System

IDE = Integrated Development Environment

RTT = Real-Time Transfer

GUI = Graphical User Interface

CAD = Computer-Aided Design

LED = Light Emitting Diode

IMU = Inertial Measurement Unit

1 Introduction

1.1 The Robot Project

This thesis builds on a project called The SLAM Robot project. The project has been ongoing since 2004 and is supervised by professor Tor Onshus at NTNU. Multiple students have been writing their specialization projects and master thesis over the years, upgrading and modifying the existing system. The general idea of this project is to use small autonomous robots to remotely map an area using wireless communication to and from a server. The server is responsible for constructing a virtual map of the robots' detected area. It does so by processing data from the robots and commanding them where to go next.

A recent upgrade in progress is to recreate the server using C++ for reasons regarding clarity and communication between the server and robots. As of the writing of this thesis, the project has six individual wheel-based robots communicating with the java-server.

Another recently intended upgrade is to include quadcopters in addition to the six robots. A quadcopter is a type of drone that flies using four rotors [13]. The purpose of the quadcopter would be to assist the mapping process from the air by equipping it with a camera. The pictures and position of the quadcopter are then sent to the server to support the mapping process.

2 System Description

2.1 Hardware

The hardware used in this project is similar to previous work done. As this project is slightly different, there are also new components. These are the used components throughout the project:
- nRF52840 SoC
- nRF51 Dongle
- HC-SR04
- AM2302 DHT22
- Motion Capture System
- Quadcopter

2.1.1 nRF52840 SoC

The nRF52840 SoC is a microcontroller created by Nordic Semiconductor and is the most advanced member in the series. The robots are currently using this microcontroller. The same microcontroller will be used in this project to make it easier to work with the system on the quadcopter. By doing so, the same software can be used only with slight modifications. The microcontroller has one MB of Flash memory and 256 kb of RAM, suitable for most advanced and complex applications. It has many different wireless communication possibilities, such as BLE, Thread, and Zigbee. The SoC can be programmed through the nRF52840 DK. Among the many peripherals and interfaces, the full-speed USB device allows for easy data transfer and power supply [30].

2.1.2 nRF51 Dongle

The nRF51 Dongle, also created by Nordic Semiconductors, is a USB development dongle for various communication styles. It supports BLE, ANT, and 2.4 GHz proprietary applications. The Dongle is connected to a USB slot in the host machine, where it is programmed using an application by Nordic Semiconductors called "nRF Connect" addressed further in Section 2.2.6. This application allows it to be used as an intermediate communication device between applications on the machine and the nRF52840 SoC. In this project, the application is a java server addressed further in Section 2.2.2, and the communication is BLE. The microcontroller and server can effectively transmit data to and from each other through this communication type.

2.1.3 HC-SR04

The distance sensor used in this project is an ultrasonic sensor named HC-SR04. This sensor uses sound as its reference medium to measure distances. It can measure distances from 2 cm to

400 cm, with a precision of ± 0.3 cm. There are two types of angles being of importance to the ultrasonic sensor. These are the effective angle, which is at 15°, and the measuring angle, which is at 30°. The effective angle determines how big of an angle the surface in front of the sensor can have relative to the sensor. An angle of 15° or more can lead to the sound waves being reflected in a different direction than where they were emitted. The measuring angle means that objects within a sector of 30° also might reflect the sound. Figure 1 shows the HC-SR04 sensor.

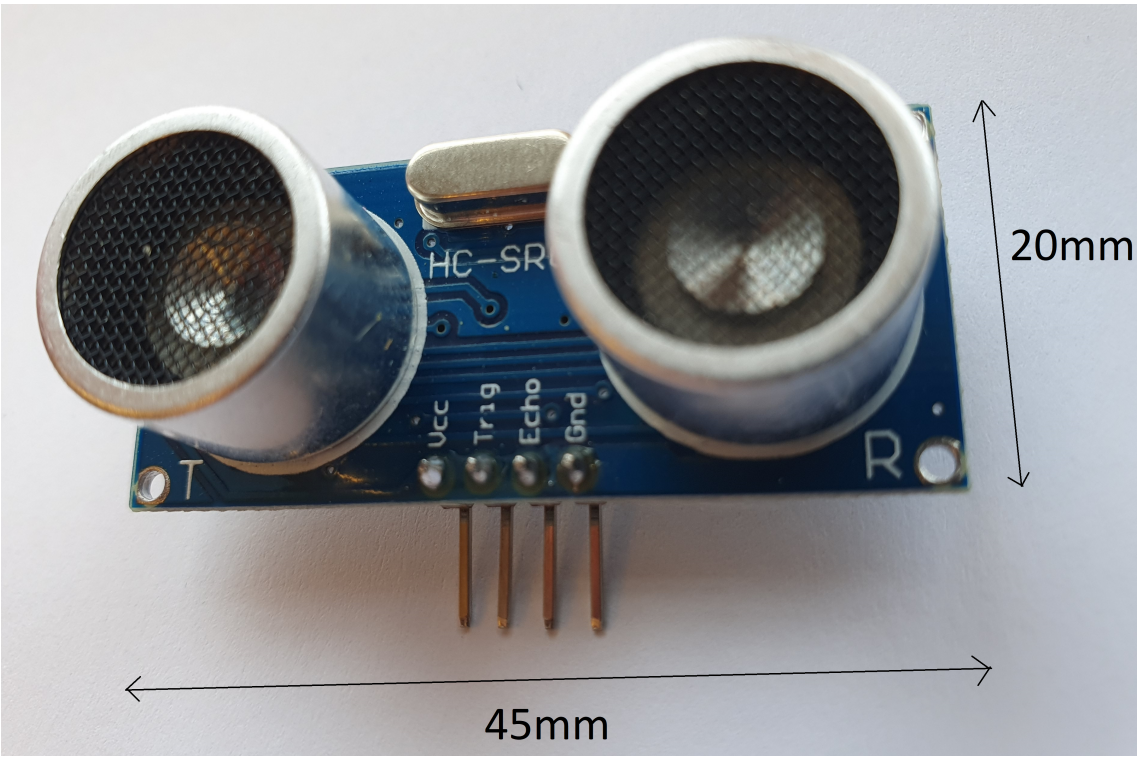


Figure 1: HC-SR04 with dimensions, its depth is 15 mm

The sensor is operated through 4 pins, called VCC, TRIG, ECHO, and GND. VCC and GND are the voltage supply and ground. Standard USB ports supply the required voltage of +5 V DC. The TRIG and ECHO pins are connected to output and input pins on a microcontroller. The sensor works the following way:

The TRIG pin activates for a duration of 12 μs. In this short period, the sensor sends out an eight-cycle sonic burst with a frequency of 40 kHz. The ECHO pin goes high afterward, meaning that current flows through it. The microcontroller registers this current and begins a timer. The ECHO pin goes low when the eight cycle burst of 40 kHz returns to the sensor after being reflected off the target. This change in the pin is registered by the microcontroller, which stops the timer. The time can be used together with the speed of sound to calculate how far away the target was [20].

2.1.4 AM2302 DHT22

The AM2302 DHT22 is a digital RH and temperature sensor. It has an operating range of 0 % - 100 % RH, and -40 °C to +80 °C. The accuracy of these measurements lies at ± 2 % RH and ± 0.5 °C [18]. The sensor can be seen in Figure 2.

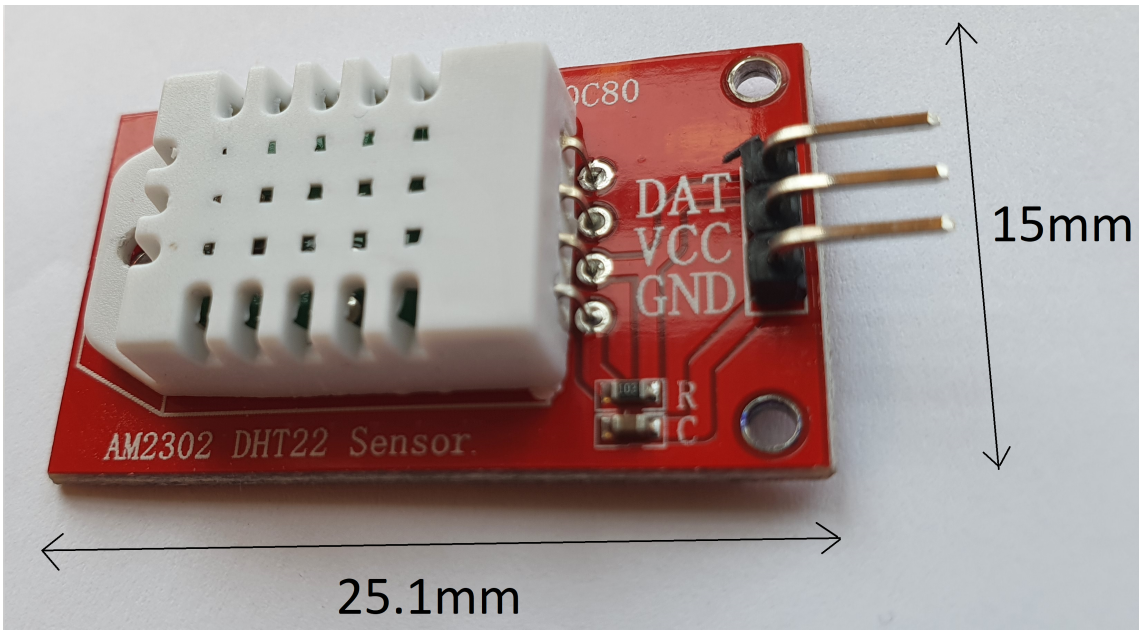


Figure 2: AM2302 DHT22 with dimensions, its depth is 7.7 mm

The three pins on the sensor are used to power and read data from the sensor. They can be connected to a microcontroller, from where it is controlled. GND is the ground, while DAT is the pin where the data is read. The VCC requires 3.3 V - 5.5 V DC to function.

2.1.5 Motion Capture System

A motion capture system is an advanced system that records any desired object's movement digitally. It was used to compare the distance sensor data with actual absolute distances. The motion capture system used is called OptiTrack, which is located at NTNU, Gløshaugen, in room B333. The system has 16 cameras that track and record small reflective spheres easily mounted on desired objects. Several spheres can be tracked relative to each other to get the distance between objects. Collection of the data through the cameras is done using a suited software called Motive [23]. OptiTrack can capture movement at a rate of 120 FPS [37], making it a suitable tool for comparing the accuracy of the sensors.

2.1.6 Quadcopter

As mentioned in the introduction, Section 1.1, a quadcopter is considered to be introduced as part of the mapping system. A quadcopter type had not yet been decided due to the new concept. During the project, all four sensors were to be tested. This test required a casing to hold them in place. While creating this casing, it was practical to create it such that it could be attached to a quadcopter for potential further testing. The project supervisor already owned a Mavic MINI drone seen in Figure 3.

Figure 3: Mavic MINI Drone, by TheBetterDay [36]
Dimensions: Length 159mm, Width 202mm, Height 55mm

For simplicity reasons, this drone was used as a candidate. It also has an in-built camera that was considered to be used. It is also relatively small, which is advantageous for mapping narrow and confined spaces.

2.2 Software

A variety of different software and programs was used throughout this project. The next sections provide a short description of the following:
- Robot application
- Server application
- Segger Embedded Studio
- Visual Studio Code
- J-Link RTT Viewer
- nRF Connect for Desktop
- MatLab
- Inventor
- Prusa Slicer

2.2.1 Robot application

The robot application is the software that controls the robot through the microcontroller. The application has been written in C over many years by different students. The primary function of the code is to communicate with the Dongle and control the hardware on the robot. Parts of the application are based on FreeRTOS, which is designed to be run on a microcontroller. FreeRTOS benefits the system by providing the microcontroller with better resources such as real-time scheduling and inter-task communication [15].

2.2.2 Server application

The server application is also written by previous students working on the project. It is written in Java. A program called Apache Netbeans IDE is used to run the server. The server's function is to create a two-dimensional map from the data received by the robots. It also controls the robots as to where to go next. The server is specifically written to fit the robots on the ground. Due to this, changes had to be made as the quadcopter hardware differs. These changes are addressed in Section 5.

2.2.3 Segger Embedded Studio

SES is short for Segger Embedded Studio. It is an IDE specifically meant for embedded systems [26]. The embedded system in this project is the nRF52840 SoC. SES enabled easy management of projects and was used for this project to manage the Robot Application. The IDE also has an effective debug tool used frequently to locate and solve issues in the code. The communication between Segger and the nRF52840 was done via USB.

2.2.4 Visual Studio Code

This source-code editor was frequently used together with Segger while debugging. It has a quick search and navigation system, which comes in handy when navigating source and header files [12]. It was also used to locate specific java code snippets in the Server Application.

2.2.5 J-Link RTT Viewer

As the microcontroller is connected to the host computer through USB and uploaded with software, the J-Link RTT viewer is used on the host computer to monitor and identify the desired data. It is the main GUI application [27] provided by Segger, used for debugging. In addition to being frequently used for debugging purposes, it has an integrated recording function.

2.2.6 nRF connect for Desktop

nRF Connect for Desktop is a tool by Nordic Semiconductor that assists the development of nRF devices [29]. It contains a variety of apps, and the one used in this project is called Programmer. This app is a tool for flash programming nRF SoCs and has been used frequently in this project for debugging the nRF51 Dongle and nRF52 microcontroller. It allows the user to easily upload new content and identify what is already on the device.

