UiO : Department of Informatics
University of Oslo

A Low-Cost Indoor Positioning System

Eurobot (2015)

André Kramer Orten

Master's Thesis Autumn 2015

A Low-Cost Indoor Positioning System

André Kramer Orten

3rd August 2015

Abstract

Eurobot is an annual robotics competition. Teams from different countries composed of mostly students but also independent teams participate. This year's title was Robomovies, and the competition were held in Yverdon-Les-Bains in Switzerland. It is the first year UiO participated in the competition, and all parts of the robot needed to be developed. The game table was also build according to the measures and specifications listed in the rules. This has been designed so that it easily can be modified to fit future year's tasks.

The main goal of this thesis is to research technologies and establish the foundation for coming participants in the Eurobot competition from University of Oslo (UiO). This thesis focus on the navigation and position system of the robot and the sensor system. A low cost beacon based positioning system using infrared codes and a rotating tower has been developed and tested. A Kalman filter was used to combine the odometry and the beacon system for a better position estimate. However, the beacon system did not work sufficiently when the robot was moving and rotating. All the code for the navigation and sensor part have been written in C++. A silicone wheel was designed in order to improve the grip ensuring that no wheelspin would occur making the position based on the odometry quite reliable. In order to avoid collision, proximity sensors were mounted both on the front and the back of the robot.

A personal Computer (PC) was used to control the robot. For the communication between all the different systems the message library Zero Message Queue (ZMQ) has been used. ZMQ is a high-performance asynchronous messaging library. It is one of the easiest message library to use, as well as being one of the fastest. And worked well in this project.

Acknowledgment

I would like to thank all team members for a good collaboration through the project.

Further i would like to thank my two supervisors, Kyrre Harald Glette and Kim Mathiassen for valuable input and ideas through the work on this thesis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robotics and automation plays an increasingly important role in the industry today[3]. Among other, is the oil industry getting more and more autonomous and the development of autonomous robots that can do several tasks will be more important. A lot of tasks that are manually done today, will be autonomous in the future. This does not necessarily mean there will be fewer jobs, but it will create a demand for other types of jobs such as operating and maintaining the autonomous robots[34]. The automation of private homes is also something that will receive more attention. These robots will do different tasks that normally would be done by a person. One application that has come into private homes today is autonomous vacuum cleaners and lawn mowers. Navigation and sensors are a really important part for a robot to be autonomous, and are therefore very important parts of robotics. Sensors are an important part for autonomous robots, and act as their eyes. They can provide vital information of obstacles and make the foundation for a robot to execute different tasks. In this master's thesis a robotics competition (Eurobot) will work as an arena for research in development of an autonomous robot. The tasks in this competition are specific, however the systems developed can be used in different settings that are not related to the competition. Therefore does this competition provide valuable research regarding the theory as well as implementation of an autonomous vehicle. It also provides an overview of what is needed in order to develop a robot from the sensors, control system and development platforms to the artificial intelligence and path-planning algorithm in order to accomplish a set of tasks in a changing environment.

1.2 Master's thesis approach

The objective of this master's thesis is to develop and create an autonomous robot to solve tasks given in the Eurobot competition 2015. As this is the first year that UiO participates in the competition all parts of the robot needs to be developed from scratch, from sensors and positioning system to power system and the programming of the robot. The main approach is to make this as low cost and simple as possible, while researching different approaches so that future Eurobot

teams from UiO can have a better starting point. The research looks at what sensors and equipment that are needed.

1.3 Research goals

The goal is to gather valuable knowledge of systems and sensors for autonomous robots, and to build the foundations for future participants for the Eurobot competition from UiO.

An autonomous vehicle will be developed for use in the Eurobot competition 2015. There will be a lot of work on the actual robot, and communication system between AI, motor controller system and position and sensors. The Eurobot competition will be used as an arena to gather knowledge in the development in autonomous robots. Even though the robot is developed specifically for this competition, the technology and systems used and developed could be used in other areas.

1.4 The Eurobot competition

This project is part of UiO's contribution to the Eurobot challenge in May 2015. Eurobot is an annual robotics competition between different teams from different European countries, where the goal is to make an autonomous robot to solve different tasks that are given each year. The tasks are each year limited to a playing table with size 3 by 2 meters, but the specific tasks are changed each year. The competition started in 1998, and took place in France. Since then, more and more teams from different countries have joined. This is the first year that UiO takes part in this competition, so every part of the robot is made from scratch. A lot of the work with the master project will go in researching and testing systems and finding out what will work and what will not work. In 2015 the competition took place in Yverdon-les-Bains in Switzerland at 22-25 of May.

1.4.1 The team

The team from UiO consists of three master students, all writing about Eurobot in their theses. We have divided the work into different parts where we have our main work areas. There are of course several areas that we worked on together to be able to get the best integration between the systems, and find the best tactics and design for the robot.

- **Bendik Kvamstad** Master student at Robotics and Intelligent Systems at the Department of Informatics, and is working on the artificial intelligence, and path planning of the robot.
- **Eivind Wikheim** Master student at Robotics and Intelligent Systems at the Department of Informatics, and is working on the control of the motors, arms and other functionality the robot needs to be able to sovle the different tasks.
- **André Kramer Orten** Master student at Robotics and Intelligent Systems at the Department of Informatics, and is working on the positioning of the robot, and the sensors that it needs.

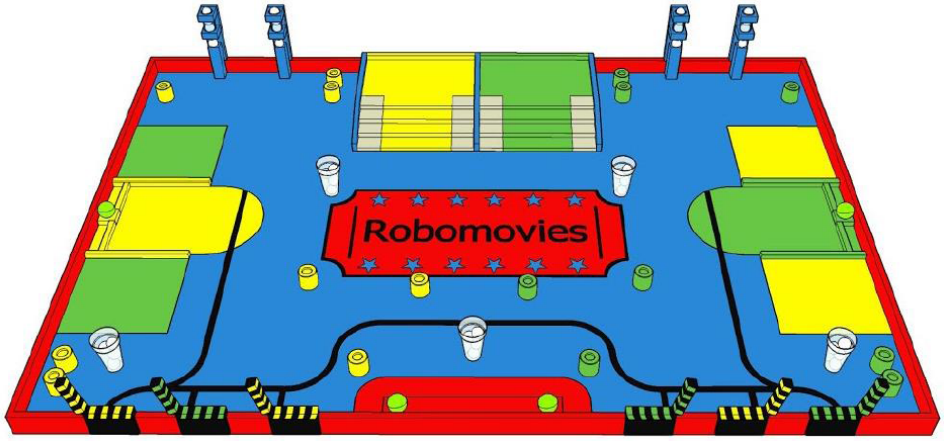


Figure 1.1: The start configuration of the playing table[23]

Task	Points
Valid stand	2
Bonus points for valid stand placed in a spotlight	3
A closed clapperboard	5
Each popcorn placed in a popcorn cup	1
For each step of walkway covered partially or totally by a red carpet	2
Bonus points for each walkway whose steps have been covered partially or totally by a red carpet	4

Table 1.1: The different tasks in the Eurobot competition with the corresponding points

1.4.2 This year's task

The title of this years competition is "Robomovies", and the details of the tasks is described in detail in the official rules [23]. The start configuration of each match are shown in Figure 1.1. The start area for each team is indicated by a colored square at the short edges of the game table. There are two teams on the playing table per round, and each teams can have 2 robots. The teams get different amount of points for different tasks based on the difficulty of the tasks, and each match lasts for 90 seconds. One of the main tasks is to pick up objects formed as cylinders, and place them on top of each other inside the colored areas corresponding to the teams color. Another objective is to pick up cups filled with ping pong balls, these should also be placed inside the team's colored area. On the game area there are two colored stairs. A robot at the top of the stairs at full time, will result in scoring extra points. The objective of the stairs is to have a robot on the top at full time, as well as a red carpet can be rolled along the stairs scoring extra points. This is where the second robot can come in handy. The last task is to close three clapperboards

on the long edge of the game table. The points for each task is described in the Table 1.1. All objects are initially placed at known locations on the game area as shown in Figure 1.1. This makes the initial path easy to calculate.

1.4.3 Qualification

Each team need to qualify before they can compete in the competition. The qualification is divided in two parts. One where the physical constraints of the robot is checked. Where the requirement of size, lasers and power source need to be approved according to the regulations as well as a stop button that should cut the power of the robot and the start function. The second part of the qualification is the practical part, where the robot needs to score at least one point as well as the anticollision works satisfactorily in order to avoid massive collisions during the competition.

1.5 Summary

This master's thesis consists of seven chapters.

1. *Introduction.* This chapter is an introductory chapter, where both the Eurobot competition and the UiO Eurobot team is described together with a motivation.

