Magnus Flaten Lindefjeld

Obstacle Detection and Avoidance in a 3D Environment

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

Master's thesis

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

NTNU
Norwegian University of
Science and Technology

Magnus Flaten Lindefjeld

Obstacle Detection and Avoidance in a 3D Environment

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

Problem Description

The objective of this thesis is to develop a method which gives a robot the ability to detect and avoid obstacles in a 3D environment. Before the start of this work, three new robots based on the nRF52840 DK were completed, which are capable of exploring and mapping an unknown environment. Currently, the robots are equipped with sensor towers consisting of 4 IR range sensors each, which provide full 360° awareness in the horizontal plane. However, the existing system is incapable of detecting hazards s.a. holes, low passageways, and small items on the ground.

More specifically, the problem is divided into the following subtasks:

1. Familiarize with, test and calibrate the nRF52840 DK robot
2. Extend the hardware with suitable sensors
3. Based on the sensor data, develop a method s.t. the robot can detect items on the ground, holes, and low passageways
4. Integrate the method into the existing system
5. Test and verify the method

Preface

This thesis is the resulting work of the course *TTK4900 - Engineering Cybernetics, Master's Thesis*. The course concludes the 5 year study program Cybernetics and Robotics at the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU).

The contents of the thesis are the efforts made to develop a method for detecting and avoiding obstacles s.a. small items on the ground, holes, and low passageways. The method is deployed on a differential drive robot based on the nRF52840 DK, which is already capable of mapping an unknown environment.

Several resources were made available at the beginning of this thesis. This includes a designated workstation at 313B in Gamle Fysikk, source code and reports from previous master's and specialization thesis students, and one robot based on the nRF52840 DK. Furthermore, assistance has been provided from both *Elektronikkverkstedet* and *Mekanisk Verksted* at the Department of Engineering Cybernetics.

I would like to extend my thanks to my adviser Thor Onshus for his supervision throughout this work, including great advise and many good discussions. Thanks also to fellow students Thomas Andersen and Martin Grønvold at 313B, who have been great help when discussing technical matters and for ensuring a good working environment.

Magnus Flaten Lindefjeld
Trondheim, January 2022

Summary and Conclusion

The objective of this master's thesis was to develop a method for detection and avoidance of obstacles s.a. small items on the ground, holes, and low passageways. A newly constructed embedded robot based on the nRF52840 DK was made available at the start of the project. Early on, previous functionality for exploration and mapping of an unknown environment was adapted onto the robot. Initial performance tests were conducted, which verified the robot's baseline functionality. The test revealed that while the robot mostly worked as expected, the reconstructed map did have cases of noise and distortion.

To make detection of the obstacles possible, a technical sensor solution was developed. Due to factors s.a. low power consumption, suitable measuring range, and small size, the Sharp GP2Y0A41SK0F IR range sensors were selected. Furthermore, a sensor rig was 3D printed s.t. six IR sensors could be mounted to robot. The sensor solution has experienced several discrepancies throughout the work, s.a. the sensors disturbing one another, and noise caused by items right outside the laser line of sight.

Based on available samples from the sensor solution, an obstacle detection method was developed. The method is easy to use, as there are few configuration parameters. False positive detections may sometimes occur, but this is due to discrepancies in the sensor solution, not the method itself. An obstacle avoidance method, which is initiated once an obstacle is detected, ensures that the robot does not collide. The obstacles are also added to the map, s.t. they are considered in subsequent planning iterations.

As the nRF52840 DK lacked ADCs for interfacing with the sensor solution, an external hardware module was developed. The module works as an external ADC by default, but also offers functionality s.a. switching of the sensor power supply and basic obstacle detection. An application of power supply switching, where an alternating switching pattern is utilized, completely solved the issue with sensors disturbing one another.

In the final performance test, the robot demonstrated the capability to explore and map an unknown environment containing all three types of obstacles mentioned in the thesis objective. In conclusion, the objective of the thesis, as stated in the problem description, is fulfilled. However, there are still some minor discrepancies present, s.a. poor mapping and cases of false positive detections.

Sammendrag og Konklusjon

Målet med denne masteroppgaven var å utvikle en metode for deteksjon og kollisjonsunngåelse av hindringer som f.eks. lave åpninger, hull eller små gjenstander på gulvet. En robot basert på nRF52840 DK ble nylig ferdigstilt, og ble tilgjengeliggjort for arbeidet med oppgaven. I starten av arbeidet ble tidligere utviklet funksjonalitet for utforskning og kartlegging av ukjente områder tilpasset og lastet opp på roboten. En innledende test ble gjennomført for å verifisere robotens funksjonalitet. Testen viste at det forekommer tilfeller av støy og forvrengninger i det rekonstruerte kartet, men at roboten hovedsakelig oppfører seg som forventet.

For å muliggjøre deteksjon av hindringer ble en teknisk sensorløsning utviklet. En avstandssensor som benytter infrarød stråling, Sharp GP2Y0A41SK0F, ble valgt på grunn av lavt strømforbruk, passende måleområde og liten størrelse. I tillegg ble en sensor rigg 3D-printet, slik at seks sensorer kunne festes til roboten. Sensorløsningen har opplevd flere avvik i løpet av oppgavearbeidet. Mer spesifikt inkluderer dette forstyrrelser mellom sensorer, og forekomster av støy når det er gjenstander rett utenfor siktelinjen til sensoren.

Ved bruk av måledata fra sensorløsningen kunne en metode for deteksjon av hindringer bli utviklet. Metoden er enkel å bruke, siden det er få konfigurasjonsparametere. I noen tilfeller forekommer det falske positive deteksjoner, men dette er hovedsakelig på grunn av avvik i sensorløsningen. Når en hindring blir oppdaget, igangsettes en metode for kollisjonsunngåelse slik at roboten ikke kolliderer. Hindringer blir også lagt til kartet slik at de tas hensyn til i fremtidige planleggingsiterasjoner.

Siden nRF52840 DK mangler ADC-er for å lese av målinger fra sensorløsningen, ble en ekstern maskinvaremodul utviklet. Standardfunksjonaliteten til modulen er som en ekstern ADC, men det tilbys også annen funksjonalitet slik som å skru av og på strømforsyningen til sensorene og grunnleggende deteksjon av hindringer. En anvendelse som benyttes er at strømforsyningen til sensorene blir slått av og på i et alternerende mønster. Dette løste problemet med forstyrrelser mellom sensorene.

I den avsluttende testen demonstrerte roboten evnen til å utforske og kartlegge et ukjent område hvor alle tre typer hindringer var til stede. Dermed kan det konkluderes med at målet med oppgaven, slik som det ble presentert i problembeskrivelsen, er oppnådd. Det er likevel noen mindre avvik som er uløste, noe som inkluderer støy og forvrengninger i kartleggingen, samt tilfeller av falske positive deteksjoner.

Table of Contents

Figures

Tables

Abbreviations

ADC	Analog-to-Digital Converter
ARQ	Automatic Repeat Request
BLE	Bluetooth Low Energy
CAD	Computer Aided Design
CCW	Counterclockwise
CW	Clockwise
DK	Development Kit
DOF	Degree of Freedom
EMR	Electromagnetic Radiation
FOV	Field of View
GPIO	General-purpose Input/Output
ICSP	In Circuit Serial Programming
IMU	Inertial Measurement Unit
IR	Infrared
JST	Japan Solderless Terminal
LED	Light-emitting Diode
LiDAR	Light Detection and Ranging
LLC	Logic Level Converter
MCU	Microcontroller
MOSFET	Metal–oxide–semiconductor Field-effect transistor
NMOS	N-type Metal-oxide-semiconductor
NTNU	Norwegian University of Science and Technology
PCB	Printed Circuit Board
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
RADAR	Radio Detection and Ranging
RTOS	Real-time Operating System
SLAM	Simultaneous Localization and Mapping

SMD	Surface Mount Devices
SoC	System on a Chip
SPI	Serial Peripheral Interface
TCAS	Traffic Collision Avoidance System
TH	Through-hole
TOF	Time of Flight
TWI	Two-Wire Interface
UART	Universal Asynchronous Receiver-transmitter
USB	Universal Serial Bus

1 Introduction

1.1 Problem Background and Motivation

In recent years, robotics has become more present in our daily lives, as the technology solves increasingly challenging problems. A common category within this field are differential wheeled robots, which can be adapted to miscellaneous applications despite their simple design. Some examples are vacuum cleaning, lawn mowing, and more specialized missions in industrial facilities. In these cases, the robots are equipped with a sensor system for gathering data about the surrounding environment, which is used to reconstruct a map. A common assumption for differential wheeled robots is that the environment is two-dimensional. However, while the robot may move in the horizontal plane, there may exist hazards s.a. holes, low passageways, and small items on the ground, which require additional consideration.

The objective of this thesis is to develop a method for detection and avoidance of obstacles s.a. holes, low passageways, and small items on the ground, which a robot equipped with a sensor system only providing horizontal perception would fail to handle.

The work in this thesis is part of the robot project at the Department of Engineering Cybernetics, which has been ongoing since 2004. The objective of the project is to use Simultaneous Localization and Mapping (SLAM) to reconstruct maps of unknown environments. The robot project has hosted numerous specialization and master's theses, and there have been multiple generations of differential wheeled robots used throughout the years, one of which was made available for this work. Figure 1.1 shows an example of a track the robot should be able to explore at the end of this thesis. The track includes representative obstacles s.a. a low passageway to the right, three small items to the left, and in the center, there is an edge which the robot can fall off.

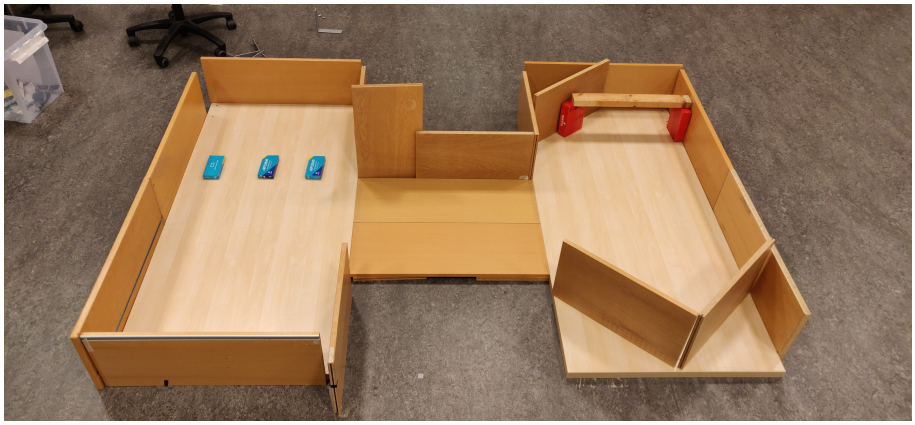


Figure 1.1: Example track which the robot should be able to explore

1.2 Previous Work

A range of different topics has been considered throughout the robot project's lifetime. Several theses were dedicated to construct new robots from scratch [1–3], and to improve or extend existing robot hardware [4, 5]. To construct a map, suitable perception sensors are required. Such sensors have not been a major focus in recent years, as IR sensors have proved reliable since the project's inception. However, work has been conducted to test other sensors s.a. LiDAR [6].

Almost all of the higher-level work relies on knowing a precise location of the robot, and the ability to specify where a robot should move. As a result, a lot of work has been put into developing and improving control and estimation methods [7–10].

A server based on Java is utilized for remote control of the robots. The development of this server started in 2016, spanning multiple theses [11–13]. More recently, a new C++ server has been in the works, as this programming language is more familiar to the students at the Department of Engineering Cybernetics. The core of the server was developed by Grindvik [14], and several students have later worked on the communication between the robots and the server [15–17], using protocols s.a. OpenThread and MQTT. Work has also been conducted to achieve a functioning SLAM algorithm on this server [18].

1.3 Contributions

This thesis makes several contributions which together gives the robot the capability to detect and avoid obstacles s.a. holes, items on the ground, and low passageways.

The first contribution is a sensor solution which provides the necessary perception s.t. these obstacles can be detected. IR sensors, similar to the ones that already exist on the robot, were found to be a suitable selection. A large part of this contribution was identifying

and reducing effects of sensor discrepancies, by designing a sensor rig addressing specific issues with the sensors.

The second contribution are methods for obstacle detection and avoidance on the robot, by utilizing data from the sensor solution. At the end of this thesis, obstacle detection can be considered completed, while obstacle avoidance can be improved further.

The third and last contribution is an additional hardware module for interfacing with the new sensor solution, as the current hardware proved insufficient. More specifically, the module can be described as an external ADC, with additional obstacle detection capabilities, reducing load on existing hardware.

1.4 Thesis Outline

Including the introduction, the thesis is divided into a total of eight chapters.

Chapter 2 – System Description reviews the theoretical baseline for the thesis, which consists of the robot system as a whole. This includes both an in-depth presentation of the robot, and descriptions of the central servers.

Chapter 3 – Making the Robot Operational goes through the customizations made for the robot to work properly. This is particularly relevant, as the exact robot model was brand new as of this semester. Additionally, testing was conducted to evaluate initial performance.

Chapter 4 – Sensor Solution explains the process of finding a technical sensor solution which makes the final objective achievable. More specifically, this includes identifying requirements, investigating and selecting suitable sensors based on said requirements, and designing a sensor rig.

Chapter 5 – Obstacle Detection and Avoidance first presents method for detection of obstacles s.a. low passageways, holes and, small items on the ground, assuming the sensor solution works as expected. Afterwards, the chapter presents a method for avoiding the obstacles once detected.

Chapter 6 – External Hardware Module contains the presentation of a hardware module which were developed in this thesis. The module works as an link between the existing hardware and the selected sensor solution.

Chapter 7 – Testing and Evaluation presents tests of the functionality developed in this thesis. This includes both testing of individual parts, and a test of the final performance which answers to the thesis objective. A discussion of the test results and the work as a whole is also conducted here.

Chapter 8 – Future Work summarizes work which was either incomplete at the end of this thesis, or which is a natural continuation of this thesis.

2 System Description

This chapter presents a baseline for the thesis, describing the existing components and functionality of the system. All relevant parts are addressed, including the robot, the two servers and in-between communication.

Firstly, an overview of the system as a whole is provided. Afterwards, the thesis reviews both the robot hardware and the source code. Lastly, the both servers are presented, mainly focusing on the Java server.

2.1 Overview

The system consists of two main components, a robot based on the nRF52840 DK, and a server based on either Java or C++. The purpose of the system is for the robot to traverse an environment which is unknown beforehand, and for the server to simultaneously reconstruct a map of said environment.

More specifically, the server makes high level decisions, which results in generating waypoints for the robot. These waypoints are either generated manually, or through a path planning algorithm. Additionally, the server reconstructs a 2D occupancy map through a Simultaneous Localization and Mapping (SLAM) algorithm, based on data from the robot.

As for the robot, the main responsibility is to navigate based on instructions from the server, gather the necessary sensor data, and send the data back to the server. The robot relies on a sensor tower with four Infrared (IR) range sensor for spatial information, and sensors s.a. an Inertial Measurement Unit (IMU) and encoders for pose estimation.

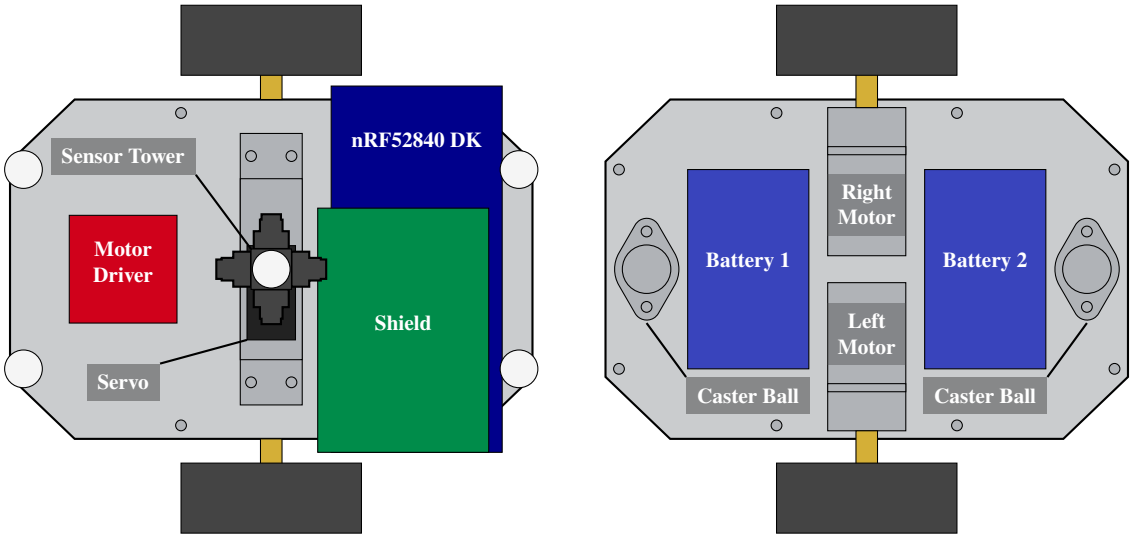
Since the robot project's inception in 2004, various robot models have been worked on. The current generation of robots was first developed by Jølsgard [3] during his specialization project in Autumn 2020, where a series of three new robots was assembled. Before the beginning of this thesis, another series of three almost identical robots was prepared. The robots are referred to by their number as follows:

- **NRF1, NRF2, NRF3:** Developed by Jølsgard
- **NRF4, NRF5, NRF6:** New robots as of this semester

The NRF4 robot was made available during this thesis work.

2.2 Robot Hardware

In this section, the NRF robot is presented in detail. At its core, the robot consists of a metal chassis with top and bottom layers, a pair of wheels and two caster balls. Figure 2.1 shows the hardware layout of the NRF robot from above. In addition to what is shown in the figure, an Inertial Measurement Unit (IMU) is located under the bottom layer, and an OLED Display is located on the shield.



(a) Top layer **(b)** Bottom layer

Figure 2.1: Hardware layout of the NRF robot

As the figure shows, the robot consists of various electrical components. An overview of these is given in Table 2.1. For the components which are the most relevant for this thesis work, a more in-depth review is provided.

Table 2.1: Hardware overview for the NRF robot

Component	Quantity	Description
Microcontroller	1	nRF52840 Development Kit
DC motor	2	Machifit 25GA370 DC 12V 110 RPM
Motor driver	1	L298 H-Bridge Dual Bidirectional Motor Driver
Battery	2	Ansmann 10.8V, 2.6Ah
OLED Display	1	PEMENOL 128 x 64 LCD with SSD1306 driver
Inertial Measurement Unit (IMU)	1	ICM20948
Servo motor	1	S05NF STD
IR range sensors	4	Sharp GP2Y0A21YK

nRF52840 Development Kit

The nRF52840 Development Kit [19] is single board development kit from Nordic Semiconductor based on the nRF52840 SoC, which has an Arm Cortex-M4 processor. The board supports various peripherals, s.a. Bluetooth Low Energy (BLE), Bluetooth mesh, Thread Zigbee, 802.15.4, ANT and 2.4 GHz. Furthermore, the board offers a range of General-purpose Input/Output (GPIO) pins for interfacing with other hardware components. Lastly, the board is equipped with an on-board SEGGER J-Link debugger, s.t. only an USB cable is necessary for flashing and debugging.

nRF Shield

A hardware shield [20] is stacked on top of the the nRF52840 DK. The shield provides peripheral connectors to other components s.a. the IR sensors, IMU, servo and OLED display. Furthermore, the shield includes a voltage regulator, for converting the battery voltage down to a 5 V logic level. However, this regulator is not in use as of the beginning of this thesis, and a regulator on the motor driver is used instead.

DC Motors with Rotary Encoders

Two 12 V DC motors [21], which are mounted on the side of the robot, are used to control robot movement. The motors are connected to a motor driver, which feeds each motor with the desired voltage based on a Pulse Width Modulated (PWM) input signal. Additionally, each motor is equipped a quadrature encoder, which yields both a velocity feedback and direction of rotation.

The NRF4, NRF5 and NRF6 robots are equipped with a slightly different pair of motors than the NRF1, NRF2 and NRF3 robots [22]. While interfacing with the motors works in the same manner, the new motors have a higher amount of encoder ticks per revolution.

Inertial Measurement Unit

An IMU is a device that consists of an accelerometer, which measures specific force, and a gyroscope, which measures angular velocity. For 9DOF IMUs, s.a. the onboard ICM20946 [23], a magnetometer is also included. Mainly, data from the device is used as input for pose estimation.

Servo Motor

A servo motor is a rotary actuator which allows for control of angular position. For the S05NF STD servo [24], the position is specified by a PWM input signal, and can be con-

trolled across a 180° interval. Contrary to the DC motors, the servo offers no feedback signal.

IR Range Sensors

An IR range sensor is a sensor which uses rays in the infrared spectrum to measure distance to an object. The robot consists of four Sharp GP2Y0A21YK0F sensors [25], which measure distances between 10 cm to 80 cm. The sensor outputs an analog voltage signal, with a direct relation to the measured distance. Figure 2.2 shows the distance measurement characteristic of a Sharp GP2Y0A21YK0F sensor. The graph is nonlinear due to the manner of which distance is measured. The sensor uses the concept of triangulation, and as a result, the sensor is more precise the closer a measured object is to the sensor.

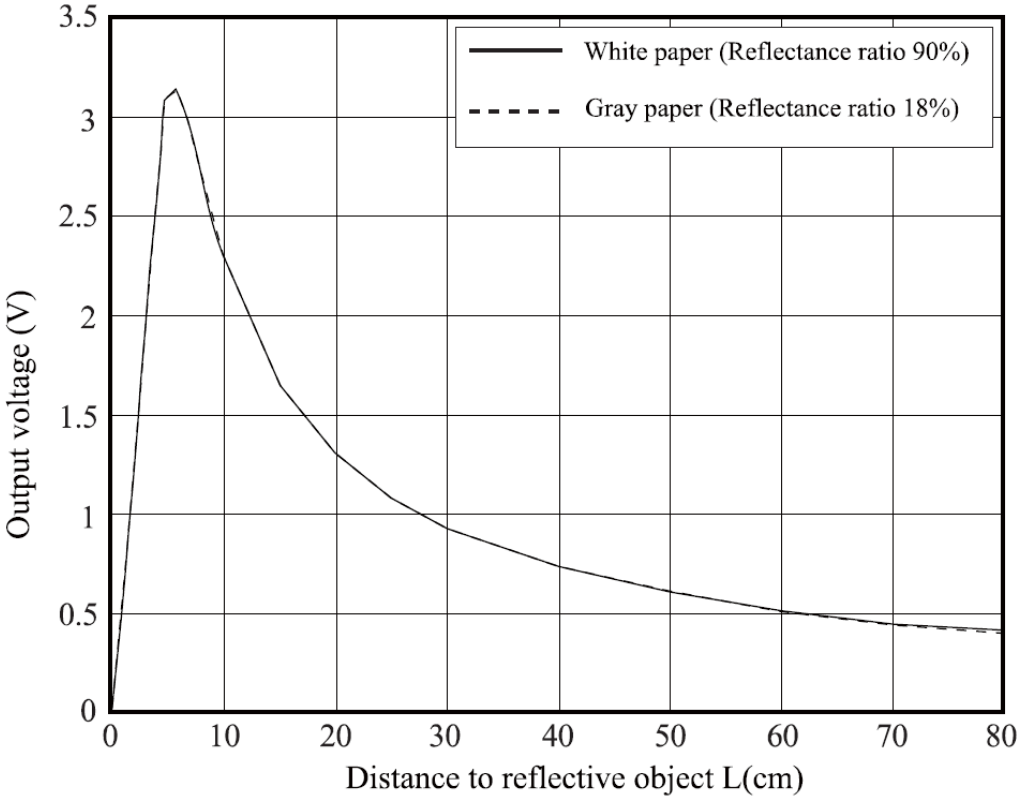


Figure 2.2: Distance measurement characteristic of the IR range sensor GP2YA21YK0F by Sharp. Adapted from Sharp GP2YA21YK0F datasheet [25].

On the robot, four Sharp IR range sensors are mounted on a rig with a 90° angle between them, such that all sensors point out in the horizontal plane. The sensor rig is further mounted on top of and rotated with the servo motor. This configuration gives the robot the necessary spatial information for mapping, and is referred to as the sensor tower.

2.3 Robot Source Code

The current version on the robot source code is fetched from Jølsgard [3], who adapted Stenset's [26] code to the NRF robot. Table 2.2 gives a overview over the folder layout of the source code.

Table 2.2: Source code folder layout

Folder	Description
`ble_communication`	Functionality for communication with the server through the BLE protocol
`config`	Configuration files
`drivers`	Drivers for interfacing with the hardware components in Table 2.1
`software`	Main source code which implements the robot functionality. The code is structured in a task-based fashion with FreeRTOS.
`test_functions`	Small FreeRTOS tasks for interfacing with and testing individual hardware components

The code in the software folder is based on FreeRTOS [27]. FreeRTOS implements a Real-time Operating System (RTOS) for microcontrollers. Compared to other hardware that runs software with real-time requirements, microcontrollers have limited memory and processing speed. Hence, FreeRTOS only implements core functionality.

FreeRTOS structures the code in a task-based manner, and hence, the code in the software folder consists of multiple tasks. A review of these tasks is provided in the next paragraphs.

User Task

The user task (`user_task`) is mainly responsible for initializing the hardware drivers. This includes the DC motors, servo, encoders and IMU. Additionally, some miscellaneous operations are conducted in this tasks as well, e.g. testing waypoint tracking when the robot is disconnected from the server.

Position Estimator Task

The position estimator task is responsible for estimating position based on sensor input from the encoders and the IMU. Two separate versions exist:

- `vMainPoseEstimatorTask`: Kalman filter with a constant acceleration model. Developed by Leithe [28] during his master's thesis in 2019.

- `vMainNewPoseEstimatorTask`: Extended Kalman filter with a model of a differential drive robot. Developed by Berglund [9] during his specialization project in 2020.

Berglund's version is the one currently in use.

Motor Speed Controller Task

The motor speed controller task (`vMotorSpeedControllerTask`) runs a PID loop to control the speed of the DC motors, using feedback from the encoders. The controller ensures each motor rotates at an almost exact reference speed, and reduces the effects of outside disturbances. Additionally, the controller also reduces any turning when the robot is tasked with driving forward in a straight line.

Pose Controller Task

The pose controller task (`vMotorSpeedControllerTask`) is responsible for bringing the robot from its current position to a specified target position, by both controlling the robot's heading and distance to the target. After a control iteration, the output signals, which consist of speed references for the two motors, are fed to the speed controller. The task is structured as a state machine, with different objectives based on the current pose, and the states are:

- `moveForward`: The robot moves directly towards the target position, where the input control signal is the remaining distance. A heading controller aids the robot in maintaining a straight path.
- `moveStop`: The robot has reached the target position, and is not moving.
- `moveClockwise/moveCounterClockwise`: The robot is not currently directed towards the target position, and the error in heading is above a certain threshold. Therefore, the robot turns a suitable direction around its center of area.

Sensor Tower Task

The sensor tower task (`vMainSensorTowerTask`) handles operation of the sensor tower. More specifically, this entails fetching measurements from the IR sensors and controlling the movement of the servo. The sensor data is also reported to the server together with the servo angle and changes in the robot position.

Communication Task

The communication task (`vMainCommunicationTask`) handles incoming messages from the server after connection has been established. Depending on message type, the robot

takes appropriate action. Most importantly, the task receives target waypoints which are relayed to the pose controller.

Other Tasks

There are a few other minor tasks, which runs in the background.

- `display_task`: Interface with the OLED display
- `microsd_task`: Interface with the micro SD card
- `vARQTask`: Required for BLE communication to function. ARQ means Automatic Repeat Request.

2.4 Java Server

The Java server is utilized for remote control of the NRF robots. The development started in 2016, and the version available today is the result of several theses [11–13].

2.4.1 Server Functionality

The server is a remote interface to the robot. It receives pose estimation data and IR sensor measurements, and by employing a SLAM algorithm, a 2D occupancy map is reconstructed. The map consists of voxels, which essentially are square tiles representing space. The voxels are divided into three types, either free, occupied or unknown. Furthermore, the server highlights free voxels within a certain range of occupied areas. These voxels are referred to as restricted, and are used for path planning purposes. While the robot is mapping an environment, the robot pose is displayed within the map.