2.2.7 MatLab

MatLab, created by the MathWorks corporation, is a robust programming and numeric computing platform that is accessed through NTNU's database of educational purpose software [19]. It has multiple purposes, whereas it has been used to process and visualize data graphically for this project. Using MatLab is due to its familiarity with other courses at NTNU and its ability to handle large amounts of data effectively.

2.2.8 Inventor

Inventor is a CAD software from the corporation Autodesk that provides the opportunity for complex and detailed 3D design, simulation, visualization, and documentation [4]. It was used to design a case to embody the different hardware described in the previous section. Similarly to MatLab, Inventor is accessed through NTNU's database of educational purpose software.

2.2.9 Prusa Slicer

Prusa Slicer is a slicer software [34]. This kind of software converts models designed on the computer into a 3D-print-friendly format. It allows the user to specify details for the print and get details such as how much the finished product will weigh and how long it will take to print. In this project, it was used to convert the created Inventor files into this 3D-print-friendly format.

3 Theory and background

3.1 Distance sensors

Having distance sensors mounted to the quadcopter can be used to locate its position. Using a GPS is not an option because the location of the quadcopter might be pinpointed indoors or in places where the GPS cannot receive a signal. The robots in this project use IR sensors to measure the distance to their surroundings effectively. These robots are relatively small and compact, making the IR-sensors short-range optimal. However, the quadcopter is more extensive, thus making the respective IR sensors range insufficient without risking flying too close to the surroundings.
Section 2.1 addresses the ultrasonic distance sensor HC-SR04. This particular sensor was used in the predecessor of this project [7]. Tests on this sensor concluded that it could measure distance no more than a few centimeters off. A few modifications to the test environment were needed before verifying the results from this previous report.

However, even though the HC-SR04 distance sensor works, it does not mean that other more compatible distance sensors should not be considered or used. This section describes distance sensors and compares them with each other and their compatibility for this project.

3.1.1 ToF principle

ToF is short for time-of-flight. It is the basic principle used in most distance sensors. Sensors based on ToF typically have an emitter and a receiver. The emitter sends out a reference medium, either sound or light. This medium bounces off the object or surface in front of it and returns to the receiver. The time it took for the medium to travel back and forth indicates the distance.

The type of reference medium used by the ToF sensor impacts its properties. Light or sound affects properties such as measuring frequency, maximum distance, and other environmental conditions.

Sound

Sensors that use sound as a reference medium are called ultrasonic sensors. They work by emitting the sound, which spreads outwards from the sensor. When the sound hits a solid object, it bounces back towards the receiver on the sensor, which identifies the sound. The emitted sound often has a specific frequency, making the receiver distinguish between random noise and the emitted signal. Calculating the distance is then done using the speed of sound. The HC-SR04 sensor used in the previous project and further in this project is an ultrasonic sensor. The measuring distance of ultrasonic distance sensors varies. They can measure distances down to a few centimeters up to 10 meters [9] or more. The common drawback is the relatively low measuring frequency which comes from the speed of sound limit [32].

Light

Light as a reference medium functions similarly to sound regarding the ToF principle. The sensor has an emitter and a receiver where light exits and enters. The type of light used varies depending on the sensor.

Sensors that use infrared light, which is in the wavelength above the visible spectrum [21], is called IR distance sensors. The robots currently use these sensors in the SLAM Robot Project. IR stands for infrared radiation, and IR sensors in general work by detecting heat radiation from the surroundings [17]. They are widely used in motion detectors to turn on lights or alarm systems. The IR sensors meant for measuring distance emit infrared light often from LEDs instead of detecting it from their surroundings. This light is distinguishable from other sources of light. By measuring the time it took for the infrared light to travel back and forth, the distance can be calculated [5].

Another typical light source used for measuring distance is a laser. The laser is a powerful beam of visible, ultraviolet, or infrared light [24]. If the laser is made up of ultraviolet light, it can measure longer distances than a sensor using infrared light [35]. This difference is due to ultraviolet light having a higher wavelength than infrared light. These sensors are called LIDAR, short for Light Detection and Ranging. A LIDAR using IR light can still measure longer distances than a standard IR sensor measured above due to the light being concentrated into a laser.

The range of light-based distance sensors typically has a more extended range than sound-based distance sensors. A laser will travel longer than infrared light generated by an LED in terms of light. Due to light traveling faster than sound, the measuring frequency of light-based distance sensors is faster than sound-based.

Its fast update rate is an advantage for capturing fast-moving details in front of it. The downside of LIDAR is the higher price relative to most distance sensors, its current draw, and the laser's harmfulness to the naked eye [33].

3.1.2 Required and satisfactory conditions

Sensing range

Choosing a suitable distance sensor depends on the environment. One such environmental condition is the required sensing range. Mounting the distance sensor at the center of the quadcopter makes measuring close distances unnecessary. Using the Mavic MINI drone as a reference mentioned in Section 2.1.6, the distance from the center of the drone to the rotors is approximately 10 cm.

The required furthest range is less obvious to pinpoint, as this depends upon various factors such as how vast the space is and if the quadcopter is to measure several sides simultaneously. There is no harm in having a sensor that can measure long distances. However, these sensors often have increased minimal measuring distance and are more expensive. Therefore, the sensor's maximum

should be as long as possible while maintaining a minimum of 10 cm.

Accuracy

The sensors should be able to measure objects in their line of sight accurately. This demand means providing correct measurements in terms of distance and avoiding measuring distance to the wrong objects. This accuracy affects the quality of the created map and the safety of the quadcopter and its equipment.

Measuring frequency

Another environmental condition is how often the sensing should occur. The sensors are attached to the quadcopter, maneuvering through the maze. The mapping is not meant to be a race. The quadcopter travels smoothly through the maze with directions sent from the server while taking pictures. This behavior means that it is not required to have a fast update rate but still fast enough to capture the quadcopter's movement.

3.1.3 Comparison

Following is a table comparing the above required and satisfactory conditions with each of the discussed sensor types. Further details about the table follow.

Sensor / **Condition**	Ultrasonic	IR distance sensor	LIDAR
Sensing range	✓	✗	✓
Accuracy	✗	✓	✓
Update rate	✓	✓	✓

Table 1: A table displaying if each sensor fulfills the respective condition

Considering Table 1 from a visual standpoint, it would seem that using a LIDAR would be the best choice for the task at hand. The importance of the three conditions is not equally weighted. The sensing range is perhaps the most important. Thus using an IR distance sensor can be excluded as a candidate for this project. It is suitable for short ranges, but this system requires a more extended range. It does exist Long Range IR distance sensors such as the Sharp GP2Y0A710YK0F [2]. This distance sensor can measure 100 - 550 cm, but this exceeds the desired minimum distance of 10 cm discussed in the previous section. Even though the Ultrasonic sensor has been marked with ✓ its range is still questionable. Light can travel further than sound because sound has to travel through a medium, and light does not. This medium requires energy to be moved, which eventually dies out. A bigger sensor could provide a longer range, but this would also increase the required power, weight, and size. Hence, if the required space to be mapped is extensive, using light might be the only option.

The ultrasonic sensor was marked with ✗ on accuracy. The reason is due to the reference medium being sound. As sound travels outwards in a cone, there is a chance it will observe objects slightly to the side of where it is pointing, as mentioned in Section 2.1.3. The further the sensor is measuring, the greater this observable area becomes. This weakness could cause problems. Light as a reference medium would, on the other hand, not experience the same problem as it travels in more or less a straight line.

Even though all sensor types have been marked with ✓ for the update rate condition, light still has a significantly faster update rate than sound. For this project, they should all be satisfactory.

The use of light versus sound as a reference medium also has more exciting properties. Both have their advantages and drawbacks in different situations. Sound is at a disadvantage where light can detect soft surfaces, edges, and angled surfaces effectively. Light is at a disadvantage where sound can measure distances through any medium, such as smoke, vapor, and dust [8]. Weighing these effects against each other would require a more detailed description of the environment.

The reason that the ultrasonic sensor was chosen in the first place might seem odd as the LIDAR theoretically turns out to be a better fit for this project and an obvious choice. A comparison like the one above was not carried out in the previous report where the ultrasonic sensor was chosen. The choice was primarily based on its highly reasonable price and availability at NTNU through Omega Verksted. Depending on the outcome of this report, there might be an idea to switch to another sensor. This shift will be briefly discussed in Section 10.

4 Improving the accuracy of the test environment

The HC-SR04 distance sensor was tested in the previous report on the topic mentioned in the Preface. Figure 4 shows one of these test results.

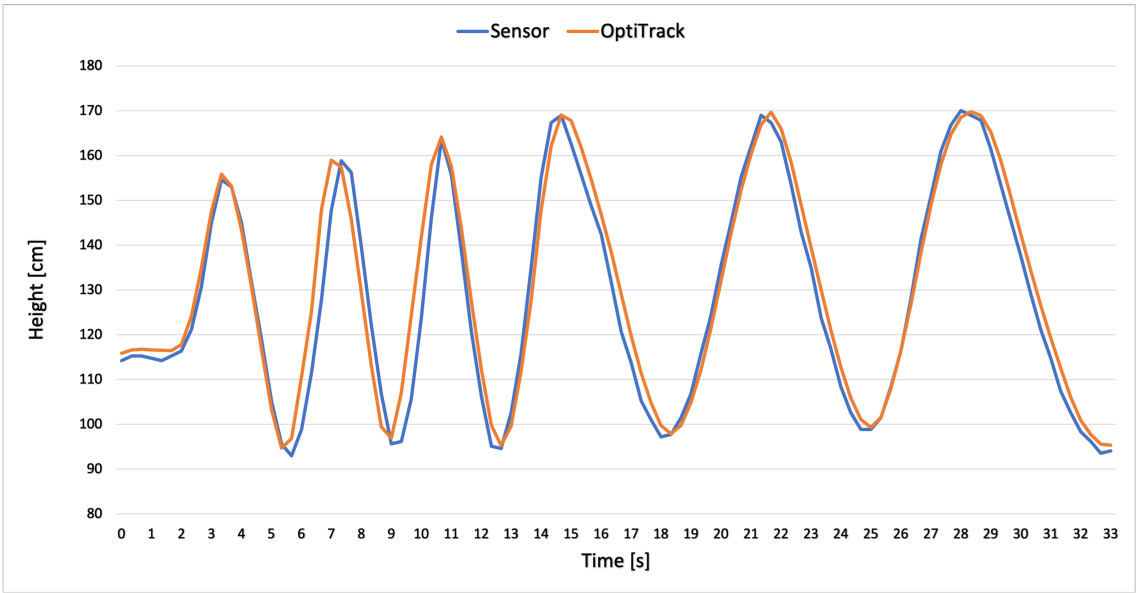


Figure 4: A graph comparing the sensors measurements with OptiTrack taken from the previous report [7]

Figure 4 compares the measurements done by the sensor with OptiTrack. These results are not completely accurate, as the test environment and recording have flaws and weaknesses. The two graphs look relatively close, but to get the exact deviation between them, one must study the graph more thoroughly.

This section provides a more detailed explanation of how the sensor's accuracy, quality of the testing, and data processing were improved. The first edition was to include a graph showing the deviation between the measurements done by the sensor and OptiTrack. This improvement made it easier to analyze the similarity which can be seen later in the results. Similar additions and improvements are explained below.

4.1 Data processing and System monitoring

The recorded distance data from the sensor and OptiTrack was previously processed using Microsoft Excel. However, as the project advanced and the data sets increased, Excel was insufficient as the processing speed drastically decreased. By processing further data in MatLab instead, the large data sets were no longer an issue.

Another shortcoming in the tests from the previous report was the sampling rate at which the debugger in Segger could collect data. The distance sensor recorded and printed approximately three samples every second to the console when using the debugger. Theoretically, if an ultrasonic sensor were placed one meter from a wall, the speed of sound would make it back and forth 170 times per second, A.1. The number of actual measurements provided by the sensor is lower due to delays in the hardware and software. However, these delays are not enough to reduce the number of measurements per second to a staggering three.

The solution to the problem was to modify the testing environment. Instead of running the program through the debugger in Segger, the software was downloaded directly onto the nRF52840 microcontroller. This adjustment allowed the software to run as it would in a real scenario, independent of the computer, which only provided the power to run it. Another logging system was used to monitor the data, namely the J-Link RTT Viewer. The result was significantly better as the system could collect over 100 samples every second. However, this many measurements leads to a quantity over quality attitude for the sensor. Many of these 100 measurements are incorrect because sound takes some time to die out. In other words, sound from one measurement can interfere with the next. To avoid this issue, the optimal number of measurements per second is around 16 [14]. A delay of 60 milliseconds between each measurement was included to reduce the number of measurements to this optimal frequency.

4.2 Improving the time aspect

Time plays a crucial role in obtaining accurate measurements. By measuring the time from when the sound left the sensor to when it returned, one can use this time to calculate the distance using the speed of sound. As sound moves fast, this time interval needs to be precise. Fortunately, the microcontroller can measure time accurately. From the previous report [7], time was measured with a precision of 31.25 microseconds (μs), meaning that the slightest noticeable change in the distance the microcontroller was able to detect was 0.5 cm A.2. As mentioned in Section 2.1.3, the HC-SR04 sensor has a precision of 0.3 cm. Obtaining this precision will take full advantage of the sensor's functionality. The precision was made ten times as precise by altering the interval in the code at which the timer counts. This alteration improved the precision from 31.25 μs, to 3.125 μs. The slightest possible change in distance was, as a consequence, reduced with a factor of ten, down to approximately 0.05 cm. This sensitivity improvement is illustrated in Figure 5.