2. *Background Theory.* Different types of sensors are described, along with filters, tools and software used in this project.

3. *Positioning and sensor systems.* This chapter shows the technique behind different positioning systems used today, and proximity sensors.

4. *Robot system implementation.* Describes the project in full. With the robot design and the robots control architecture. This chapter briefly touches the Artificial Intelligence part which was developed by Bendik Kvamstad, and the motor control system which was developed by Eivind Wikheim.

5. *Design of the positioning system.* Shows the implementation of the positioning system that was developed for the robot.

6. *Evaluation/Experimental results.* This chapter shows the findings of the different systems and shows why design choices of the different system were chosen. Constraints and weaknesses with the system will also be presented.

7. *Discussion.* The last chapter of the thesis consists of a conclusion section, and a future work section including a proposed opponent detection system and positioning system.

Chapter 2

Background Theory

This chapter will provide the relevant background theory for the project. Firstly a brief introduction in the field of robotics. Next is description of different sensor classes followed by theory of filtering, and explanation of the Kalman filter which has been used in the project. Lastly, the tools and programs used will be described.

2.1 Robotics

Robotics is yet a young research field. It combines engineering from fields like electrical, mechanical and computer science. Also psychology and ethics will play a part in the development of robotics[17]. A lot of industry is today automated in order to increase efficiency. Usually stationary robots (Manipulators) are being used for assembly lines. They are bolted to a base frame and are programmed to perform it set of task really fast and accurately. The manipulators uses kinematic chains in order to describe the position for each of the joints and the tool (end-effector). Compared to mobile robots that can move around the manipulators have a limited work space. Mobile robot are capable of moving around in the environment, but the requirements to sense the environment is much higher than for manipulators. Mobile robots needs a lot of sensors in order to avoid collisions and take smart path choices. Mobile robots are well suited to operate where humans cannot, like oil platforms. The Mars rover that is currently driving around on Mars is a good example of how mobile robots can be used. Mobile robots needs locomotion mechanism to move through the environment. Legs, belts or wheels are widely used[13]. Among the different locomotion wheels are the most used. It provides high accuracy and stability, but a lot of wheels suffers when in rough terrain. Legs are well suited for moving in rough terrain, and a lot of research involves the development of walking algorithms for legged robots using evolutionary programming. The algorithms often tries to optimize the walking pattern in order to make the robot walk as efficient as possible, or be able to re-calibrate the walking pattern if it looses one of its legs. However a legged robot has a higher requirements for stability than a wheeled robot, and wheeled robots are more suited for most tasks.

2.2 Positioning for autonomous robots

An important aspect for autonomous robots is to enable them to navigate. Every robot needs navigation to be able to move around, and to perform its set of tasks. Most autonomous vehicles use the encoders in the motors to calculate their position. The encoders have high accuracy, but they will easily accumulate errors. These errors will grow larger and larger over time. This can be a result of spinning wheels or from bumps in the road. Therefore other positioning systems, with reference to the global environment, are applied to correct the accumulation of error. For outdoor positioning Global Positioning System (GPS) is often used, but for indoor positioning GPS cannot be used since this system require line of site to satellites. Methods such as bluetooth trilateration[48] and localization using RFID[44] are different approaches to indoor positioning that has been researched further in this master thesis.

In the Eurobot competition there has been different approaches to the localization part. Since the rules allow each team to place three fixed beacons, most of the systems are based on the principle of triangulation.

2.3 Sensors

For mobile robots, we need information about the surroundings to be able to move in the best possible way in an unknown and changing environment. The key element for a robot to gather this information is through different kinds of sensors. To be able to pick the best sensors for each task, we need to understand the physical principle behind different types of sensors. Therefore, we usually classify sensors in different classes[27] according to how they sense the environment, and what they sense.

2.3.1 Passive

Passive sensors, are sensors that do not send out any energy to measure the environment. They sense the natural energy around themselves[27]. This means that we can only use passive sensor when we have detectable energy around us. If we are to use accelerometers, we have to measure the energy from the movement of the robot. If we want to measure the distance to an object in the environment, there are many different passive sensors to apply[28]. We have for example cameras, where the energy is in form of illumination[16], that makes it possible for the cameras to get useful data.

2.3.2 Active

Active sensors, unlike passive sensors, project energy towards the target to be measured. The reflected signal is then detected and processed further to produce a result that can be used[16]. With the use of active sensors, we can easily direct the measure exactly towards the object we want to measure. For instance, if we want to measure the distance to an object, we can use an ultrasound sensor, see section 3.4.1, which sends out a ping, and measures the time before the echo

returns. This way it is able to calculate the distance by looking at the flight time of the signal.

2.3.3 Proprioceptive

The sensors that measure values internally in the robot are classified as proprioceptive sensors. This kind of sensors can for example be the internal motor encoder, battery status, accelerometer, and so on. Proprioceptive sensors are not very suitable for the purpose of measuring the positioning relative to the environment, since we will not get any information about the environment and what is happening around the robot with the use of proprioceptive sensors. The motor encoders are one type of proprioceptive sensors, and with the use of this we can calculate the position. Internal Measurement Unit (IMU) is also another type of proprioceptive sensor. The main responsibility for proprioceptive sensors is to monitor self maintenance, and controlling internal status[49].

2.3.4 Exteroceptive

The sensors that measures the environment, and everything surrounding the robot are classified as exteroceptive sensors. Examples of exteroceptive sensors could be Cameras, Contact sensors or compass. Sonar is a widely used exteroceptive sensor. Distance measurement with the use of ultrasound, is also a type of exteroceptive sensing, and is an easy way for measuring the distance to an object. Exteroceptive sensors are suited for path-planning, and to locate different obstacles and generate maps of the world.

2.3.5 Error from sensors

There exist many different sources to error when dealing with sensors. Every set of sensors will have some kind of noise that will influence the result. Random noise will have an effect on most of the sensors that we are working on, as well as noise created by the sensor itself. As described in section 2.3.3 the use of motor encoders as a position system will accumulate a lot of error, since we only measure the rotation of the wheels to determine the position. If the wheel is stuck, or it slips on the ground, the encoders will have another belief of how far we have moved than the actual movement. This kind of position estimation is usually called odometry, and is what we will be referring to later on. If we let the robot drive with only the positioning from the encoders, there is a chance that we have drifted from the real position[11]. It is important to have correct measures of the wheels diameter and the length between them, in order for the most correct estimate of the position and to minimize the error.

There are lot of other errors that can occur when working with sensors. Noise from ambient light will affect many sensors, for instance infrared proximity sensors. In an environment where there are several sensors on different robots this may also lead to a source of noise. This is because exteroceptive sensors might use the same energy source in order to make their measure, and other sensors pick up this energy source.

2.4 Filtering

All system and sensors have some amount of noise. By filtering the output from the sensors, we wish to minimize this noise. There exist numerous methods to do this, and two examples will be described below. Moving average filter (MA) and Kalman filter. The moving average filter is actually just a smoothing algorithm, and do not take use of how much amount of noise the system has. The Kalman filter is optimal in the sense of minimum mean square error (MMSE). The filter require knowledge of the error proliferation.

2.4.1 Moving Average filter (MA)

Moving Average filter (MA) is the most common filter used in digital signal processing. It is also one of the easiest to understand and implement. The filter can be useful for reducing noise and irregularities that can occur from wheelspin and abrupt transitions. In Figure 2.1 we have made a really simple MA filter to show the effect. Here we use a filter with a length of 3: $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$. We are then convolving it with the measurements. This works by averaging the three and three points from the input signal [52, chapter 15], to produce the new point in the output signal. One of the main disadvantages with the moving average filter is that in order to filter out errors that are significant large, we need a relatively large window. This introduce a delay to our system and makes it more vulnerable to sudden changes, which can be a drawback for a system where we want to rapidly change our direction.