Upon a new robot connection to the server, the user specifies an initial position and orientation. The user also selects which mode to use. There are two separate modes:

- **Manual Waypoints:** The user manually inputs a target waypoint by specifying a (x, y) coordinate in cm. Immediately afterwards, the waypoint is sent to the robot.
- **Automatic Waypoint Generation:** Waypoints are generated through a path planning scheme, which functionality exists on the server. In this mode, the server regularly generates and sends waypoints until exploration is completed, at which point planning stops automatically.

2.4.2 Communication between Server and Robot

The communication protocol which connects the server to the robot is Bluetooth Low Energy (BLE). Figure 2.3 shows the devices involved. From the robot, the onboard BLE

module on the nRF52840 DK communicates with a nRF51 Bluetooth Dongle. This dongle is connected to a USB port on a computer running the Java server.

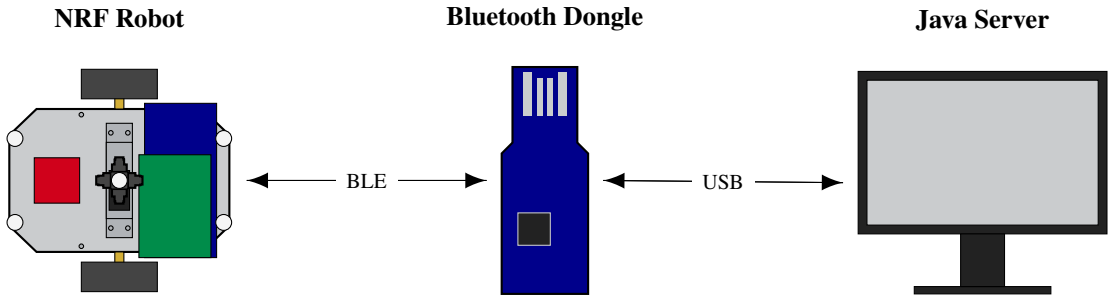


Figure 2.3: Communication interface between the Java server and NRF robot

For communication, the robot and server uses a custom set of messages. Table 2.3 provides an overview of the types of messages, and a short description of their content.

Table 2.3: Message overview for Java server communication

Sender	Message Type	Variant	Description
Robot	Handshake		Initial message with information about the robot
	Update		Robot position, orientation, sensor tower heading, sensor values
	Status	Idle	Notification after reaching a waypoint
Server	Order		Target waypoint to the robot
	Status	Handshake Confirmed	Confirmation of Handshake message
		Pause Robot	Pause robot while maintaining connection
		Unpause Robot	Unpause if previously paused
		Robot Finished	Disconnect from server

2.5 C++ Server

A new C++ has been in the works since 2019, which will eventually replace the Java server. The core functionality of the server was developed by Grindvik [14] in 2019. Since then, several master's and specialization theses have extended the server, with functionality s.a. SLAM [18] and communication with robot (OpenThread and MQTT) [15–17]. As of now however, no reliable way of communication with the NRF robots exists, hence the server is not adapted in this thesis.

3 | Making the Robot Operational

In this chapter, all the steps to make the NRF robot operational are explained. From previous work, the robot should be able to autonomously map an unknown environment in cooperation with the server. Furthermore, mapping should be accurate, which means the performance of the estimator and controllers need to be verified. Since the NRF4, NRF5 and NRF6 robots are new as of this semester, inspection of all hardware and software is required. This process was conducted in cooperation with two other students, Skinstad and Andersen, also working on the NRF robot.

The chapter is structured as follows. Firstly, the robot dimensions are updated, the sensors are calibrated, and the hardware components are inspected. Afterwards, the PID controllers are tuned, and lastly, three initial performance tests are conducted.

3.1 Robot Dimensions

The first step of making the robot operational was to measure its dimensions. This is important as they are used for pose estimation. The dimensions are shown in Table 3.1, and are slightly different from the previous generation of robots.

Table 3.1: Dimensions of all NRF robots

Property	NRF1-3	NRF4-6
WHEELBASE_MM	157	170
ROBOT_TOTAL_WIDTH_MM	186	196
ROBOT_TOTAL_LENGTH_MM	194	194
WHEEL_DIAMETER_MM	65	67

3.2 Hardware Components

The next step of making the robot operational was to calibrate the sensors and inspect the other hardware components.

3.2.1 IR Sensors

The IR sensors equipped on NRF4, NRF5 and NRF6 were all calibrated. To calibrate an IR sensor, a way of converting the output voltage from the sensor into a distance is required. The most common approaches are lookup tables and regression analysis, and as for this work, the latter approach is used.

Regression Model

To model the nonlinear relation between output voltage and measured distance (see Figure 2.2), the power law model in Equation (3.1) is considered.

$$d(U) = \beta U^{\alpha} \tag{3.1}$$

Here, d is the measured distance, U is the output voltage in mV, and β and α are parameters. A regression analysis can identify β and α. First, the power law model is converted into the linear regression model in Equation (3.2),

$$y = ax + b + \epsilon \tag{3.2}$$

where ϵ is an error term. The relationship between power law model in Equation (3.1) and the linear model in Equation (3.2) is given in Equation (3.3).

$$y = \ln d(U), \quad x = \ln U, \quad a = \alpha, \quad b = \ln \beta \tag{3.3}$$

Data Acquisition Method

In Leithe [28], data for calibration was acquired through an automatic test setup, where the sensors where calibrated while mounted on the robot. However, as encoders and other components on the NRF robot were not yet ready, this method was not applied. Furthermore, a criticism of Leithe's method is that errors in the position estimation and encoder measurements will permeate into the IR calibration.

Instead, a manual test setup was prepared to acquire data for the regression analysis. On a flat surface next to a wall, lines were drawn 5 cm apart, starting from 10 cm and ending at 80 cm from the wall. For each position, a dataset of 200 measurements was recorded.

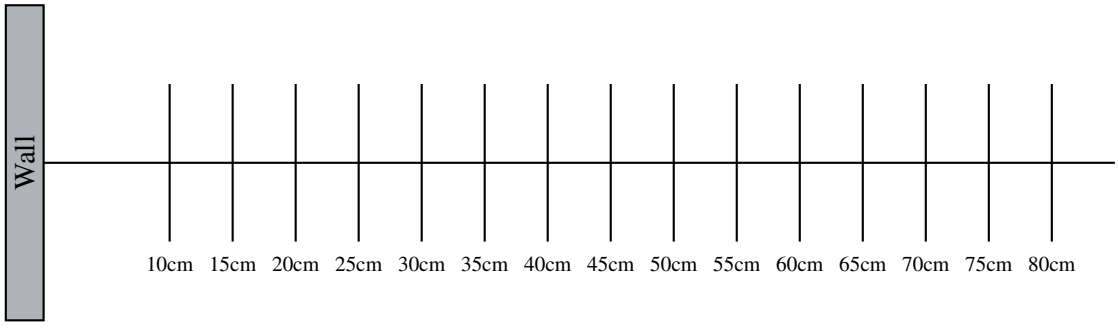


Figure 3.1: Manual IR sensor calibration setup

Sensor Parameter Identification

Lastly, a regression analysis was conducted with the acquired datasets to identify the model parameters. The average of each dataset was used as an input data point in the method. The resulting parameters for NRF4, NRF5 and NRF6 robots are given in Table A.1 in Appendix A. The table also includes the R value from the regression. Note that the parameters converts the voltage in mV to distance in mm.

Furthermore, to confirm that the model accurately represents the input data, the input data points and the resulting model for Sensor 1 on NRF4 are displayed in Figure 3.2. As the figure shows, the function accurately models the input data points with only minor deviations.

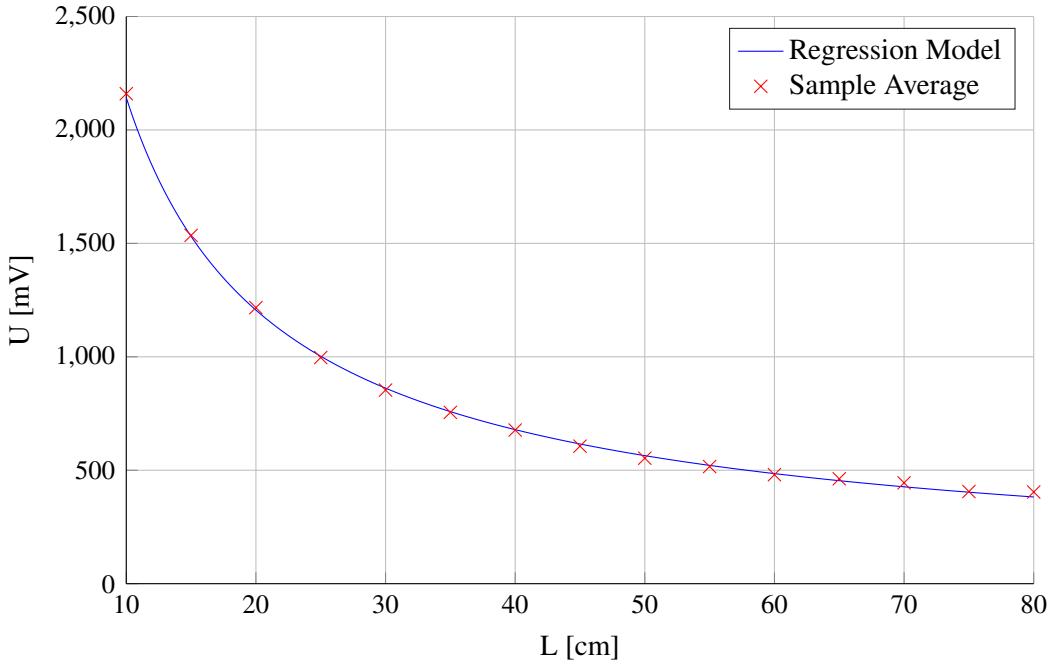


Figure 3.2: Result of function fitting for Sensor 1 on the NRF4 robot. The plot shows input data points from the averages of 200 samples each, and a variant of the power-law model in Equation (3.1), with specific parameters α and β.

3.2.2 Rotary Encoders

The rotary encoders were calibrated to ensure the robot traverses the same distance as it estimates. There are two relevant configuration parameters:

- `ENCODER_TICKS_PER_ROT`: Number of encoder ticks per one revolution of the wheel
- `WHEEL_FACTOR_MM`: The ratio between `WHEEL_CIRCUMFERENCE_MM` and `ENCODER_TICKS_PER_ROT`

For calibrating `ENCODER_TICKS_PER_ROT`, a line test was conducted. The exact procedure is as follows. Firstly, prepare a reference distance, which in the conducted test was 3.5 m. Secondly, drive the robot in a straight line of equal length to the reference distance. After the robot has stopped, record how far the robot traveled. Note both the estimated distance L_{est} and measured distance L_m. Thirdly, update the parameter `ENCODER_TICKS_PER_ROT` according to Equation (3.4). Lastly, verify the distance by running the test once more. L_{est} and L_m should now be equal.

$$\texttt{ENCODER\_TICKS\_PER\_ROT} := \frac{L_{est}}{L_m} \times \texttt{ENCODER\_TICKS\_PER\_ROT} \tag{3.4}$$

The resulting values of `ENCODER_TICKS_PER_ROT` for the NRF robots are given in Table 3.2. The NRF1-3 values are included to emphasize the difference between the two types of DC motors.

Table 3.2: Encoder ticks per revolution value for each of the NRF robots

Property	**NRF1-3**	**NRF4**	**NRF5**	**NRF6**
`ENCODER_TICKS_PER_ROT`	224	817	782	800

3.2.3 Servo Motor

In the robot source code accessed at the start of this work, the conversion from degrees to PWM input signal to the servo was not accurate. The conversion is given by Equation (3.5),

$$\Delta T = \Delta T_0 + k\theta \tag{3.5}$$

where ΔT is the pulse width, ΔT_0 is the pulse width at 0°, θ is the angle, and k is a constant.

The servo can only rotate at an interval of 180°. When tasked with rotating from 0° to 90°, as was the case in `vMainSensorTowerTask`, the servo reached a dead end. To solve this issue, the zero servo angle was shifted with $-90°$ by modifying ΔT_0.

3.2.4 Inertial Measurement Unit (IMU)

No particular changes was done to the IMU. The robot employs an automatic calibration scheme at startup, which minimizes the bias in both the accelerometer and gyroscope measurements.

3.3 PID Controllers

In total, there are five PID-controllers that were tuned for smooth control of the NRF4, NRF5 and NRF6 robots. These are:

- Speed controller, left motor
- Speed controller, right motor
- Pose controller, distance controller
- Pose controller, heading controller (while idle)
- Pose controller, heading controller (while driving)

The tuning was conducted by Andersen. See his specialization project report [29] for more information.

3.4 Testing of Initial Performance

After inspection of hardware components and tuning of PID controllers, several tests were conducted to verify the system performance. All the tests were conducted with the assistance of the Java server.

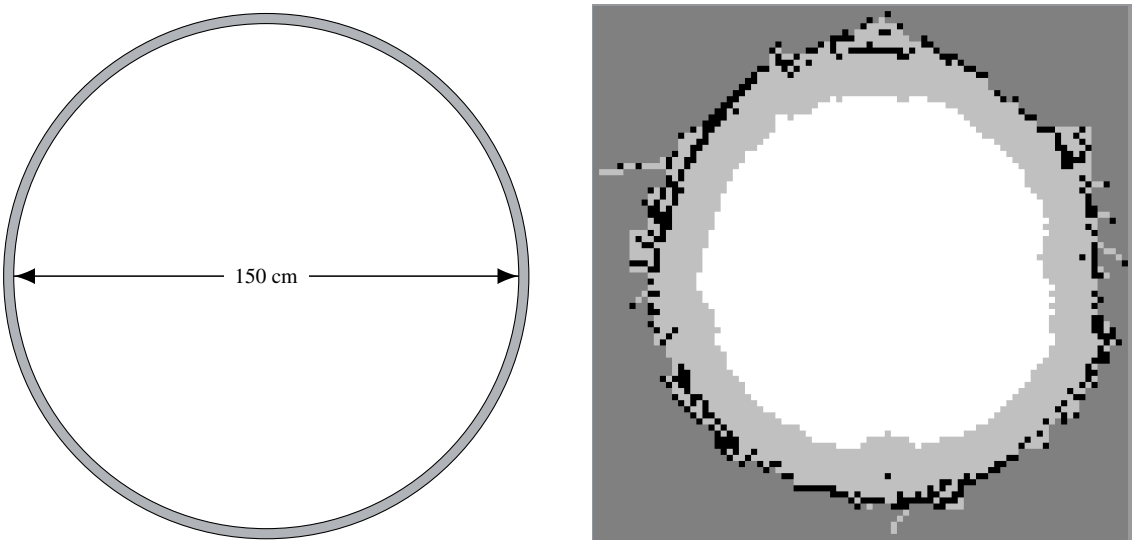
3.4.1 Test 1: Circular Track

The first performance test was conducted on a circular track. Figure 3.3a shows a sketch of said track, which can be characterized by its simple topology. The robot was tasked with navigation and mapping of the track, and no information about the track was provided beforehand.

The test was set up with the following configuration:

- **Server:** Java server
- **Mode:** Automatic Waypoint Generation
- **Voxel Size:** 2 cm
- **Robot:** NRF4
- **Starting Position:** Arbitrary

Figure 3.3b shows the occupancy map of the circular track upon completion. All in all, the robot performed well, experiencing no navigation problems. As for mapping, the shape of the wall, i.e. black voxels, is similar to the original, round shape of the circular track. However, there are some additional noise included.



(a) Sketch of the track **(b)** Mapping by the NRF4 robot

Figure 3.3: A sketch of the circular track with actual sizes, compared to the map constructed by the NRF4 robot after traversal.

3.4.2 Test 2: Maze

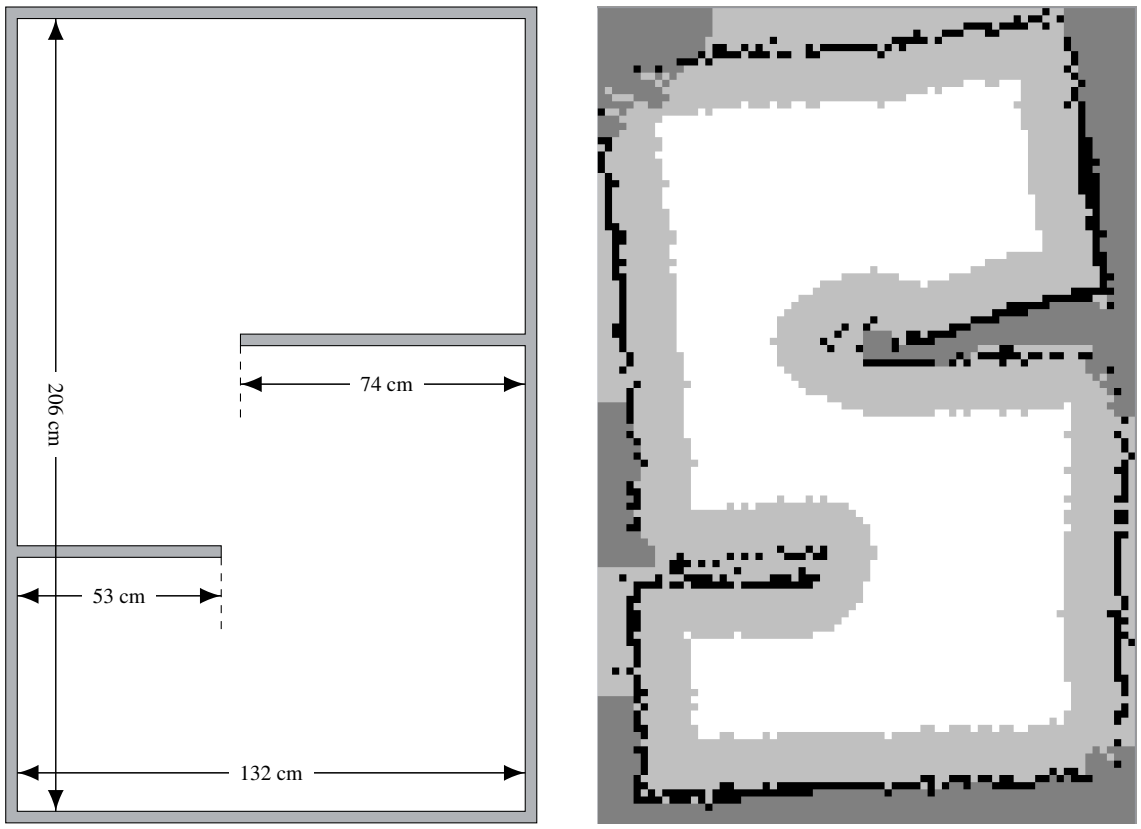
The second performance test was conducted on a larger and more topologically advanced track, referred to as the maze. Figure 3.4a shows a sketch of the maze. Once more, the robot was not provided any information about the track prior to the test.

The test was set up with the following configuration:

- **Server:** Java server
- **Mode:** Automatic Waypoint Generation
- **Voxel Size:** 2 cm
- **Robot:** NRF4
- **Starting Position:** Lower left corner, orientation towards the right

Figure 3.4b shows the resulting map after the robot has completely navigated and mapped the maze. While the maze is more topologically advanced than the circular track, the robot had few problems with navigation. However, the robot did require more time for exploration.

Once again, the robot experienced some of the same issues with a noisy representation of the walls. However, the representation of straight walls seems to be more consistent than curved ones. Furthermore, a second observation is that the upper part seems distorted when compared to the lower part.



(a) Sketch of the track **(b)** Mapping by the NRF4 robot

Figure 3.4: A sketch of the maze with actual sizes, compared to the map constructed by the NRF4 robot after traversal.

3.4.3 Test 3: Square

The last performance test was a square test to to verify the robot movement by utilizing an external tracking system. As the previous test revealed, the pose estimation may not be entirely accurate, and as a result, external tracking data should provide a better tool for evaluation.

The test was conducted in the 3D Movement Lab (B333) at NTNU, which has installed a movement tracking system from OptiTrack [30]. The robot was tasked with driving a path shaped as a square with 1 m side lengths. In total, one clockwise and one counterclockwise square test were conducted.

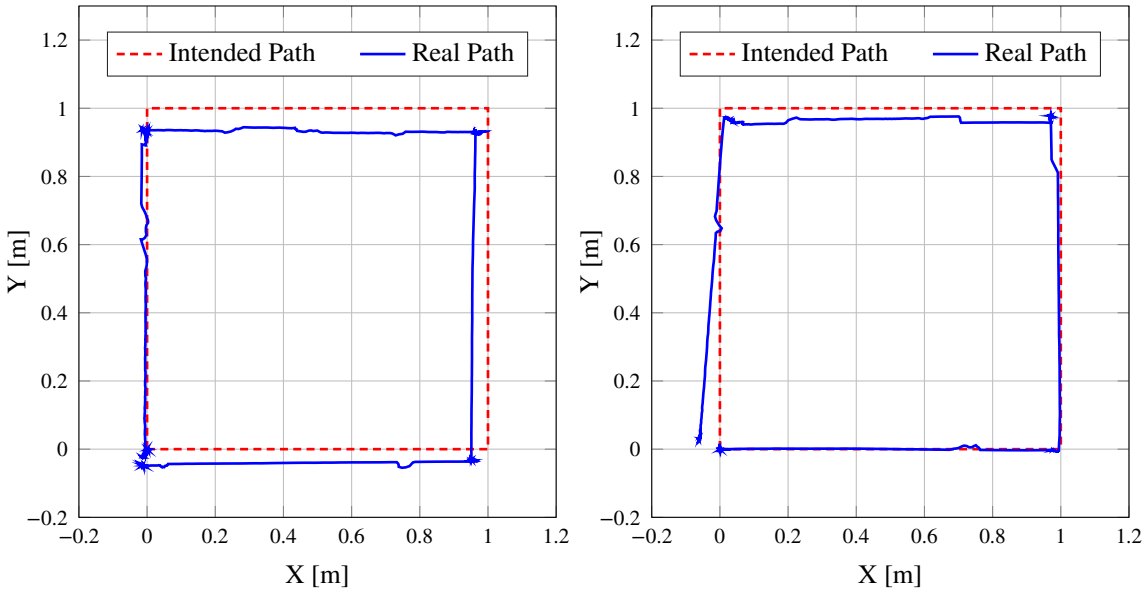
The test was set up with the following configuration:

- **Server:** Java server
- **Mode:** Manual Waypoints
- **Robot:** NRF4
- **Starting Position:** Arbitrary, as the coordinate system is shifted and rotated later, s.t.

the first line segment is parallel to the intended path. However, while the CCW-test starts by the robot moving forward, the CW-test starts by the robot turning 90°.

- **Tracking System:** Motive (software) with OptiTrack cameras

The results of the CW and CCW tests are shown in Figure 3.5. To evaluate the performance, the primary properties in consideration are the lengths of the line segments and the degree of turning.



(a) Clockwise **(b)** Counterclockwise

Figure 3.5: The result of the 1 m square test, including both the clockwise and the counterclockwise paths. The real paths are found by utilizing a camera tracking solution from OptiTrack [30].

In the CW-test, each turn is consistently close to 90°. However, there are discrepancies in the lengths of line segments, which lead to a deviation of more than 5 cm from the intended path. Eventually, the robot stops at a distance of 4.9 cm from the starting position.

In the CCW-test, the line segments are closer to 1 m. As a result, the real path tracks the intended path closely. However, after the last turn, the orientation is slightly inaccurate. Eventually, the robot stops at a distance of 6.9 cm from the starting position.

3.4.4 Discussion

The three initial performance tests have revealed a few discrepancies. Test 1 (see Figure 3.3b) showed that the constructed map consists of noisy representations of the walls. Test 2 (see Figure 3.4b) revealed distortion in the constructed occupancy map. Test 3 (see Figure 3.5) confirmed that the robot does not move exactly as intended.

A likely source of error is inaccurate pose estimation. Firstly, the distortion in Test 2 and the exaggerated turning in Test 3 may indicate bias in the estimated attitude. As the gyroscope, which is the main sensor for attitude estimation, only measures angular velocity, attitude bias may occur eventually. Secondly, another source of error may be wheel slip, which has been observed during testing. Wheel slip will result in the robot overestimating the traveled distance.

A less likely source of error is the IR sensor measurements. During calibration, the sensors were non-moving, and the ray and the measured wall were perpendicular. Hence, when deployed on the sensor tower, the IR sensors may experience other discrepancies affecting the measurements.

In summary, the tests verify that the robot performs reasonably well, i.e. the robot is now operational. However, there are clear deviations from an ideal behavior. While these deviations may jeopardize the final performance of the system, further optimization of current performance is not prioritized.

4 | Sensor Solution

This chapter presents a technical sensor solution which makes detection of obstacles possible. As of the start of this work, the robot collects spatial data about the surrounding environment through four IR range sensors, which provides perception in the horizontal plane. However, to achieve awareness of obstacles at varying heights, the hardware will be extended with new sensors.

The chapter is structured in the following manner. Firstly, a requirement analysis is conducted. Secondly, different sensor technologies are reviewed, products realizing these technologies are evaluated, and a specific sensor is selected. Thirdly, a sensor rig is designed based on the selected sensor. Lastly, the sensor solution is evaluated through proof of concept testing, as later work depends on a working solution.

4.1 Requirement Analysis

In this section, requirements of the sensor solution are identified. The analysis is divided into two parts. Firstly, the thesis presents a functional specification of how the obstacle detection system as a whole is expected to work. Afterwards, the thesis identifies all factors which should contribute to the final design of the sensor solution.

4.1.1 Functional Specification

The functional specification describes how the system is expected to work. Here, the obstacle detection and avoidance system as a whole is considered. However, some design factors of the sensor solution derive from the specification. The functional specification is as follows:

1. The system should detect non-moving obstacles in the robot's path which would prevent the robot from continued traversal. This includes all of the following:
    - Obstacles on the ground
    - Overhead obstacles
    - Negative obstacles i.e. holes

2. The system should not impose new speed constraints on the robot, and obstacle detection should work well when traversing at maximum speed.
3. The system should change exisiting functionality in two ways:
    - Upon detection of an obstacle, control towards the current waypoint is aborted.
    - Previously detected obstacles should be considered when planning paths.
4. When an obstacle is detected, the robot should not collide.

4.1.2 Product Design Factors

After presenting the functional specification, design factors of the sensor solution can be identified. While most of the factors stems from a practical point of view, some are derived from the functional specification. Table 4.1 provides all the design factors that where found.

Table 4.1: Design factors for the sensor solution

Factor	**Explanation**
Power Consumption	The robot has limited battery capacity ($2 \times 2600\,\text{mA}$), Therefore, the power consumption should be low.
Weight	The solution should be lightweight. High weight leads to several issues s.a increased DC motor power consumption and friction on the caster balls.
Size	The added hardware must correspond to the current robot size ($194\,\text{mm} \times 196\,\text{mm} \times 178\,\text{mm}$).
Cost	The total of cost of the solution should not be too high. Existing robot hardware can be used as a reference.
Obstacle Detection	The sensor must be able to detect all three types of obstacles described in the first functional requirement.
Precision	Sensor data with a large variance can lead to false positives in the collision detection, and will generally result in a less accurate detection system. Hence, it is essential that the variance is low.
Data Acquisition Speed	To fulfill the first and second functional requirements, the solution must provide enough data.
Measuring Range	To fulfill the fourth functional requirement, the selected sensors must have a suitable measuring range, s.t obstacles can be detected in time.
Hardware Constraints	The nRF52840 DK should be able to interface with all the hardware components in the solution, without experiencing any issues which causes the onboard program to fail.

4.2 Review of Range Sensors

To select suitable sensors, a review of existing technologies was conducted. More specifically, the category of interest is range sensors, which measure the distance to the surrounding environment from the sensor origin. The sensors are non-intrusive, as there are no contact with the objects which are perceived. Instead, various types of waves are utilized, the most common being mechanical waves s.a ultrasound, or electromagnetic radiation (EMR). In this section, information was gathered from Springer Handbook of Robotics [31].