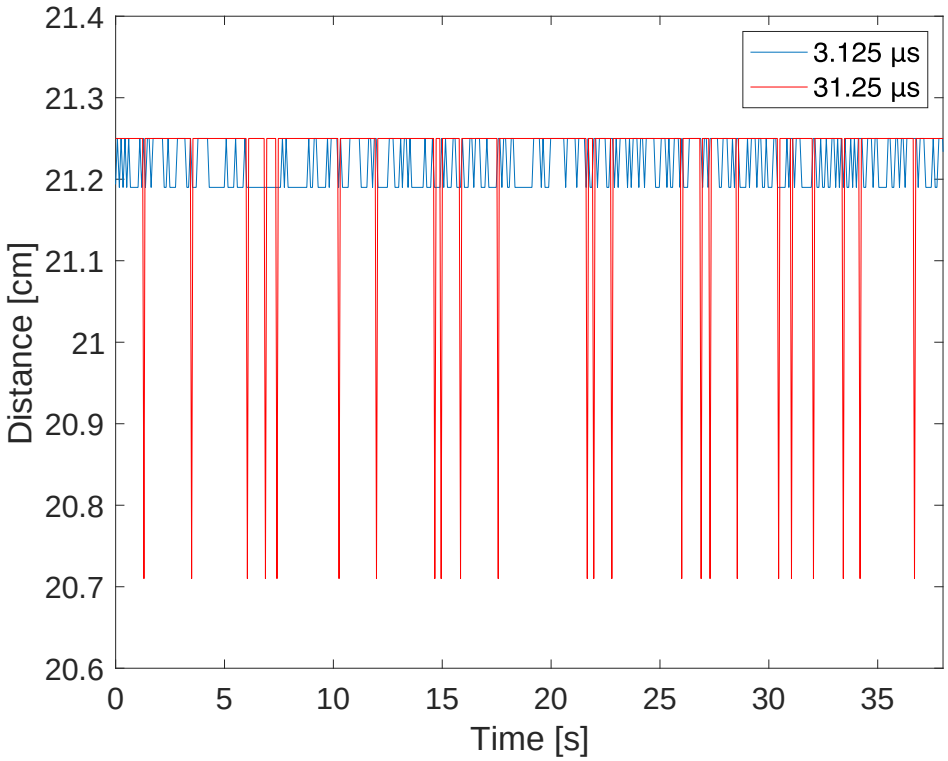


Figure 5: The improved timer illustrated

Figure 5 shows how the sensor operated in the same static conditions over 40 seconds, the only difference being the precision of the timer. The red graph illustrates the measured distance when the timer precision was 31.25 μs. The slightest noticeable change here is, as explained, 0.5 cm. The more accurate timer of 3.125 μs is shown in blue, where the slightest noticeable change is 0.05 cm. Using a more accurate timer provides a more exact change in distance. The microcontroller does provide the opportunity for even greater precision down to 62.5 nanoseconds [31], which would make the precision 50 times better than 0.05 cm. However, implementing this caused unknown errors and is unnecessary as the improved precision of 0.05 cm is highly precise relative to 0.3 cm and therefore sufficient for obtaining accurate measurements.

Another improved aspect when it comes to time was that the tests from the previous report [7] could not accurately compare data from the sensor with data from OptiTrack. There was no exact way of knowing when the measurements from the sensor had taken place. This shortage led to a poor comparison, where the two sets of data were compared at only approximately equal times. Solving this issue required a counter in the microcontroller to extract the corresponding time for each measurement. However, this counter proved to be off by 0.5 seconds for every 60 seconds passed, which accumulated throughout the tests. The result was a continuously increasing deviation for the measurements from the distance sensor and OptiTrack. Correcting this error was done in MatLab after the data was sampled.

4.3 The speed of sound

As mentioned in the section above, distance is calculated using the measured time together with the speed of sound. This formula is seen in Equation 1.

$$\text{Distance} = \text{Time} \cdot \text{Speed of sound} \tag{1}$$

In the previous report, [7], the speed of sound was set to be a constant of 340 m/s. However, the speed of sound is a changing variable depending on different factors, such as temperature and density of air. These factors, along with others, can be combined into a general equation for the speed of sound in gasses, shown in Equation 2, A.3.

$$\text{Speed of sound} = \sqrt{\frac{Cp}{Cv}\frac{R}{M}T} \tag{2}$$

Cp is the specific heat at constant pressure, Cv is the specific heat at constant volume, R is the universal gas constant, M is the molecular weight of the gas, and T is the temperature. The general procedure for using this formula is to locate the variables in pre-calculated tables. As for this project, the accuracy will still be acceptable if a simplified formula is used where temperature and RH are the only variables considered. This formula can be seen in Equation 3 [10] and provides negligible error. This equation is used in the code to obtain the speed of sound.

$$\text{Speed of sound} = 331.4 + 0.606 \cdot T + 0.0124 \cdot RH \tag{3}$$

With the temperature and RH considered, it remains to see the actual effects these factors have on the measurements.

Figure 6 shows how a change in temperature and RH affects the measured distance. The dotted graphs in both figures show the average of all the points of the respective color. These dotted lines are used as a reference point to see how the distance is affected when a change in temperature and RH occurs. The upper graph shows the change in the measured distance when the room temperature increases from 18° Celsius to 22° Celsius, a relatively small change in temperature that might occur in an arbitrary room. This small change in temperature results in the speed of sound going from 342.3 m/s to 344.7 m/s when using the formula in Equation 3 with 0 % RH. Hence, a temperature increase of 4° Celsius increases the speed of sound by 2.4 m/s, which again results in a 0.46 cm change in measured distance. This temperature change is relatively small, whereas a more significant change would cause a greater error. This error is not optimal considering the sensor's resolution of 0.3 cm.

Change in RH has less impact on the measured distance. On the bottom graph, the change in the measured distance becomes 0.27 cm when the RH goes from 0 % to 100 % RH. This increase in RH

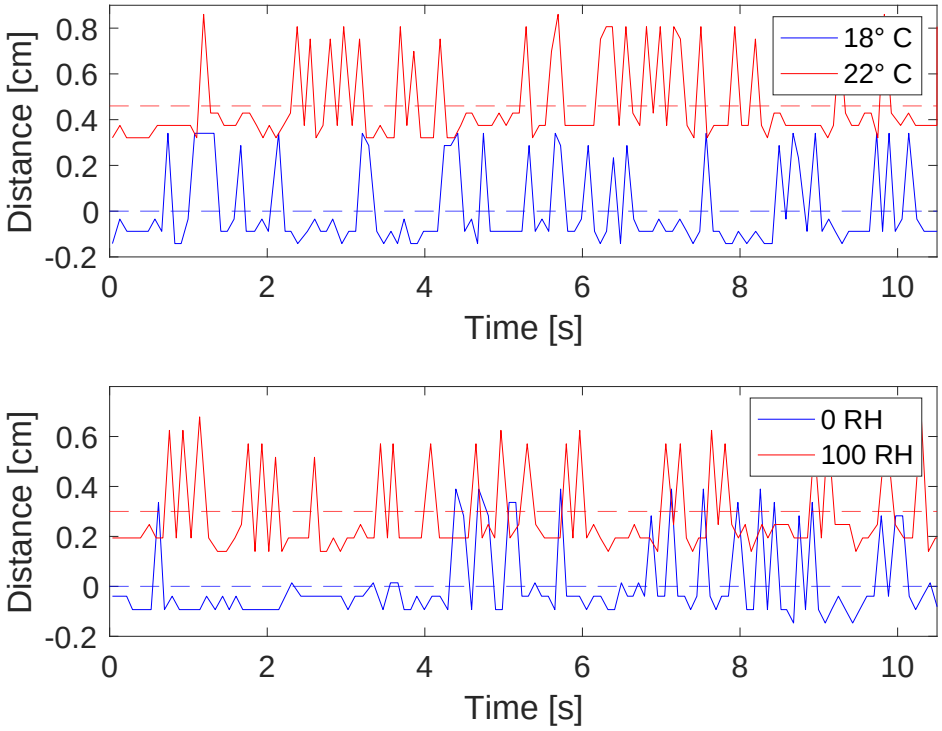


Figure 6: Measured distance is affected by changes in temperature and RH

will be based on the formula in Equation 3 with a constant temperature of 18° Celsius, resulting in the speed of sound going from 342.3 m/s to 343.5 m/s. This change provides an error less than the sensor's resolution of 0.3 cm. Considering that this is the maximum possible change in RH, it should not significantly affect the measurements.

The AMD2302 DHT22 thermostat was implemented with the sensor and microcontroller to account for possible temperature changes. As mentioned, the RH does not have such a significant impact on the measured distance. However, as the thermostat also comes with a hydrometer able to measure RH, there is no reason not to include it to minimize error further. This thermostat was made part of the microcontroller's start-up routine. This addition means that the thermostat measures temperature and RH every time the microcontroller is started. These values are further used in Equation 3 to calculate the speed of sound in the surrounding area. This speed is used to calculate the distance acquired by the ultrasonic sensor.

4.4 Correcting OptiTrack´s view

As mentioned in Section 2.1.5, the motion capture system OptiTrack is used to track the position of objects relative to each other. It does so by using small reflective spheres attached to the object it is tracking, in this case, the sensor. A minimum of three spheres fixed relative to each other are

required for OptiTrack to characterize it as a rigid body. OptiTrack can then gather each sphere's position in the rigid body. From the previous report, [7], these dynamics were misunderstood when performing tests. It was believed that OptiTrack used triangulation when locating the position of the rigid body. Figure 7 illustrates how this was done wrong. The assembly in the figure was a temporary solution made with cardboard for testing the sensor together with OptiTrack.

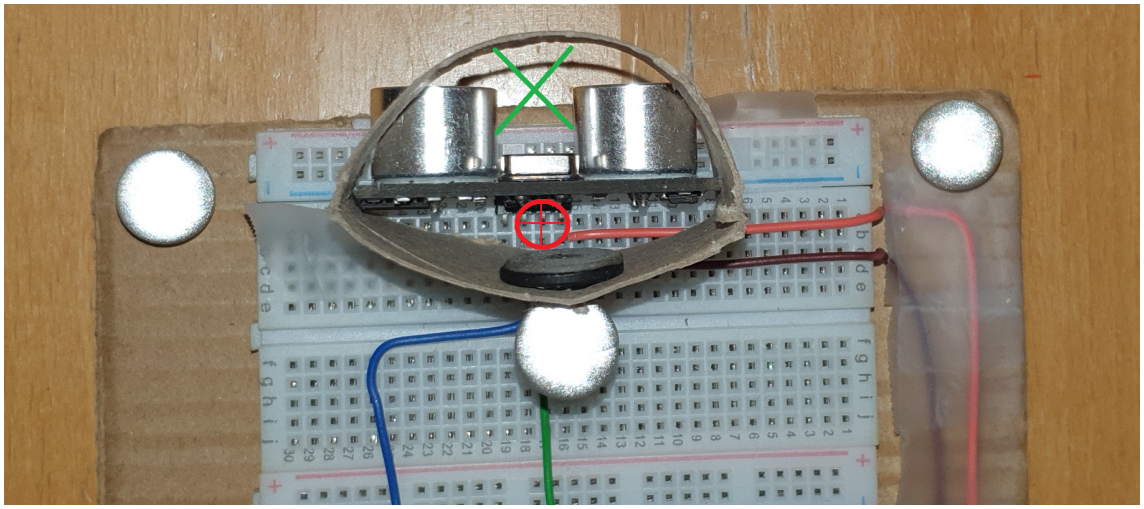


Figure 7: Illustration of the incorrect adjustment done in the previous report

The green cross marks the approximate point where the sensor emits its sound waves. It is preferable to place a sphere and gather its position data at this point. The red circle shows the location where it was believed that OptiTrack was gathering data. The distance between the triangulation point in red and the sensor in green was subtracted from the distance data afterward. However, in reality, the gathered data from OptiTrack provided the position of all three spheres shown in white. This realization meant that the compensation done in the previous report was incorrect. The correct way would be to compensate for the distance between the position of the sensor to the sphere whose position data was being used. An illustration of how this correct compensation was done can be seen in Figure 11b in Section 7.1.

5 Altering software

The microcontrollers used on the vehicles are all implemented with the same software. This software is written to control and monitor all the components attached to and used by the robot it is controlling. Some of these components are wheels, IMU, motors, and LEDs. These components are not yet part of the quadcopter. Some of these components might be implemented later, such as an IMU. However, only four distance sensors and one temperature- and humidity sensor are connected to the microcontroller. This change of components means that all of the software that operates these components can be removed. If not removed, the code will not run due to the hardware not being detected and providing data. By removing these bits of code, the software also becomes more manageable and easy to navigate, freeing some space on the microcontroller. In addition to removing bits of the code, some additions were made for the new hardware. These additions are briefly described below.

5.1 main.c

This file is where the pins on the microcontroller connected to the sensors are configured. The pins are configured as output or input depending on whether the pin is responsible for sending or receiving signals. The timer used to measure the signals from the sensor and convert them to distance was configured here. This same timer is also used as a time reference for plotting graphs.