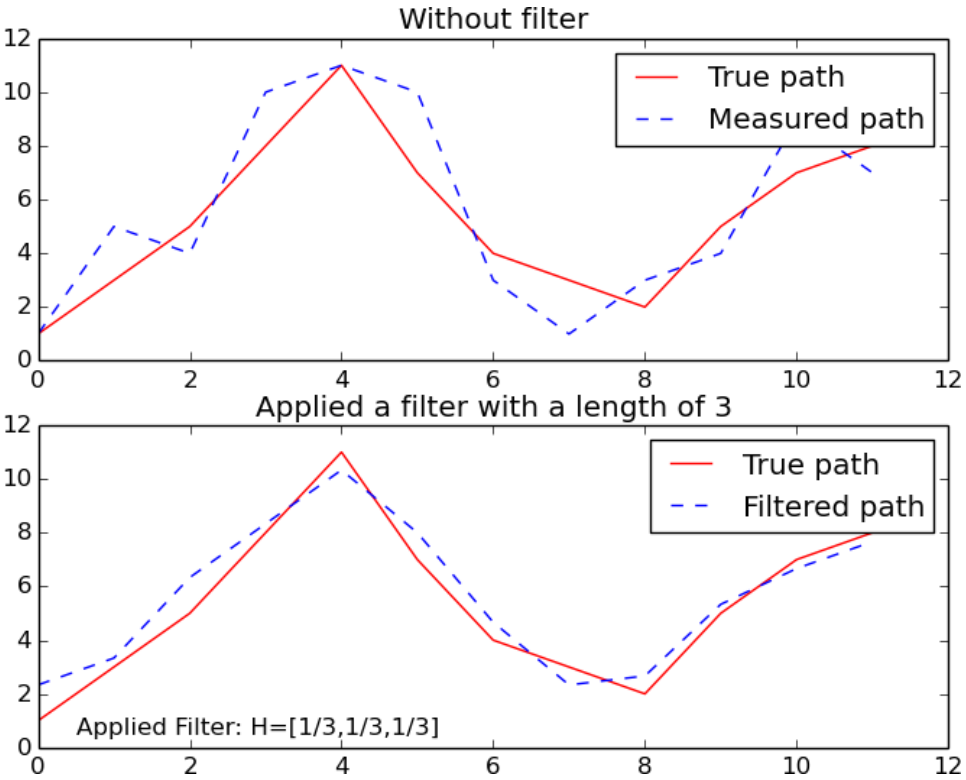


Figure 2.1: Moving average filter on the blue line from the top figure. The red line is the correct, and wanted path

Mathematically the MA filter can be written as:

$$y[i] = \frac{1}{N} \sum_{j=0}^{N-1} x[i+j] \tag{2.1}$$

Where y[i] is the output signal after i samples, x[m] is input after m samples, and N is the number of points in the moving average window

2.4.2 Kalman filter

The Kalman filter uses the systems dynamic model to be able to predict each new state. It looks at the noise over time in order to form an uncertainty matrix of the system. This tend to be more precise than basing the uncertainty on fewer samples. It can also be used to combine measurement from several types of sensors. By looking at the difference between the two estimates and with the use of the known uncertainty of each sensor, it can form a better estimate of the measured value. Based on this property the Kalman filter is a commonly used sensor fusion algorithm. The Kalman filter is divided into two phases. The prediction phase, and the update phase. In the prediction phase the new state is predicted based on the old state, and the current action performed, this step can be seen in eq. (2.2). Equation (2.3) shows the predicted (a priori) estimate covariance. In the update phase the state is updated based on the measurements, and the uncertainty matrices for these measurements. Equation (2.4) decribes the Kalman gain. The update (a posteriori) state estimate is described in eq. (2.5). The last part of the filter is the update (a posteriori) estimate covariance, seen in eq. (2.6). The filter can handle measurements from different sensors and use this to form a better state prediction. The work flow of the Kalman filter can be seen in Figure 2.2.

The Kalman filter bases the prediction on the linear trajectory, and therefore nonlinear systems need to be linearized in order to be used.

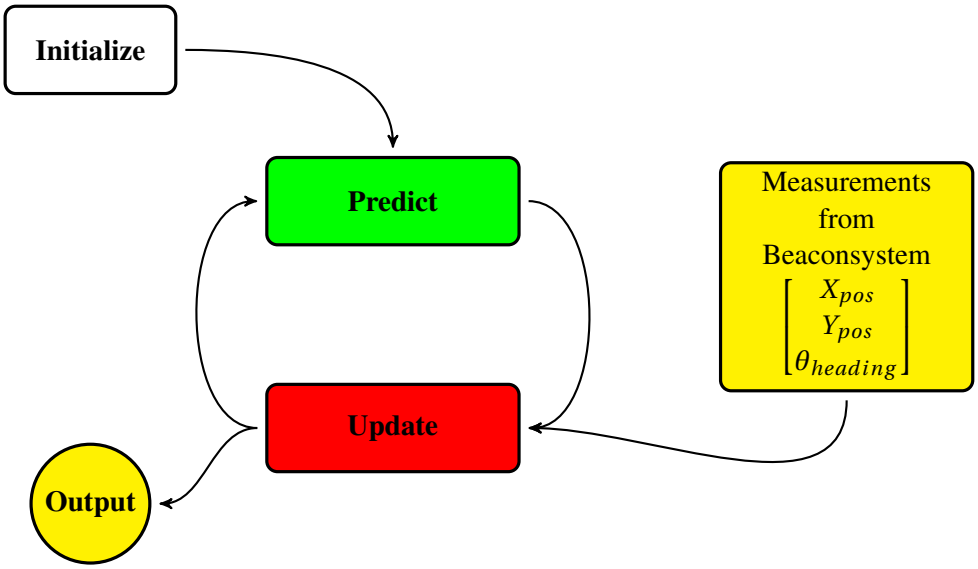


Figure 2.2: Flow in a Kalmanfilter

Prediction

$$\hat{X_k} = A_k X_{k-1} \tag{2.2}$$

$$\hat{P_k} = A_k P_{k-1} A_k^T + \Gamma_k Q_k \Gamma_k^T \tag{2.3}$$

Update

$$K_k = \hat{P_k} H_k^T (H_k \hat{P_k} H_k^T + R_k)^{-1} \tag{2.4}$$

$$X_k = \hat{X_k} + K_k(Z_k - H_k \hat{X_k}) \tag{2.5}$$

$$P_k = (I - K_k H_k)\hat{P_k} \tag{2.6}$$

2.4.3 Extended Kalman filter

The difference between a linear Kalmanfilter and an extended Kalmanfilter is just the transition functions. To take the non linearity into account, we need to linearize all of the transition matrices. We obtain the linearized transition matrices by taking the Jacobian of the time derivative [10, 36].

$$H_k = \frac{\partial h}{\partial x}\hat{x_k} \tag{2.7}$$

$$A_k = \frac{\partial f}{\partial x}\hat{x_k} \tag{2.8}$$

State model

There are many ways to describe the state model for our system. For our purpose we need three states to process between the update and prediction phase of our Kalman filter. The x and y position, and the heading of the robot forms the three states that will be carried between the update and prediction state. These states are used because they are easy to read out from the system. We can calculate all the states from the encoders, and they can be provided by a global beacon system. With the use of a compass, we also have access to the orientation from another set of sensors.

To start, we use position from the encoders to fill the states. In eqs. (2.9) to (2.11) the state of the position and heading are described as the system function $f(k) = [f_x, f_y, f_\theta]^T$. These are the same as that will be described more in depth in section 3.3.1.

$$f_{x,k} = x_k = x_{k-1} + D_c \cos(\theta_{k-1}) \tag{2.9}$$

$$f_{y,k} = y_k = y_{k-1} + D_c \sin(\theta_{k-1}) \tag{2.10}$$

$$f_{\theta,k} = \theta_k = \theta_{k-1} + \frac{D_r - D_l}{d} \tag{2.11}$$

The equations needs to be placed inside the transition matrix A_k to describe the transition from previous state to current state. Equation (2.12) describes the Jacobian of function $f(k)$

$$A_k = \begin{bmatrix} 1 & 0 & -D_c \sin(\theta_k) \\ 0 & 1 & D_c \cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

This is as we see a non-linear system, but it is being modeled as a linear system for each time step we look at the value from the encoders. The actual transition from the previous state to the current state happens in the prediction phase in eq. (2.2). Equation (2.3) describes the a priori estimate covariance projected forward to the next step with the added system covariance Q_k. Γ transforms the estimate covariance to fit the filter covariance.

Estimate covariance matrix

The estimated covariance matrix is a measure of the estimated accuracy of the state estimate. To be able to predict the covariance matrix in eq. (2.3) we need only the error associated with the states. The error is used in the filter to judge how good the measurements are. The state uncertainty matrix Q_k is a diagonal matrix with the associated errors in the diagonal elements. These errors are not random. They come from wheel slip, inaccuracies in the model of the robot measurements or looseness or backlash in the motors.

$$Q_k = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{2.13}$$

The variance of the error makes the diagonal elements of the uncertainty matrix in eq. (2.13). For encoders the error will grow bigger as we move, and therefore it might be given in percentage of the measurement.

Measurement Model

The measurement model, is the model of the measurements done by the positioning system, and other sensors.

$$Z_k = \begin{bmatrix} X_{Beacon} \\ Y_{Beacon} \\ \theta_{Beacon} \end{bmatrix} \tag{2.14}$$

The measurement matrix may contain numerous sensory data to be interpreted in the filter. The important part is to make the measurement matrix and the measurement uncertainty matrix match the model matrix. Equation (2.14) show one possible way to describe the measurement model for a global beacon system.