4.2.1 EMR-based Sensors

Range sensors based on EMR normally utilize light in either the infrared or the visible spectrum, and sometimes low-frequency radiation s.a. radio waves. Generally, higher frequency allows for higher precision, at the cost of measuring range. Some of the common types of sensors within this category are IR-sensors, LiDAR, RADAR and depth cameras. In broader terms, there are two fundamental techniques for range detection, i.e. triangulation and Time of Flight (TOF).

Triangulation

Triangulation is a way of range sensing where the angle of a signal determines the range. Stereo cameras use this technique, by matching features from slightly different viewpoints. However, as these devices require the capacity for image processing, the more lightweight IR sensor will be used to explain triangulation. This sensor consists of a emitter that emits bursts of IR waves, and a detector that detects waves reflected off objects. The method for calculating range is given in Equation (4.1),

$$\tan\theta = \frac{d}{b} \tag{4.1}$$

where b is the baseline distance between the emitter and detector, d is the distance to the object which waves are reflected off, and θ is the angle between the two segments. Figure 4.1 shows a illustration of how this technique works. Essentially, θ is indirectly measured, b is known, and hence, d can be calculated.

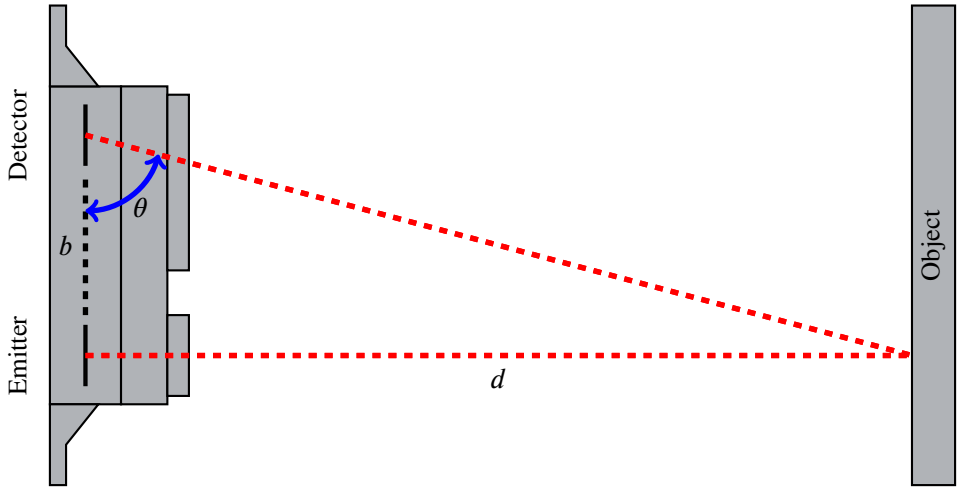


Figure 4.1: Sketch of triangulation with an IR sensor

Triangulation-based sensors have a rapidly decreasing sensitivity when increasing the distance to a perceived object. Hence, the sensors are unsuitable for measuring far-away objects. Conversely, triangulation-based sensors are often very precise at close range.

Time of Flight

Time of Flight (TOF) is a sensing method that determines range by measuring the time it takes for a signal to travel from a sensor to an object and return. Hence, similar to the triangulation method, the sensors consist of a emitter and detector. For direct TOF sensors, the travel time is measured by a precise chronometer. Afterwards, the distance is calculated according to Equation (4.2),

$$d = \frac{ct}{2} \tag{4.2}$$

where d is the one way distance, c is the speed of the signal, and t is the measured time of travel. For EMR-based sensors, c is the speed of light, and hence, the measured time interval is particularly short. Therefore, the measured distance is usually less precise than for triangulation based sensors. However, the technique allows for a longer measuring range.

4.2.2 Ultrasonic Sensors

Ultrasonic sensors are another type of range sensors. For this category of sensors, the most common technique is TOF. For usage in obstacle detection, the time until the first detected echo is always considered. In some ways, ultrasonic sensors function in a similar fashion to EMR-based TOF sensors. However, there are notable differences due to the properties of sound waves.

Firstly, the speed of sound, which is 343 m s^{-1} at 25°C, is significantly slower than speed of light. Hence, there are less strict timing requirements, which allow for more precise range measurements. Secondly, as sound waves cannot be concentrated in the same way as EMR, ultrasonic has a wider measuring angle. As a result, the signal weakens further away from the source, which limits the measuring range.

Lastly, sound waves are reflected cleanly off smooth surfaces. Consequently, when the waves reflect of a surface from a wide angle, no echo returns to the sensor. To consider an example, Figure 30.5a in Springer Handbook of Robotics [31] shows that most of the signal is reflected when the angle between the waves and the surface norm is less than 15°.

4.2.3 Product Investigation

An investigation was conducted into identifying specific products that realize the sensing technologies described above. Table 4.2 gives a summary of the sensors that were selected, and includes properties s.a. measuring range and sample frequency. The products are described in more detail below.

Table 4.2: Overview of range sensors

Sensor	Type	Technique	Measuring Range	Frequency	Consumption
GP2Y0A51SK0F			2 cm - 15 cm	60 Hz	12 mA
GP2Y0A41SK0F			4 cm - 30 cm	60 Hz	12 mA
GP2Y0A21YK0F	IR	Triangulation	10 cm - 80 cm	26 Hz	30 mA
GP2Y0A02YK0F			20 cm - 150 cm	26 Hz	33 mA
GP2Y0A710K0F			100 cm - 550 cm	60 Hz	30 mA
LiDAR Lite v3	LiDAR	TOF	n/a - 40 m	650 Hz	135 mA
RPLIDAR A1M8	2D LiDAR	Triangulation	15 cm - 12 m	8000 Hz	400 mA*
HC - SR04	Ultrasonic	TOF	2 cm - 4 m	40 Hz	15 mA

*Motor: 100 mA, Scanner: 300 mA

Sharp IR Range Sensors

Sharp has a product lineup of triangulation-based IR range sensors. The sensors differ slightly by measuring range, sample frequency and power consumption (see Table 4.2). These sensors are familiar to the robot project, as the GP2Y0A21YK0F is currently utilized on the NRF robots. See more at the product page [32].

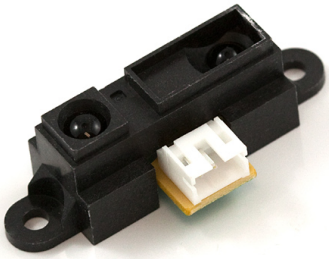


Figure 4.2: Sharp GP2Y0A21YK0F

LiDAR Lite v3

LiDAR Lite v3 is a range sensor from Garmin. LiDAR is an abbreviated form of Light Detection and Ranging. The device works by emitting a concentrated laser signal, and measuring the TOF of the signal. The sensor has a higher frequency than the Sharp sensors, with 650 Hz in fast mode, and the power consumption is 135 mA during continuous operation. For communication, the sensor offers two alternatives, which are I2C and PWM. See more in datasheet [33].

In a master's thesis from 2018, Jensen [6] employed the LiDAR Lite v3 on a robot, and compared the sensor to the GP2Y0A21YK0F IR sensors. The study found that the IR sensors produced a more precise map.

Figure 4.3: LiDAR Lite v3

RPLIDAR A1M8

The RPLIDAR A1M8 is a laser scanner from Slamtec which rotates on a motor, and conducts a full 360° scan. While the device is advertised as a LiDAR, the rangefinding technology is triangulation. The device is quite large, with dimensions 98.5 mm×70 mm×60 mm,

and weights 170 g. While the power consumption is high, the device offers a high sample frequency. The communication interface is UART. See more in datasheet [34].

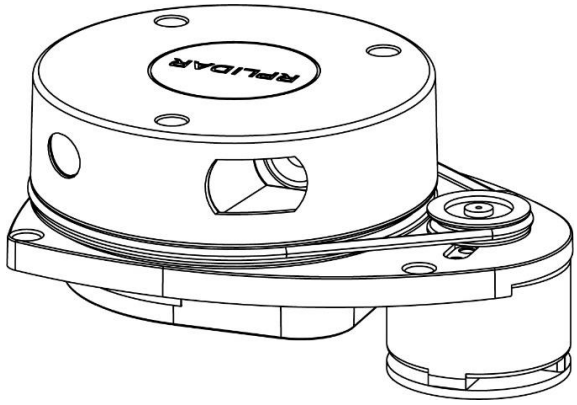


Figure 4.4: RPLIDAR A1M8

HC - SR04 Ultrasonic Sensor

The HC - SR04 is an ultrasonic sensor based on the TOF principle. The device is of a comparable size to the Sharp sensors, and consumes a similar amount of power. The sensor has two GPIO-pins of relevance, one input pin that triggers a burst of sound waves, and an output pin that signals when the sensor detects echo. See more in datasheet [35].

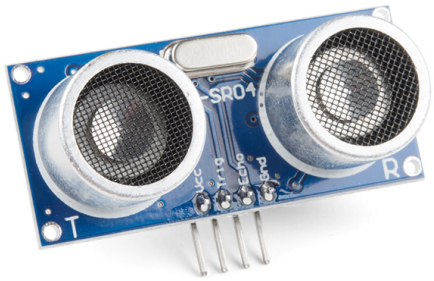


Figure 4.5: HC-SR04

4.3 Range Sensor Selection

After both the requirement analysis and an investigation into range sensing are conducted, a specific range sensor can be selected. The evaluation of the sensors is purely based on the design factors presented in Section 4.1.2.

The LiDAR Lite v3 main advantages is the power consumption, which is quite low when also considering the high sample rate. However, the main drawback is due to the TOF

ranging technique, which leads to less precise measurements than for other sensors considered. Furthermore, this sensor is more suited for long range detection, than the shorter range which is required here.

The RPLIDAR A1M8 is based on triangulation, and therefore, the sensor is more precise. Furthermore, as a motor is included, the sensor should provide most of the required data by simply mounting the device to the robot in a non-horizontal configuration. However, while the sensor will work, the drawbacks are a high weight and power consumption, in addition to its large size. Consequently, the sensor was not selected.

The HS-SR04 ultrasonic sensor has several desirable attributes. The precision is good enough, and the cost, power consumption and weight are low. An issue however, occurs when detecting negative obstacles. From an engineering perspective, a negative obstacle is the absence of ground. Since ultrasonic sensors require an almost perpendicular angle between the sound wave and the surface, the HS-SR04 would have to be mounted directly downwards. Additionally, the sensor may be noisy when used for indoor applications.

Ultimately, the Sharp IR range sensors were selected. These sensors are lightweight, small and low-cost. They are also more precise compared to the other sensors except the RPLIDAR. As for the specific model, the GP2Y0A21YK0F was initially pursued, since multiple devices of this type were available from previous projects. However, GP2Y0A41SK0F [36] was eventually utilized, due to a more suitable range and lower power consumption. A point to consider is that the nRF52840 DK lacks the required ADCs for interfacing with any more of these sensors. Hence, additional hardware will have to be developed before the sensors can be fully utilized.

4.4 Sensor Rig Design Process

Based on the selected range sensor, several sensor rig prototypes were proposed. Essentially, the objective of this process was to find a configuration of N sensors which could detect the three obstacles types in the functional description (see Section 4.1.1).

The design procedure kept to a few common principles. Firstly, the sensors only keep track of areas in front of the robot, as the robot moves forward in straight lines. Secondly, the added components should not block the view of the existing sensor tower. Lastly, detection of overhead obstacles was kept separate from other types of obstacles, from Prototype 2 onwards. As the thin sensor tower is the only part of the robot at significant risk of collision from such obstacles, only a single upwards oriented sensor is needed.

In total, six prototypes were developed. In summary, Prototypes 1 and 2 generated a 2D sample array by rotating IR sensors on a servo, while Prototypes 3 and 4 used a sensor array to generate the data. Prototype 5, which was the pursued solution, considered a decoupling approach, where detection of each obstacle is considered separately. However, this prototype was later modified, resulting in Prototype 6, as the lack of ADC restricted proper testing at this stage. The thesis will not considered specifics about the discarded

prototypes, but will instead discuss in broader terms. SolidWorks [37] was used to design the prototypes, and the resulting CAD models are given in Appendix B. To manufacture the models, Ultimaker 2+ 3D printers [38] at Make NTNU [39] were used.

4.4.1 Servo-based Approach

Firstly, a servo-based approach was pursued. Figure 4.6 shows how this was intended to work, where an IR sensor is mounted on a servo rotating back and forth. When the robot moves forward, the ground in front of the robot is sampled. As the servo movement introduces an additional variable, MATLAB simulations in Appendix C aided the design.

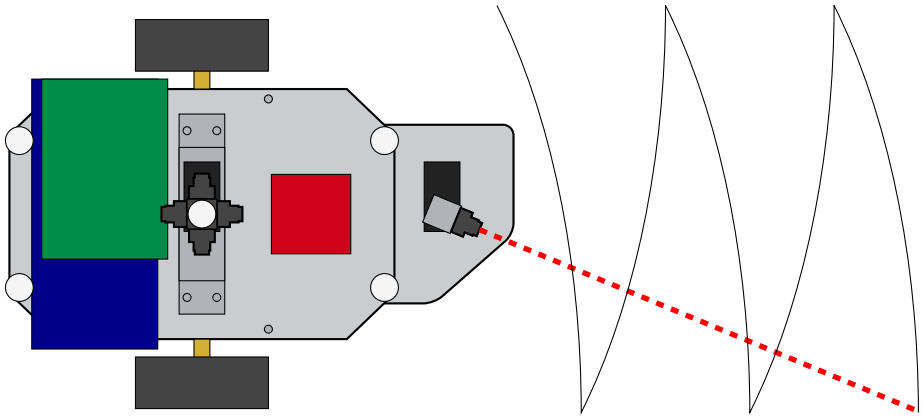


Figure 4.6: Conceptual visualization of the servo-based approach

Two prototypes were made using this approach. Prototype 1 consisted of a downwards and an upwards oriented sensor, both mounted at 22.5°. This solution proved to be insufficient, as the sensor could not sample a sufficient amount of data. Hence an improved version, with three downwards oriented sensors, was developed. Several issues were revealed during this process, which ultimately lead to the servo being scratched. The issues were:

- **Voltage drop:** When the servo is accelerated, a large amount of current is drawn. This resulted in a voltage drop, as the 5 V-regulator on the motor driver could not deliver enough current. The regulator at the NRF shield was then utilized, as it was rated for a higher current. However, while this solved the issue, the second 5 V-regulator struggled with heat dissipation.
- **Sample gaps and rotation speed:** While the sensor configuration in Prototype 2 samples most of the ground in front of the robot, MATLAB simulations revealed that there were still minor gaps at the sides. Figure C.2 in Appendix C shows this phenomenon. Additionally, the servo rotated slightly slower than intended at design stage, further increasing these gaps in practice.
- **Tilt angle:** A small 22.5° tilt angle was necessary to reduce the servo movement interval. To elaborate, a large angle would mean the servo would have to rotate even

faster in the same amount of time. As a result, some additional noise was induced in the measurements, due to the small angle between the ground and the infrared ray.

4.4.2 Sensor Array-based Approach

Secondly, a sensor array-based approach was pursued. Figure 4.7 shows how this is intended to work, where multiple IR sensors are mounted in a downwards angle towards the ground. An disadvantage with this method is that more sensors are required, compared to the servo-based approach, to minimize the gaps between the infrared rays.

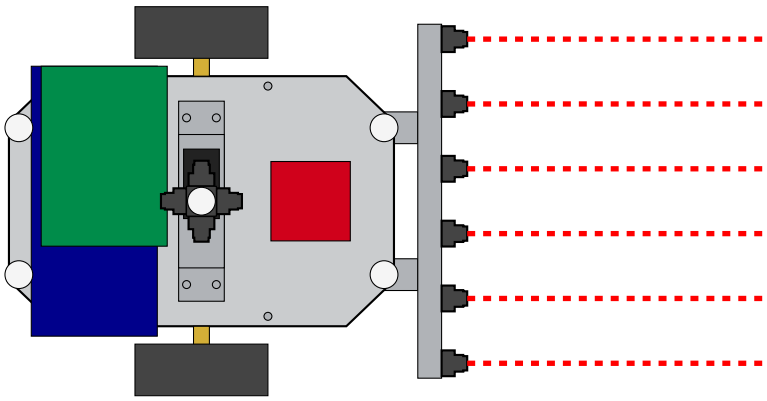
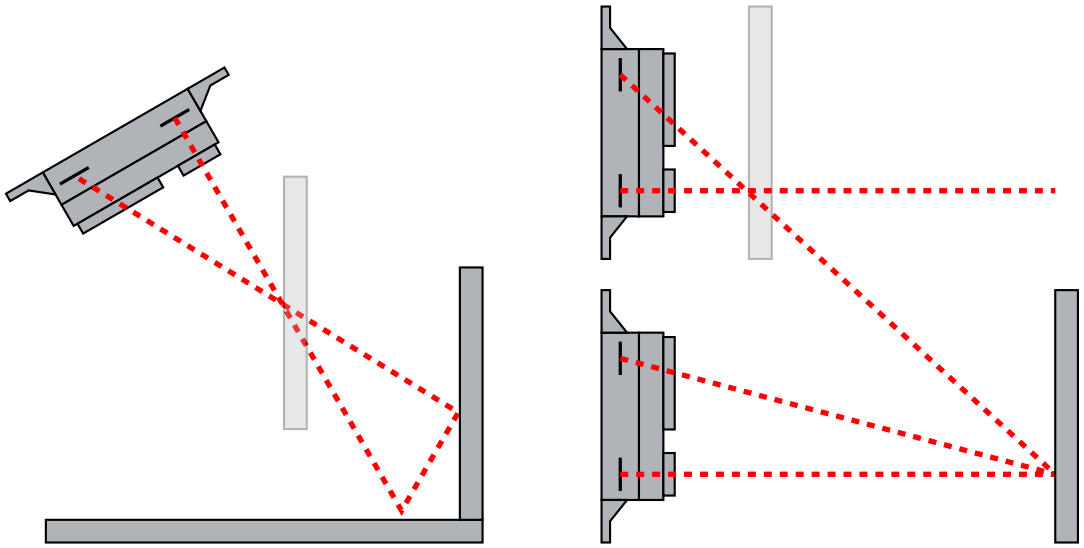


Figure 4.7: Conceptual visualization of the sensor array-based approach

Using this approach, two prototypes where made. In Prototype 3, the sensors were mounted in a vertical manner, with the emitter above the detector. In Prototype 4, the sensors were mounted in a horizontal manner, with the emitter and detector located at the same height.



(a) Reflection **(b)** Disturbance from other sensors

Figure 4.8: Discrepancies experienced with the Sharp IR sensors

Two types of discrepancies were found during this process, and these are visualized in Figure 4.8. When mounted in a vertical manner, the IR-sensors experienced reflections of objects outside the line of sight. As a result, objects triggered change in the sensor output long before intended. When mounted in a horizontal manner, the sensors experienced disturbances from other sensors. Upon further investigation, it was discovered that the disturbances were reduced when the sensors were mounted at different angles. However, due to these discrepancies, the sensor array-based approach was discarded.

4.4.3 Decoupled Obstacle Detection Approach

Lastly, a solution was pursued, by decoupling the detection of each type of obstacle. The identified sensors were as follows:

- **Negative obstacles:** 3 downwards oriented sensors. One for each of the wheels, and one for the caster ball.
- **Overhead obstacles:** 1 upwards oriented sensor, to cover for the sensor tower.
- **Ground obstacles:** 2 horizontally oriented sensors, positioned close to the ground.

To reduce the effect of reflection, all sensors were mounted in a horizontal manner. Furthermore, the design heavily focused on reducing disturbances between sensors. Therefore, the sensors were mounted in different orientations, as opposed to the sensor array-based prototypes. The rig consists of two 3D printed parts, and Figure 4.9 shows a conceptual illustration of the prototype as a whole. The downwards and upwards oriented sensors are located around the height of the upper metal plate, and are mounted to the first 3D printed part. The horizontally oriented sensors are mounted to the second 3D printed part, which is located close to the ground. Additionally, each sensor is assigned a number, counting from the four already existing sensors.

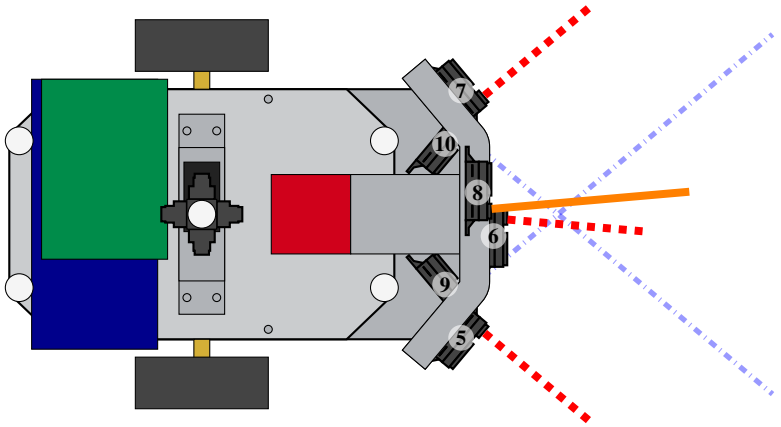


Figure 4.9: Conceptual visualization of the decoupled obstacle detection approach, i.e. Prototype 5 with enumerated sensors. The red rays point downwards, the orange ray points upwards, and the blue rays point horizontally.

Prototype 5 solves one issue which has not been mentioned thus far. During testing, Sensors 9 and 10 experienced false measurements, due to being mounted close to the ground. To elaborate, the sensors functioned as expected when an object was inside their measuring range. However, when the object was removed, the sensors measured a shorter distance than expected. While the exact cause of this phenomenon is still unknown, it is assumed that some kind of IR radiation reflects of the ground. The solution to this issue was to limit the view of the ground, by adding FOV limiters around the detector. The CAD model in Figure B.5b shows how this was done.

4.5 Proof of Concept

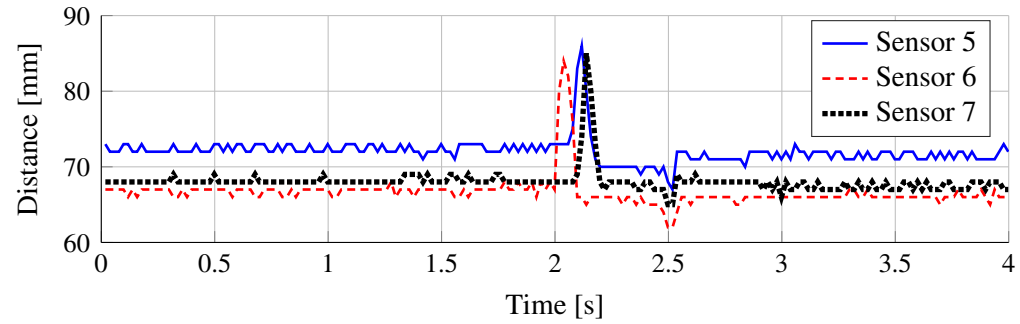
As an additional hardware module is necessary to utilize the sensors, due a lack of ADCs on the nRF52840 DK, proof of concept testing was necessary at this stage. The objective of this testing procedure is to verify that the sensor solution will work. Furthermore, a secondary objective is to discover any immediate discrepancies, s.t. any additional hardware requirements can be identified. The testing will be limited, as only four out of the six sensors can be tested simultaneously. However, this should be sufficient when testing each obstacle type separately, and for investigating whether any sensors disturb each others.

Before the test, Sensors 5-10 were calibrated, with a similar method to Section 3.2.1. However, as the GP2Y0A41SK0F has a shorter range, the step size was changed to 2 cm. Note that the sensors need to be recalibrated again once an additional hardware module is introduced.

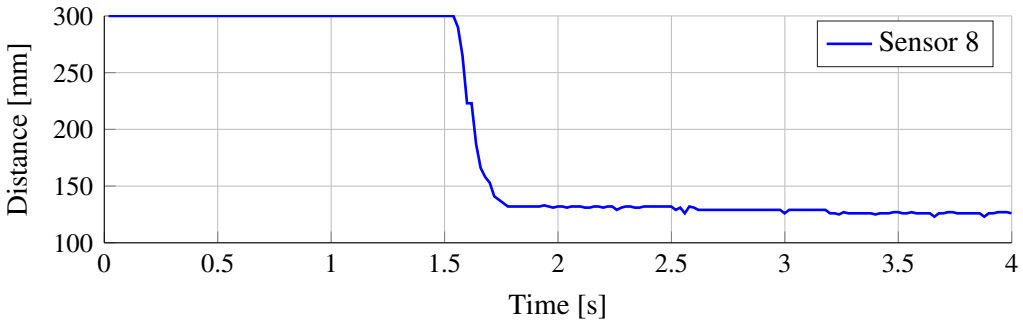
4.5.1 Obstacle Test

Firstly, a simple obstacle test was conducted. The main idea was to study sensor data upon the robot coming across each of the three categories of obstacles. In this test, the discrepancies were not in focus. Therefore, sensors responsible for another obstacle type were disconnected from power. Each test iteration was conducted in the following manner:

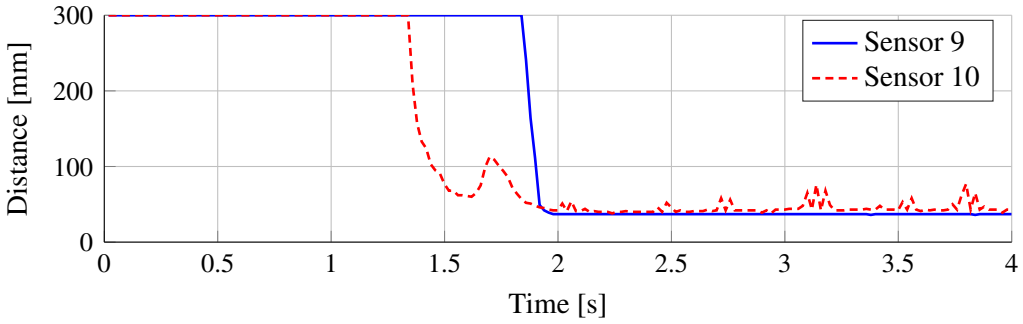
1. Connect a selection of sensors to power
2. Disconnect any other sensors
3. Prepare an obstacle of the correct type
4. Position the obstacle in front of the robot
5. Manually drive the robot 50 cm while recording sensor data



(a) Negative obstacle



(b) Overhead obstacle



(c) Ground obstacle

Figure 4.10: Time series of sensor data for three different obstacles types. The data was generated by the robot moving towards the obstacles at 0.25 m s^{-1}.

Figure 4.10 shows the output sensor data from the three test iterations. When studying the time series, it important to note that the robot essentially collided with the obstacle, as there exists no obstacle avoidance functionality as of yet. Therefore, phenomena occurring after an obstacle is first observed can be neglected, e.g. noise in Figure 4.10c.

There is a clear change in sensor data when obstacles are perceived by the sensors. This is especially apparent for Sensors 8, 9 and 10. Additionally, the amount of noise is low, which means small changes in the sensor data can be detected. This is important when detecting negative obstacles with Sensors 5, 6 and 7, where the distance to ground is constantly

monitored.

Another observation is that the distance to ground in Figure 4.10a is different for the three sensors, although it should be the same. While this is likely from inaccuracies in calibration, a ground detection scheme upon startup could be considered.