Another critical alteration was done in the main.c file regarding threads. A thread named MainSensorTowerTask had to be given a higher priority than it had before. This thread is responsible for measuring distances using IR sensors. These sensors have a different input type than what the ultrasonic sensor used for this project has. Immediately after the ultrasonic sensor has emitted a sound wave, it emits a signal to the respective input pin in the microcontroller. This signal is active until the sensor receives the sound wave. The distance can be calculated by measuring how long this signal is active. The issue of having the MainSensorTowerTask thread set to a low priority was that the code responsible for detecting the start of this signal had the chance of being deprioritized due to higher priority tasks needing the microcontroller's attention. If this happened immediately before the signal from the ultrasonic sensor was initiated, the code would never detect the signal, thus being stuck. This detection was done via a while-loop. A time-interrupt was also introduced in the while-loop to prevent this from happening. Increasing the thread's priority means that other threads are being set back in the queue, which could lead to problems, but this change did not seem to cause problems.

5.2 HC_SR04.c and HC_SR04.h

This software is responsible for controlling the included hardware for this project. The source file contains the initialization of the timer and the functions used to calculate the distance, temperature, humidity, and speed of sound. The header file references the functions. Amongst other functions, the while-loop responsible for detecting the signals from the ultrasonic sensor mentioned in the previous section is found in this source file.

A problem regarding the timer in the source file was encountered. Previously the timer used in HC_SR04.c was NRF_TIMER1. Integrating this timer with the robot application turned out to cause an overlap with the FreeRTOS-timer. This overlap was fixed by using NRF_TIMER2 instead of NRF_TIMER1.

The code used to read the data from the AM2302 DHT22 Temperature and Humidity sensor is also included in this file [18].

5.3 SensorTowerTask.c

The robot vehicles use four IR sensors to gather distance data. The function used to gather this IR data was switched out with a function using the ultrasonic sensors instead. This function is defined in HC_SR04.c. SensorTowerTask.c also contains the function that communicates with the server. This function contains different input arguments. Four of these input arguments are distance data previously taken from the IR functions, now taken from the ultrasonic sensors. The other input arguments, such as tower angle and movement, were constricted as no hardware provided this data. The input argument named xhat, representing movement, had to be artificially iterated during the testing in Section 8.2 to simulate movement.

5.4 robot_config.h

The different pins were configured with names and values in this header file. Some global variables used across files were also defined here, such as the speed of sound and elapsed time.

5.5 Javaserver

An issue arose during the testing regarding maximum distance. The java-server was not able to map anything beyond 60 cm. It turned out to be restricted by a variable called sensorRange in the java-server with a value of 60. This variable was increased to 400, which is the maximum range for the ultrasonic sensor. The results looked promising after changing this variable as the server began to map above 60 cm. However, it stopped at a distance of 127 cm. This new limit lies in the type of data sent to the server. The function in SensorTowerTask.c responsible for communicating the

distance with the server has input arguments of the datatype uint8_t. This datatype can provide values between 0 and 255. Theoretically, this means that the server should be able to map up to 255 cm. The limit of 127 cm is caused by the server using a datatype named signed 8 bit with the maximum value of 127. Locating and redefining this maximum distance remains unsolved and is elaborated in Section 10.

6 Case creation and assembly

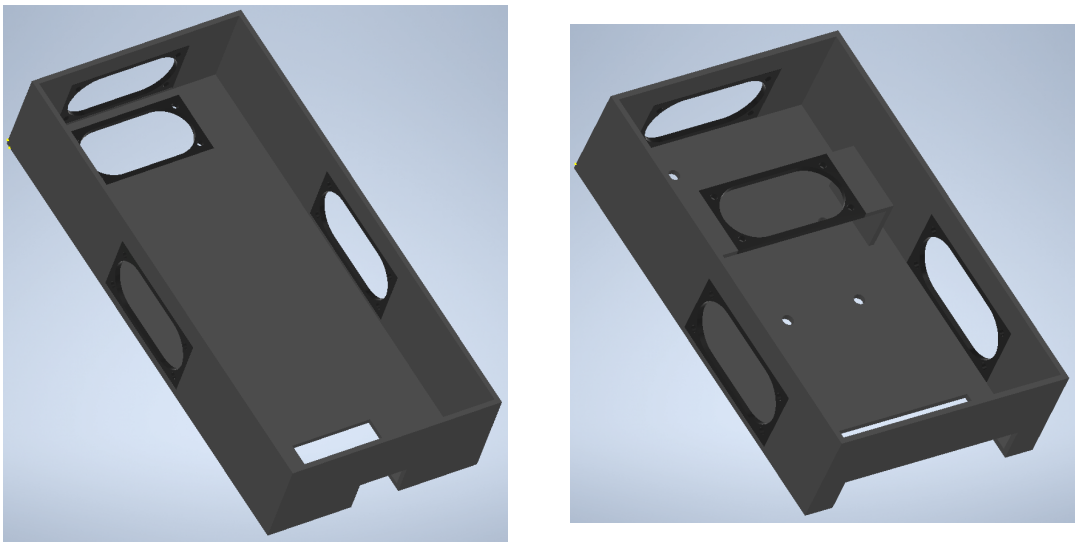
A casing for all the hardware was created to test all four sensors. The purpose of the casing is to hold the hardware in place, protect it from external damage, and function as a fitting to the quadcopter. The casing was designed based on the hardware, and the dimensions of the Mavic MINI addressed in Section 2.1.6.

6.1 Design and weight

It would be best to attach the casing under the quadcopter to maintain stability. The first thought was to create the case small enough not to obscure the built-in camera on the underside of the Mavic MINI. At the same time, a smaller case would mean less weight which is beneficial for an aerial vehicle.

The first design of the case was done in Inventor and shown in Figure 8a. The four oval darker, colored holes are where the ultrasonic sensors are fastened. Two sensors point to each side of the case, one sensor points down to the ground, and the last sensor points forward to measure what is in front of the quadcopter. The microcontroller is to be fastened at the visible bottom inside the case. The length of this design was 18.4 cm, which means that with the underside of the Mavic MINI being 14 cm, the casing would either obscure the camera or make the quadcopter unstable. The bottom oval hole meant for a downward-pointing sensor was relocated to a plateau in the middle of the casing seen in Figure 8b to downsize the case. This plateau was high enough for the microcontroller to still fit beneath by inserting it through a wider opening at the back. With this new design, the case had to be flipped 180° to measure the distance downwards. This altering of the design reduced the size of the casing to 14.5 cm. This reduction is still longer than the Mavics MINI underside of 14 cm, and the reason for not creating it any smaller was due to the weight addressed below.

The weight turned out to be a more significant issue than expected. According to a payload test for the Mavic-drones [11], the Mavic MINI was able to lift 181 grams, but with a heavy toll on the engines. The casing where to be 3D-printed using PLA, which is a type of plastic [25] that weighs 1.24 g/cm^3 [16]. The total weight of the hardware consists of five sensors weighing approximately 10 g each, the microcontroller weighing approximately 70 g, and a battery for the power supply weighing 67 g, A.5. This weight of the hardware sums up to a total of 177 g. By placing the casing model shown in Figure 8b into the 3D-printing program PrusaSlicer, the weight was shown to be 72 g. Together with the hardware, the total weight becomes 249 g, exceeding the earlier mentioned limit of 181 g. This total weight is not considering minor weight contributors such as bolts, nuts, cables, headers, and a breadboard.



(a) Design 1, 18.4 cm

(b) Design 2, 14.5 cm

Figure 8: Reducing the length of the casing

The thickness of the casing walls was reduced from 3 mm to 2 mm to reduce the weight. This reduction did not help more than a few grams due to the 3D print not being solid but instead having a 15 % infill in the walls. Further actions could be done to reduce the weight, such as hollowing out parts of the structure, using a slightly lighter 3D-print filament, or replacing the battery with a lighter one. However, hollowing out parts could reduce the stiffness of the casing to an undesired level, and the available lighter filaments would not reduce the weight enough.

Lighter batteries exist, but the decision was to go with the current design with the already implemented thickness reduction and switch to another stronger quadcopter when necessary. This decision was based not only on the excessive weight but also on the fact that more equipment will likely be mounted to the quadcopter in the future, making it require even more power. More equipment could also mean that a larger casing is required, resulting in the Mavic MINI being too small for this project.

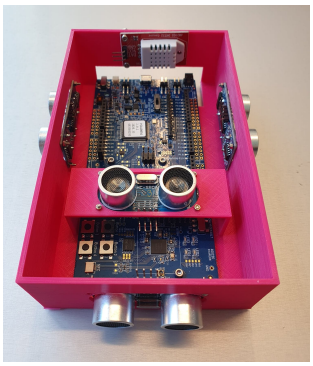
6.2 Mounting hardware and cables

The finished 3D-printed casing can be seen in a stunning color in Figure 9a. The four ultrasonic sensors, microcontroller, and temperature- and humidity sensor were fastened using M1.7 bolts and nuts, seen in Figure 9b. The temperature- and humidity sensor had to be fastened to an unused wall as it was not considered when designing the case.

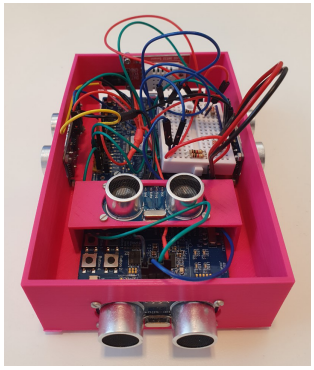
Headers were soldered onto designated locations on the microcontroller so that jumper cables could be attached to connect the hardware further. The cables providing 5 V and ground could be soldered together to create a common power- and ground supply. The common power supply is sufficient as only one sensor is active at a time. The finished assembly containing all the connected hardware can be seen in Figure 9c.

When the ultrasonic sensor is activated, it receives 5 V. This same amount of voltage is also received by the echo-pins on the nRF52840 microcontroller. This voltage is an issue as the echo-pins have a maximum input voltage of 3.9 V [28]. Continuously providing the input pins with 5 V could damage them. Hence a voltage divider was created to reduce the output voltage from the sensors [6]. This voltage divider reduced the returning voltage from the sensors from 5 V to 3.4 V, which satisfies the maximum of 3.9 V. The voltage divider can be seen in Figure 9d. A more detailed schematic drawing and the calculations can be seen in A.4.

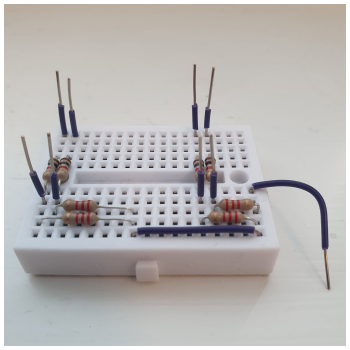
(a) Empty casing



(b) Attached hardware



(c) Connected hardware



(d) Voltage divider

Figure 9: Casing before and after hardware was fastened and connected

7 Preparing the test environment

Section 4 discusses improvements regarding the accuracy of the ultrasonic sensor. This section describes how the accuracy was tested using two different environments.

7.1 One-dimensional test setup

The data from the sensors have to be compared relative to something to comprehend their accuracy, which is where OptiTrack comes into play. As mentioned in Section 2.1.5, small reflective spheres are used to gather this data. Figure 10 shows three items used as references for OptiTrack to gather data.