Measurement transition matrix

The measurement transition matrix is one of the main matrices in the Kalman filter. This is what handles the sensor fusion, and creates the relations between the measured values and the states of the filter. With the use of this matrix, we choose what values should be used from each set of sensors. The relation is described in eq. (2.15).

$$Z_k = H_k \hat{X_k} \tag{2.15}$$

Where z_k is the measurements and H_k forms the measurement state to the system state x_k. So if we want to add in values from a compass for the heading of the robot, the measurement matrix would look like eq. (2.16).

$$H_{compass} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

This is because the compass only has the measure of the heading and can be described as the function $v(k) = \theta_{compass}$. When we take the Jacobian of v(k) we get the matrix described in eq. (2.16). For a global beacon system we can describe $h(k) = [h_x, h_y, h_\theta]^T$ where the values are described in Equations (2.17) to (2.19).

$$h_x = x_{beacon} \tag{2.17}$$

$$h_y = y_{beacon} \tag{2.18}$$

$$h_\theta = \theta_{beacon} \tag{2.19}$$

When we take the Jacobian of $h(k)$, we end up with the measurement matrix for the encoders as described in eq. (2.20)

$$H_{beacon} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

Since the measurements from the beacon system are given in the same state as the filter, the transition matrix becomes the identity matrix, and just describes a mapping between the filter states, and the beacon states.

Measurement uncertainty

Like the state uncertainty matrix, the measurement uncertainty will provide information to the filter about the quality of the data from the measurement, and tells the filter how much it should trust this data. This uncertainty matrix takes the same form as the state uncertainty matrix as a diagonal matrix with the variances as the diagonal element, as can be seen in eq. (2.21).

$$R_k = \begin{bmatrix} \sigma^2_{Beaconx} & 0 & 0 \\ 0 & \sigma^2_{Beacony} & 0 \\ 0 & 0 & \sigma^2_{Beacon\theta} \end{bmatrix} \tag{2.21}$$

Some sensors provide the standard deviation in the data sheet. For the beacon system we have based the errors on tests made. These tests can be seen in section 6.2.1.

Kalman gain

The Kalman gain is perhaps the most important part of the Kalman filter. This is what minimizes the error. When the Kalman gain is optimal, it yields the minimum mean square error. As described in eq. (2.4) it processes both the estimate covariance from the prediction model, as well as the measurement uncertainty.

State update

In the state update, the new position estimate is calculated based on the error between the system, and the measurement model. $(Z_k - H_k \hat{X_k})$ describes the deviation between the measured values, and the system values. The kalman gain is then multiplied with the error, and added to the position estimate. The last step, described in eq. (2.6), is the a posteriori update of the estimate covariance matrix based on the kalman gain.

2.5 Tools and programs used

Through the project different types of programs have been used. From designing the 3D models of the robot, driving the beacon tower, and controlling the stepper and the IR-LEDs.

2.5.1 SolidWorks

Most of the parts of the robot have been designed in Solidworks. Solidworks is a solid modeling computer-aided design (CAD) software. This software have been used for rapid prototyping of the different parts of the robot. SolidWorks makes it easy to design different parts separately, and design components that can easily be assembled later on. It makes parts easy to replace or change later in the design process. In SolidWorks you often start off by sketching the part in 2D, and extrude the part from a selected plane in order to get the drawing in 3D. It is one of the most used 3D modeling program currently in use, and it offers a lot of powerful tools to make our part more detailed. Every part can be assembled in one assembly, and relations between the parts may be set to get an insight in what components will look like. Physical constraints can be set, to make the design as real as possible. SolidWorks also offers a simulation tool with physics engine, allowing the user to identify whether the design gives the desired movement, as well as finding constraints and errors in the design.

2.5.2 3D-printing

All parts except the robot frame have been 3D printed. This has allowed us to rapidly design new parts for testing, and to make the different components such as motors and micro controllers fit the robot frame. There are several possible

technologies for 3D printing. In fused deposition modeling (FDM) we add the material to the workspace by melting it with a high temperature nozzle [2]. For overhangs support material is added when printed. The support material is then removed when the print is done. This can either be done manually or with a caustic bath[32]. Another method is stereolithography. In this method the building material starts as a liquid polymer, and we apply a UV light at the parts we want the 3D model. This is done to solidify the liquid polymer. This creates the layer, and the model is lowered further down, and UV light is again applied to create more layers of the 3D model. The UV light polymerizes the resin, and hardens the material on that layer [19]. This process is repeated until we have the final model.

To get our model from SolidWorks to be printed we convert the model as a printable file. STL (Standard Tessellation Language) files are one of the most commonly used, and is supported by most of the 3D modeling softwares. This file format describe the surface geometry of the object to print, and is built by triangles.

3D printing makes designing parts relatively fast and cheap. It makes it possible to make mounts for motors, wheels and other parts needed in the project. Figure 2.3b shows different parts during a 3D print.

2.5.3 Stratasys Fortus 250

Stratasys Fortus 250 shown in Figure 2.3a is based on fused deposition modeling (FDM). This is the printer used for all of the 3D printed parts on the robot. It uses ABS plastic for the model, and needs support material under overhangs that are more than 30° degrees. This 3D-printer prints with a resolution of 0.178-0.33 mm. Beacuse it may be hard to get rid of the support, the design of the parts should therefore be designed to minimize the use of support.

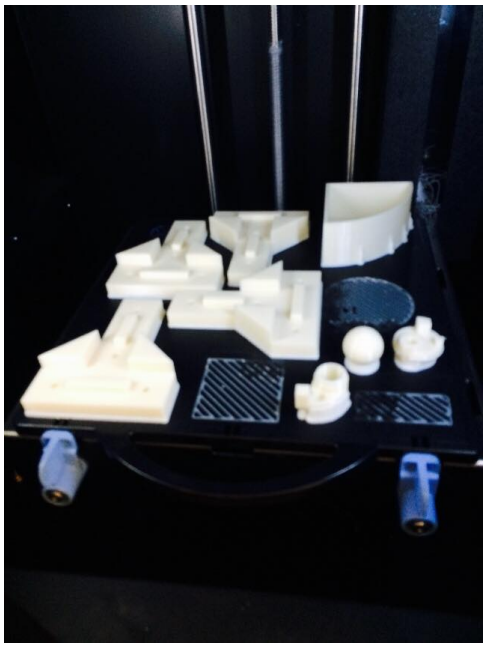
2.5.4 Processing

Processing is mainly a Java based development environment that is originally developed for teaching computer programming. With processing it is easy to plot and make illustrations in real time. The serial library in processing is easily combined with the serial communication from Arduino. Therefore processing was used in this thesis for the live plotting and visualization of the position in real-time. This made debugging and optimization of the beacon tower code more efficient.

2.5.5 Arduino

We have used Arduino[4] through the project to easily reprogram different parts of the robot, and to make this process fast, easy and cheap. For the beacons an Arduino nano has been used to get the beacons as small as possible. One of the Arduino used can be seen in Figure 2.4. For the beacon tower an Arduino Duemilanov with an Atmel 328p chip is being used. This Arduino controls the stepper motor, and registers the angles between all of the beacons. Based on these angles the arduino then calculates the position of the robot as described in section 3.2.

(a) Stratasys Fortus 250 3D printer



(b) Different parts being 3D printed with Fortus 250 3D printer

Figure 2.3: (a) shows the 3D printer used in the master, and (b) shows a print with the Fortus 250 printer

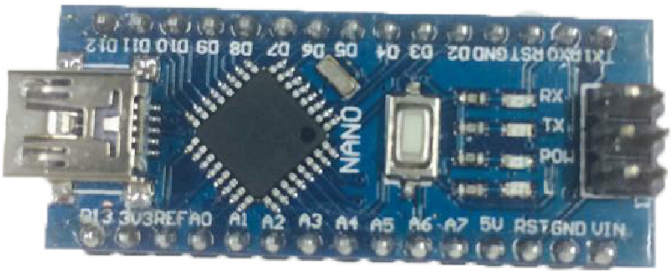


Figure 2.4: Arduino nano, used in the beacons

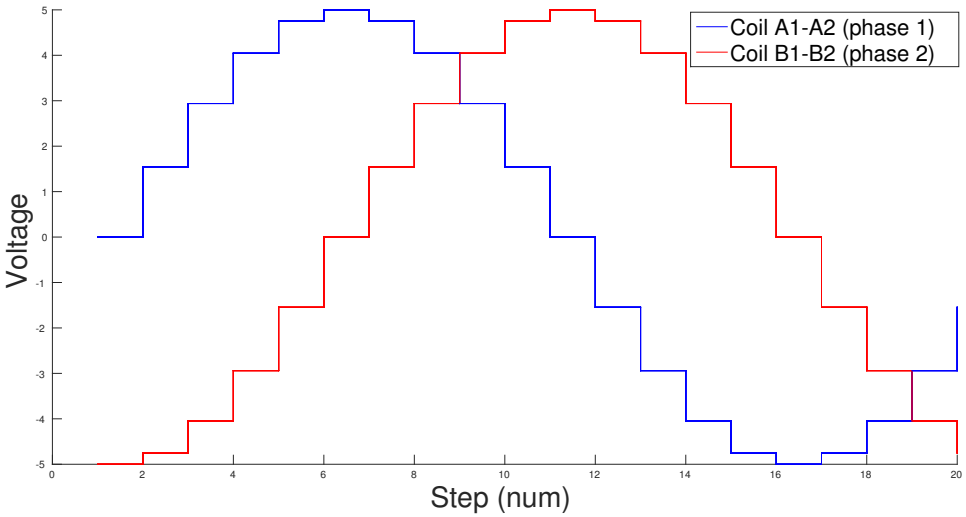


Figure 2.5: Illustration of a typical microstepping wave where the waveforms of the two coil pairs will approach 90 deg phase shifted sinusoidal. -5 to 5V