4.5.2 Sensor Disturbance Test

Secondly, tests to reveal disturbances between sensors were conducted. The procedure went as follows. Initially, let the robot be idle and remove any obstacles. For each group of sensors, power on one additional sensor outside the group, and check if any disturbances occur. Afterwards, repeat the test, but move an obstacle around and try to provoke a disturbance. The results found where as follows:

- When no obstacles are present, there are no disturbances
- Sensors 9 and 10 can be disturbed by Sensor 6 when there are obstacles present
- In all other cases, no disturbances were observed

To comment on the magnitude of the noise, an example is included. Figure 4.11 shows noise in the measurements from Sensor 9, induced by Sensor 6. The example shows that the magnitude can be relatively large, and the signal is practically unusable when the disturbance occurs. was

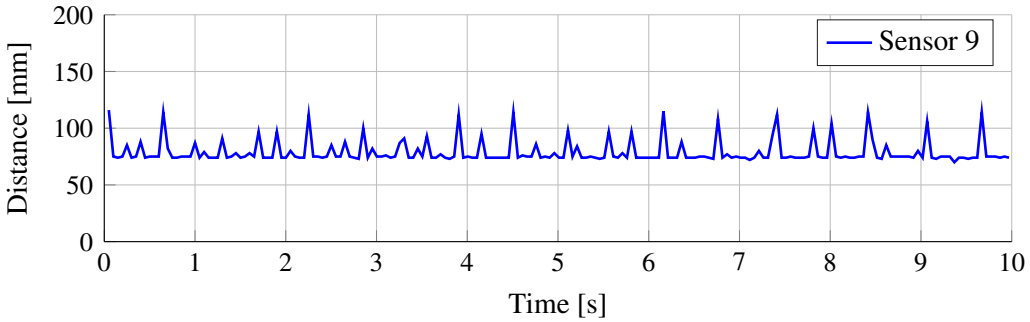


Figure 4.11: Time series showing a case of Sensor 6 disturbing Sensor 9 when an obstacle is present

In conclusion, the disturbances between sensors are still present, even in Prototype 5, which specifically addresses these issues. However, as they only appear when obstacles are present, the sensors should still give sufficient data for simple obstacle detection. When using the data in other ways, e.g. if adding obstacles in the map on the server, the data may be insufficient still. Hence, filtering or other noise-reducing measures should be considered.

4.6 Final Solution

After completing the additional hardware module (see Chapter 6) proposed when selecting the Sharp IR sensors, testing revealed that the horizontally oriented sensors in Prototype 5 are less robust than what was believed initially. Essentially, the sensors yield measurements leading to frequent false positive detections. A more elaborate discussion of this topic is provided in Chapter 7.

Therefore, Prototype 6 is proposed, where two more downwards oriented sensors replace the horizontally oriented sensors. This solution is more similar to the sensor array-based approach, as there are now five downwards oriented sensors, and one upwards oriented sensor. However, due to the different orientations of the sensors, and noise-reducing features developed in Chapter 6, disturbances between sensors do no longer jeopardize the system.

Figure 4.12 shows the prototype mounted onto the robot. The prototype consists of two parts. Part 1 is the same as in prototype 5, with three downwards oriented sensors and one upwards sensors. Part 2 is fixed two Part 1, and provides an additional two downwards oriented sensors. The sensors are enumerated according to the numbers in the figure.

A consequence of switching from Prototype 5 to Prototype 6 is that the solution may be incapable of detecting particularly small or thin items, e.g. dices or thin posts. To elaborate, there are gaps between the points which the sensors perceive, where items can slip between. Detection of negative obstacles, which has the most damage potential upon failure, is unaffected.

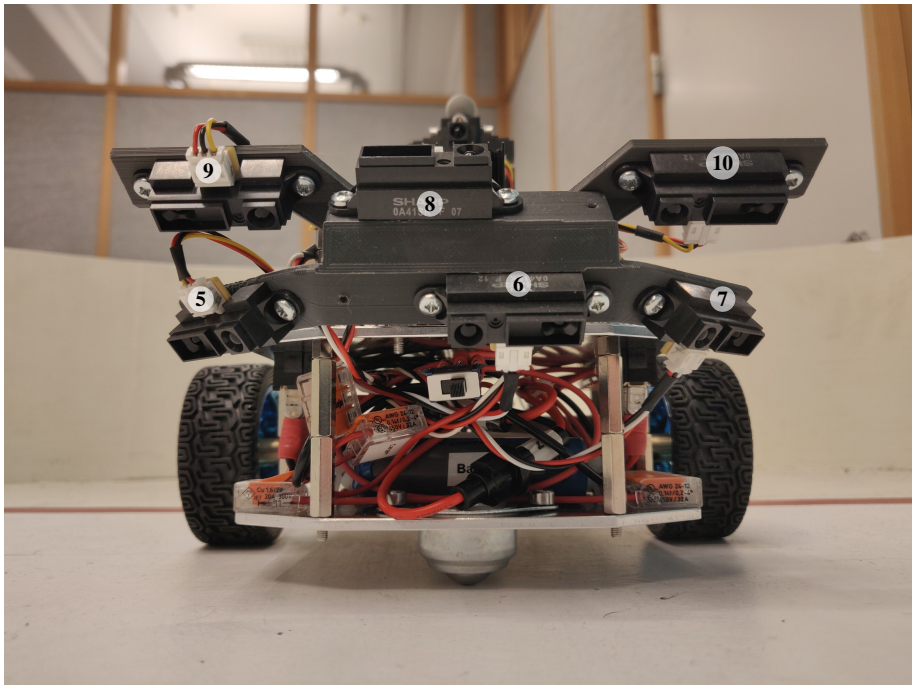


Figure 4.12: Final sensor solution with enumerated sensors

5 | Obstacle Detection and Avoidance

In this chapter, the obstacle detection and avoidance functionality is explained. The content is based on the available data from the sensor solution found in Chapter 4. The chapter is structured as follows. Firstly, the coordinate systems are defined. Afterwards, the obstacle detection method is presented. Lastly, the obstacle avoidance measures are explained.

5.1 Coordinate Systems

Previously, navigation of the NRF robots has been conducted in the 2D plane, as the robots move on a flat surface. However, as this thesis seeks to detect obstacles at different heights, the coordinate systems need to be extended with a third dimension.

5.1.1 World frame

The world frame is a fixed reference frame defining the map axes. The frame is defined by the three directions x_w, y_w and z_w, where the z-axis is oriented upwards. Two vectors are defined in world frame:

- Let $\mathbf{r}_w = [x_w \ y_w \ z_w]^T$ be an arbitrary position in world frame
- Let $\mathbf{r} = [x \ y \ 0]^T$ be the position of the NRF robot. The position is defined at the center of the robot, but at the ground plane ($z_w = 0$).

The origin of the frame is specified on the Java server, by setting $\mathbf{r}$ upon connection with a robot. The orientation is set s.t. the robot axes are parallel with the axes of the body frame, which is presented in the next paragraph.

5.1.2 Body Frame

The body frame is a reference frame which is fixed to the body of the NRF robot. The origin position of this frame is located on the ground, right below the center of the robot (i.e. sensor tower). Furthermore, the axes are defined as follows

- x_b - Direction toward the front of the robot
- y_b - Direction towards the left wheel
- z_b - Direction straight upwards

A position in body frame can be described by the vector $\mathbf{r}_b = [x_b\ y_b\ z_b]^T$. To transform a point from body to world frame, the following equation can be used

$$\mathbf{r}_w = \mathbf{r} + \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{r}_b \tag{5.1}$$

where θ is the orientation of the robot.

5.1.3 Sensor Frame

The sensor frame is a reference frame which is fixed to an IR range sensor. Each of the new sensors on the robot, enumerated from 5 to 10, is defined in its own sensor frame. Let x_{sk} be the x-axis of the kth sensor. The y- and z-axes are left undefined, as the x-axis is defined along the laser line of sight. Hence, y_{sk} and z_{sk} is equal to 0 for all sensor measurements. A sensor frame is defined by the following:

- A position $\mathbf{S}_k = (s_{xk}, s_{yk}, s_{zk})$ in body frame
- A CCW rotation β_k around the z-axis.
- A downwards rotation α_k around the new y-axis. Defined $\alpha = 0$ when the sensor is pointing in a horizontal direction.

Consequently, a transformation from the frame of sensor k to body can be described by

$$\mathbf{r_b} = \mathbf{S}_k + \begin{bmatrix} \cos\alpha_k \cos\beta_k \\ \cos\alpha_k \sin\beta_k \\ -\sin\alpha_k \end{bmatrix} x_{sk} \tag{5.2}$$

5.2 Obstacle Detection

This section presents a method for IR sensor-based obstacle detection. In this thesis, an obstacle is considered detected when the robot collects a sample which would indicate a forthcoming collision or near-collision, given the assumption of a straight path. To describe this mathematically, let l, w and h be the dimensions of the NRF robot. Furthermore, let δ_x, δ_{z^-} and δ_{z^+} be margins, s.t near-collision can be defined clearly. Then, the region in body frame where ground and overhead obstacles can be detected is given in Equation (5.3).

$$-\frac{w}{2} - \delta_x < y_b < \frac{w}{2} + \delta_x \quad \cap \quad \delta_{z^-} < z_b < h + \delta_{z^+} \tag{5.3}$$

For negative obstacles, the region of detection is given in Equation (5.4).

$$z_b < -\delta_{z^-} \tag{5.4}$$

However, it is assumed that the sensor is mounted s.t. perceives the ground in front of the robot, and not on the sides or behind. This is the case for the downwards oriented sensors in Prototypes 5 and 6, which are capable of detecting negative obstacles.

To evaluate the samples, two approaches were considered:

1. Transform a sample into body frame, and evaluate the sample in this frame.
2. Treat the problem in sensor frame, in one dimension. More specifically, for each sensor, divide the measuring range into intervals, where the interval decides whether or not a sample indicates detection.

While either approach would be equally effective, the latter was selected, as analysis and signal processing would be slightly simpler. Additionally, some obstacle detection is offered on an additional hardware module (see Chapter 6). Hence, the second approach would reduce the communication load between the module and the nRF52840 DK.

5.2.1 Detection in Sensor Frame

The main idea behind obstacle detection in sensor frame is to identify which samples would indicate either collision or near-collision, by defining a set of intervals. If an interval indicates that an obstacle is detected, it is referred to as a detection interval.

The intervals are parameterized by a reference distance d_r and a threshold d_δ. The main idea is that d_r is the exact distance to the boundary between where a collision would occur, and where the robot would barely avoid the obstacle. As the sensor measurements are not exact, d_δ adds extra leeway. Assuming d_r and d_δ are identified, a detection mode is selected. The modes define which of the intervals are detection intervals, and which are not. The available detection modes are defined in Figure 5.1.

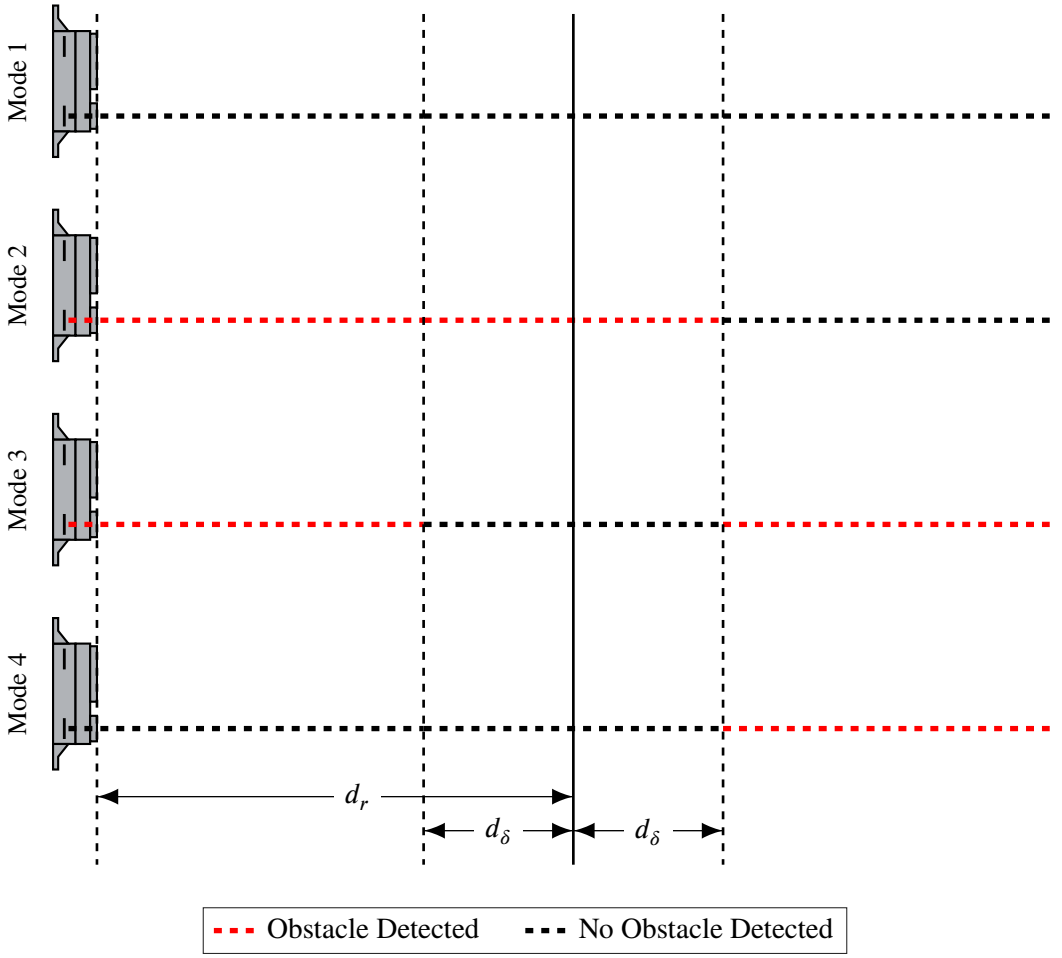


Figure 5.1: The four obstacle detection modes, defining the intervals which determine whether an obstacle is considered detected

Mode 1 is a default mode, where no obstacle is detected. For detection of overhead obstacles with upwards oriented sensors, or for detection of ground obstacles with horizontally oriented sensors, obstacles inside a certain range are considered detected. Therefore, Mode 2 is applicable. For detection of negative obstacles with downwards oriented sensors, it is necessary to detect the absence of the ground. If Mode 4 is used, only negative obstacles are detected. Mode 3 allows for additional detection of ground obstacles for these sensors.

5.2.2 Offset Adjustment Calibration

In the proof of concept testing of the IR sensors in Section 4.5, it was found that despite calibration, the measured distances were slightly inaccurate. For the downwards oriented sensors (5, 6 and 7), where the distance to ground is normally measured, accurate samples are particularly important. Therefore, an offset adjustment calibration scheme is proposed.

Initially, for each of the downwards oriented sensors, an offset Δd_k is calculated, where k

denotes the sensor. Equation (5.5) shows how this is done,

$$\Delta d_k = \frac{1}{N} \sum_{i=1}^{N} (d_k[i] - d_{rk}) \tag{5.5}$$

where $d_k[i]$ is a sample, d_{rk} is the reference distance, and N is the total number of samples during calibration. The samples are collected under the assumption of there being no obstacles in front of the robot. After completion, the offset is subtracted from all subsequent samples, as in Equation (5.6).

$$d'_k[i] = d_k[i] - \Delta d_k \tag{5.6}$$

Lastly, adjusted samples $d'_k[i]$ are evaluated during obstacle detection.

5.2.3 Parameter Identification

For each sensor, the reference distance d_r can be identified by projecting the regions in Equations (5.3) and (5.4) onto the individual sensor frames. To reiterate, d_r is a distance to a boundary between where collision would occur, and where the robot would barely avoid it. There are four potential boundaries in body frame, depending on the sensor orientation. These are $y_b = -w/2$, $y_b = w/2$, $z_b = 0$ and $z_b = h$. Then, given that the sensor position is known, the reference distance d_{rk} for the kth sensor can be calculated according to

$$d_{rk} = \begin{cases} \dfrac{s_{yk} - b_k}{\sin \beta_k} & \text{for horizontally oriented sensors} \\ \dfrac{s_{zk} - b_k}{\sin \alpha_k} & \text{otherwise} \end{cases} \tag{5.7}$$

where b_k is the value of the boundary for sensor k. Similarly, the detection threshold $d_{\delta k}$ can be calculated according to

$$d_{\delta k} = \begin{cases} \dfrac{\delta_{z^-}}{\sin \left\| \alpha_k \right\|} & \text{for downwards oriented sensors} \\ \dfrac{\delta_{z^+}}{\sin \left\| \alpha_k \right\|} & \text{for upwards oriented sensors} \\ \dfrac{\delta_y}{\sin \left\| \beta_k \right\|} & \text{for horizontally oriented sensors} \end{cases} \tag{5.8}$$

This simplifies the number of configuration parameters down to only δ_{z^-}, δ_{z^+} and δ_y. Scaling by the sensor angle also makes setting the threshold values more intuitive, as one can consider the margins along the y- and z-axes in body frame. Sensor noise and desired leeway should be taken into consideration when selecting appropriate values.

5.3 Obstacle Avoidance

Based on the obstacle detection method presented in the previous section, an obstacle avoidance method can be proposed. Most importantly, the responsibility of this method is to take appropriate action once an obstacle is detected. This includes aborting the current waypoint, and stopping the robot before colliding. In missions where no obstacles are detected, the robot should act the same as before, and the method should never be engaged. A point to consider is that the method presented here is not a complete Traffic Collision Avoidance System (TCAS), as it only handles non-moving obstacles. Other situations, s.a. avoidance of other robots, are outside the scope of this thesis.

5.3.1 Core Functionality

The main principle behind the proposed method is utilizing the allowable moves in the pose controller, by forcible interjection from the time an obstacle is detected, until it is avoided. As a reminder, the pose controller consists of four states, which are `moveForward`, `moveStop`, `moveClockwise` and `moveCounterclockwise`.

Figure 5.2 shows how obstacle avoidance is intended to work. More specifically, it shows a scenario where the robot moves forward, before it detects an obstacle and applies the appropriate action, which is to stop. In an attempt to return back to normal, the robot turns around its center of area. The desired outcome of this move is the robot ending up in a direction where it detects no obstacle, s.t. it can start moving forward again. Alternatively, the robot can end up in another direction where an obstacle is still being detected.

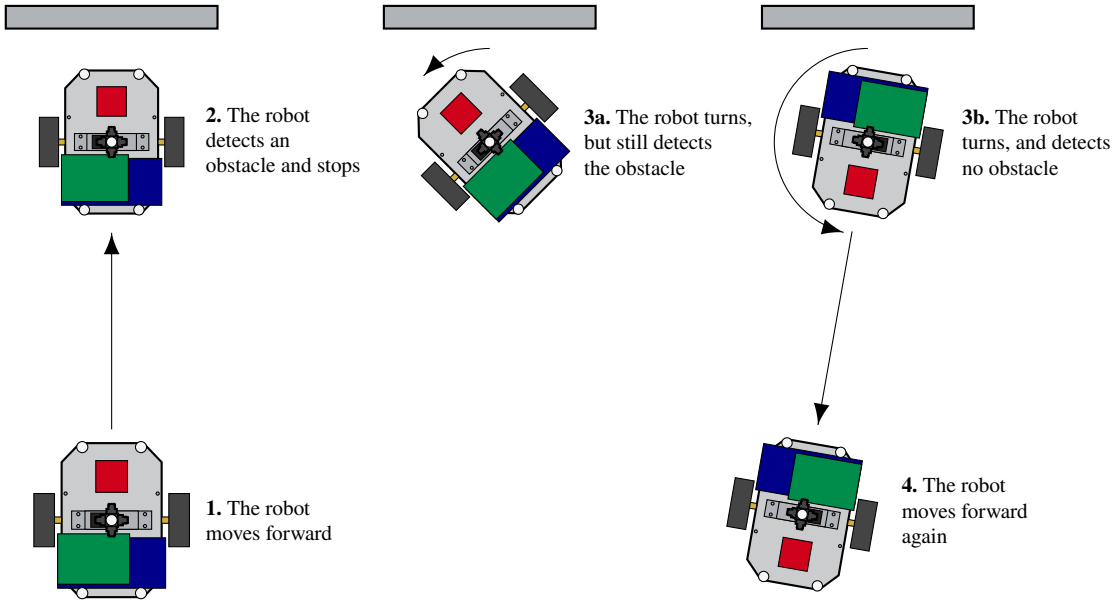


Figure 5.2: Illustration of how the core part obstacle avoidance works. There are two scenarios, one where the robot fails to immediately return to normal operation, and another where it succeeds.

To relate the above explanation to the pose controller, `moveForward` is the only state in which obstacle avoidance is necessary, while `moveStop`, `moveClockwise` and `moveCounterclockwise` are legal states after obstacle avoidance is already initiated. An assumption is made here, which is that turning is always collision-free. In reality, there exist situations where turning can result in the NRF robot colliding. However, a completely collision-safe avoidance system would require a more advanced sensor solution, or a different robot shape. Similar robots s.a. robotic vacuum cleaners avoid this problem as they are completely circular.

5.3.2 Adding Obstacles to the Java Server

A missing part in the core obstacle avoidance functionality is how the robot is supposed to turn s.t. it ends up in a direction where it can move freely forward once more. This challenge is more difficult than simply turning until no obstacle is detected, as there is nothing that prevents the path planner from guiding the robot back to a similar state. Consequently, the planner needs to be aware of the obstacle, s.t it can attempt turning in directions which are not yet observed by the sensor solution. This is achieved by adding the obstacle to the map on the Java server.

To add obstacles in the map, several modifications were made. Firstly, a new message was defined, as the existing update message for the IR sensors was insufficient. Table 5.1 shows the content of the message. Note that the sensor data parameter is assumed to be in the horizontal plane, hence a projection is required before sending the data. Additionally, for negative obstacles, the distance is always equal to the reference distance before projection, as the obstacle is the absence of ground.

Table 5.1: Obstacle update message

Message	**Parameters**	**Unit**
Obstacle Update	Position (x,y)	cm
	Orientation	deg
	Sensor Angle	deg
	Sensor Offset (x,y)	cm
	Sensor Data	cm

Furthermore, some minor modifications were made to the Java server. Without going into details, this included transforming the samples according to the rotations described in Section 5.1. Additionally, when an obstacle is added to the map, only the specific location of the obstacle is marked as occupied. Originally, an issue occurred after adding sample data, where the samples from the sensor tower overwrote the obstacles. This was solved by adding obstacles permanently to the map. As a result, estimation bias and faulty detections will have a permanent impact, which will jeopardize long-running exploration missions. However, a more sophisticated solution to this problem could not be found within the time frame of this thesis.

5.3.3 State Machine

The obstacle avoidance functionality was implemented as a state machine, consisting of three states. Figure 5.3 shows a diagram of the state machine. Monitoring is the default state, where the robot conducts normal operation. When an obstacle is detected while moving forward, the state machine switches to the stopping state. After a stop is confirmed, the waiting state is entered, where the state machine waits for an orientation where no obstacle is detected. In this state, all moves except forward movement are allowed.

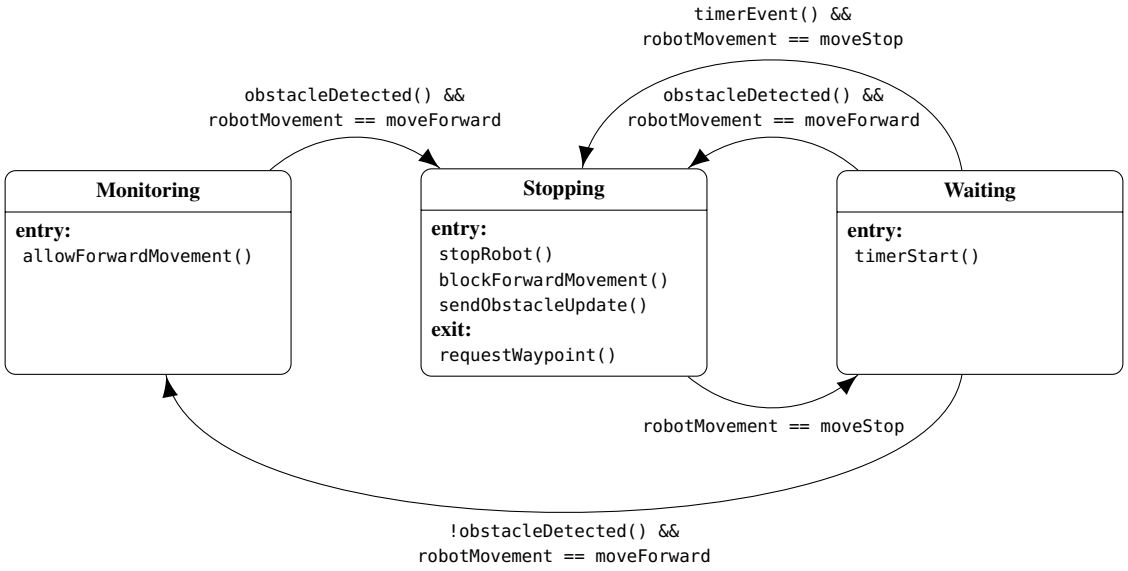


Figure 5.3: Obstacle avoidance state machine diagram

In addition to what is previously presented, the state machine requires the ability to block forward movement. After receiving a new waypoint, stopping and completing the turn towards this waypoint, the state machine must confirm that forward movement is safe. Otherwise, small bursts of forward movement occur, which can lead to collision. Another necessary addition to the state machine was a timeout option for the waiting state. The reason for this is that occasionally the server aborts a waypoints, which leads to the robot being stuck in this state. In this case, a new waypoint needs to be requested.

6 External Hardware Module

This chapter presents an external hardware module for interfacing with the sensor solution. As was discovered during the sensor selection in Chapter 4, the nRF52840 DK lacks the necessary ADCs for connecting with more IR sensors. Therefore, the need for an additional MCU with extra ADCs was identified.

The chapter is structured as follows. Firstly, a plan for the module is made, describing what the hardware must offer, and some initial considerations. Secondly, the hardware and interfaces on the external hardware module are presented, before the process of designing the PCB is explained. Lastly, onboard software and drivers are presented.

6.1 Planning the Module

Before selecting components and designing a PCB, it is necessary to identify the requirements of the module. The following were found:

- **ADC**: The module have to read analog values from six Sharp GP2Y0A41SK0F sensors in total. Therefore, the selected MCU needs at least six GPIO-pins with ADC functionality. Both 10-bit and 12-bit ADCs are acceptable.
- **Communication interface**: A common interface like I2C or SPI should be considered when communicating with the nRF52840 DK.
- **Logic level**: The nRF52840 DK runs on 3.3 V, while the Sharp sensors require 5 V. Therefore, appropriate measures are required for safe communication with existing hardware while reading sensor values at the same time.
- **Sensor switch**: In the proof of concept testing in Chapter 4, noise was experienced when utilizing multiple IR sensors simultaneously. A way of removing this issue is by using a switching circuit, where the power supply is turned on and of s.t. sources of noise are temporarily removed. Hence, hardware for such functionality should be offered.
- **Onboard functionality**: The module should offer some obstacle avoidance functionality, to lessen the load on the nRF52840 DK and the I2C/SPI bus.

6.2 Hardware and Interfaces

In this section, the components on the PCB are selected and the interfaces between them are established. A general philosophy throughout this stage is convenience, i.e. choosing the easier option to save time if the decision is inconsequential to the module requirements. An overview of the hardware components and the interfaces between them, which were found during this stage, are given in Figure 6.1.