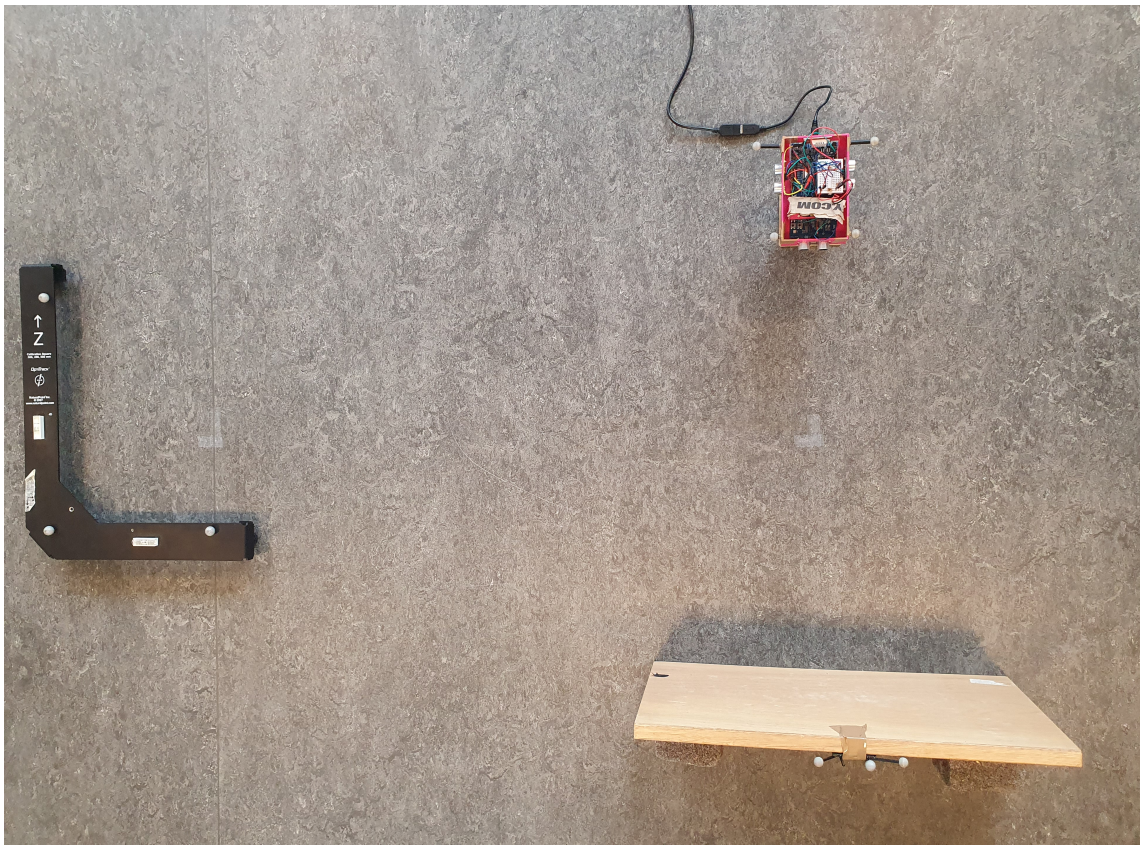
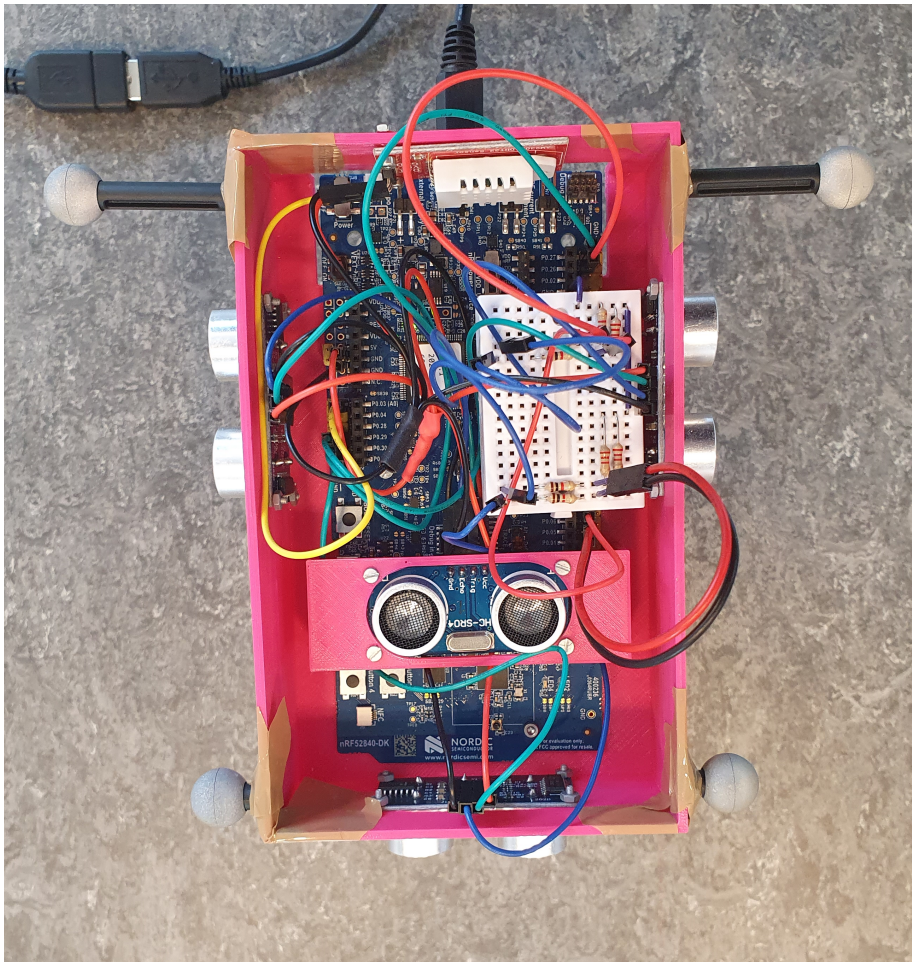
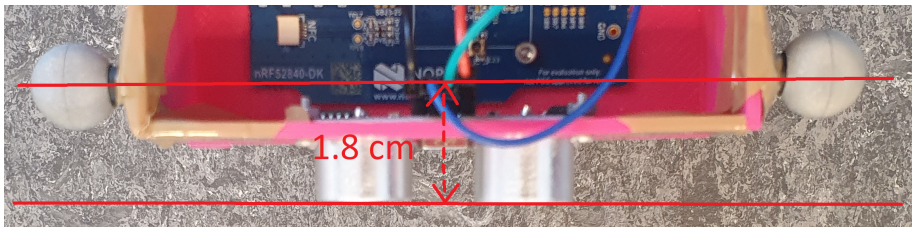


Figure 10: Calibration square to the left, wall and casing with sensors to the right

The calibration square contains three reflecting spheres and functions as a base point for OptiTrack. It tells the system where the ground is and is a reference for every other sphere in the room. The wall and casing can also be seen to have spheres attached to them. As mentioned in Section 4.4, the distance between the spheres and sensor must be altered when processing the data afterward. Figure 11 shows a closer look at the casing.



(a) Close up of the casing showing four spheres attached



(b) The distance between the spheres and sensor has to be altered accordingly

Figure 11: Close up of the casing in the setup shown in Figure 10

For OptiTrack to track and record the position of the casing, the four spheres seen in Figure 11a could, in theory, be reduced to three. The only reason for having four spheres was a matter of symmetry and simplifying the process of visually locating the body in Motive, the program used to manage OptiTrack. Figure 11b shows how the distance needs to be altered when comparing the data. This same alteration had to be done with the spheres attached to the wall seen in Figure 10. These spheres were attached a short distance from the wall's surface, which is a reflective point for the sensors.

The test was conducted by simultaneously launching the Motive and J-Link Viewer recording software before moving the casing with the sensors back and forth from the wall. The results are shown and discussed in Section 8.1.

7.2 Three-dimensional test setup

The previous one-dimensional test setup tests the accuracy of one sensor. As mentioned in the problem description, the goal is to find a solution to measure the position of the quadcopter. The setup seen in Figure 12 was created to test how all four sensors combined would provide an estimate of the quadcopter's position.

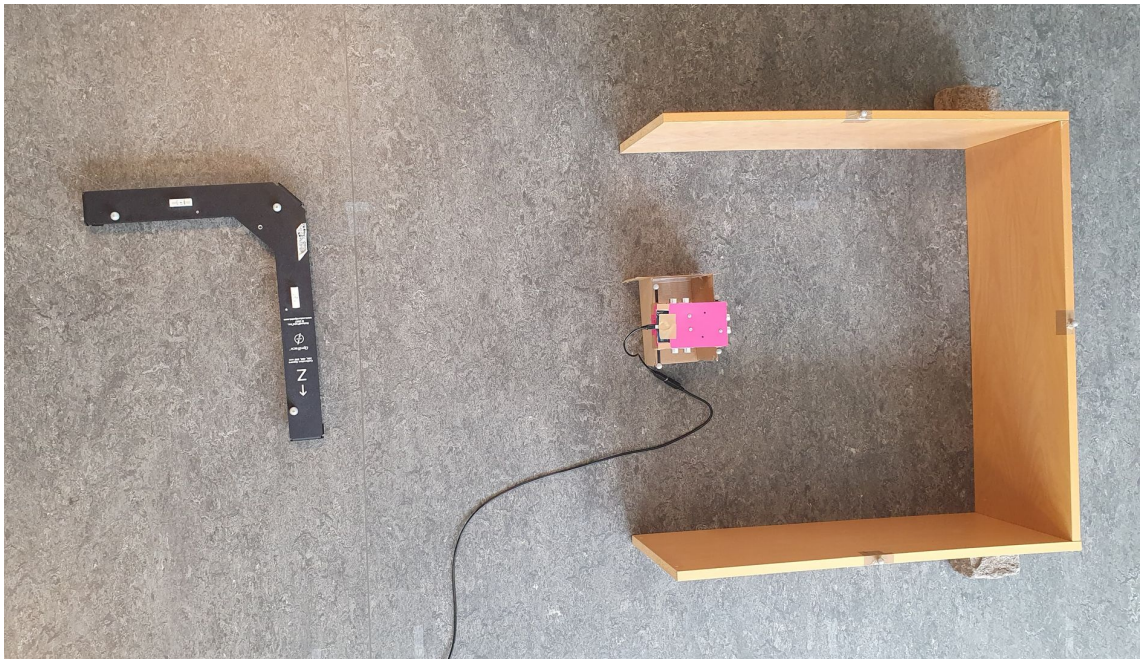


Figure 12: Calibration square to the left, walls and casing with sensors to the right

This setup looks similar to Section 7.1 above for a one dimensional test. The difference lies in the number of sensors being active. All four sensors are active for this three-dimensional test, thus requiring more surfaces to gather measurements. These surfaces are the walls in Figure 12. Each wall contains a reflective sphere mounted on the top. These spheres have the same purpose

mentioned in the one-dimensional test, to get the exact position between the sensors and walls.

A minor alteration of the four spheres was required. It proved somewhat problematic for OptiTrack to track the spheres if they were close to one of the walls. The reason was the vision of too many cameras being obstructed by the walls. Fixing this issue required a repositioning of the spheres. Figure 13 shows how the spheres were placed higher, making them visible enough for OptiTrack to track them continuously.

Figure 13: Placement of spheres for the three dimensional test

Performing the test was done similarly to the one-dimensional test, only in this scenario, the case was moved around in all directions, restricted by the three walls. The results are discussed in Section 8.3.

8 Results

One important thing to note regarding the following results is the sensor's distance to the walls. In the tests, the sensors and walls are closer than what might occur in a real mapping scenario. Even though the ultrasonic sensors might not suit larger spaces, a similar setup and software could be used if switching sensor types.

8.1 Results from the one-dimensional test

The results from the test conducted in Section 7.1 can be seen in Figure 14.

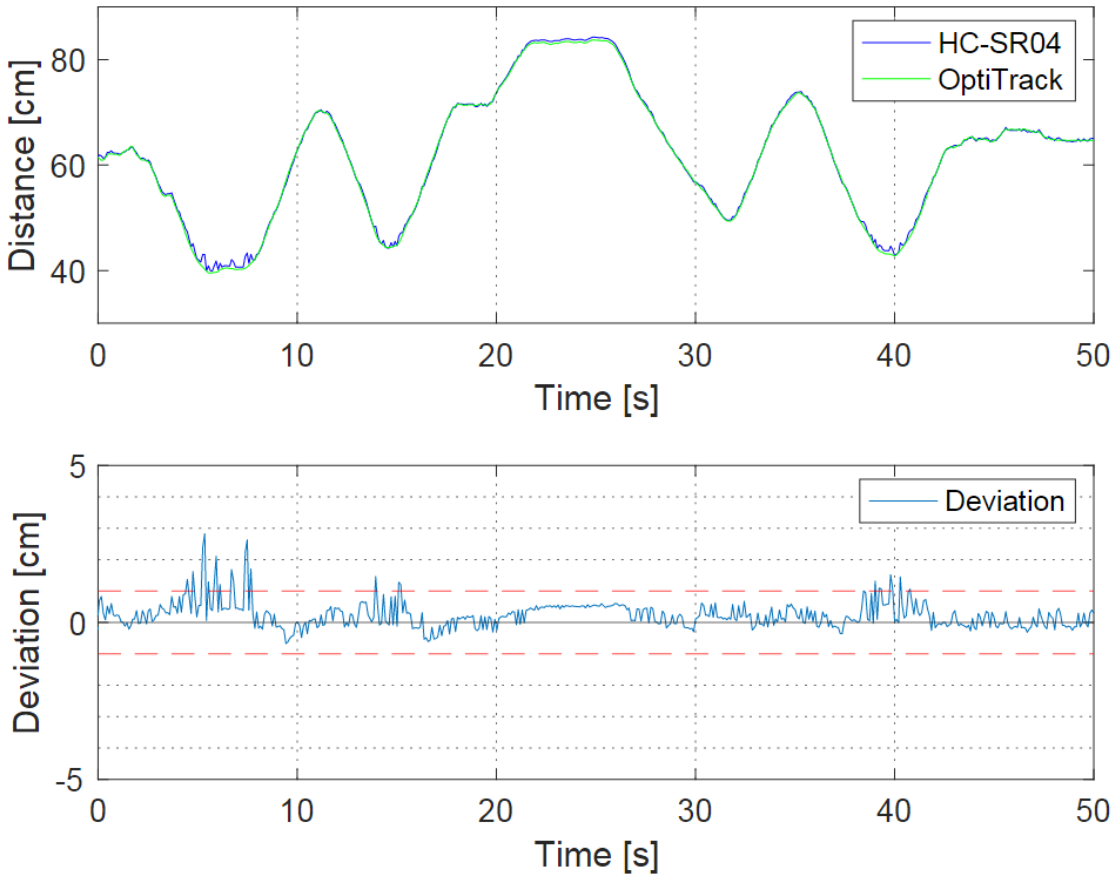


Figure 14: The sensors measurements compared to OptiTrack

The green graph in the top plot shows the correct distance of the ultrasonic distance sensor relative to the wall recorded by OptiTrack. The blue graph in the top plot shows the distance provided by the sensor. The bottom plot shows the deviation between the green and blue graphs. Two red dotted lines at +1 cm and -1 cm are included as a visual reference.

The more valuable part of Figure 14 is the deviation graph at the bottom. The deviation fluctuates mainly about zero between the two red dotted lines, which means that the sensor provides accurate measurements of ± 1 cm. Whether this is accurate depends on the server's ability to handle these

received measurements. The type of data that is being sent to the server is integers. This type means that whether the sensor reads 9.6 cm or 10.4 cm, both will be interpreted as 10 cm. Thus, measurements containing decimals are excessive. Hence, the deviation shows that the server would have an error of ± 1 cm in most cases. The only digression of this particular test would be in the first 10 seconds, where the deviation is close to 3 cm, providing a matching error in the server map.