2.5.6 Stepper motor

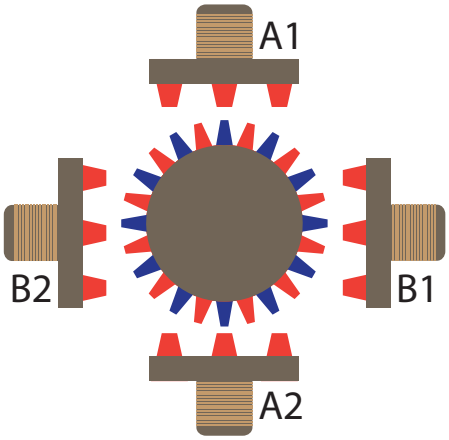
In this thesis Mercury motor SM-42BYG011-25, seen in Figure 2.6a has been used to spin the positioning tower. It was also used on an early version of the robots lift. The stepper motor is a two phase hybrid. It has a low cost and it is reliable and easy to control. The motor is made up with a steel shaft. At the shaft a permanent magnet is attached, and two iron disks are placed on each side of the magnet making each one an extension of the pole it is welded on. These disks will describe the steps of the motor. The stator is made of soft iron equipped with four electromagnetic coils in a cross pattern describing two different pair/phases, hence two-phase stepper. The coils in same phase are aligned with each other, seen in Figure 2.6b. The first and second pair of the coils are not perpendicular, but they are placed 90° degrees + the angle of one step of the motor. This is due to how a stepper motor operates. When the coils in the same phase are applied voltage, the north pole of the rotary disk is attracted to each of these. This is described as one step. When the shaft has stepped one step, it turns the first coils in the first pair off, and then turns on the coil in the second pair. This continues as long as you want the motor to run.

There are typically three different modes of stepping, full stepping, half stepping and micro stepping. When the coils are turned on and off pairwise, as described above we have full stepping. Half step mode is when we energize both phase 1 and phase 2 between each normal step. This requires more power than the full stepping mode, but offers twice the resolution. For microstepping we are controlling the current in the coils to be able to obtain an even higher resolution. Often the current is controlled using a sinusoidal function, as can be seen in Figure 2.5. The motor driver EasyDriver offers a microstepping with 8 steps per original step. Since the stepper motor do not have any feedback, it is possible for the stepper to jump a step if its shaft meets some resistance.

Since SM-42BYG011-25 has a disk with 200 steps the resolution in full stepping is $360/200 = 1.8°deg$ which is not usable for the purpose of accurate

positioning. With half stepping the resolution was doubled, but still gives a bit low resolution. This is why micro stepping has been used, where the EasyDrive offers 8 steps per step: $360/200 * 8 = 0.225° deg$.

(a) Stepper motor



(b) When pair A is induced, it attracts the closest tooth and vice versa with pair B

Figure 2.6: (a) shows the stepper motor used, while (b) illustrates the principle behind the stepper, with the teeth and four magnets used in order to step the motor

Chapter 3

Positioning and sensor systems

Different approaches to the positioning problem exists. They can be classified in two classes, relative positioning and absolute positioning. The difference between the two is from what point of view they solve the problem. These methods and technologies will be investigated further in this section, along with the theory of some common proximity sensors.

- **Relative positioning**
  Odometry
  Inertial navigation
- **Absolute positioning**
  Compass
  GPS
  RFID
  Bluetooth and iBeacons
  Landmark recognition
  Beacon tower

3.1 Positioning

In all aspects of robotics, we need to be able to position the robot relative to the environment. For stationary robots, we can use the base frame of the robot in order to reference the robot arms and the end effector of the robot[53, chapter 2]. In mobile robots this is one of the most challenging tasks. We first need to have a reference frame which is stationary in the environment. A lot of navigation system uses Global Positioning System (GPS) to navigate, and uses latitude and longitude to reference where in the world it is located. GPS can give an accuracy of up to 1 meter, and is highly reliable for outdoor use. The drawback is that GPS is a ”line of sight” system, and signals can often be blocked by tunnels, mountains, or if GPS-receiver is located indoors[6]. Since the Eurobot competition take place indoor, we need some system other than GPS to locate the robot relative to the environment. For our problem, we can describe the environment in three dimensions, as a vector $[X_R, Y_R, \theta_R]^T$ where X_R is the position of the robot in X direction, Y_R is the position of the robot in Y direction, and θ_R is the heading of the robot. We define bottom left of the game area as the origo shown in Figure 3.1.

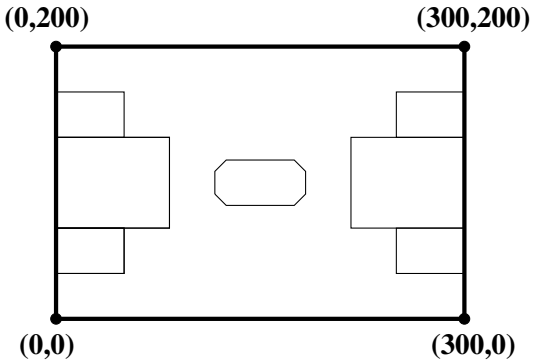


Figure 3.1: The coordinates of the playing table

.

3.2 Triangulation

Triangulation is a method of calculating the position with the use of angle calculations. Triangulation has been used to measure various properties like length, and position through history. The ancient Egyptians used triangulation to measure the heights of the pyramids using the angle of its shadow[56]. There exists numerous triangulation algorithms, some of these algorithms are described in [9, 15, 22, 25]. The algorithms are often classified in four classes based on the approach they use to solve the problem: *Geometric Triangulation, Geometric Circle Intersection, Iterative methods, Multiple beacons triangulation*. The triangulation algorithms belonging to the geometric triangulation class applies a lot of trigonometric computations to solve the problem, this is why these types of algorithms often are called trigonometric triangulation. For the geometric circle intersection class, the intersection of circles passing through the beacons and the robot is calculated to find the point. Iterative methods linearizes trigonometric relations. The idea here is to minimize the error by changing the weights of the linear equations. For the multiple beacons triangulation, measures from several beacons are used for minimizing the error. This is a more general group where several algorithms are located. Though if the setup only contains three beacons or if the computing power is low, either Geometric triangulation, or Geometric circle intersection algorithms are the most suited. There are some drawbacks to algorithms in these two classes. The largest issue is due to the trigonometric function used. When the parameters to the trigonometric functions are $(\pi, \frac{\pi}{2}, 0)$ we might end up with dividing by 0, or meeting other constraints. When the robot is located on the same circle as all the beacons, no calculation is possible due to trigonometric constraints. This case is seen in Figure 3.2, and will be shown more thoroughly in the following subsection.

3.2.1 Tienstra formula

The Tienstra formula is a triangulation algorithm, and will be used in the position calculation. It is unknown who came up with the Tienstra formula first, but it

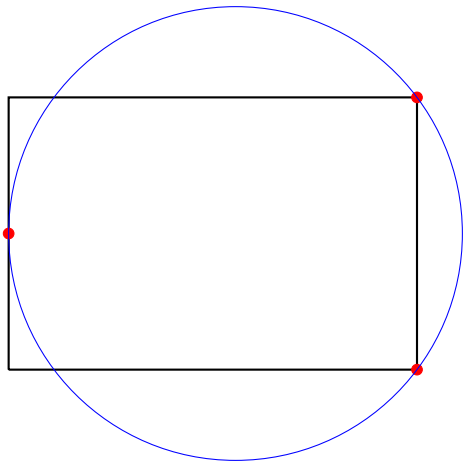


Figure 3.2: Impossible to calculate the position by either geometric or circle intersection triangulation algorithms of the points that lies on the circle

has its name from professor J. M. Tienstra (1895-1951). Tienstra taught the use of barycentric coordinates at Delft University of Technology, and this is probably why the formula is named after him[47]. The formula has been used in area as land survey to map the landscape.