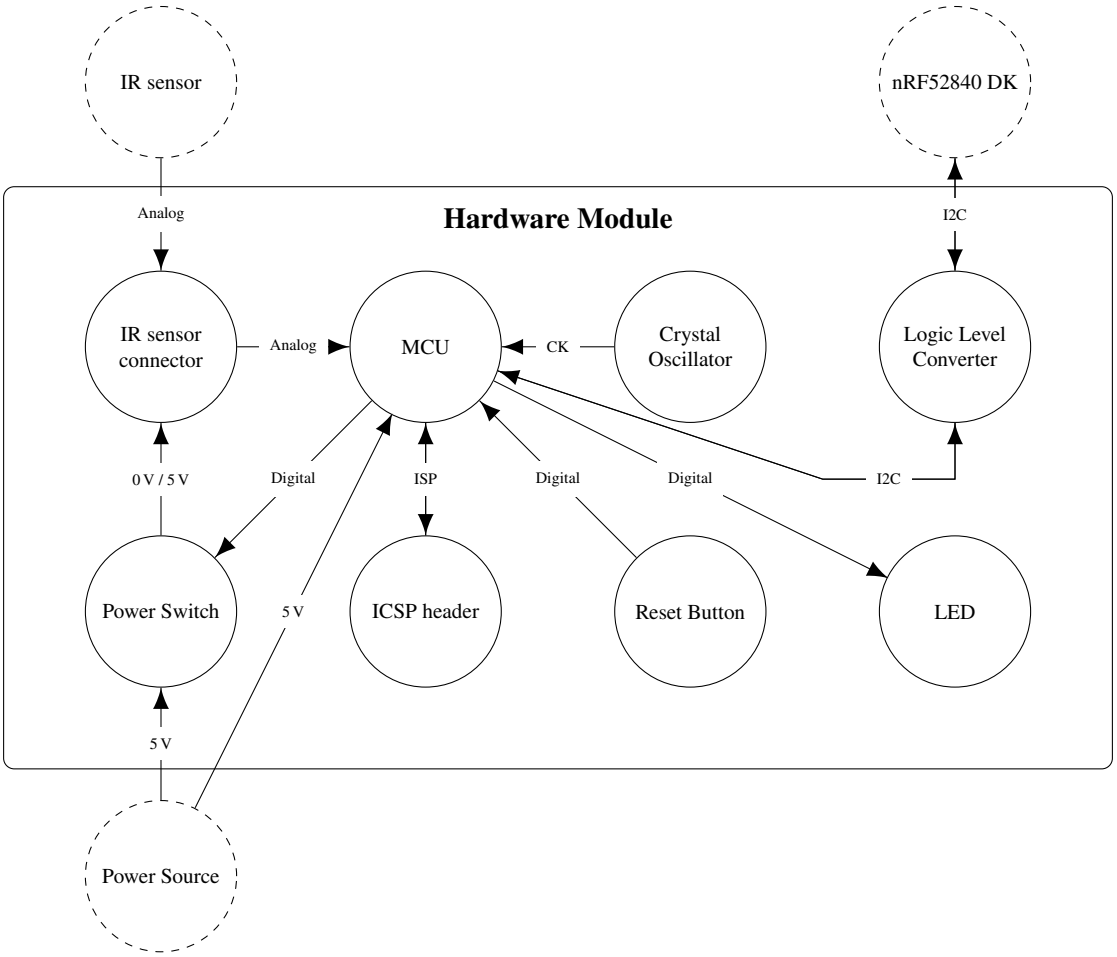


Figure 6.1: Overview of components and interfaces

6.2.1 Logic Level

A decision early on was whether the module should be powered by 3.3 V or 5 V. There are advantages and drawbacks with either solution. If a 3.3 V logic level is utilized, the MCU can be directly connected to the nRF52840 DK. Conversely, one voltage divider would be required for each sensor to protect the MCU, resulting in 12 additional resistors on the PCB. If a 5 V logic level is used, voltage dividers can be scratched. Instead, a logic level

converter is needed when interfacing with the nRF52840 DK. Ultimately, a 5 V logic level was selected, as it seemed the more convenient option.

6.2.2 Communication Protocol

Another decision was whether to use the I2C or SPI protocol for communication with existing hardware. The I2C protocol was eventually selected, due to the following reasons. Firstly, the protocol has been more regularly used throughout the robot project, and hence, a I2C master driver already exists on the nRF52840 DK. Secondly, I2C pins are available several places on the nRF shield, as opposed to a single header, which gives more freedom when designing peripheral connectors.

6.2.3 Selection of Components

When selecting components, the main philosophy in this project is availability. Therefore, components from previous students in the robot project, and institutions s.a. *Omega Verksted* (`https://www.omegav.ntnu.no/`) and *Elektronikkvekstedet* at the Department of Engineering Cybernetics, were checked before ordering online.

Microcontroller

The selected MCU is the ATmega168-20AI [40], which is an MCU in the AVR series from Atmel. This circuit has eight ADC-pins, but two of these are assigned to the SCL and SDA lines in the I2C protocol. Furthermore, the ATmega168-20AI is a 5 V MCU. As for the exact model selection, the near-identical ATmega328PB was originally considered. However, this MCU was not available at *Omega Verksted*. One of the advantages of an AVR-based microcontroller, is that software and drivers can be developed while the PCB is in production, by using an Arduino board.

Logic Level Converter

To connect the ATmega168 to the nRF52840 DK through I2C, a logic level converter is required. The role of this component is to shift a digital 3.3 V signal to a 5 V signal, and vice versa. Since I2C consists of two bidirectional lines (SDA and SCL), a bidirectional logic level converter with two channel is required. Figure 6.2 shows the circuit for one channel of a logic level converter. The circuit consists of one N-Channel MOSFET (NMOS) and two pull-up resistors.

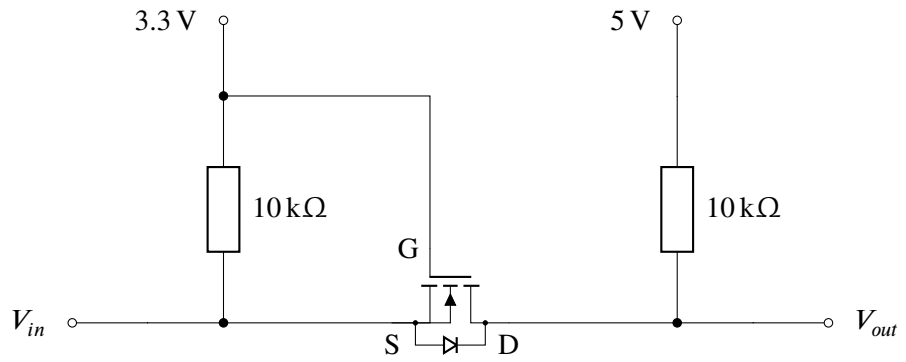


Figure 6.2: Bidirectional logic level converter

Since only two channels are needed, the logic level converter was built from scratch. Due to availability at *Omega Verksted*, the selected transistor is the MMBF170 [41]. I2C communication was tested with the circuit, and was confirmed to work. However, the BS170 was used during testing, which is the TH version of the same transistor.

Sensor Switch

The main idea behind a sensor switch was to reduce noise between the IR sensors, which proved an issue during the development of the sensor solution (see Chapter 4). The first idea was to directly use the output from a GPIO-pin. However, while the power consumption was inside the datasheet's specification, there were doubts whether this would provide a steady, low-noise power supply. Therefore, a transistor circuit was selected instead.

Figure 6.3 shows the switching circuit, which consists of an NMOS transistor, a pull-down resistor and a gate resistor. The load R_L represents the sensor. Once again, the MMBF170 [41] transistor was selected, and the circuit was tested with the complementary BS170 before PCB design. As for the gate resistor, the size was selected s.t. the current can never surpass 40 mA, which is the absolute maximum supply current according to the datasheet [40]. Equation (6.1) shows the calculation.

$$R \geq 5\,\text{V}/40\,\text{mA} = 125\,\Omega \tag{6.1}$$

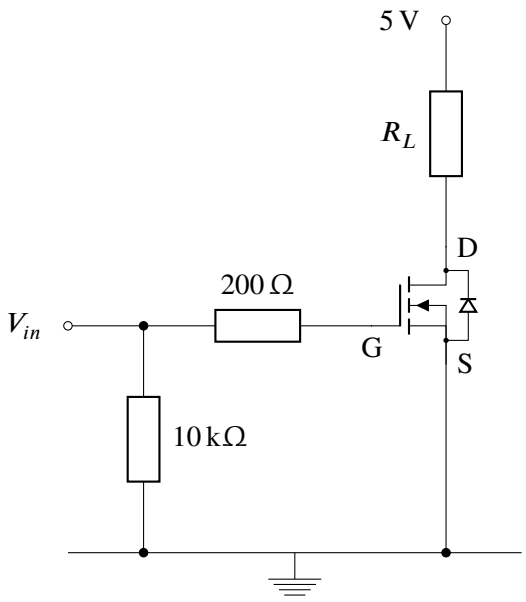


Figure 6.3: NMOS switch circuit

Other Components

Two LEDs are provided on the module, where one indicates power supply, and the other can be programmed by the user. Furthermore, several decoupling capacitors are added close to the ATmega168 VCC pins.

For common components s.a. resistors, capacitors and LEDs, SMD components were selected. The main reason for this was space considerations, as the number of resistors became significant due to the logic level converter and the switching circuits. The 0603 version ($0.6" \times 0.3"$) was selected, as it was the commonly available size at *Omega Verksted*.

Lastly, a 16 Hz-oscillator was included for precise timing in the ATmega168, and a reset button was also added for convenience.

6.2.4 Peripheral Connectors and Power Supply

After the hardware components were selected, connectors for peripherals and power supply were chosen. In compliance with existing IR sensors, 3-pin JST sockets were selected. Additionally, an In Circuit Serial Programming (ICSP) header was required to program the ATmega168.

For the I2C connectors, there exist two dedicated 4-pin headers on the nRF shield, where one is male and one is female. However, the female header is dedicated to the OLED display. As for the power supply, 5 V and GND signals are available through the headers on the nRF shield. During the PCB design stage, the module was connected to exisiting hardware as a shield, making both power supply and I2C available through stackable headers.

6.3 PCB Design

This section goes through the PCB design process step by step, including schematics, PCB layout, and production of the board. A PCB design program called KiCAD [42] was utilized, as the nRF shield was developed using this software previously.

6.3.1 Schematics and Layout

The first step of the process is to create a schematic of the electronic circuit, which is done in KiCAD's Schematics software EESchema. Most of the used component symbols were found natively in Eeschema. However, more advanced components s.a. the ATmega168 and 3-pin JST connectors were fetched from the Mouser Components Library [43]. The next step was to associate symbols with PCB footprints, which are an arrangement of pads or through-holes for attaching components to the circuit board. For the symbols from Mouser Components Library, this was done automatically.

Subsequently, a layout for the circuit board was created in PCBnew, which is KiCAD's PCB layout program. The standard of 2 layers was selected. This step consists of 3 stages. First, a board outline is defined, which determines the shape of the PCB. Secondly, footprints are placed on the board, in a way s.t. components are easily connected later. Lastly, connections called traces are drawn between the components, and holes called vias are used to connect the bottom and front layers. It is important that traces are sufficiently wide, especially for the power supply. Additionally, there were multiple requirements originating from the manufacturer. Without going into details, this regarded matters s.a. trace width and size of vias.

Both the front and bottom layers of the PCB, and the circuit schematics, are provided in Appendix D.

6.3.2 Production and Assembly

A total of six PCBs were ordered from *Elektronikk og prototypelaboratoriet* at the Department of Electronic Systems. The manufacturer did not offer silk screen, and hence, there is no print on the PCBs. In total, the time between the order inquiry and completion was approximately two weeks. However, there is a large variety in delivery time depending on the demand.

After receival, one board was soldered at *Elektronikkverkstedet*, Department of Engineering Cybernetics. The SMD components were soldered by a person at the lab, while TH-components were soldered by the author of this thesis. Figure 6.4 shows the completed PCB stacked on top of the nRF shield.

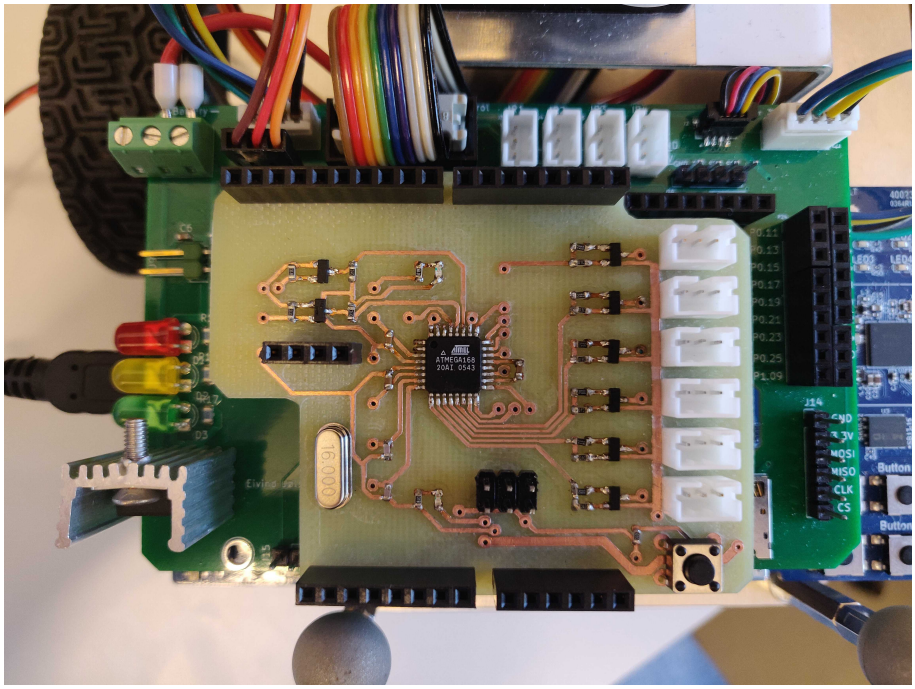


Figure 6.4: Assembled PCB stacked on top of the nRF shield

6.4 Source Code

This section presents the source code on the external hardware module, including both drivers and software. Figure 6.5 shows a simplified class diagram, which includes the main components of the source code. Note that the programmable LED is excluded, as interfacing with this component is done through macros, which are available across all c files.

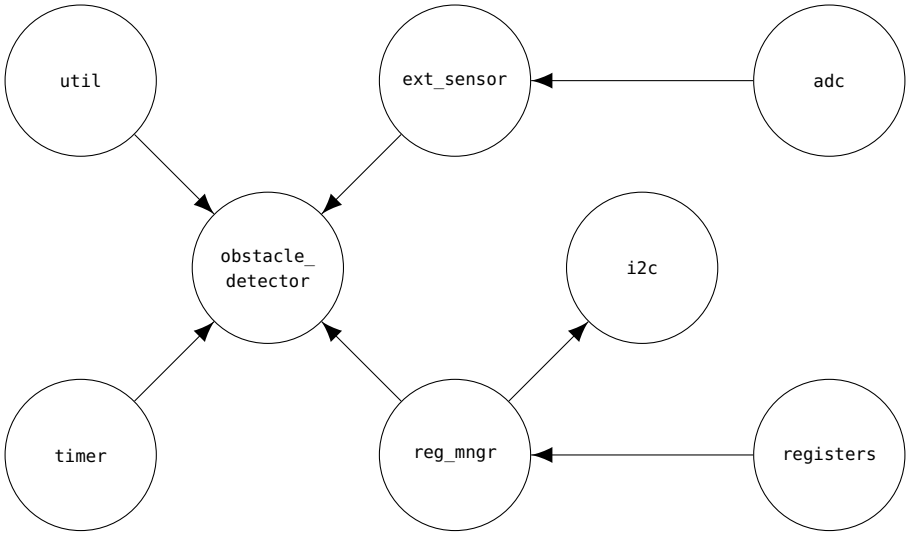


Figure 6.5: Class diagram of source code

6.4.1 Drivers

`i2c`

The I2C slave driver handles communication with the master driver on the nRF52840 DK board, using the Two-Wire Interface (TWI) protocol. The driver is interrupt-based, s.t. an event is handled as fast as possible when it occurs. The data is fetched from and relayed to the other parts of the system through callback functions, and is not directly handled in the driver. Three callback functions are required:

- A function for handling a received byte
- A function for fetching a requested byte
- A function to signal end of transmission

`adc`

An ADC driver was implemented to read the output voltage from the IR sensors. On the ATmega168, there is a single 10-bit ADC, which can read the input from eight different channels. A multiplexer selects which channel to read from. Hence, the driver provides functionality for ADC setup, channel selection, and reading the ADC registers.

`ext_sensor`

The external sensor driver unifies all functionality for interfacing with the IR sensors. Firstly, the driver offers functionality for turning the sensors on or off, i.e. the driver interfaces with the NMOS switches. Secondly, the driver works as layer between the ADC

driver and software, where the main addition is a simple filter. The main reason for including such filter was that the ADC experiences noise. This noise is especially prevalent in the first few measurements after switching the ADC multiplexer. Consequently, a fixed number of ADC-readings is initially discarded when measuring the sensor signal. Afterwards, the final measurement is found by calculating the median of another fixed number of readings. Lastly, the driver can access a sensor in the aforementioned manners by simply specifying the number (1-6) of the sensor.

`timer`

The timer driver implements timing functionality. The driver offers start and wait functionality, where the waiting time is set when the timer is started. Timer1, which is a 16-bit timer on the ATmega168, is utilized.

6.4.2 Software

`registers`

The register module implements simple storage on the external hardware module, including read and write operations for the `uint8_t`, `uint16_t` and `float` data types. The registers are divided into user banks 0-6, to streamline access to individual sensor settings. The registers themselves are documented in further detail in Appendix E. However, the functionality which these registers enable is described more concisely in this subsection.

`reg_mngr`

The register manager's main responsibility is providing synchronized access to the registers. The main reason for this is to avoid race conditions, as both the obstacle detector routine and the I2C interrupt routine require access to the registers. The register manager provides the following functions:

- **I2C callback functions**: The register manager implements the three callback functions for handling a received byte, fetching a requested byte, and ending a transmission. When a byte is received or requested, a busy flag is set, which is in turn reset when the transmission is ended.
- **Load settings function**: An interrupt-safe function is provided for loading copies of a selection of registers. As a copy is loaded, the settings will only change when this function is called, and not when new data arrives from the I2C driver.
- **Store data function**: An interrupt-safe function is provided for storing samples and obstacle detection status in the registers.

To ensure the load and store functions are interrupt-safe, two synchronization functions were implemented. Figure 6.6 shows the implementation. Essentially, these function ensures that interrupts are disabled only after the busy flag is cleared.

```
void reg_mngr_semaphore_take(uint16_t timeout_10_us) {

    while (1) {
        cli();

        if (!reg_busy_flag || timeout_10_us <= 0) {
            reg_busy_flag = 0;
            break;
        } else {
            sei();
            _delay_us(10);
            timeout_10_us--;
        }
    }
}

void reg_mngr_semaphore_give() {
    sei();
}
```

Figure 6.6: Synchronization functions in the register manager

`obstacle_detector`

The obstacle detector routine implements most of the functionality the external hardware module has to offer, by utilizing the drivers and other software modules. By default, the module functions as an external ADC, where the input values on the six ADC channels are read. To utilize more advanced features, the register settings must be changed. See Appendix E for more information.

The first available feature is the ability to turn the power supply for each of the sensors on and off, as specified by the user. For example, this feature may be utilized to turn of all sensors when there are multiple robots in close proximity, s.t. the robots do not disturb each others.

A second feature is continuous switching of the power supply, where the sensors can be turned on and off repeatedly in specified patterns. The main application of this is alternating switching, which prevents the IR sensors from disturbing each others. Figure 6.7 shows a timing diagram of how alternating switching between two sensors may work. The key part of the example is that the sensors are never turned on simultaneously. To determine a switching period, timing requirements from the sensors are considered. According to the GP2Y0A41SK0F datasheet [36], there is a delay of 5 ms between when a sample is recorded and the corresponding change in output voltage. Therefore, as the sample period of the sensor is (16.5 ± 3.7) ms, a switching period of at least 25.2 ms should be considered.

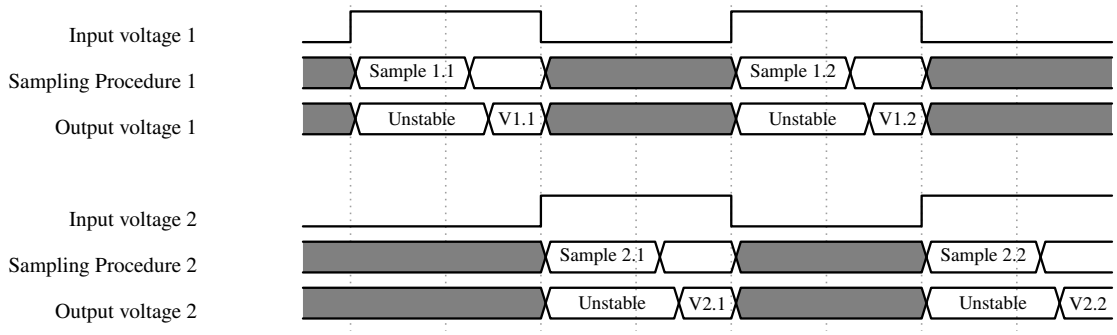


Figure 6.7: Example of how alternating switching removes disturbances between two sensors

The last and most important feature is obstacle detection, which functionality is already described in Sections 5.2.1 and 5.2.2. To enable obstacle detection for a single sensor, the detection mode must be set to any legal mode (1-4). Detection mode 0 is the default, which means detection is disabled, and the output data format is raw ADC readings.

For the other detection modes, the output data is converted into mm. If desirable, an offset can be calibrated, as described in Section 5.2.2. Lastly, the output data is evaluated based on the reference distance and the threshold. A detection status, which indicates whether or not an obstacle is detected, is determined according to the detection mode. See Appendix E for further detail on how to set up obstacle detection on the external hardware module.

7 | Testing and Evaluation

In this chapter, tests are conducted to evaluate the parts and functionality developed in the thesis. Firstly, the external hardware module is tested and verified. Secondly, the sensor solution and the obstacle detection method is tested. Thirdly, a final performance test of the system as a whole is conducted, to evaluate whether all individual parts work together. Lastly, the test results are discussed.

7.1 Testing of External Hardware Module

In this section, the external hardware module, which is presented in Chapter 6, is tested and verified. As most of the obstacle detection and avoidance functionality depends on data from this module, it is essential that this part works close to what was intended.

7.1.1 Continuity on the PCB

The first test stage was conducted to discover any errors in manufacturing, by utilizing the continuity function on a multimeter. The objective of this test is confirming that traces properly connects components in accordance with the PCB layout, and to discover any faulty short-circuitry. No problems were found during this operation. However, if more PCBs are assembled, the test should be repeated.

7.1.2 Hardware Components and Drivers

The second test stage was to examine whether the hardware components and the drivers works as expected. At this stage, the test method was mainly using onboard software for testing hardware, as most of the software was already developed on an Arduino Uno while waiting for the PCB being manufactured. As a result, prototypes of the hardware components had already been tested, as briefly mentioned in Chapter 6. Regardless, a final set of tests, confirming whether all components and corresponding drivers works, was necessary. Table 7.1 provides a short description of the individual tests, and the resulting outcome.

Table 7.1: Tests of individual hardware components and drivers on the external hardware module

Component	Test Description	Verdict
ICSP Programming	Flash a simple program to ATmega168 in Microchip Studio, using the ICSP protocol. Additionally, confirm whether debugging works with debugWIRE.	Passed
Power Supply	Check that the power supply works as expected.	Failed
LED	Apply power supply to the circuit, and check if the power indicator is lit. Afterwards, blink the programmable LED at a specific frequency.	Passed*
Reset Button	Reset program with button, and check the reaction of the programmable LED	Passed
Logic Level Converter	In turn, apply high and low voltage on both sides of the LLC, and read the other side. Repeat for both LLC channels.	Passed
I2C	Send and receive data from the nRF52840 DK.	Passed
IR Sensor	Read data from all 6 ADCs in turn while the corresponding IR sensor is connected. Continuously fetch the data from the nRF52840 DK, and print to terminal. Move an item in front of the sensor, and check if the output data changes.	Passed*
NMOS Power Switch	Power on and off each of the NMOS transistors and read voltage over the corresponding IR sensor. The resulting voltage should be 5 V when the power is on, and 0 V when the power is off. Furthermore, study the response with an oscilloscope.	Passed**

*Software-related failures, which were fixed
**Slow transient response when the switch is turned off, which needs further consideration

In the power supply test, a major error was discovered. In the design, the hardware module is connected to the 5V pin of the nRF52840 DK. This pin however, is not connected directly to the same 5V voltage regulator as the rest of the system. As a result, the nRF52840 DK stopped working properly when the USB was disconnected, likely due to an excessive current draw. The ultimate solution was to connect the hardware module to the voltage regulator on the nRF shield, by utilizing a breadboard wire. This is a separate power supply than for the servo, as the servo was found to generate significant noise in the IR sensor measurements.

Two software-related errors were also found during these tests. Firstly, when blinking the programmable LED, the frequency was much lower than expected. The cause of this error was that the internal oscillator was set as default. Hence, the solution was to change the fuses governing this setting, s.t. the external oscillator could be utilized. The second error was a simple bug in the LED macros, which also toggled two of the NMOS switches.

Lastly, when studying the response of the NMOS switch in the oscilloscope, which is

shown in Figure 7.1, the turn-off response is quite slow compared to the turn-on response. When the sensor is connected, the voltage goes below 1 V in less than 0.5 ms, but after 5 ms the voltage still has not reached zero. This may cause a problem for the continuous switching feature on the external hardware module, as a requirements for this to work is that two sensors are never turned on simultaneously. In case the switching feature fails, or requires a significant delay to properly remove sensor disturbances, the switching circuit should be reconsidered.

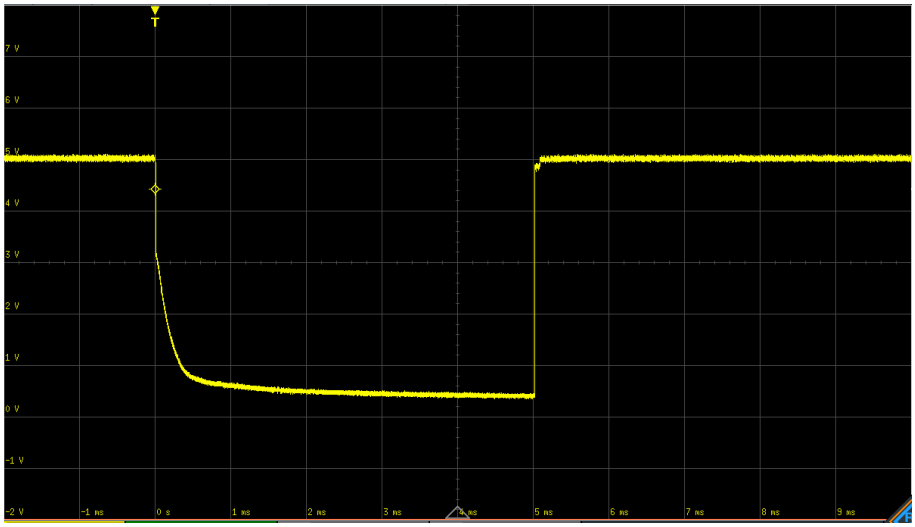
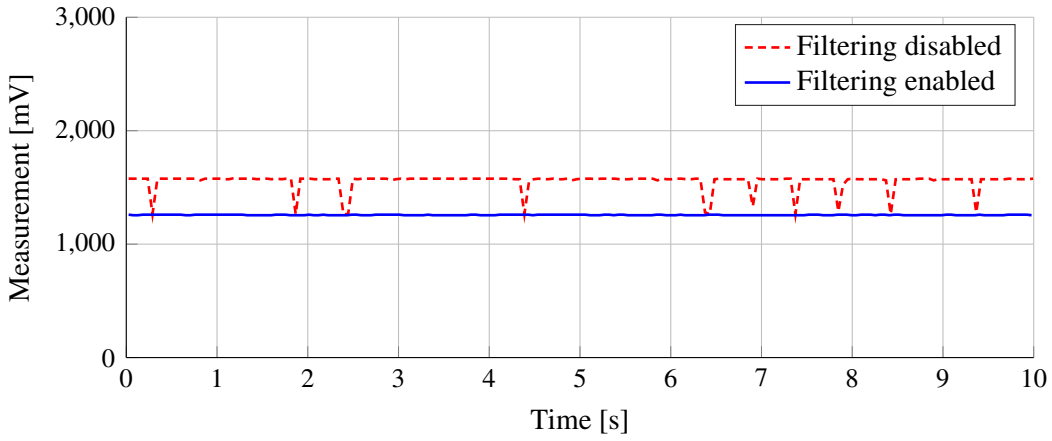


Figure 7.1: Transient response of the voltage over an IR sensor when the corresponding NMOS switch is toggled, measured with an oscilloscope. Here, the sensor is first turned off, before it is turned on again 5 ms later.