8.2 Server testing

The objective of this test was to ensure that the server receives the ultrasonic distance measurements and correctly plots them as surfaces. Establishing a connection with the server required adapting the software to fit the new hardware, as addressed in Section 5. The testing could begin after successfully establishing a connection with the Dongle and server. The same setup and method as the previous test was used. The only addition to this test was the server running and receiving data from the microcontroller. The resulting map created by the server can be seen in Figure 15.

Figure 15: Map created by the java server

The server uses three nuances to represent different entities in the respective area. White is the open space available to the robots, and black is a solid surface. The light grey is the unreachable area close to a surface limited by the size of the robot body. In contrast, the darker grey is the undiscovered area typically found behind a surface.

The java server consists of various variables provided by the different sensors on the robot mentioned in Section 5. One of these variables is the robot's movement measured by the rotation of the wheels. This rotation allows the server to get a perspective on where the robot is at all times and map at different locations. As for this test, there are no wheels providing movement. The server will believe that the robot is standing still when mapping without this movement, resulting in a plain black dot whose position is constantly overwritten. Such a map would not be handy for later comparison. Acquiring the surface shown as a black line in Figure 15 required mimicking movement. Hence a continuously iterating variable was created to simulate movement in one direction. This iteration ensured that the varying black line in Figure 15 was not constantly overwritten by new data. To get a perspective on how accurate the representation in the server is, OptiTrack was used simultaneously. Figure 16 shows how the sensor performed compared to OptiTrack.

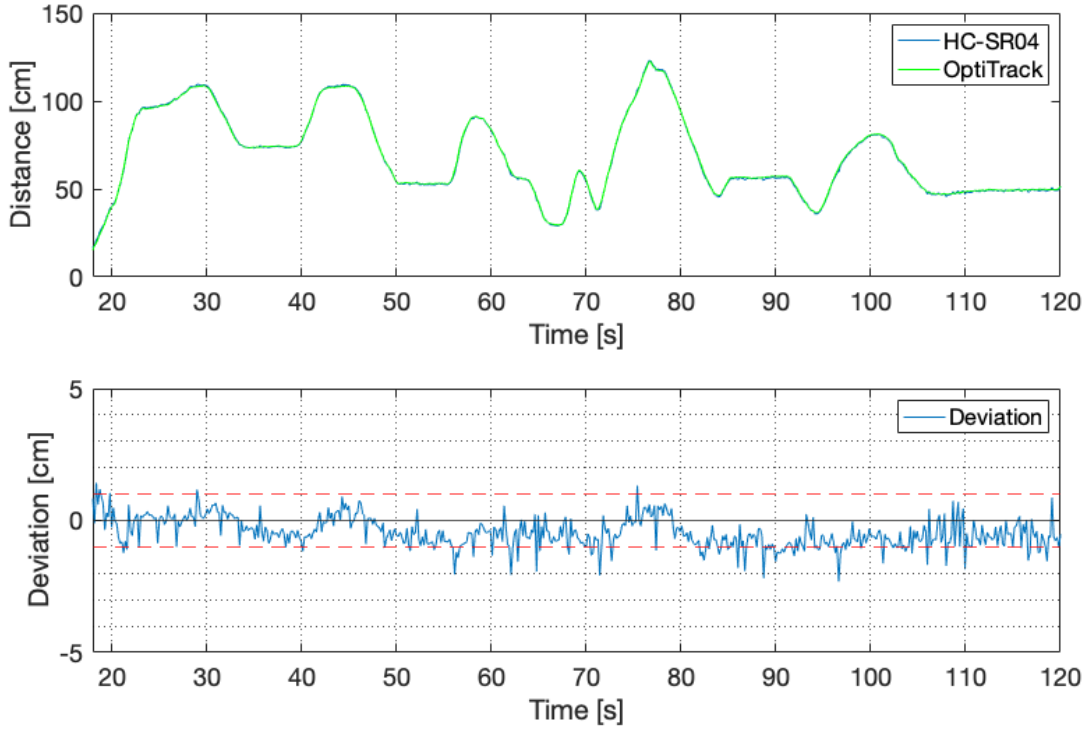


Figure 16: The sensors measurements compared to OptiTrack. Recorded at the same time the server was mapping in Figure 15

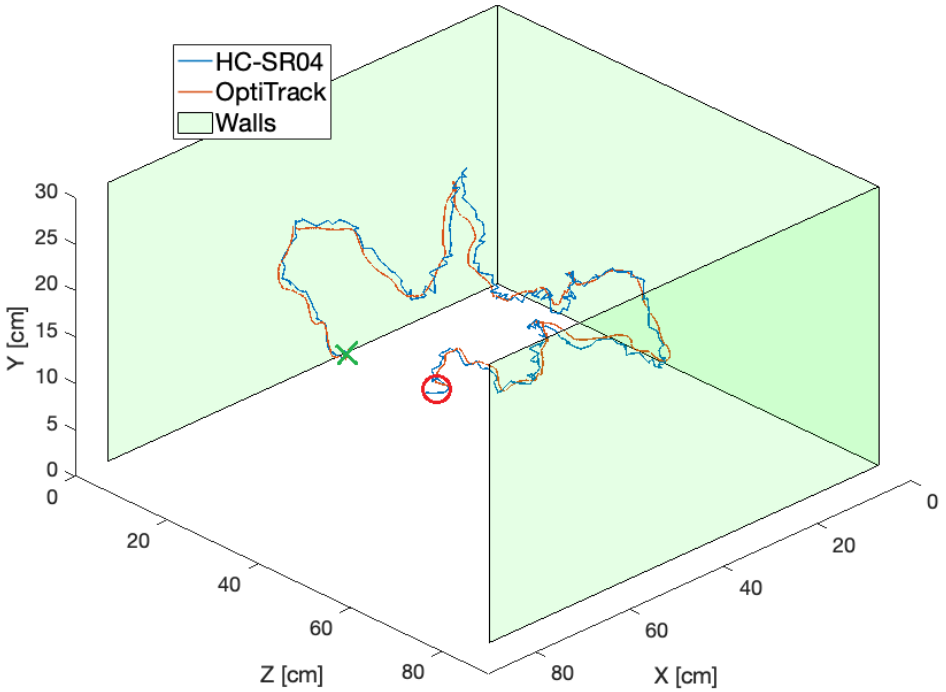
The shape of the graphs in Figure 15 and 16 appear to be similar. This similarity indicates that the server successfully receives and identifies sensor measurements as surfaces.

The sensor provides reasonably accurate distances, as seen at the bottom in Figure 16. The figure represents 568 measurements from the sensor, whereas 481 are between the two red dotted lines. In other words, the sensor provides measurements with an accuracy of ± 1 cm, 85% of the time.

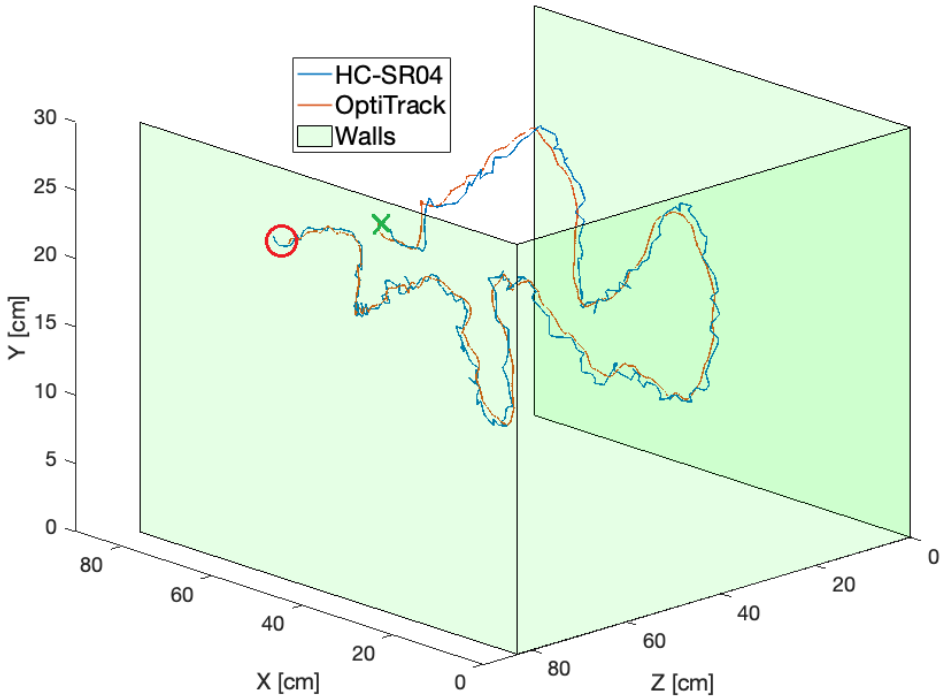
Section 8.1 discussing the first conducted test mentioned the communication between the server and sensor having the datatype of integers. This functions as a filter that diminishes accuracy by neglecting decimals. Furthermore, the accuracy of the map created by the java server is based upon the size of the pixels seen in Figure 15. The size of each pixel illustrated by the server is 2 cm as a default but can be changed in the server's code. This pixel size means that the accuracy of ± 1 cm could lead to one pixel off.

8.3 Results from the three-dimensional test

The results from the three dimensional test addressed in Section 7.2 can be seen in Figure 17. The results are presented from two different angles. The three green surfaces represent the three walls from the setup. They are made transparent in the figure to visualize the trajectory of the graphs.



(a) First view



(b) Second view

Figure 17: Estimated position based on sensor values compared to actual position

There are two trajectories in Figure 17. The blue line represents the estimated position of the casing generated by the sensors. The orange line represents the actual trajectory of the casing generated by OptiTrack. The red circle represents the start of the movement, and the green cross represents the end. As mentioned in Section 6.1 there are four ultrasonic sensors mounted on the case. All sensors were active and gathered data during the test, but the resulting plot is only based on three sensors. As one sensor points to either side of the setup, they are theoretically providing the same measurements needed for the position in this axis.

By visually inspecting Figure 17, the two trajectories appear to have a similar path. The path generated by OptiTrack has a smoother path due to its fast update rate and consistency. The sensor-generated path is less consistent and has more rapid and sudden movements. In Section 4.1 it was stated that the optimal amount of measurements per second was 16. This amount hindered the sound from one measurement to influence the next and was done by introducing a delay of 60 ms between each measurement cycle. When using four sensors simultaneously, this became an issue. When a sensor had gathered one measurement, it had to wait for the remaining three to finish with their 60 ms delay. This delay led to each sensor only providing around two measurements per second. Few measurements would not be a problem if the quadcopter moved slowly, but the sensor's update rate became too slow with a more realistic pace.

The results in Figure 17 do not have a 60 ms delay between each cycle but rather a 20 ms delay. This reduction increased the measurements per second for each sensor to six. As for the sound from one measurement influencing the next, this did not become an issue. When reducing the delays to smaller than 5 ms, the sound from previous measurements affected the subsequent measurement. However, as 20 ms provided a decent amount of measurements per second, the delay was not examined and decreased further.

Figure 17 provides an idea regarding the accuracy of the estimated position. To get a more comprehensive interpretation, Figure 18 illustrates the deviation between the two graphs in Figure 17 from start to end. The deviation is shown for each direction in the three-dimensional space, hence the X, Y, and Z-axis.

The deviation in Figure 18 is represented in the same manner as the deviation seen in Figure 14 and Figure 16. The red dotted lines illustrate where the deviation between the two graphs is more than one cm. The deviation for all three axes looks similar to the deviation seen in the previous results. Considering that the server interprets all data as whole numbers by rounding up or down, the deviation is mostly either zero or one centimeter. It becomes two centimeters in a few situations, for example, on the X-axis 33 seconds into the test.