The concept of Barycentric coordinates was first introduced by August Ferdinand Möbius in 1827[24]. The Barycentric system defines a point of a simplex as the mass center of masses placed along the vertexes of the simplex. In Figure 3.3 The point is located in the center of the triangle. This correspond to masses that is located in the middle of the vectors AB, BC and CA.

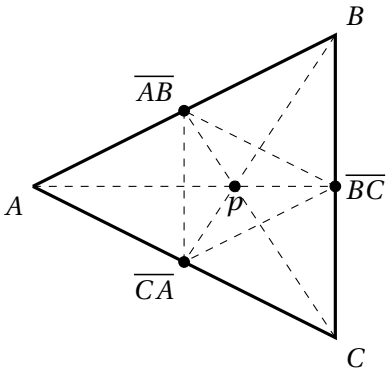


Figure 3.3: Triangle with barycentric masses shown as dots

The Tienstra formula uses this to calculate where the point is located by looking at the mass center along the known vectors AB, BC and CA.

Tiestra formula is a trigonometric triangulation method, called Geometric triangulation in section 3.2. The computation requires a lot of mathematical operations, which leads to relatively slow computation time, compared to other triangulation methods [1]. But it is easy to implement and use. A complete proof

[1] $http://www2.ulg.ac.be/telecom/triangulation/$

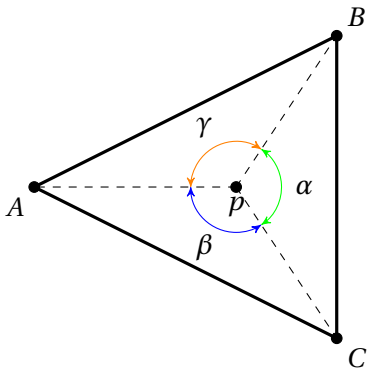


Figure 3.4: A point p with corresponding angles between the three beacons/points A,B and C

can be seen in [47]. The X_R and Y_R position can be achieved from eq. (3.1) and eq. (3.2).

$$X_R = \frac{K1X_A + K2X_B + K3X_C}{K1 + K2 + K3} \tag{3.1}$$

$$Y_R = \frac{K1Y_A + K2Y_B + K3Y_C}{K1 + K2 + K3} \tag{3.2}$$

The different $X_{A,B,C}$ are the corresponding X coordinate of the beacons, and $Y_{A,B,C}$ are their corresponding Y coordinates. K1, K2 and K3 are found in Equations (3.3) to (3.5), where the α, β, γ are shown in Figure 3.4

$$K1 = \frac{1}{cot(A) - cot(\alpha)} \tag{3.3}$$

$$K2 = \frac{1}{cot(B) - cot(\beta)} \tag{3.4}$$

$$K3 = \frac{1}{cot(C) - cot(\gamma)} \tag{3.5}$$

From these sets of equations we clearly see the restriction when the point is located on the arc of the circle passing through all the beacons. In Figure 3.5 angle A is equal to α resulting in $\frac{1}{0}$ for K1 giving an invalid result. This is a result from the inscribed angle theorem.

3.3 Trilateration

Trilateration is similar to triangulation based on three fixed beacons. What separates the two methods of positioning, is that in trilateration, we use the distances from the object to the beacons to calculate the position of the object, and not the angles between them. GPS is as mentioned an example of a trilateration system. Prior to GPS, LORAN was a long range navigation system developed by the Americans at Massachusetts Institute of Technology. There were fixed stations that sent out low frequent radio signals (90-110KHZ), and the position was

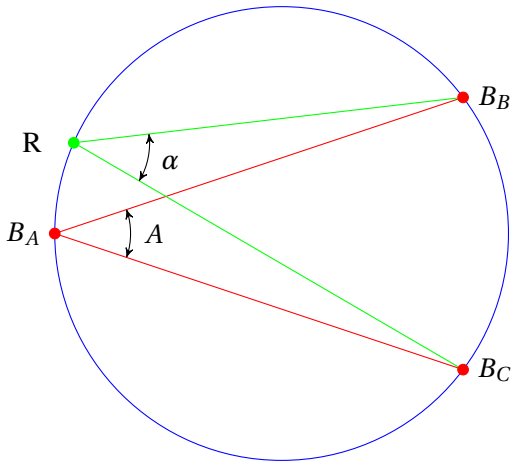


Figure 3.5: From the inscribed angle theorem all points located on the arc from B_C clock wise to B_B forms the same angle to the two points.

calculated based on the flight time of the signal. The accuracy of the system could get up to 100 meters. Decca was a similar system planned to use for navigation around mines outside the Norwegian coast. The accuracy of this system was up to 10 meters. Decca was like LORAN used mainly for navigation of ship, and fishing boats. Like GPS the distance is calculated from the flight time of the signal. Both LORAN and Decca have been replaced by GPS.
For indoor positioning, GPS and other systems where the base stations is not in line of sight can not be used. Microsofts RADAR [5], AT&T Active Bat [54] or using the signal strength received from Wi-Fi access points[18] are examples of different methods accomplishing indoor positioning with the use of trilateration. In trilateration we find the point from the intersection of three circles, with radius as the distance to the point. For the case where we have perfect information as shown in Figure 3.6 the point will be the intersection between the three circles. In the case where we have some errors from the reading Figure 3.7, the circles do not intersect at a given point, and they might not intersect at all. In this case an estimation method is often used to find the point that minimizes the distance to all of the circles. Different mathematical approaches exists like the Least Square Estimation (LSE) [29]. An efficient least-square algorithm is described in [58].

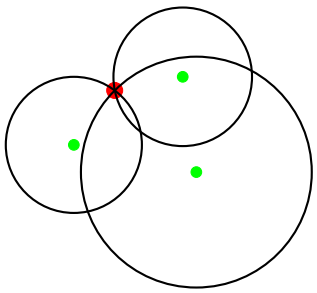


Figure 3.6: One unique intersection point

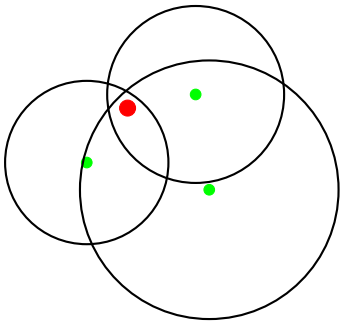


Figure 3.7: No unique intersection point

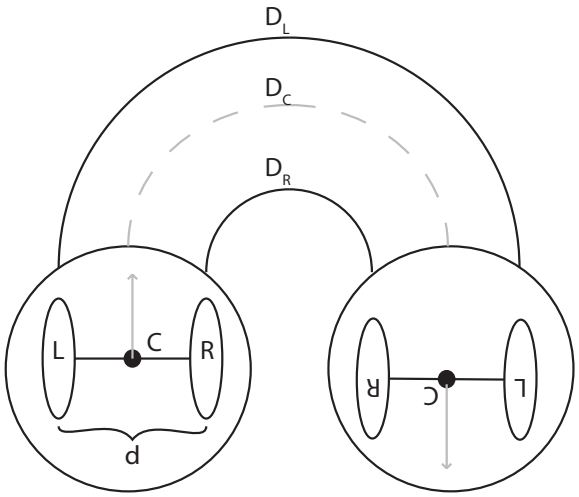


Figure 3.8: Unicycle robot, with 2 wheel

3.3.1 Odometry

Odometry is the most widely used navigation method for mobile robots[12]. It provides high accuracy with high sampling rates. However since this method uses proprioceptive sensors described in section 2.3.3, like the encoders in the motors, this method does not provide any information from the environment than what we initially started with. But it is on the other hand an easy and cheap way to get an estimate of the position, unlike many other advanced positioning systems.