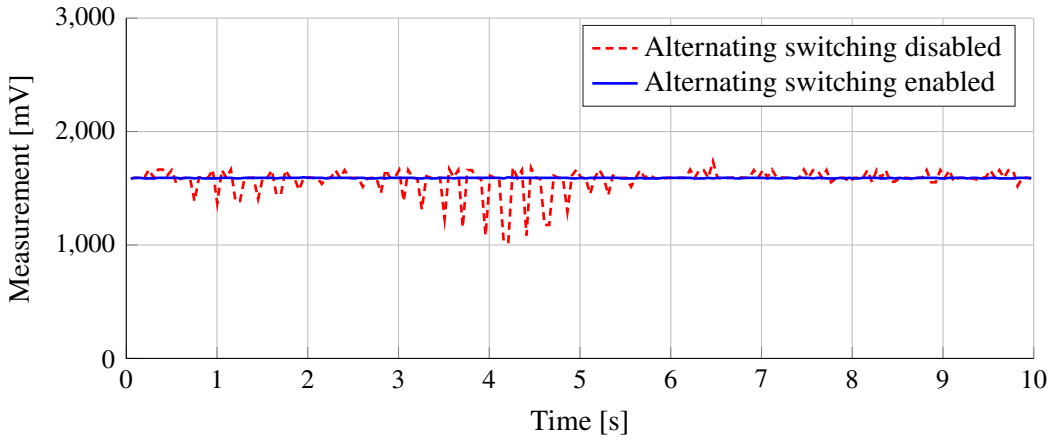
7.1.3 Noise-reducing Features

The main objective of the third stage of tests was to show the quality of the processed IR measurements which the external hardware module provides to the nRF52840 DK. One of the reasons for such tests is that the higher-level functionality s.a obstacle detection requires reliable, low-noise signals to function properly. Furthermore, a 10-bit ADC is now utilized, as compared to a 12-bit ADC when testing with the nRF52840 DK previously.

As described in Chapter 6, there are two features that specifically target IR sensor noise. The first is a simple filter to remove noise from the ADC. A test was conducted to show that this filter do indeed remove noise from the measurement. Figure 7.2a shows a time series where the filter is enabled, as compared to a time series where only a single ADC-reading is recorded. The plot shows that the filter gives a stable output signal, and that it removes what can be described as initial bias in the ADC readings.



(a) ADC noise filtering



(b) Alternating switching

Figure 7.2: Demonstration of the two noise-reducing features implemented on the external hardware module. The first plot **(a)** shows filtering of the ADC readings, while the second plot **(b)** demonstrates that alternating switching removes disturbances between sensors.

A second test was conducted to demonstrate that alternating switching can remove disturbances between sensors. These types of disturbances was previously shown to be an issue in the proof of concept testing of the sensor solution (see Section 4.5.2). Similar to that test, an obstacle was positioned s.t. one sensor is disturbing another. With this configuration, two time series were recorded, one with alternating switching enabled, and one with normal sensor data. Figure 7.2b shows the two time series, and the plot confirms that alternating switching works as intended.

7.2 Testing of Sensor Solution and Obstacle Detection

In this section, test results demonstrating the performance of the sensor solution and the obstacle detection system are presented. Results for these two categories are closely associated, as false detections mostly originate from discrepancies in the sensor solution, and not from the obstacle detection method itself. Furthermore, the contents of this test stage can be viewed as results the thesis wants to highlight, either to verify that some core parts of the system works as expected, or to show discrepancies which does not directly present themselves in the final performance test.

7.2.1 Setup and Configuration

At this stage, all six sensors were tuned based on ADC readings from the external hardware module. The method was similar to tuning procedure in Section 3.2.1, with a spacing of 2 cm instead due to the shorter measuring range of the GP2Y0A41SK0F.

The detection system is set up according to the parameters in Table 7.2. As both Prototypes 5 and 6 are tested, both parameter configurations are included. To give further explanation, the detection threshold and reference distance are determined according to the guide in Section 5.2.3, and offset calibration is enabled for all downwards oriented sensors. The threshold values are calculated based on $\delta_{z^-} = 5$, $\delta_{z^+} = 10$ and $\delta_y = 5$, where δ_{z^+} is slightly larger to for extra leeway in low passageways. Note that all distance units are in mm.

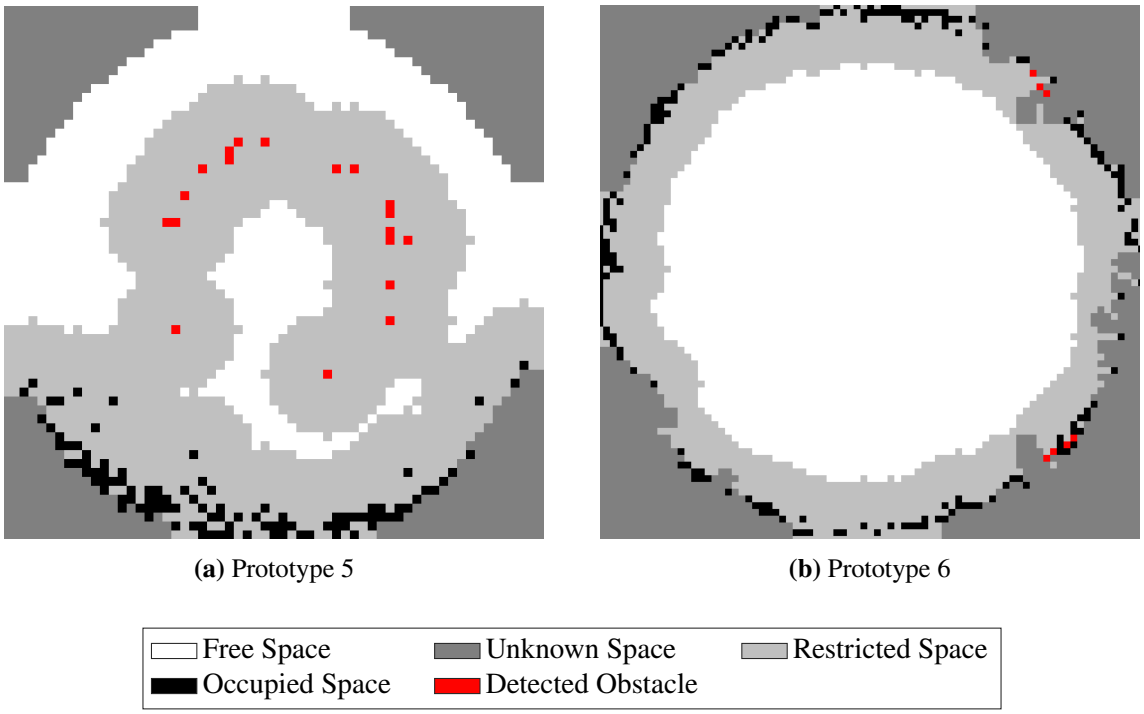
As for the power switching, there are a total of two cycles. Sensors 5-7 and Sensors 9-10 are switched on and off in an alternating pattern, while Sensor 8 is always turned on. This is the case for both prototypes.

Table 7.2: Obstacle detection setup for Prototypes 5 and 6

		Sensor 5	Sensor 6	Sensor 7	Sensor 8	Sensor 9	Sensor 10
Prototype 5	**Sensor Type**	Down	Down	Down	Up	Horizontal	Horizontal
	Offset Calibration	On	On	On	Off	Off	Off
	Switching Cycle	1	1	1	1 & 2	2	2
	Reference Distance	86	86	86	110	215	206
	Detection Threshold	7	7	7	14	9	7
Prototype 6	**Sensor Type**	Down	Down	Down	Up	Down	Down
	Offset Calibration	On	On	On	Off	On	On
	Switching Cycle	1	1	1	1 & 2	2	2
	Reference Distance	86	86	86	110	102	102
	Detection Threshold	7	7	7	14	6	6

7.2.2 Sensor Rig Prototypes

One of the most important aspects of the obstacle detection system as a whole is that when no additional obstacles are present, the robot should explore the environment as it normally does. As an initial test of sensor rig Prototypes 5 and 6, the NRF4 robot was tasked with exploring the circular track once for each prototype. In this case, a performance similar to the mapping and exploration of the circular track in Section 3.4 would be ideal.



(a) Prototype 5 **(b)** Prototype 6

Figure 7.3: Two reconstructed maps of the circular track, with Prototypes 5 and 6 used for obstacle detection respectively. In this case, there are no additional obstacles present, and any detected obstacles are represented by red voxels.

Figure 7.3 shows the two resulting maps after the robot has attempted to navigate and map the circular track. For mapping with Prototype 5, the robot experiences frequent false positive detections, which results in the robot failing to fully explore the map. The sensors in question that causes these false detections are Sensors 9 and 10, which are the horizontally oriented sensors close to the ground.

When mapping with Prototype 6, the horizontally oriented sensors are no longer utilized, and consequently, the robot manages to completely navigate and map the track. Additionally, the results are similar to the test results in Section 3.4. On some occasions however, the sensor solution does detect the wall, as there are groups of red voxels two places in the map.

As Prototype 5 experiences a high amount of false positive detections, this prototype was discontinued. In the remaining tests, Prototype 6, also referred to as the final solution, was

utilized.

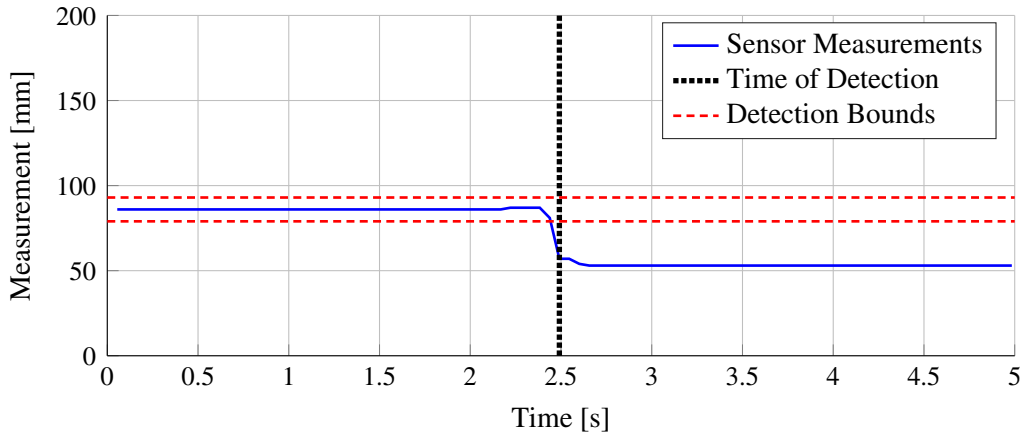
7.2.3 Detection of Individual Obstacle Types

The next step is to confirm that the system is capable of detecting each of the three obstacle types. To reiterate, these are overhead obstacles s.a. low passageways, ground obstacles s.a. small items, and negative obstacles s.a. holes in the ground.

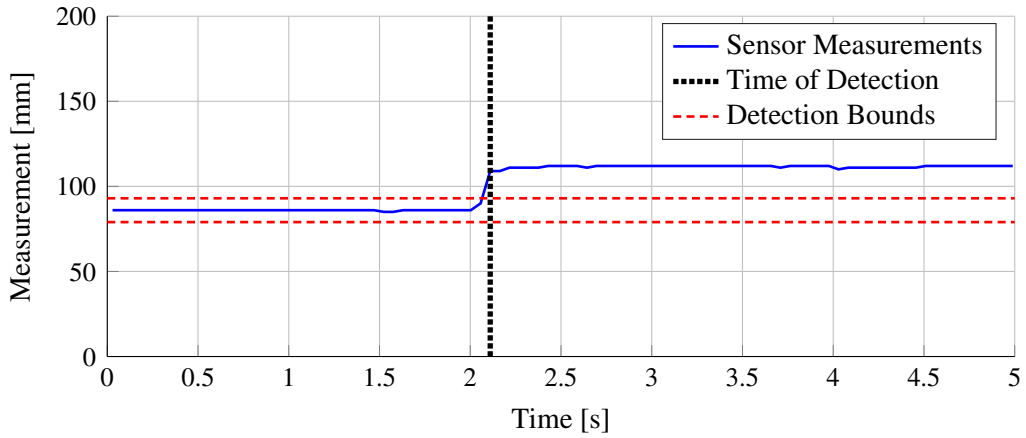
The test was set up as follows. In turn, obstacles of each type is positioned right in front of the robot. The robot is manually tasked with tracking a waypoint, s.t. the robot would collide with the obstacle if detection and the subsequent emergency stop fails.

Figure 7.4 shows the resulting measurement time series for a representative sensor in each of the three cases. For negative and ground obstacles, this is a downwards oriented sensor, while for overhead obstacles, this is an upwards oriented sensor. The plots also include the detection bounds determined by the reference distance and detection threshold according to Table 7.2. To confirm that detection works as expected, the first time of detection is marked in each of the plots.

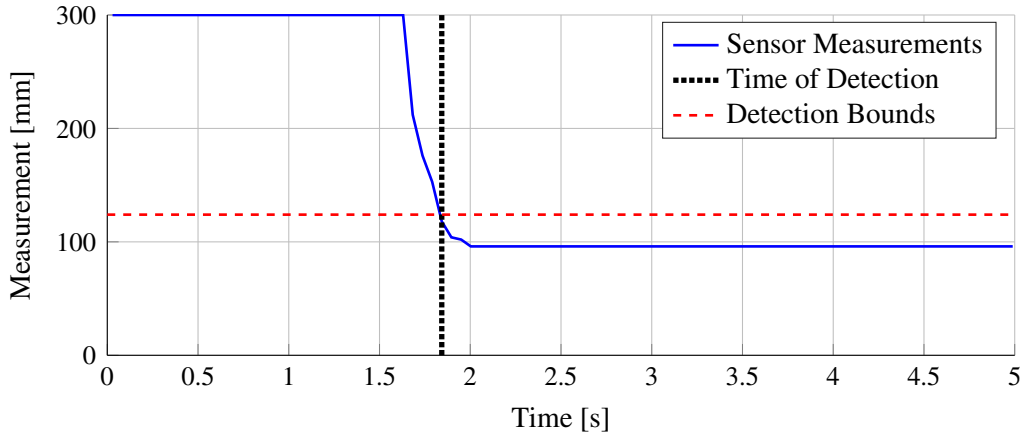
The results confirm that the system indeed manages to detect the three types of obstacles. Furthermore, the detection occurs right after a new measurement crosses one of the detection bounds.



(a) Ground obstacle



(b) Negative obstacle



(c) Overhead obstacle

Figure 7.4: Time series of measurements from a representative sensor, when detecting each of the three obstacle types.

7.2.4 Overview of Discrepancies

Several discrepancies with the obstacle detection system has been discovered throughout the work, which a user of the system should be aware of. As discrepancies do not always reveal themselves in larger tests, an overview is given in this subsection.

Firstly, the upwards oriented sensor experiences erroneous measurements when the robot is applied certain poses. These are measurements which affect the detection system, as the sensor measures a much shorter distance than what is the actual case.

Figure 7.5 shows an example of a pose where false detections occur. In this case, the wooden wall is slightly outside the laser line of sight. The resulting sensor measurements become very noisy, which results in the detection system yielding several false positives. This occurs despite the wooden wall being far outside the reference distance of the upwards oriented sensor. When the wooden wall is removed, or an item is placed inside the laser line of sight given the same pose, the noise disappears completely.

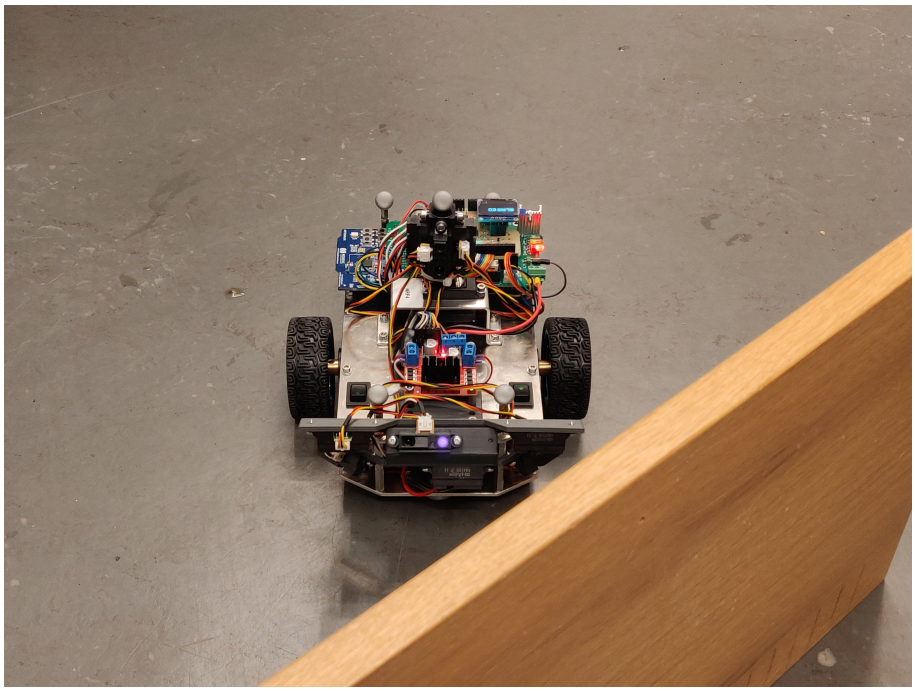


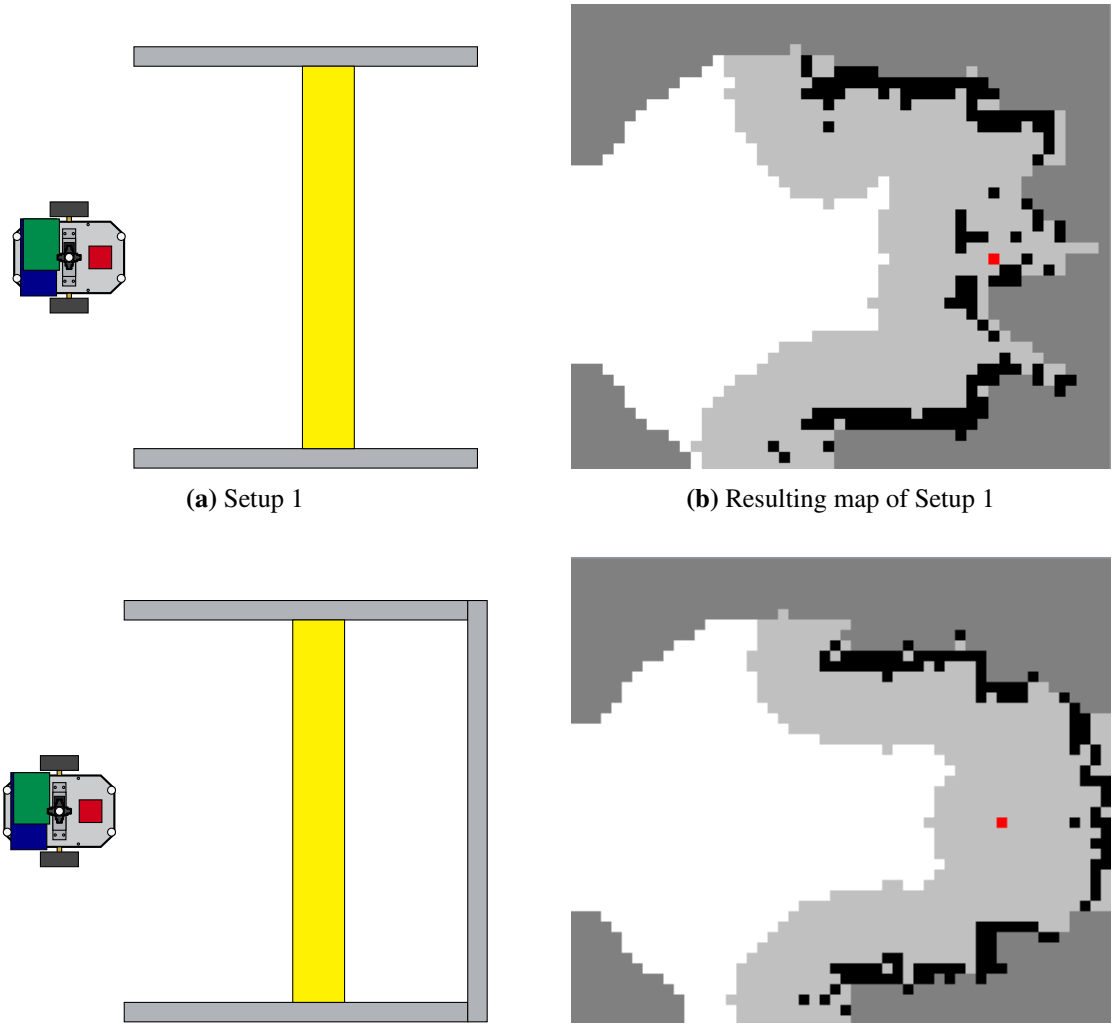
Figure 7.5: Robot pose where the detection system experiences a false positive

Secondly, another discrepancy was discovered when testing mapping with an overhead obstacle present. Similar to erroneous measurements of the upwards oriented sensor, the IR sensors of the sensor tower perceived the overhead obstacle, despite it being outside the laser line of sight. To demonstrate this, a test was conducted where the robot is manually tasked with moving towards an overhead obstacle, and stops upon detecting the obstacle.

Figure 7.6a shows a sketch of the setup, and Figure 7.6b shows the resulting map on the Java server. As the map shows, the IR sensors on the sensor tower perceived the obstacle,

albeit rather noisily, and consequently, added occupied voxels to the map.

A second test was also conducted where there is a wall behind the obstacle, as shown Figure 7.6c. The resulting map is given in Figure 7.6d. In this test, the IR sensors on the tower perceived the added wall instead, and only the upwards oriented sensor detects the obstacle.



(a) Setup 1

(b) Resulting map of Setup 1

(c) Setup 2

(d) Resulting map of Setup 2

Figure 7.6: Mapping and obstacle detection in two simple environments with an overhead obstacle present. Setup 1 **(a)** consists of two regular walls and an overhead obstacle, while Setup 2 **(b)** includes an additional wall. **(b)** and **(d)** show the resulting maps of the respective setups.

Lastly, the IR sensors will give erroneous measurements when perceiving low-reflective surfaces. A test was conducted to demonstrate this, where an area of the ground in front of the robot was covered with black insulation tape. The robot was then manually tasked with moving past this area, while recording the measurements of a representative sensor. Note that obstacle detection was disabled during this test.

Figure 7.7 shows the recorded measurements, compared to a recording of the similar test with a normal, flat surface. When the sensor perceives the black insulation tape at around 1.7 s, the measurements change substantially, and would certainly cause a false positive detection.

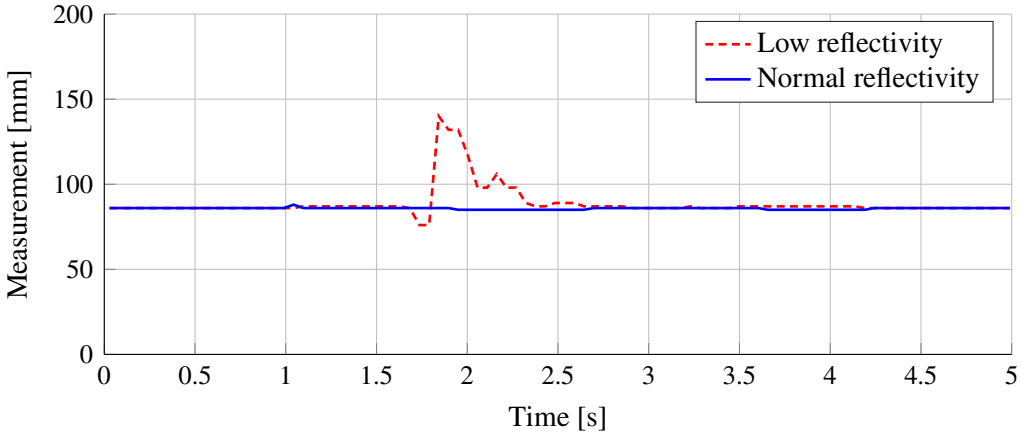


Figure 7.7: Time series showing how the sensor measurements change when an IR sensor perceives a low-reflective surface, i.e. black insulation tape.

7.3 Testing of Final Performance

Finally, the obstacle detection and avoidance system developed throughout this thesis was tested while simultaneously mapping an environment. A track containing each of the three classified obstacle types, referred to as the obstacle track, was prepared. Figure 7.8 shows a sketch of the track. Note that the overhead obstacle was placed closed to a wall, s.t. the sensor tower would not detect this obstacle (see more in Section 7.2.4).

The test was set up with the following configuration:

- **Server:** Java server
- **Mode:** Automatic Waypoint Generation
- **Voxel Size:** 2 cm
- **Robot:** NRF4
- **Starting Position:** Lower left corner, orientation towards the right

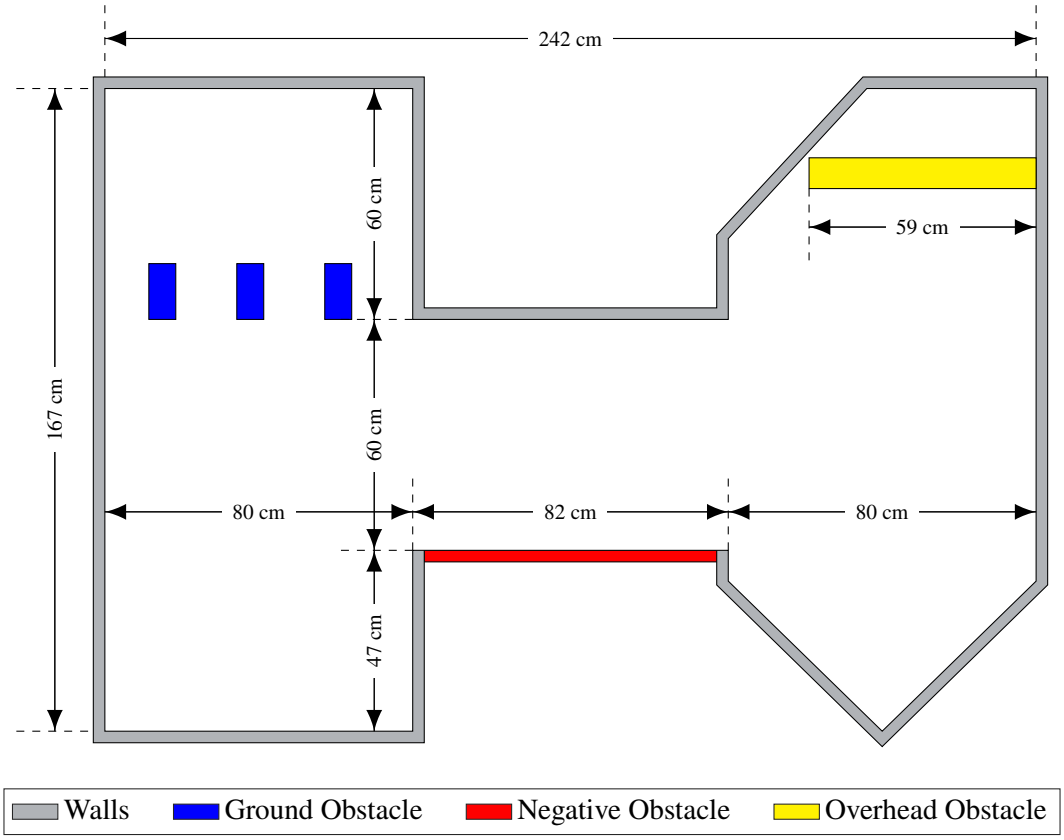


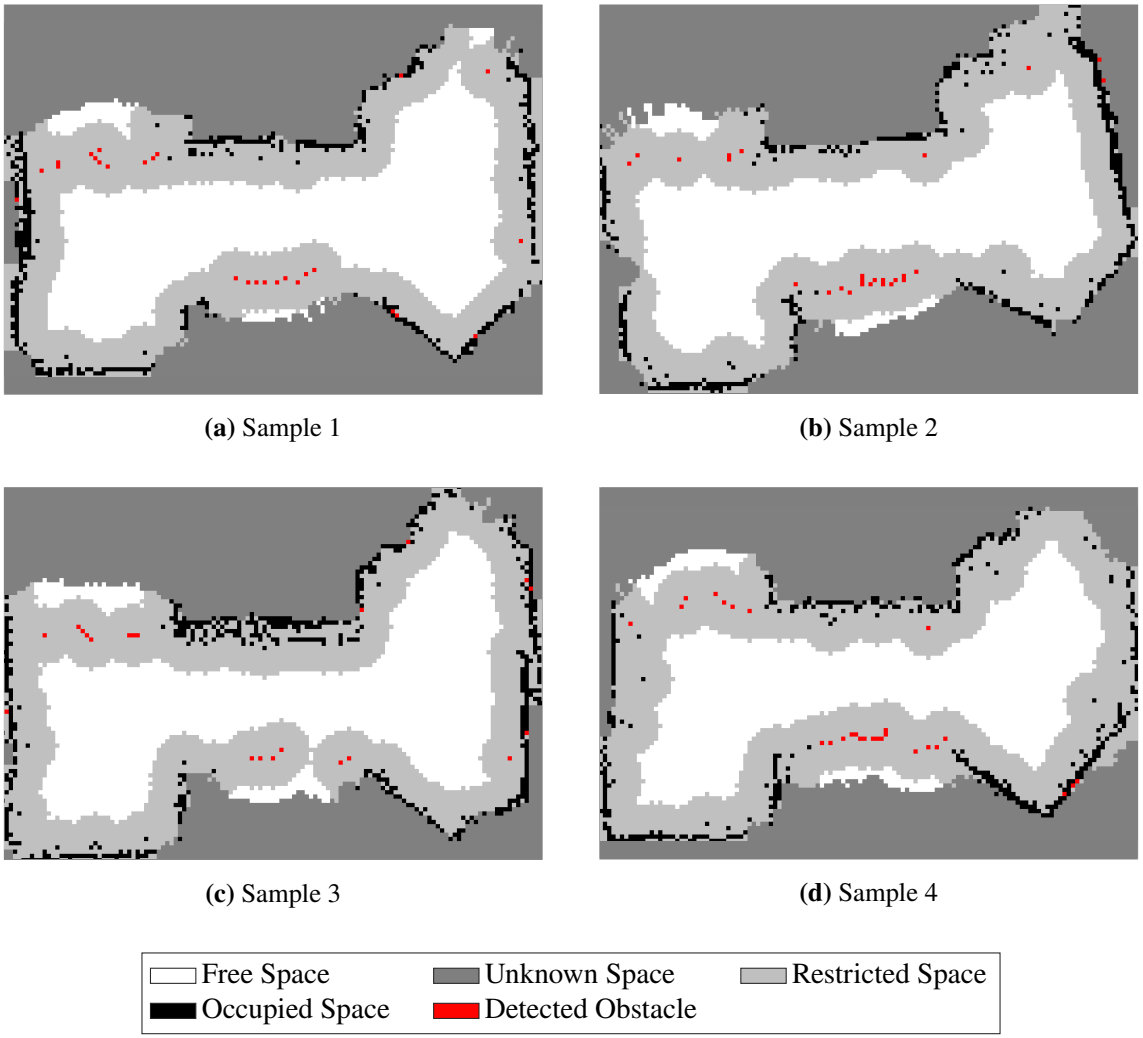
Figure 7.8: Sketch of the obstacle track

In total, four exploration missions were conducted, and the resulting maps on the Java Server are given in Figure 7.9. The obstacles detected through the obstacle detection method are marked as red voxels in the map, and these are treated equally to occupied voxels during subsequent planning iterations.