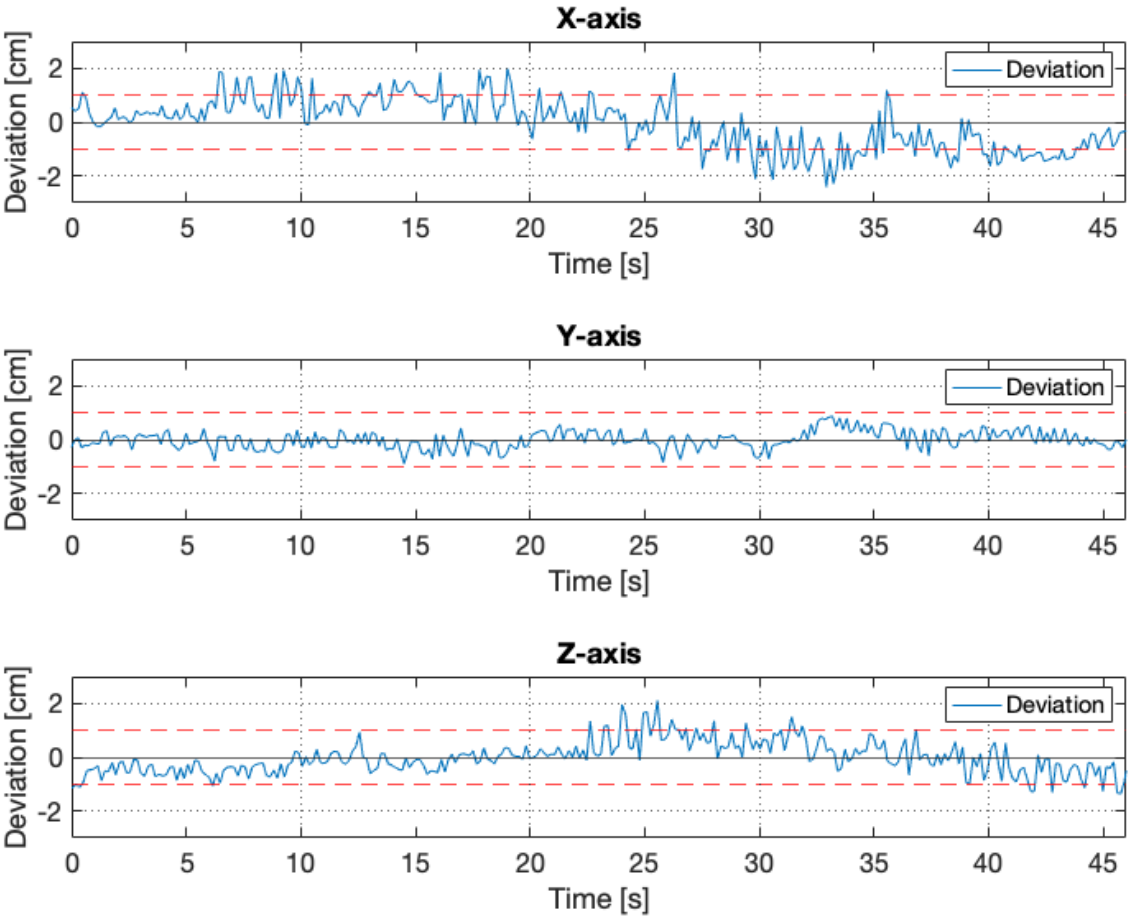


Figure 18: The deviation between the graphs seen in Figure 17. The deviation is represented in each of the three axis, X, Y and Z.

8.4 Stationary measurements

Both the one-dimensional and three-dimensional tests were conducted while continuously moving the casing. The casing was kept motionless at certain points during the tests, such as 25 seconds into the one-dimensional test seen in Figure 14. The corresponding deviation is seen to be more or less stationary. It seems that dynamic effects influence the deviation when moving the casing. These dynamic effects are most likely small changes in the angle of the sensor towards the respective surface and slight variations due to the measuring frequency of the sensor not precisely matching OptiTracks measurements. To see whether movement influences the deviation, consider Figure 19. When performing the three-dimensional test in Figure 17, the casing rested stationary above the ground for about ten seconds before being lifted and recorded. Figure 19 shows what the deviation looked like during these first ten seconds when the casing was completely static.

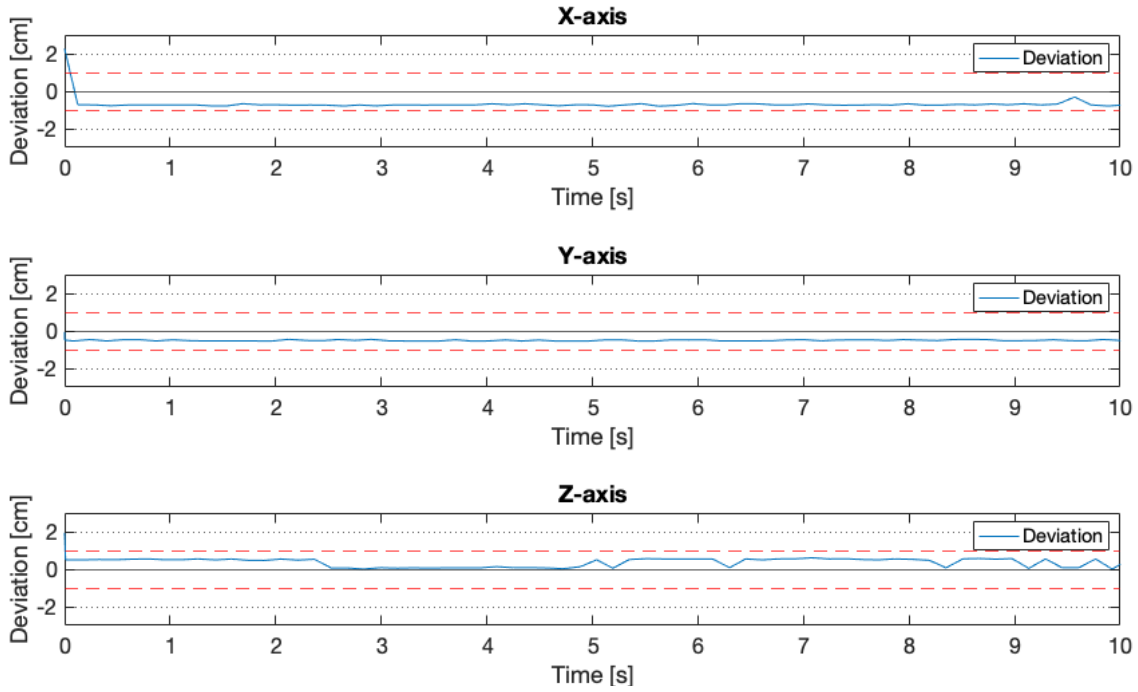


Figure 19: The deviation between the ultrasonic sensor and OptiTrack during the first ten seconds of the three-dimensional test. These first ten seconds is not illustrated in Figure 17

Figure 19 shows a stable deviation never exceeding one centimeter. This test confirms that movement decreases the stability of the measurements. In reality, the casing will be fastened to a quadcopter. The resulting position estimate might be more stable if the quadcopter is hovering while taking pictures rather than moving at a slow speed.

9 Discussion

Consider the two objectives stated at the beginning of this thesis. The first objective was to investigate using ultrasonic sensors to provide an accurate position estimate for a hovering quadcopter in a three-dimensional space. To best conclude this objective, consider the results from the three-dimensional test in Section 8.3. The sensors provided the position of the moving quadcopter with an error of $\pm$ 2 cm .

Due to the pixels building the map being 2 cm, this error can cause the map to be wrong by one pixel in either direction. There is no definite requirement for accuracy now, but there have been considerations to make the map more accurate regarding the project. This alteration would mean decreasing the java server's pixel size and potentially enabling the server to handle decimals. In such a scenario, the ultrasonic sensor would be insufficient in dynamic environments with an error of $\pm$ 2 cm.

However, it was confirmed from stationary measurements that the sensor's accuracy had less deviation when standing completely still. The deviation might be similarly low if the quadcopter can hover reasonably still. If this is the case, the ultrasonic sensor could be an acceptable candidate for measuring position. In a scenario where the ultrasonic sensor provided excellent results with an accuracy of $\pm$ 0.3 cm, its use would still be restricted by its maximum range of 4 meters, as mentioned at the beginning of this section.

A sensor that can measure longer distances is needed to fulfill this objective. A light-based sensor with equal or better precision would be ideal.

The second objective was to look into and modify existing software, making it suitable for collecting and transmitting data to a server regarding new hardware. This objective required altering the existing software. The results from Section 8.2 demonstrated this was possible. The server received the measured distances successfully and was able to map them. However, the limit to large distances being communicated remains an issue.

10 Further Work

10.1 New sensors

Using ultrasonic sensors comes with limitations. In this case, they would be unable to detect the walls in a large area due to the limited sensing range. In addition to this, they cannot provide accurate measurements enough for the server. If the idea is to create an even more accurate map in the future, a different sensor is needed, such as a light-based LIDAR. This change of sensor would drastically increase the update rate of the measurements and the accuracy.

One such sensor that could replace the HC-SR04 currently in use is the VL53L4CX Time of Flight Distance Sensor [3]. It has a range of 1 - 6000 mm, making it able to detect two meters beyond what the ultrasonic distance sensor can. It also uses a narrow light source, making it excellent for only detecting objects directly in front of it. The only drawback would be a higher price. Switching to a light-based sensor would also require rewriting the software and having a more accurate timer as light travels faster than sound.

10.2 Software

The software would not only require a rewriting if the sensors are to be switched. Plenty of removing, including and altering of the existing code remains for it to be optimal. One such example is the changes in priorities that were done. This change was more a test to see if it would work rather than a professional evaluation and should receive a second opinion. Rewriting the software should also be done to handle the sensors more professionally. As of right now, they are simply providing measurement after measurement. By designing and applying an intelligent filter where the measurements would be processed concerning previous measurements, the estimated position could be more accurate.

Perhaps the most critical unsolved problem is the server's maximum distance of 127 cm, as addressed in Section 5.5. This problem is required for the entire length of the distance sensors to be available for the mapping process. Upgrade to a longer-range sensor is useless if this problem is not fixed.

In Section 4.2, it was mentioned that MatLab was used to correct an error in the timer. This correction was done for the sole reason of getting a correct comparison for the graphs. However, correcting afterward is not an option if this timer is to be used in real-time. The amount of time the microcontroller has been running might not be required, but if it were, the delay on the timer would need a permanent fix.

10.3 Casing

If the casing is to be redesigned, specific improvements should take place. The first thing to consider is the position of the sensors and if it is necessary with four. It might be beneficial to create a similar rotating tower mounted on the robots on the ground. The wires connecting all the hardware were slightly in the way of the upward-facing sensor. This web of wires can be avoided by creating a slightly larger case. This larger case should also be able to hold a battery and have a better spot for the voltage divider seen in Figure 9c. The need for a voltage divider was first noticed after a while. This inattentiveness resulted in pin P0.06 on the nRF52840 being defective. The temperature- and humidity sensor does not have a voltage divider. It is not frequently active compared to the sensors, but it should have a voltage divider to prevent more pins from being defective.

10.4 IMU

The next piece of hardware that should be considered as part of the system could be an IMU. Using this IMU together with the sensors could provide better accuracy and function as a tool for measuring the quadcopter's movement. Using a quadcopter would also provide more realistic results when performing tests.

As mentioned in Section 8.4, the angle of the sensor towards the surface affects the accuracy. This angle is not accounted for, but an IMU could be used to resolve this issue.

List of Figures

1 HC-SR04 with dimensions, its depth is 15 mm 3

2 AM2302 DHT22 with dimensions, its depth is 7.7 mm 4

3 Mavic MINI Drone, by TheBetterDay [36] Dimensions: Length 159mm, Width 202mm, Height 55mm 5

4 A graph comparing the sensors measurements with OptiTrack taken from the previous report [7] 13

5 The improved timer illustrated 15

6 Measured distance is affected by changes in temperature and RH 17

7 Illustration of the incorrect adjustment done in the previous report 18

8 Reducing the length of the casing 23

9 Casing before and after hardware was fastened and connected 24

10 Calibration square to the left, wall and casing with sensors to the right 25

11 Close up of the casing in the setup shown in Figure 10 26

12 Calibration square to the left, walls and casing with sensors to the right 27

13 Placement of spheres for the three dimensional test 28

14 The sensors measurements compared to OptiTrack 29

15 Map created by the java server 30

16 The sensors measurements compared to OptiTrack. Recorded at the same time the server was mapping in Figure 15 31

17 Estimated position based on sensor values compared to actual position 32

18 The deviation between the graphs seen in Figure 17. The deviation is represented in each of the three axis, X, Y and Z. 34

19 The deviation between the ultrasonic sensor and OptiTrack during the first ten seconds of the three-dimensional test. These first ten seconds is not illustrated in Figure 17 35

20 A schematic drawing of the voltage divider seen in Figure 9d 44

21 NTNU powerbank 5 V 45

Bibliography

[1] Khan Academy. *Voltage divider*. URL: https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-voltage-divider. (accessed: 25.04.2022).

[2] Acroname. *Sharp GP2Y0A710YK0F Long Range IR Distance Sensor*. URL: https://acroname.com/store/long-range-distance-sensor-r316-gp2y0a710yk. (accessed: 11.04.2022).

[3] Adafruit. *Adafruit VL53L1X Time of Flight Distance Sensor*. URL: https://www.adafruit.com/product/5425. (accessed: 13.04.2022).

[4] Autodesk. *Inventor: Powerful mechanical design software for your most ambitious ideas*. URL: https://www.autodesk.com.au/products/inventor/overview?term=1-YEAR&tab=subscription. (accessed: 31.03.2022).

[5] Benne de Bakker. *How to use a SHARP GP2Y0A710K0F IR Distance Sensor with Arduino*. URL: https://www.makerguides.com/sharp-gp2y0a710k0f-ir-distance-sensor-arduino-tutorial/. (accessed: 02.04.2022).

[6] Steve Bertrand. *Voltage divider*. URL: https://metacpan.org/pod/RPi::HCSR04. (accessed: 25.04.2022).

[7] Jonas Bjerke. 'Position Measurement for Quadcopter'. In: (2021).

[8] Roderick Burnett. *Ultrasonic vs Infrared (IR) Sensors – Which is better?* URL: https://www.maxbotix.com/articles/ultrasonic-or-infrared-sensors.htm. (accessed: 08.06.2022).

[9] Roderick Burnett. *Understanding How Ultrasonic Sensors Work*. URL: https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm#:~:text=Ultrasonic%5C%20sensors%5C%20are%5C%20suitable%5C%20for,multiple%5C%20range%5C%20measurements%5C%20per%5C%20second.. (accessed: 07.04.2022).