For the robot as shown in Figure 3.8 we need to know as accurate as possible the diameter of the wheels, and the length between the two wheels. The more accurate these values are, the less is the error that will be accumulated. The center between the two wheels, will be the origo of the coordinate frame for the robot. In eq. (3.6) the center displacement is expressed from the displacement of both the left and right wheel.

$$D_c = \frac{D_l + D_r}{2} \tag{3.6}$$

We can now derive the position and heading after time t from the previous position and heading. $[x_{t-1}, y_{t-1}, \theta_{t-1}]^T$. The d in eq. (3.9) is the length between the two wheels, as illustrated in Figure 3.8

$$x_t = x_{t-1} + D_c \cos(\theta_{t-1}) \tag{3.7}$$

$$y_t = y_{t-1} + D_c \sin(\theta_{t-1}) \tag{3.8}$$

$$\theta_t = \theta_{t-1} + \frac{D_r - D_l}{d} \tag{3.9}$$

From the encoders we know that we get N "ticks" per revolution. To calculate the displacement of each wheel, we look at eq. (3.10) and 3.11. These calculations calculate the circumference of the wheels, and multiply it with how many ticks we have read out, divided by the number of ticks the encoder gives out per revolution.

$$D_l = 2\pi R \frac{\Delta tick_l}{N} \tag{3.10}$$

$$D_r = 2\pi R \frac{\Delta tick_r}{N} \tag{3.11}$$

We find $\Delta tick$ as the last tick from the encoders, minus the tick from last reset of the encoders. The encoder value since last reset is denoted $tick_{l/r}$ and the newest value of the encoders are denoted $\hat{tick}_{l/r}$.

$$\Delta tick_{l/r} = \hat{tick}_{l/r} - tick_{l/r} \tag{3.12}$$

As described in section 2.3.5, we will accumulate a lot of error over time by using only the wheel encoders for positioning. To get a more accurate estimate of the position, we want to apply an absolute position system in order to correct any errors that the encoders may introduce. Even with perfectly tuned values we can have external factors like crash or wheelspin effecting the accuracy of the odometry.

3.3.2 Inertial Measurement unit (IMU)

A device containing both a gyroscope and an accelerometer is commonly called Inertial measurement unit (IMU). In an IMU we have one or more accelerometers that are detecting the acceleration in a given direction, while one or more gyroscopes detects the orientation and rotation of the IMU. These values can then be used in order to calculate the movement. IMUs are more and more used in navigation, and IMU-enabled GPS devices are used to keep track of the position, even when GPS signals are unavailable[57]. IMU is a dead-reckoning sensor, and will therefore in the same matter as odometry, accumulate error over time. IMU might get bias on the input, and if not removed this will lead to large error especially for the accelerometer. The bias is the difference between real value and the output. A constant error in the acceleration is for the accelerometer integrated and will result in a linear error in velocity, and a quadratic error in position. For precise IMUs the price is rather expensive while a cheaper IMUs are more unstable.

3.3.3 Compass

Magnetic compass is able to give out the heading with a relatively high accuracy. The fact that compass bases its measurements on the earth's magnetic field, makes it vulnerable to magnetic fields from nearby electric devices. The placement of the compass is important to minimize the noise. Equipment like stepper motors that uses a magnet to step will interfere with the compass. The compass therefore needs to be placed as far as possible from this kind of equipment.

3.3.4 Global Positioning System (GPS)

GPS is frequently used in outdoor navigation. It was first developed for the U.S. military as a satellite-based navigation system[41]. Later it was opened for civilian usage, and is today one of the most used positioning system, and GPS receivers can be found in cars, cellphones etc. GPS is based on time, and each of the satellites are carrying an atomic watch which is synchronized to a base station every day. The GPS satellites transmit there current position, and time. The position is calculated with the use of trilateration as described in section 3.3, and the distance to the satellite is calculated from the flight time of the signal. The GPS receiver needs signal from four satellites to be able to calculate the position. The position is then calculated by solving a set of equation where longitude, latitude, height and time are the different unknowns. GPS is a line of sight system, which means that the satellites needs to be in a straight line with out anything blocking them in order to get a signal. When indoors the GPS receiver is unable to get information from the satellites which makes the calculation of the position impossible. In mobiles, and other GPS modules the system remember the last known position if no GPS signal is received. A technical report[33] submitted to Federal Aviation Administration showed that the horizontal accuracy of GPS often is within 1 meter.

3.3.5 Radio-frequency identification

Radio-frequency identification (RFID) is a method to store and retrieve data from small units (RFID-chips). The RFID-chips contains small antennas which makes it capable of receiving and answering a radio signal from a RFID sender. The main function is to retrieve information id from a tag (transponder).

There are two methods for communication between readers and tags. Passive RFID tag, using an inductive coupling that drains energy from the electromagnetic waves of a nearby RFID receivers. Or an active RFID tag that has its own power source. An active RFID has a longer range, but needs to have the battery changed once in a while to be operational. It is also a lot larger, to fit the battery. There are also two different active RFID tags, commonly called RFID transponders, and Beacons. The difference between this two, is that a beacon continuously broadcast there ID, while an active RFID transponder only transmit when in range of a RFID reader. RFID is today being used in a lot of equipment from car keys and passport to mobile phones.

Another application for RFID is for indoor localization[51]. There are different approaches to use RFID as a position system, but the most common is to calculate the distance to a RFID transponder from the received signal strength Indication (RSSI)[14]. Different algorithms that is based on RSSI can be applied, as spotON [30] which is used to map the signal strength to a given distance, before a lateration algorithm can be used to calculate the position.

3.3.6 Bluetooth and iBeacons

For indoor positioning bluetooth is a technology that is more and more being used, due to the increase in number of devices supporting bluetooth. The distance to each

beacon can be measured using the Received Signal Strength Indication (RSSI) or Link Quality Indicator (LQI) [8]. The most accurate is the RSSI, where the distance is based on the received signal strength, while the LQI estimates how easily the received signal can be modulated considering the noise in the channel. Neither of the methods are very good for accurate positioning, due to easily scattering, or reflection of signals.

iBeacons is Apple's bluetooth positioning system [7]. Apple was among the first to take this technology into use for this purpose. iBeacons uses Bluetooth Low Energy (BLE). BLE is a bluetooth mode used to pair devices with low energy sources, smart watches is another application using this technology. The difference between BLE and common bluetooth protocol is the minimum energy needed to discover devices. This means that the transfer of data will be slower for BLE, therefore devices using BLE have minimal data transmitting. iBeacons are not very accurate, and do not offer high precision, but can be used for applications to know if devices are nearby. In a many office buildings bluetooth has been used for locating in what rooms the device is located.

3.3.7 Landmark recognition

For an autonomous robot to position itself, it needs to relate the position to some reference point. These points can be certain topological points in the environment where the position is known. The landmarks could be features in the landscape or other houses where the position is known. This method often requires a lot of image processing, and powerful perception systems [40]. One of the difficult tasks is to recognize the templates we want to find. Indoors the landmarks we use may be doors, walls and corridors. This requires a lot of pre calculations and classification in advance. In the Eurobot competition the landmarks could be represented as the beacons. Task of the robot is then to locate each of these landmarks relative to the robot. The idea of placing beacons is used a lot for ships and aircrafts. They can then apply a lateration algorithm to calculate there position relative to the beacons. In ships, the position was found be looking up the length from the landmarks in a tables, and read out the position.

Omnidirectional camera for beacon recognition

One solution that we looked into, was to use an omnidirectional camera. From the image we got out, we could locate the three beacons, and find how many pixels in the picture they where placed. We would then be able to calculate the angle θ_i relative to $beacon_i$ as seen in eq. (3.13)

$$\theta_i = \frac{360}{totHorisontalPixel_i} \tag{3.13}$$

One of the major advantages for this system is that it do not have any moving parts which can introduce noise. And since all the beacons are seen at the same time, this system would not get error if the robot is rotating or moving. The cons are that this method needs advanced image analysis to locate each of the beacons. This requires heavier computational power[26]. The beacons should be made so that it is easy to classify them. This could be done with sharp colors. The identification

of the beacon is then just to look at the color. But if there are other objects in the room with the same colors, this might be classified as a beacon. Template matching might be a better solution. The beacons can contain a geometric figure, and template matching can be used to match where in the picture the beacons are located [35]. A very easy template matching method is based on convolution. A set of pixels containing the structure we are looking for in the picture is moved over the the pixels in the picture. For each step, the difference is calculated, the point with the lowest score, will then be the position of the template we are looking for in the picture. This method uses the sum of absolute differences (SAD), which is a method to measure the similarity between image blocks. In eq. (3.14) we can assume we are looking at a grayscale image with value from [0-255] and the (x,y) is the position of the pixel in the main picture. T_{row} and T_{col} is the number of row and column in the template. We take the difference between each pixel in the original picture with the corresponding pixels in the template.

$$SAD(x,y) = \sum_{i=0}^{T_{row}} \sum_{j=0}^{T_{col}} |(x+i, y+j) - T(i,j)| \tag{3.14}$$

The template is then moved through all the pixels in the picture, and the same procedure is done for each new position eq. (3.15)

$$\sum_{x=0}^{P_{row}} \sum_{y=0}^{P_{col}} SAD(x,y) \tag{3.15}$$

OpenCV [2] offer a lot of real-time application, like template matching. And is a well documented library for this use.