With a few exceptions, the robot performed as intended. Most importantly, collision was avoided throughout all the exploration missions. In all four samples, both negative and ground obstacles were always detected, and obstacle avoidance was then conducted. As for the overhead obstacle, detection did not occur in Samples 3 and 4.

In addition to the detection of the three obstacle types, the samples shows several examples of detected obstacles at other locations in the maps. In some cases, the corresponding red voxels are part of the surrounding wall. In Samples 2 and 4, there are red voxels which are close to the wall, but a certain distance apart from it.

Lastly, the issues which where present in the initial performance tests (see Section 3.4) are also present in the final performance test. This includes noisy representation of the walls, which is the case in all four samples. Additionally, there are some distortion in the maps, e.g. a wall in Sample 3 which is added to the map twice at slightly different locations.



(a) Sample 1

(b) Sample 2

(c) Sample 3

(d) Sample 4

Figure 7.9: Four map samples of the obstacle track, after exploration with the NRF4 robot

7.4 Discussion

As a reminder, the objective of this thesis was to to develop a method for detection and avoidance of obstacles s.a. holes, low passageways, and small items on the ground, s.t the NRF robot could explore environments with such obstacles present. The discussion will consider whether the objective is fulfilled, but also discuss specifics regarding the individual parts of the work.

7.4.1 External Hardware Module

For the most part, the external hardware module works the way it was intended in Chapter 6. From a functional perspective, the module offers all the functionality it was meant

to, but minor workarounds had to be considered due to hardware issues.

Power supply switching of the IR sensors addresses the issue that the sensors often disturb each others. Alternating switching, which is the main application of this feature, is shown to completely remove this disturbance in Figure 7.2b, despite the non-optimal response of the NMOS switch in Figure 7.1. As a result, disturbances between between sensors, which posed a significant adversity in the sensor rig design process, no longer affect the system. While alternating switching has proven to be a useful tool, it is still recommended to minimize the number of switching cycles. To elaborate, the sample period scales with the number switching cycles, and needs to be low enough s.t. an obstacle can be detected in time. Based on the test results, the total of two cycles utilized in this thesis, seems to be a suitable number. A second application of power supply switching is turning off sensors for extended periods of time. While the usage of this application was not thoroughly considered in this thesis, it may prove helpful in a multi-robot system, i.e. to avoid disturbing sensors on other robots.

When sampling with the ADC, the sampled signals initially experienced significant noise. The origin of this noise is unknown, but a missing capacitor on the ATmega168 AREF pin, which is usually good practice to include, may have contributed to the issue. Software filtering measures were attempted to remove the noise, which included discarding ADC readings and using the median to remove outliers. The test result in Figure 7.2a confirms that the filtering method works. Aside from this issue, the ADC seems to yield sensor measurements of sufficient quality.

The only major issue in the design was the power supply, as the 5V signal was fetched from the 5V pin of the nRF52840 DK, instead of the voltage regulators. As this pin could not deliver sufficient current, a temporary workaround is utilized. Instead, the hardware module is connected to the voltage regulator on the nRF shield via a breadboard wire. For high-quality sensor measurements, a stable power supply is required. Therefore, it is recommended that the servo, which is a significant source of noise, is connected to a separate power supply from the IR sensors.

7.4.2 Sensor Solution

A lot of thought was put into finding a sensor sensor solution capable of detecting ground, overhead and negative obstacles. The Sharp GP2Y0A41SK0F range sensor is arguably a good selection out of the ones considered. The main reasons are its high precision compared to the other sensors, and sufficiently high sample frequency, which was particularly important for alternating switching to work. Additionally, a total power consumption of approximately $6 \times 12\,\text{mA} = 72\,\text{mA}$ is a quite low compared to that of other alternatives. In retrospect, the selected sensor can also be concluded as a good choice as it actually ended up working for detection purposes. See Section 4.3 for a more in-depth discussion, which also includes more information about the other sensors considered.

However, the sensor has several undesirable attributes which made usage more tedious

than expected initially. As already discussed, units of this sensor tend to disturb one another when perceiving a similar area. In the sensor rig design process, configurations reducing this issue were therefore heavily sought after. With the development of alternating switching of the power supply however, this issue is now considered solved. A second issue for the sensor rig design was a reflection issue, which Figure 4.8a shows an illustration of. The solution to this problem was simpler, as it mostly disappeared when mounting the sensors in a horizontal manner, i.e. the emitter and detector are at the same height.

A third issue, which a user of the sensor solution should be aware of, is that the sensors cannot properly measure distance to low-reflective surfaces, as the test result in Figure 7.7 demonstrates. While the datasheet does not specifically address this issue, the sensors are only tested for 18% and 90% reflectance ratios, and give similar results in both cases [36]. Therefore, the sensors should only be assumed to work in the interval between these ratios.

The last issue, which is considered unsolved at the end of this thesis, occurs when an item is positioned right outside the laser line of sight. In this case, the resulting sensor output contains significant noise, which can cause issues for the obstacle detection system as a whole. Most of the sensor issues outside the ones that are already mentioned, can likely be accredited to this discrepancy. This includes the noise issues experienced with the horizontally oriented sensors which were part of sensor rig Prototype 5. In this case, the ground is the item that caused the issue. As an attempt to solve the issue, FOV limiters were added around the sensors. However, while this seemed to solve the issue in the proof of concept tests (see Section 4.5), later test results in Section 7.2.2 revealed that the solution was less robust than what was initially believed. While more time could have been used to specifically address this issue, the sensors were replaced with two downwards oriented sensors, as these had been proven to provide more stable sensor signals.

At the end of the thesis, the upwards oriented sensor still struggle with issues caused by the last sensor issue. More specifically, the upwards oriented sensor experiences this when applied certain poses, e.g. as Figure 7.5 shows. Changing the orientation of the sensor may help alleviate the issue. Otherwise, another sensing technology should be considered as replacement. Regardless, the consequences of the issue are not detrimental, as it occurs quite rarely, and only close to walls. On the few occasions the issue does occur, obstacle avoidance is initiated, and the robot continues operation.

7.4.3 Obstacle Detection

The obstacle detection functionality presented in Chapter 5 can be said to work as expected. As the test results in Section 7.2.3 shows, the robot now has the capability to detect all the obstacles mentioned in the thesis objective. The method is also easy to use, as most configuration parameters can be determined according to the sensor position and orientation (see Section 5.2.3). The only tunable variables are the three margin parameters δ_{z^-}, δ_{z^+} and δ_y, which can be found by studying measurement noise, or by simple experimentation.

The additional offset calibration feature ended up being quite important, as all downwards oriented sensors utilizes the feature. In the final sensor solution, this accounts for five out of six sensors. The reason why offset calibration is important, is that it removes bias in the sensor measurement around the reference distance. Therefore, slight differences in IR calibration, which was shown to be present in the proof of concept testing of the sensor solution (see Section 4.5), does not affect the detection system. Offset calibration may also remove other constant disturbances which have not been discovered.

A general theme during testing was that when false positive detections occur, the sensor solution was liable, not the obstacle detection method itself. Sensor solution discrepancies are already discussed in Section 7.4.2. The obstacle detection method may also fail to detect particularly thin items s.a. posts, as there are gaps between the downwards oriented sensors which the items can slip in between.

7.4.4 Obstacle Avoidance

The obstacle avoidance method presented within Chapter 5 also functions as expected. Most importantly, the method ensures the robot does not collide once an obstacle is detected, i.e. as the final performance test in Section 7.3 shows.

While the method has experienced no major issues during testing, several theoretical issues does exists. Firstly, the robot can collide during turning, despite the fact that turning is assumed to be a safe move within this thesis. For turning to be completely collision safe however, the robot would have to be completely circular.

The second issue is that when an obstacle is added to the Java server, s.t. the path planner can take the obstacle into account, the obstacle is added permanently. Therefore, for large-scale environments, where the estimated robot pose may drift over time, the added obstacles may cause problems for navigation.

Lastly, this method only considers non-moving obstacles, and may have to be extended to avoid moving obstacles s.a. other robots. Hence, the method cannot be considered a complete Traffic Collision Avoidance System (TCAS).

7.4.5 Final Performance

The final performance test in Section 7.3 demonstrates that the robot now can detect and avoid obstacles s.a. holes, low passageways, and small items on the ground. The test was conducted while the robot simultaneously explored an unknown environment, as it was previously capable of. As no collision occurred during exploration, the test is considered a success.

Several discrepancies did reveal themselves in this test. As discussed in Section 7.4.2, sensor noise affected the upwards oriented sensor in certain robot poses. In the final performance test, this occurred in two of the four samples. However, as this only happened in

proximity to the walls, the issue did not have a great impact on the overall performance. Another discrepancy, which did not really affect the system in any negative way, was that the robot detected the wall when moving too close. Furthermore, the test results also revealed that the robot is lacking in terms of mapping, as noise and distortions in the map were still present. As these issues have not been a priority throughout this work, no efforts of improvement have been made. Therefore, the initial performance tests in Section 3.4 are referred to for more discussion on these issues. Lastly, the robot only detected the overhead obstacle in two out of the four samples. This is not a discrepancy, as the developed functionality is an obstacle detection and avoidance system, not a full 3D mapping solution. In this work, obstacles are avoided upon detection, but the robot does not actively seek areas which has not yet been perceived by the sensor solution.

In conclusion, the final performance test confirms that the thesis objective has been fulfilled. The robot is now capable of exploring more advanced environments, with obstacles located in all three dimensions.

8 | Future Work

This chapter suggests topics for future work. More specifically, future work is natural continuations of this thesis, areas which were not prioritized, or discrepancies which can be fixed and improved upon.

8.1 Improvements to Existing System

Both the initial and the final performance tests revealed that the maps which the NRF4 robot constructed were prone to issues s.a noise and distortion. As the NRF5 and NRF6 robots are constructed in the same manner, it is assumed these robots experience similar issues. Therefore, future work should aim to improve the robot's mapping capabilities.

While the robot hardware can generally be considered to be of high quality, IR sensors will often experience noise when connected to the same power supply as the servo. While the sensor solution is connected to a separate power supply, the IR sensors on the tower are not. Hence, if hardware improvements are planned at some point, the servo should be connected to a power supply separate from other hardware components.

As for the software on the nRF52840 DK, there exist several global variables in the source code, s.a. `gHandshook` and `gPaused`. This is bad practice, and can be confusing for students working on the robot project, and consequently, future work should aim to remove these variables.

Lastly, communication with the Java server can be quite complicated, as the robot can often suddenly lose connection. As a result, time and resources are wasted, even for projects only indirectly related to the server communication. Improvements to the `ble_comminication` source code, or the adaption of new protocols s.a. Thread should be one of the main priorities in the coming years.

8.2 Improvements to Obstacle Detection and Avoidance

While the work conducted in this thesis can be mostly considered completed, there exist parts which can be improved upon. Most importantly, development of a second version

of the external hardware module can be considered, which includes proper power supply connectors. Furthermore, despite sensor measurements being quite stable after filtering, a capacitor should be connected to the AREF pin for good practice.

Another improvement is to find a better way of adding obstacles to the Java server. As of now, the obstacles are added permanently, which means than noise from the sensors, or bias in pose estimation, will cause great problems for robot navigation over time. Test results have also shown that sensor noise does appear in certain robot poses, i.e. from the upwards oriented sensor. Minor changes to the sensor solution or software filtering may be considered to remove the noise from the map. Lastly, in-depth verification with the OptiTrack positioning system on B333 should be conducted, to further evaluate the system performance.

8.3 Continuations of this Thesis

Utilizing the obstacle detection and avoidance system developed throughout this thesis, future work should explore more applications of the system. One such application, is obstacle detection and avoidance in a multi-robot system. A complete Traffic Collision Avoidance System (TCAS) would be able to handle both the non-moving obstacles considered in this thesis, while simultaneously avoid collision with other robots.

A second continuation is to develop functionality s.t. the obstacle detection and avoidance system can work with the C++ server. Before this however, basic mapping and exploration using the C++ server, as was demonstrated with the Java server in the initial performance tests in Section 3.4, must be working.

Bibliography

[1] J. Stüper, "Lego mindstorms ev3 robot," Specialization Thesis, NTNU, Trondheim, 2015.

[2] S. R. Nilssen, "Arduino slam robot," Specialization Thesis, NTNU, Trondheim, 2017.

[3] E. Jølsgard, "Embedded nRF52 robot," Specialization Thesis, NTNU, Trondheim, 2020.

[4] E. Leithe, "Embedded nrf52 robot," Specialization Thesis, NTNU, Trondheim, 2018.

[5] S. R. Nilssen, "Developement of a real-time embedded control system for slam tobots," Master's Thesis, NTNU, Trondheim, 2018.

[6] S. M. K. Jensen, "Autonom-robot med lidar-sensor," Master's Thesis, NTNU, Trondheim, 2018.

[7] E. Gulbrandsen, "Controlling the nrf52-robot using matlab generated code," Specialization Thesis, NTNU, Trondheim, 2020.

[8] E. Sjøvold, "Position estimation on an nrf52-robot using matlab," Specialization Thesis, NTNU, Trondheim, 2021.

[9] G. Berglund, "Embedded nrf52840 dk robot," Specialization Thesis, NTNU, Trondheim, 2020.

[10] K. Håland, "Improving navigation in the nrf52 robot," Master's Thesis, NTNU, Trondheim, 2021.

[11] M. G. Rødseth and T. E. S. Andersen, "System for self-navigating autonomous robots".," Master's Thesis, NTNU, Trondheim, 2016.

[12] E. Thon, "Mapping and navigation for collaborating mobile robots," Master's Thesis, NTNU, Trondheim, 2016.

[13] K. Lien, "Embedded utvikling på en fjernstyrt kartleggingsrobot," Master's Thesis, NTNU, Trondheim, 2017.

[14] T. Grindvik, "Creating the foundations of a graphical slam application in modern c++.," Master's Thesis, NTNU, Trondheim, 2019.

[15] M. Mullins, "Nrf52 with openthread," Master's Thesis, NTNU, Trondheim, 2020.

[16] S. Murad, "Implementing mqtt for nrf52840," Master's Thesis, NTNU, Trondheim, 2021.

[17] A. Hunshamar, "C++ server for simultaneous localisation and mapping in robotic system," Master's Thesis, NTNU, Trondheim, 2020.

[18] M. S. Mullins, "Implementation of simultaneous localisation and mapping in robotic system using the improved rao-blackwellized particle filter," Master's Thesis, NTNU, Trondheim, 2020.

[19] Nordic Semiconductor, *nRF52840 DK*, https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk, Accessed: 2021-09-07.

[20] E. Jølsgard, "Shield for embedded nrf52840-dk robot," Specialization Course Report, NTNU, Trondheim, 2020.

[21] Machifit, *Machifit 25GA370 DC 12V Micro Gear Reduction Encoder Motor*, https://www.banggood.com/Machifit-25GA370-DC-12V-Micro-Gear-Reduction-Encoder-Motor-with-Mounting-Bracket-and-Wheel-p-1532242.html?cur_warehouse=CN&ID=6157423, Accessed: 2021-09-17.

[22] Uxcell, *uxcell DC 12V 220RPM Encoder Gear Motor*, https://www.amazon.com/gp/product/B078HYX7YH/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1, Accessed: 2021-09-07.

[23] Invensense, *ICM-20948 datasheet*, https://cdn.sparkfun.com/assets/7/f/e/c/d/DS-000189-ICM-20948-v1.3.pdf, Accessed: 2021-09-07.

[24] DGServo, *Servo - Generic Metal Gear (Micro Size)*, https://www.sparkfun.com/products/14760, Accessed: 2021-09-24.

[25] Sharp Electronics, *Sharp GP2YA21YK0F datasheet*, https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf, Accessed: 2021-09-07.

[26] A. Stenset, "Nrf52 robot with openthread," Master's Thesis, NTNU, Trondheim, 2020.

[27] FreeRTOS, *Real-time operating system for microcontrollers*, https://www.freertos.org/index.html, Accessed: 2021-09-08.

[28] E. Leithe, "Embedded nrf52 robot," Master's Thesis, NTNU, Trondheim, 2019.

[29] T. Andersen, "Feature extraction implementation for nrf52840-based robot," Specialization Thesis, NTNU, Trondheim, 2021.

[30] Optitrack, *Motion Capture Systems*, https://optitrack.com/, Accessed: 2021-09-21.

[31] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2016, ISBN: 978-3-319-32550-7. DOI: 10.1007/978-3-319-32552-1.

[32] Sharp, *Distance Measuring Sensor*, http://global.sharp/products/device/lineup/selection/opto/haca/diagram.html, Accessed: 2021-09-27.

[33] Garmin, *Lidar Lite v3 Operation Manual and Technical Specifications*, https://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf, Accessed: 2021-09-27.

[34] Slamtec, *RPLIDAR A1*, https://www.slamtec.com/en/Lidar/A1, Accessed: 2021-09-27.

[35] Elecfreaks, *Ultrasonic Ranging Module HC - SR04*, https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf, Accessed: 2021-09-27.

[36] Sharp Electronics, *Sharp GP2YA41SK0F datasheet*, https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a41sk_e.pdf, Accessed: 2021-11-09.

[37] Dassault Systems, *SolidWorks*, https://www.solidworks.com/, Accessed: 2021-09-29.

[38] Ultimaker, *Ultimaker 2+ Connect*, https://ultimaker.com/3d-printers/ultimaker-2-plus-connect, Accessed: 2022-01-05.

[39] Make NTNU, *Make NTNU*, https://makentnu.no/, Accessed: 2022-01-05.

[40] Atmel, *ATmega48/V / 88/V / 168/V*, https://no.mouser.com/datasheet/2/268/Atmel-2545-8-bit-AVR-Microcontroller-ATmega48-88-1-1315326.pdf, Accessed: 2021-11-09.

[41] ON Semiconductor, *BS170 / MMBF170 N-Channel Enhancement Mode Field Effect Transistor*, https://www.onsemi.com/pdf/datasheet/mmbf170-d.pdf, Accessed: 2021-11-09.

[42] KiCad, *KiCad EDA - A Cross Platform and Open Source Electronics Design Automation Suite*, https://www.kicad.org/, Accessed: 2021-11-09.

[43] Mouser Electronics, *FREE HIGH QUALITY PCB LIBRARIES FOR ECAD TOOLS*, https://mouser.componentsearchengine.com/pcb-libraries.php, Accessed: 2022-01-02.

A | IR Sensor Calibration Parameters

Table A.1: IR sensor parameters for the NRF4, NRF5 and NRF6 robots

	Sensor	β	α	R
NRF4	Sensor 1	778908	-1.168136	0.9995
	Sensor 2	1013082	-1.203120	0.9997
	Sensor 3	1043658	-1.206453	0.9989
	Sensor 4	875430	-1.180489	0.9996
NRF5	Sensor 1	1050635	-1.207230	0.9991
	Sensor 2	758062	-1.162092	0.9997
	Sensor 3	1068250	-1.206257	0.9995
	Sensor 4	1120601	-1.210874	0.9997
NRF6	Sensor 1	1026143	-1.206862	0.9987
	Sensor 2	874521	-1.186358	0.9988
	Sensor 3	1393666	-1.247142	0.9990
	Sensor 4	1051848	-1.209854	0.9997

B | Sensor Rig CAD Models

B.1 Prototype 1

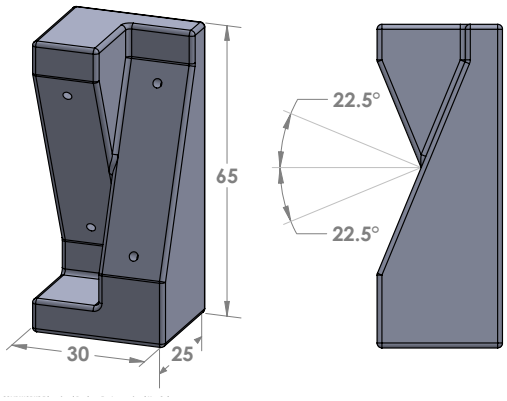


Figure B.1: CAD model of Prototype 1 with dimensions [mm]

B.2 Prototype 2

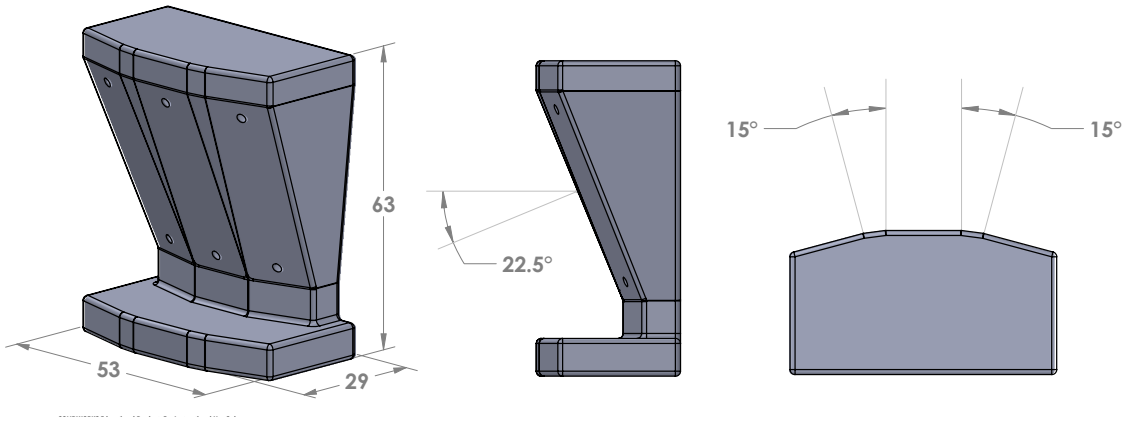


Figure B.2: CAD model of Prototype 2 with dimensions [mm]

B.3 Prototype 3

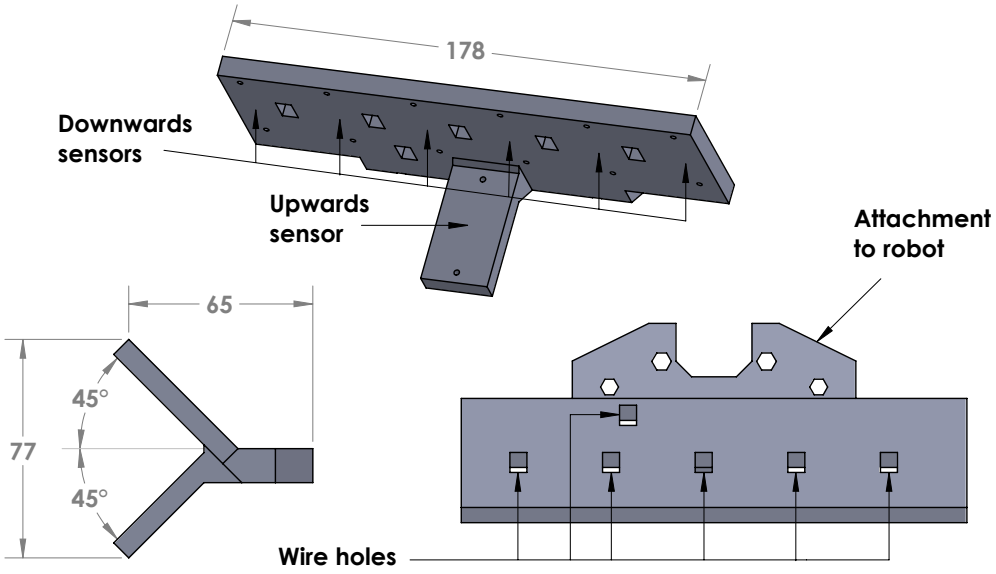


Figure B.3: CAD model of Prototype 3 with dimensions [mm]