[10] Scott Campbell. *How to set up an ultrasonic range finder on an arduino*. URL: https://www.circuitbasics.com/how-to-set-up-an-ultrasonic-range-finder-on-an-arduino/. (accessed: 13.04.2022).

[11] Sean Captain. *The dead lift contest*. URL: https://dronedj.com/2020/06/01/dji-mavic-air-2-can-lift-a-lot-more-than-its-body-weight/. (accessed: 05.04.2022).

[12] Visual Studio Code. *Visual Studio Code FAQ*. URL: https://code.visualstudio.com/docs/supporting/FAQ. (accessed: 31.03.2022).

[13] droneblog. *Drone vs Quadcopter: What are the Differences?* URL: https://www.droneblog.com/drone-vs-quadcopter-what-are-the-differences/. (accessed: 24.05.2022).

[14] elecFreaks. *Ultrasonic Ranging Module HC - SR04*. URL: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. (accessed: 13.04.2022).

[15] FreeRTOS. *What is FreeRTOS?* URL: https://www.freertos.org/about-RTOS.html. (accessed: 01.04.2022).

[16] Robert French. *The density chart of all 3D materials.* URL: https://bitfab.io/blog/3d-printing-materials-densities/. (accessed: 05.04.2022).

[17] InfraTec. *Infrared Sensor - IR Sensor.* URL: https://www.infratec.eu/sensor-division/service-support/glossary/infrared-sensor/#:~:text=An%5C%20infrared%5C%20sensor%5C%20(IR%5C%20sensor,systems%5C%20to%5C%20detect%5C%20unwelcome%5C%20guests.. (accessed: 02.04.2022).

[18] Thomas Liu. *Digital relative humidity and temperature sensor AM2302/DHT22.* URL: https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf. (accessed: 12.04.2022).

[19] MathWorks. *Math. Graphics. Programming.* URL: https://www.mathworks.com/products/matlab.html. (accessed: 31.03.2022).

[20] Elijah J. Morgan. *HCSR04 Ultrasonic Sensor.* URL: https://datasheetspdf.com/pdf-file/1380136/ETC/HC-SR04/1. (accessed: 05.04.2022).

[21] NASA. *What are Infrared Waves?* URL: https://science.nasa.gov/ems/07_infraredwaves#:~:text=Infrared%5C%20waves%5C%2C%5C%20or%5C%20infrared%5C%20light,change%5C%20channels%5C%20on%5C%20your%5C%20TV.. (accessed: 02.04.2022).

[22] OpenStax. *Speed of sound.* URL: https://phys.libretexts.org/Bookshelves/University_Physics/Book%5C%3A_University_Physics_(OpenStax)/Book%5C%3A_University_Physics_I_-_Mechanics_Sound_Oscillations_and_Waves_(OpenStax)/17%5C%3A_Sound/17.03%5C%3A_Speed_of_Sound. (accessed: 12.05.2022).

[23] OptiTrack. *Optical motion capture software.* URL: https://optitrack.com/software/motive/. (accessed: 12.04.2022).

[24] Planet-Science. *What is a laser?* URL: http://www.planet-science.com/categories/over-11s/technology/2012/01/what-is-a-laser.aspx. (accessed: 24.05.2022).

[25] Tony Rogers. *Everything You Need To Know About Polylactic Acid (PLA).* URL: https://www.creativemechanisms.com/blog/learn-about-polylactic-acid-pla-prototypes. (accessed: 05.04.2022).

[26] Segger. *Embedded Studio — The leading multi-platform IDE.* URL: https://www.segger.com/products/development-tools/embedded-studio/. (accessed: 31.03.2022).

[27] Segger. *J-Link Tools - RTT Viewer.* URL: https://www.segger.com/products/debug-probes/j-link/tools/rtt-viewer/. (accessed: 31.03.2022).

[28] Nordic Semiconductors. *Absolute maximum ratings.* URL: https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps_nrf52840%5C%2Fabs_max_ratings.html&cp=4_0_0_8. (accessed: 25.04.2022).

[29] Nordic Semiconductors. *nRF Connect for Desktop*. URL: https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop. (accessed: 31.03.2022).

[30] Nordic Semiconductors. *nRF52840*. URL: https://www.nordicsemi.com/Products/nRF52840. (accessed: 10.05.2022).

[31] Nordic Semiconductors. *TIMER — Timer/counter*. URL: https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%5C%2Ftimer.html. (accessed: 13.04.2022).

[32] Senix. *Ultrasonic Sensor FAQs*. URL: https://senix.com/faqs/. (accessed: 07.04.2022).

[33] Shawn. *Laser Distance Sensors: LIDAR*. URL: https://www.seeedstudio.com/blog/2019/12/23/distance-sensors-types-and-selection-guide/. (accessed: 02.04.2022).

[34] Prusa Slicer. *PrusaSlicer introduction and download*. URL: https://www.prusa3d.com/page/prusaslicer_424/. (accessed: 31.03.2022).

[35] Study.com. *Why do longer wavelengths travel farther than shorter wavelengths?* URL: https://study.com/academy/answer/why-do-longer-wavelengths-travel-farther-than-shorter-wavelengths.html. (accessed: 24.05.2022).

[36] TheBetterDay. *DJI MAVIC mini Combo DIY Kit*. URL: https://wordpress.org/openverse/image/f947240f-db40-4bca-9683-f0b1aa1acdcc/. (accessed: 12.05.2022).

[37] Tracklab. *OPTITRACK FLEX 13*. URL: https://tracklab.com.au/products/brands/optitrack/optitrack-flex-cameras/optitrack-flex-13/. (accessed: 12.04.2022).

A Appendix

A.1 Number of measurements

"Theoretically, if an ultrasonic sensor was placed 1 meter from a wall, the speed of sound would make it back and forth 170 times per second A.1."

$$\begin{aligned} \text{Number of measurements} &= \frac{\text{Speed of sound} \cdot \text{Time}}{2} \\ \text{Number of measurements} &= \frac{340\frac{m}{s} \cdot 1\,m}{2} = 170\frac{1}{s} \end{aligned} \tag{4}$$

A.2 Smallest noticeable change

"From the previous report [7], time was measured with a precision of 31.25 microseconds, meaning that the smallest noticeable change in distance the microcontroller was able to detect was 0.5 cm A.2."

$$\begin{aligned} \text{Noticeable change} &= \frac{\text{Speed of sound} \cdot \text{Time interval}}{2} \\ \text{Noticeable change} &= \frac{340\frac{m}{s} \cdot 31.25 \cdot 10^{-6}\,s}{2} = 0.53125cm \end{aligned} \tag{5}$$

A.3 Speed of sound

"These factors along with others can be combined into a general equation for the speed of sound in gasses, shown in Equation 2, A.3."

To get a closer look on what factors contributes to the speed, consider Equation 6 [22], which assumes the speed of sound in an ideal gas. The variables are as follows, c is the speed of sound, γ is the adiabatic index, P is the pressure and ρ is the density of the gas. As the equation applies to ideal gases, it is further assumed that the air surrounding the sensor is an ideal gas.

$$\text{c} = \sqrt{\gamma \frac{P}{\rho}} \tag{6}$$

By rewriting the ideal gas law shown in Equation 7, the fraction including P and ρ in Equation 1 can be simplified to Equation 8.

$$\text{PV} = nRT \tag{7}$$

$$c = \sqrt{\gamma RT} \tag{8}$$

The adiabatic index and R can be further rewritten, giving a more detailed formula for the speed of sound, shown in Equation 9. Cp is the specific heat at constant pressure, Cv is the specific heat at constant volume, R is the universal gas constant, M is the molecular weight of the gas and T is the temperature.

$$c = \sqrt{\frac{Cp}{Cv}\frac{R}{M}T} \tag{9}$$

A.4 Voltage divider

"This voltage divider was able to reduce the returning voltage from the sensors from 5 V to 3.4 V, which satisfies the maximum of 3.9 V. The voltage divider can be seen in Figure 9d. A more detailed schematic drawing together with the calculations can be seen in A.4."

Source of formula: [1].

$$V_{\text{out}} = V_{\text{in}} \cdot \frac{R2}{R1 + R2}$$
$$V_{\text{out}} = 5V \cdot \frac{2.2k\Omega}{1k\Omega + 2.2k\Omega} \approx 3.44V \tag{10}$$

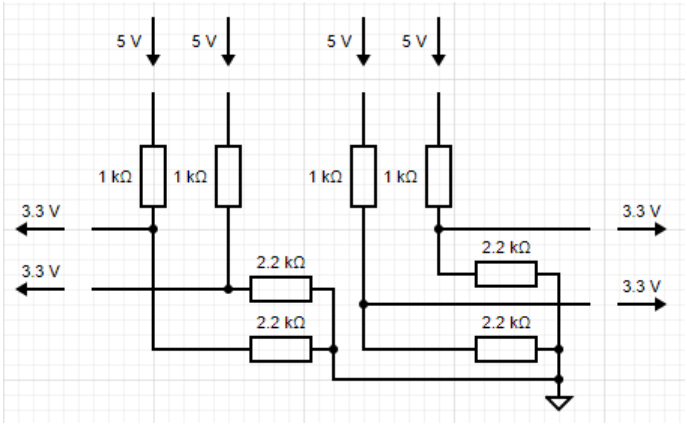


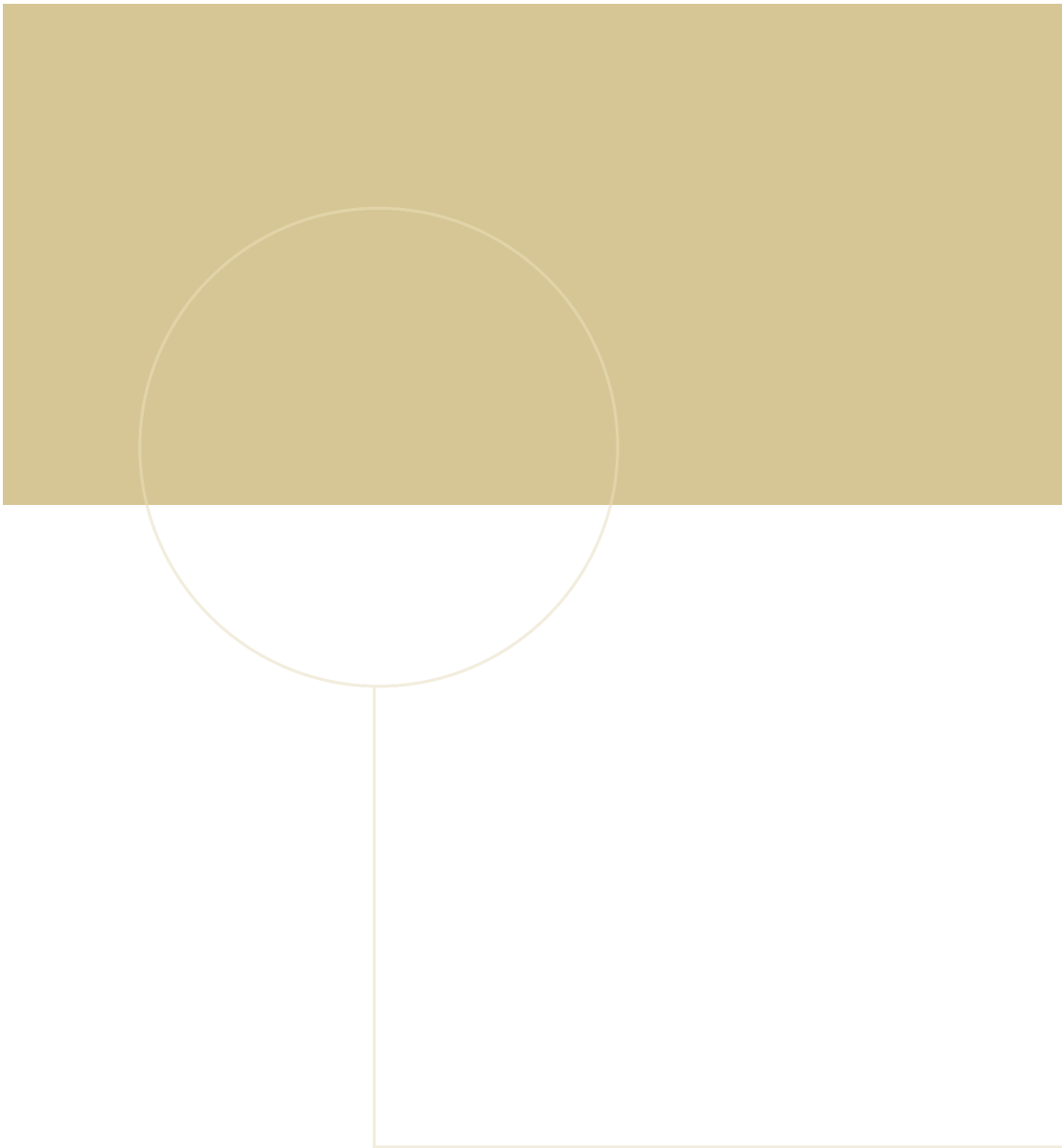
Figure 20: A schematic drawing of the voltage divider seen in Figure 9d

A.5 Battery

"The total weight of the hardware consists of five sensors weighing approximately 10 g each, the microcontroller weighing approximately 70 g, and a battery for the power supply weighing 67 g, A.5."

Figure 21: NTNU powerbank 5 V

The power bank seen in Figure 21 was considered being used if the casing were to be wirelessly mounted to the quadcopter.



NTNU

Norwegian University of
Science and Technology