B.4 Prototype 4

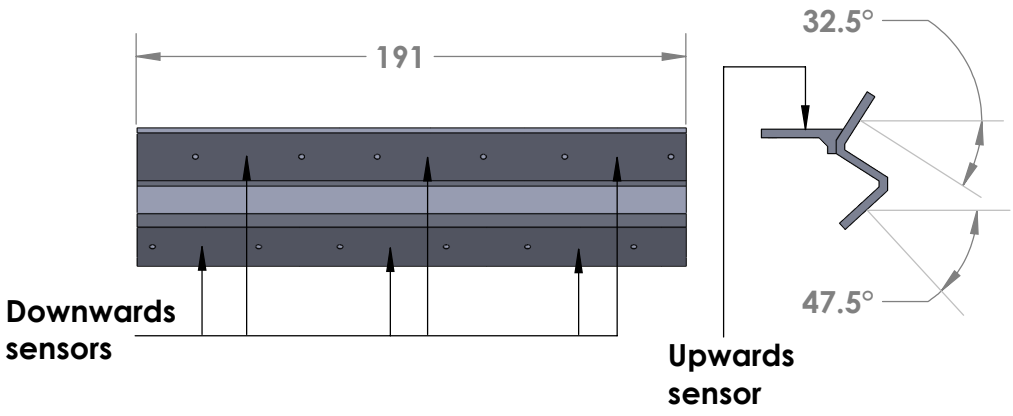
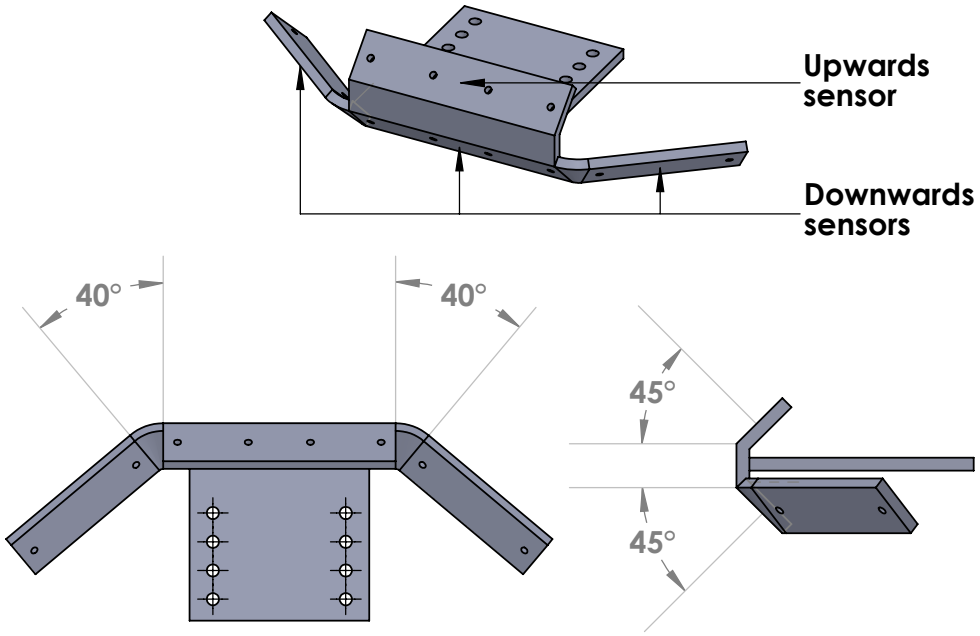
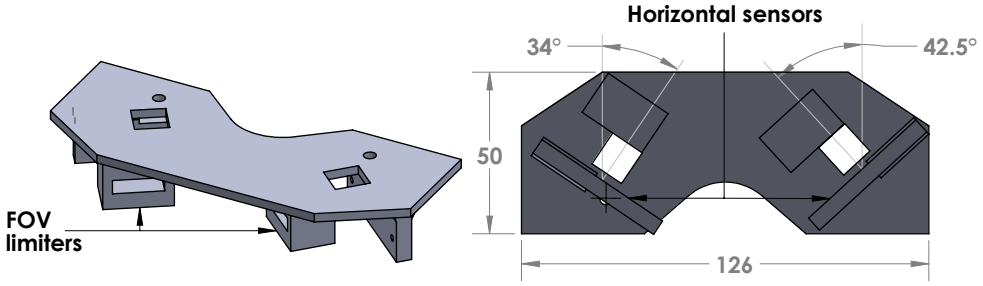


Figure B.4: CAD model of Prototype 4 with dimensions [mm]

B.5 Prototype 5



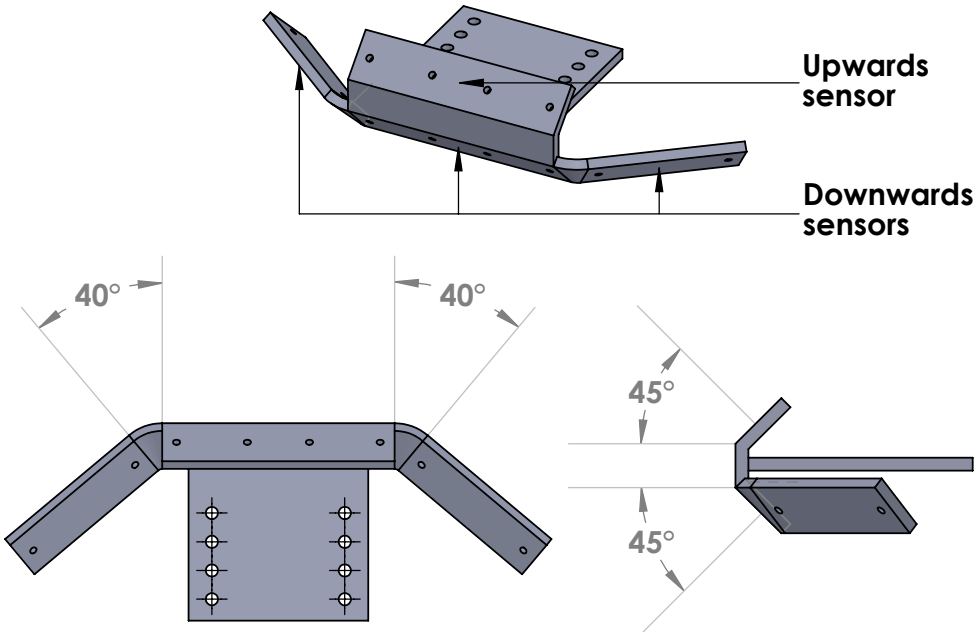
(a) Part 1



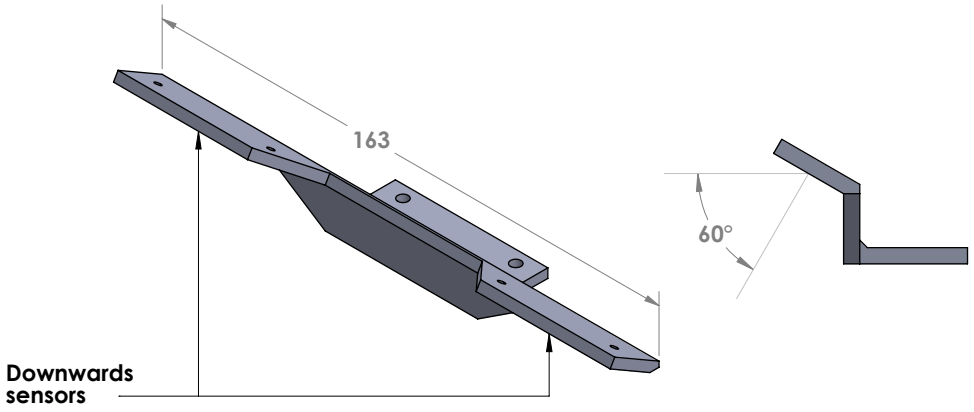
(b) Part 2

Figure B.5: CAD model of Prototype 5 with dimensions [mm]

B.6 Prototype 6



(a) Part 1



(b) Part 2

Figure B.6: CAD model of Prototype 6 with dimensions [mm]

C | Sensor Rig Simulations

C.1 Prototype 1

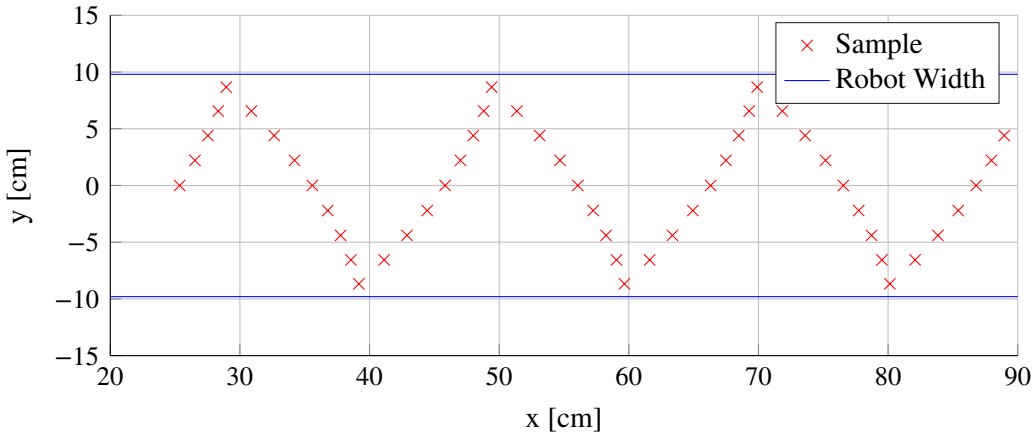


Figure C.1: Simulation of how samples are scattered on the ground plane with the sensors in Prototype 1. The robot moves at a maximum velocity of 0.32 m s^{-1} along the x-axis, and the servo moves 5° per measurement every 0.04 s, across a 45° interval. The vertical position of the IR sensor is 10.5 cm above the ground, and the downwards angle is 22.5°. The simulation time is 2 s.

C.2 Prototype 2

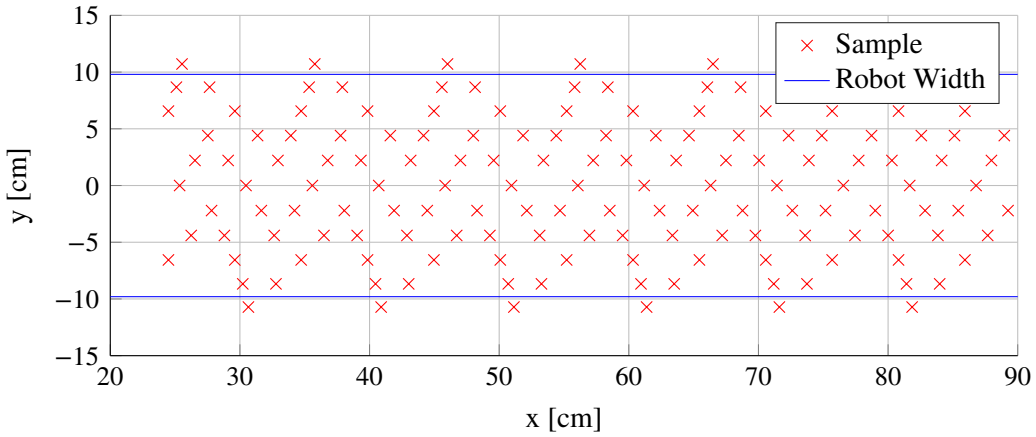


Figure C.2: Simulation of how samples are scattered on the ground plane with the sensors in Prototype 2. The robot moves at a maximum velocity of 0.32 m s^{-1} along the x-axis, and the servo moves 5° per measurement every 0.04 s, across a 30° interval. The vertical position of the IR sensors is 10.5 cm above the ground, and the downwards angle is 22.5°. The sensors are mounted with 15° between them. The simulation time is 2 s.

D | Schematics and PCB Layout

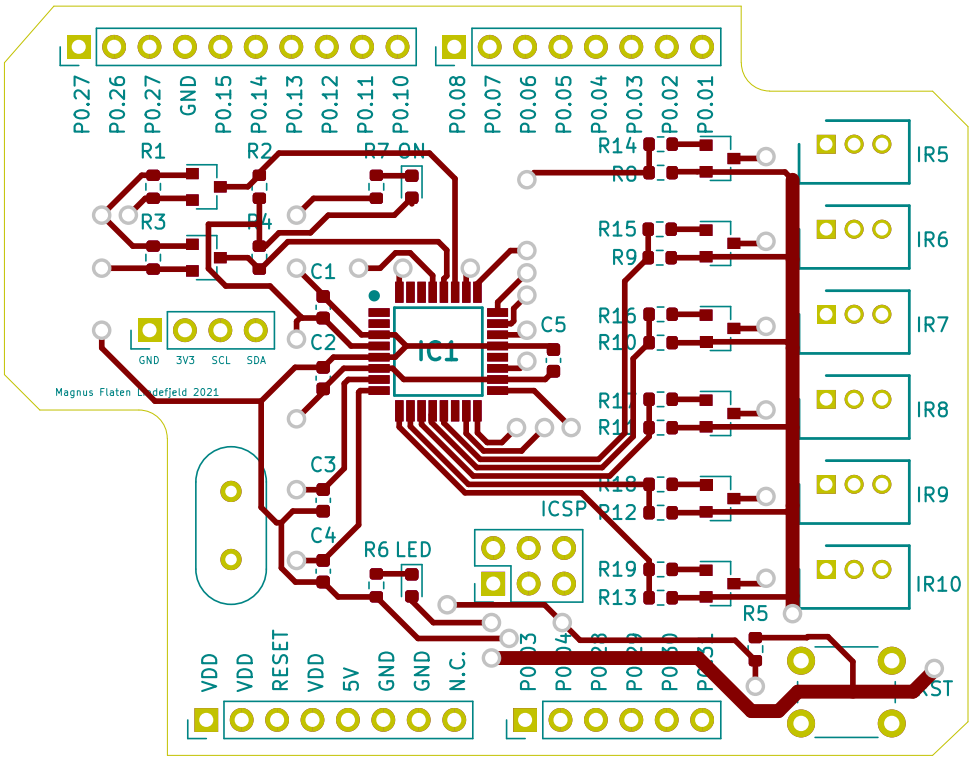


Figure D.1: Front layer of PCB

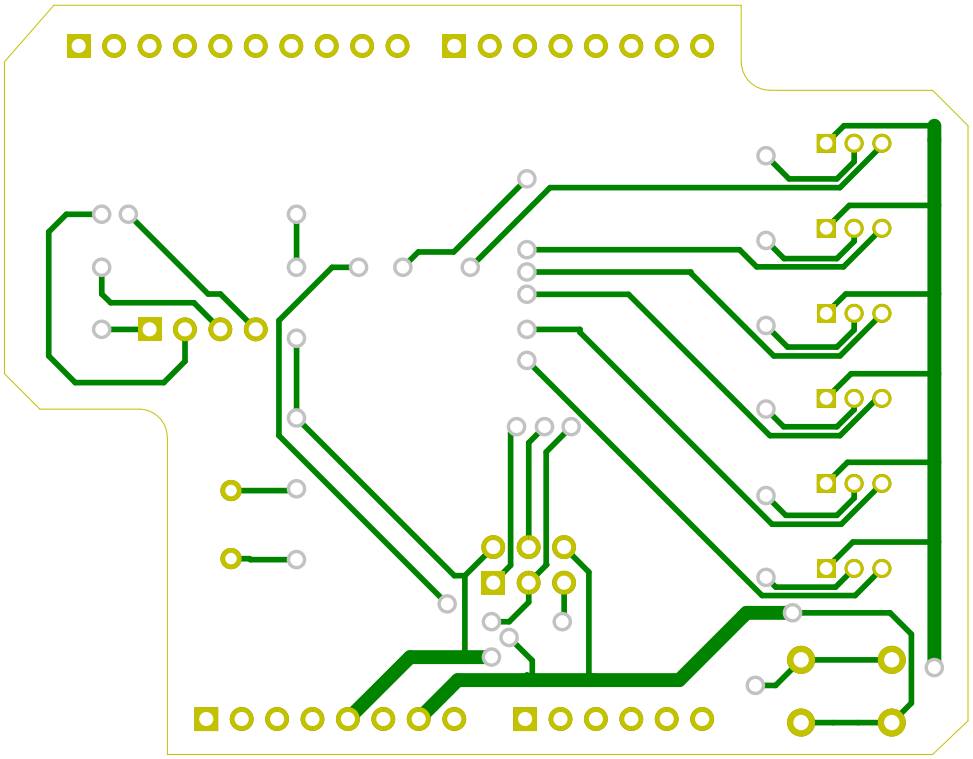


Figure D.2: Bottom layer of PCB

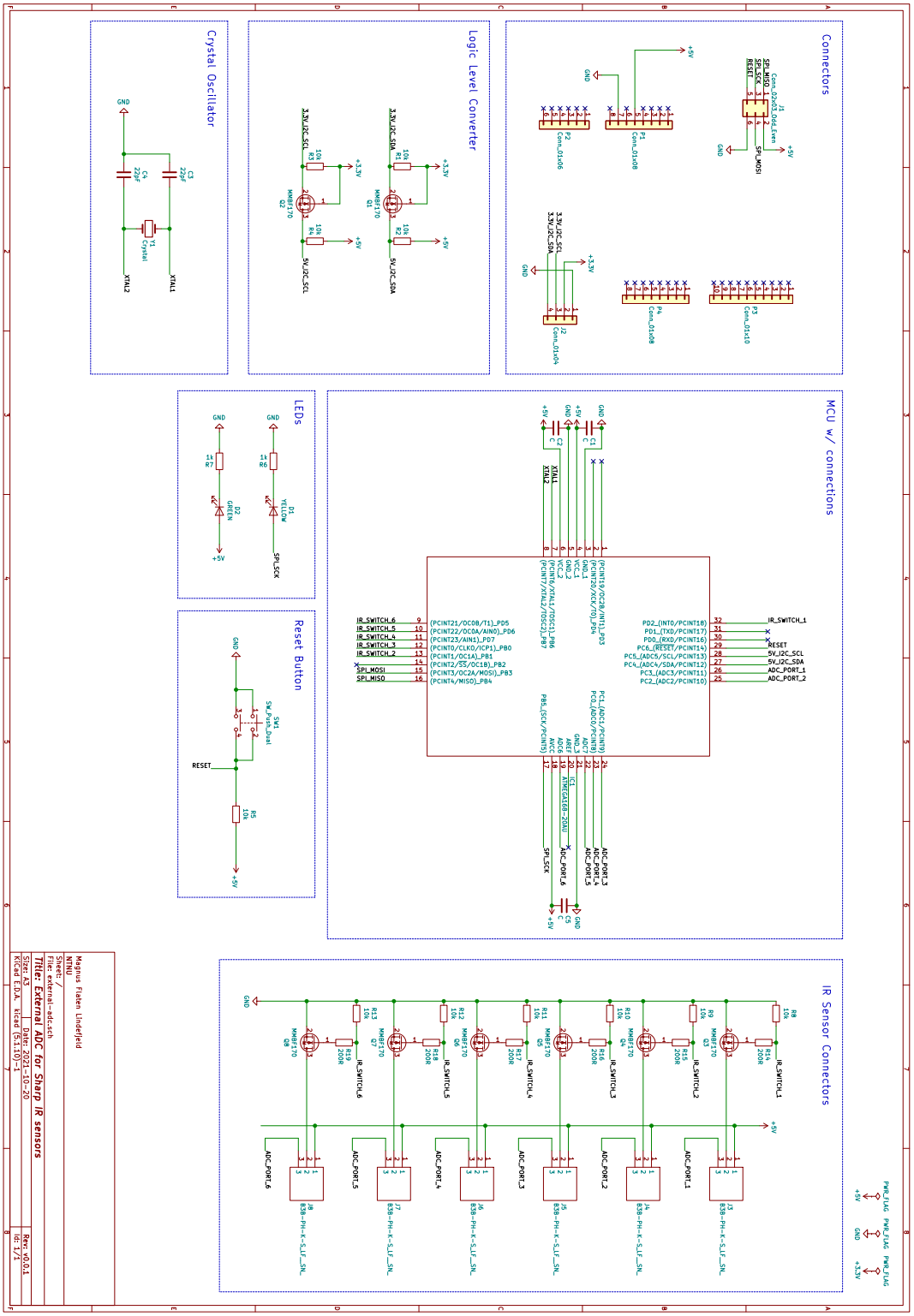


Figure D.3: Schematics for external hardware module

E External Hardware Module Registers

E.1 Register Map

Table E.1: User Bank 0

ADDR (Hex)	ADDR (Dec)	Register Name	Serial I/F
00	0	BANK_SEL	R/W
01	1	WHO_AM_I	R
02	2	SENSOR_TYPE	R/W
03	3	POWER_TRIG	R/W
04	4	TOTAL_SWITCHING_CYCLES	R/W
05	5	CAL_TRIG	R/W
06	6	TOTAL_CAL_SAMPLES	R/W
07	7	DETECTION_STATUS	R
08	8	EXT_SENSOR_1_DATA_H	R
09	9	EXT_SENSOR_1_DATA_L	R
0A	10	EXT_SENSOR_2_DATA_H	R
0B	11	EXT_SENSOR_2_DATA_L	R
0C	12	EXT_SENSOR_3_DATA_H	R
0D	13	EXT_SENSOR_3_DATA_L	R
0E	14	EXT_SENSOR_4_DATA_H	R
0F	15	EXT_SENSOR_4_DATA_L	R
10	16	EXT_SENSOR_5_DATA_H	R
11	17	EXT_SENSOR_5_DATA_L	R
12	18	EXT_SENSOR_6_DATA_H	R
13	19	EXT_SENSOR_6_DATA_L	R

Table E.2: User Bank 1-6

ADDR (Hex)	ADDR (Dec)	Register Name	Serial I/F
01	1	`EXT_SENSOR_X_POWER_MODE`	R/W
02	2	`EXT_SENSOR_X_SWITCHING_CYCLE`	R/W
03	3	`EXT_SENSOR_X_DETECTION_MODE`	R/W
04	4	`EXT_SENSOR_X_CAL_ENABLE`	R/W
05	5	`EXT_SENSOR_X_DIST_REF_H`	R/W
06	6	`EXT_SENSOR_X_DIST_REF_L`	R/W
07	7	`EXT_SENSOR_X_THRESH_H`	R/W
08	8	`EXT_SENSOR_X_THRESH_L`	R/W
09	9	`EXT_SENSOR_X_DIVIDER_1`	R/W
0A	10	`EXT_SENSOR_X_DIVIDER_2`	R/W
0B	11	`EXT_SENSOR_X_DIVIDER_3`	R/W
0C	12	`EXT_SENSOR_X_DIVIDER_4`	R/W
0D	13	`EXT_SENSOR_X_EXPONENT_1`	R/W
0E	14	`EXT_SENSOR_X_EXPONENT_2`	R/W
0F	15	`EXT_SENSOR_X_EXPONENT_3`	R/W
10	16	`EXT_SENSOR_X_EXPONENT_4`	R/W

E.2 Bank 0 Register Descriptions

Register 0 - `BANK_SEL`

This register specifies which bank to address. There are seven available banks in total. Bank 0 holds common settings and all output data. Banks 1-6 hold individual configurations for the IR sensors.

Register 1 - `WHO_AM_I`

This register holds the I2C address, which is `0xA2`. To check whether there is a connection to the external hardware module, it is recommended that this register is read.

Register 2 - `SENSOR_TYPE`

This register specifies the sensor type, which is used to set the sample period of the module. The external hardware module offers support for two sensors. The values that can be set in this register are:

0. `SENSOR_UNKNOWN` - Sample period: 52.9 ms. Default option.
1. `GP2Y0A21YK0F` - Sample period: 52.9 ms
2. `GP2Y0A41SK0F` - Sample period: 25.2 ms

Register 3 - `POWER_TRIG`

This register is used to turn on or off all sensors. There are three values associated with this register

0. `POWER_TRIG_OFF` - Trigger is off. The register is always reset to this value after the register is written to. Default option.
1. `POWER_OFF` - All sensors are powered off.
2. `POWER_ON` - All sensors are powered on.

Register 4 - `TOTAL_SWITCHING_CYCLES`

This register specifies the total number of switching cycles, where each cycle lasts one sample period. Switching can be used to power on and off sensors in succession to reduce interference between them. If 0 (`SWITCHING_OFF`) or 1 is written, switching is turned off. Note that switching has lower precedence than the `EXT_SENSOR_X_POWER_MODE` registers.

Register 5 - `CAL_TRIG`

This register triggers offset calibration, which can used for sensors where accuracy is especially important (see Section 5.2.2). When 1 is written to this register, calibration is triggered. Then, for all sensors where `EXT_SENSOR_X_CAL_ENABLE` is set to 1, offset is calibrated. The following values are associated with this register.

0. `CAL_TRIG_OFF` - Trigger is off. Default option.
1. `CAL_TRIG_ON` - Trigger is on. While the calibration is in process, this option is set.

Register 6 - `TOTAL_CAL_SAMPLES`

This register specifies the total number of samples which are averaged to find the adjusted offset. The default option is 200 samples.

Register 7 - `DETECTION_STATUS`

This register holds the detection status for all six sensors. If a bit i is set high, then sensor $i+1$ has detected an obstacle. Bit 6 and 7 are not used. By default obstacle detection is not enabled. To enable obstacle detection for a single sensor, the following registers has to be modified for each sensor:

- `EXT_SENSOR_X_DETECTION_MODE`
- `EXT_SENSOR_X_DIST_REF`
- `EXT_SENSOR_X_THRESH`

- EXT_SENSOR_X_DIVIDER
- EXT_SENSOR_X_EXPONENT

Register 8-19 - EXT_SENSOR_X_DATA

These registers hold 16-bit output data from the sensors. The format of the output data is determined by the content of the EXT_SENSOR_X_DETECTION_MODE registers. For DETECTION_MODE_0, the raw 10-bit ADC reading is written to the corresponding register. Otherwise, the data in mm is written.

E.3 Bank 1-6 Register Descriptions

Register 0 - EXT_SENSOR_X_POWER_MODE

This register specifies whether the sensor is powered on or off. The content of the register can be changed by writing directly, or using the POWER_TRIG register in bank 0. The valid options for the EXT_SENSOR_X_POWER_MODE are

1. POWER_OFF - The sensor is powered off.
2. POWER_ON - The sensor is powered on. Default option.

Register 1 - EXT_SENSOR_X_SWITCHING_CYCLE

This register specifies which cycle the sensor is turned on when switching is enabled. The cycle must be less or equal to the value in the TOTAL_SWITCHING_CYCLES register. If the value is 0 (SWITCHING_OFF), the sensor is always turned on.

Register 2 - EXT_SENSOR_X_DETECTION_MODE

This register configures the detection mode of the sensor, which determines if and how an obstacle is detected. Additionally, the register also determines the format of the output data written to the EXT_SENSOR_X_DATA registers. The available detection modes are:

0. DETECTION_MODE_0 - Raw ADC data. Obstacle detection disabled. Default option.
1. DETECTION_MODE_1 - Data in mm. Obstacle detection disabled.
2. DETECTION_MODE_2 - Data in mm. Obstacle detection enabled.
3. DETECTION_MODE_3 - Data in mm. Obstacle detection enabled.
4. DETECTION_MODE_4 - Data in mm. Obstacle detection enabled.

For exactly how detection works in detection modes 2-4, see Figure 5.1.

Register 3 - `EXT_SENSOR_X_CAL_ENABLE`

This register specifies whether offset adjustment calibration is enabled for the sensor. Calibration occurs when `CAL_TRIG` register is set, and the number of samples are determined by the `TOTAL_CAL_SAMPLES` register. Calibration is performed with the specified switching settings, which may prolong the operation. Furthermore, ensure that the sensors are turned on during calibration, i.e. the `EXT_SENSOR_X_POWER_MODE` registers holds `POWER_ON`. The available options for this register are:

0. `CAL_OFF` - Calibration disabled. Default option.
1. `CAL_ON` - Calibration enabled

Register 5-6 - `EXT_SENSOR_X_DIST_REF`

These registers specify the reference distance in mm which is used for obstacle detection. The value is unsigned. For how the reference distance ties into obstacle detection, see Figure 5.1.

Register 7-8 - `EXT_SENSOR_X_THRESH`

These registers specify the threshold in mm which is used for obstacle detection. The value is unsigned. For how the threshold ties into obstacle detection, see Figure 5.1.

Register 9-12 - `EXT_SENSOR_X_DIVIDER`

These registers specify the divider for converting ADC readings to mm. See Section 3.2.1 for how this conversion works. The value of the divider is a float, and is stored in the registers as four bytes. Appendix E.4 shows and example of how a float is converted into four bytes and vice versa.

Register 13-16 - `EXT_SENSOR_X_EXPONENT`

These registers specify the exponent for converting ADC data to mm. See Section 3.2.1 for how this conversion works. The value of the divider is a float, and is stored in the registers as four bytes. Appendix E.4 shows and example of how a float is converted into four bytes and vice versa.

E.4 Float to Bytes Conversion Example

```
void registers_write_float(float data, uint8_t addr, uint8_t bank) {

    uint8_t bytes[sizeof(data)];
    memcpy(&bytes, &data, sizeof(data));

    for (uint8_t i = 0; i < sizeof(data); i++)
        registers_write_uint8(bytes[i], addr + i, bank);

}

float registers_read_float(uint8_t addr, uint8_t bank) {

    float data;
    uint8_t bytes[sizeof(data)];

    for (uint8_t i = 0; i < sizeof(data); i++)
        bytes[i] = registers_read_uint8(addr + i, bank);

    memcpy(&data, &bytes, sizeof(data));

    return data;
}
```

Figure E.1: Example of conversion from float to four bytes and vice versa

Magnus Flaten Lindefjeld

Obstacle Detection and Avoidance in a 3D Environment

NTNU
Norwegian University of
Science and Technology