3.4 Distance measures

Distance measurement, or proximity measures is an important part of autonomous vehicle. They are used for several areas in robotics, where the most used application is for collision avoidance. There exists several different ways of measuring the distance. In this section the proximity sensors based on ultrasound, infrared and laser will be described.

3.4.1 Ultrasound

The principle behind the use of ultrasound for distance measure is rather simple. By sending out a sound wave (PING), we can measure the time from the ultrasound signal was sent, and until we receive the reflected PING. The ultrasound sensor consists of a sound source, that transmits the PING, and a sound receiver that receive the echo [43]. We can also measure the strength of the received PING, and compare it with the strength of the sent PING.

For ultrasound to work well, there are several factors that are important to take into account. First one is the surface of the object we want to measure the distance to. An object with a hard surface will reflect the sound much better than an object with a soft surface. This means that the range of the sensor is longer if the object

[2] http://opencv.org/

Figure 3.9: LV-MaxSonar-EZ3 ultrasound sensor

is hard, since less sound will be absorbed by the object. One problem when using ultrasound to measure the distance, is that you can not be absolutely sure what you measure the distance to since the sound spreads out. This means that if the ultrasound sensor is mounted to low on the robot, the floor will reflect some sound, and some of the readings may be feedback from the floor. You will also get an invalid distance measure, if what you measure is in a steep angle as can be seen in Figure 3.10. Then the echo will not be reflected back to the sensor again. Another source of error, is if the object is to small to reflect enough sound back to the sensor.

(a) Object is normal to the ping

(b) Object is in an angle relative to the ultrasound source

Figure 3.10: Angle problem in ultrasound

In the project, LV-MaxSonar-EZ3 ultrasound sensor has been used, this sensor is shown in Figure 3.9. The operation voltage for this sensor is between 2.5V-5.5V. The characteristics of the sensor is relatively linear, but the scaling factor depends on what voltage is applied. The maximum range of the sensor is 645cm. The signal operates on 42KHz frequency, and the sampling rate is 20Hz resulting in a fast updating rate. The beam is quite narrow for this types of sensors, but it still has a width of 1m, giving a wide detection area for the robot. There are different pins that can be used to extract the data from the ultrasound sensor. The different pins are listed in Table 3.1. The two most common methods is the analog pin, where the distance can be described from the strength of the returned signal, and the pulse width method where the distance can be described from the flight time of the signal. The difference between the two can be seen in section 6.2.3.

The distance can be calculated as follows in eq. (3.16) for the PW method, or with eq. (3.17) for the analog voltage method.

$$distance(cm) = \frac{pulsewidth(\mu s)}{pulsewidthScalingFactor} \tag{3.16}$$

MaxSonar pin out scheme	
GND	Circuit ground
V+	Supply voltage
TX	Transmit pin, RS232 serial bus
RX	Receive pin, RS232 serial bus
AN	Analog voltage output pin
PW	Pulse width output pin
BW	Leave it open or hold low for serial output on the TX output

Table 3.1: Pin-out scheme for the MaxSonar ultrasound sensor

$$distance(cm) = \frac{analogvoltage(V)}{analogScalingFactor} \tag{3.17}$$

One widely used application for ultrasound is parking sensors for cars. Many modern cars uses arrays of ultrasound sensors in the bumpers for ranging objects in front of the car to avoid collision when parking.

3.4.2 Infrared

Figure 3.11: The sharp 2Y0A21 IR-distance sensor

Infrared sensors are widely used in robotics as proximity sensors to detect nearby object, and to avoid collisions. Compared to Ultrasound they are often cheaper, and have a faster update rate, however they are more effected by noise from the environment. This means, that for outdoor use the ambient light will act as a source of noise, and the readings will be affected. But for indoor use were the light setting is not as fluctuating they are quite reliable. Similar as for ultrasound, the surface to what we measure will impact the reflected signal and the readings we get. The sharp IR sensor is a cheap and accurate IR range sensor with a low power consumption. The sensor is suited for small spaces and to detect nearby objects. The sharp IR-sensors seen in Figure 3.11 uses triangulation to find the distance to the object. The sensor starts by emitting an IR-light, and waits for this light to hit an object and be reflected back. From the reflected signal the angle α shown in Figure 3.12 is measured, and the distance can easily be calculated. At the receiver part of the sensor we find a PSD-sensor (Position Sensing Detector) with a charged-coupled array (CCD). The reflected light charges up capacitors in the CCD, and the angle can be calculated by looking at how much of the CCD array that has been illuminated[1]. Angle α is then calculated as shown in eq. (3.18), where x is the distance of the CCD array that has not been hit by the reflected IR

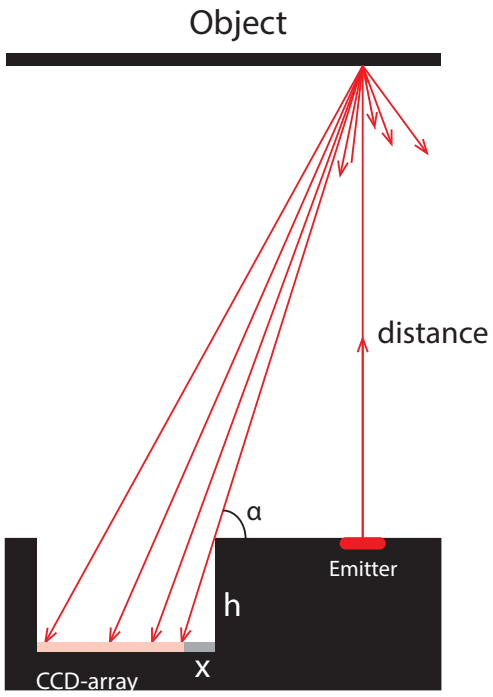


Figure 3.12: Principle behind proximity measurement with the IR sensor

light. This method is quite reliable. The range of the sensor is limited to where close the object needs to be placed before the reflected IR-light do not hit the CCD array, and how far away before it hits the entire CCD array. This results in a non-linear function compared to the results of the ultrasound sensor, which is linear. A simple scaling factor can therefore not be used when computing the distance.

$$\alpha = atan2(x, h) \tag{3.18}$$

By combining measures from Infrared and ultrasonic sensors we can achieve more precise distance measures, and are able to create a reliable collision avoidance system[20].

3.4.3 Laser

Distance measure with lasers works by emitting a laser pulse towards the object. The laser beam is then reflected back to a photocell in the sensor and the time interval between the transmitted and received signal is used to calculate the distance. Laser sensors are a lot more accurate than for instance ultrasound and infrared proximity sensors. The accuracy of a laser sensor is $\approx 2mm$, while compared to ultrasound the accuracy is based on the length of the distance to the object, where the margin of error is around 0.5% of the distance, but this might vary for different sensors. This makes laser more suitable for ranging over large distances, since the performance do not decrease over large distances, as for ultrasound and infrared. Another advantage with laser sensors is that they emit visible light, unlike infrared. This makes it easier to see what objects the sensor is ranging, which again makes the debugging and tuning of the sensor system

easier. A disadvantage of laser compared to ultrasonic, is the ability to measure the distance to transparent object like water or glass. They are also a lot more expensive than both infrared and ultrasonic sensors.

Chapter 4

Robot system implementation

In this chapter the project in full will be described. The chapter briefly touches the motor control system developed by Eivind Wikheim and the Artificial Intelligence developed by Bendik kvamstad.

4.1 Open Control Architecture

The control architecture is based on a Personal Computer (PC), the robot (*Mario*), the positioning system, and finally the communication system between all of the components. From this point the robot and *Mario* will be used interchangeably to describe the robot. The communication system is an important part of the project to easily be able to communicate between all of the different parts in the project. In this project we have used Zero Message Queue[31] which is an open source messaging library.

4.1.1 Zero Message Queue

Zero Message Queue (ZMQ) is a message-oriented middleware library. It supports different types of communication such as multicast and TCP, and is therefore suitable to use as a concurrency framework. Another advantage with ZMQ is that it can be broker less, which means that it does not need any server for sending messages. ZMQs brilliant implementation of its message batching[42] helps decreasing the number of traversal through the stack compared to sending each message on the I/O bus as in an ordinary message-queue. This feature improves ZMQs throughput and latency significantly compared to other message libraries like RabbitMQ and QPID. In [50] five message queue libraries have been compared with each other, showing different trends for the different libraries. This shows the ZMQs outstanding performance compared to many of the libraries using message brokers. A research of several communication platforms was done by CERN to replace their old system Remote Device Access (RDA) [21]. The comparison of different libraries showed that ZMQ was among the best in both latency, throughput and stability. ZMQ is designed for communication with low latency for high message throughput. The fact that ZMQ runs on most modern platforms and support language bindings for most programming languages[1] makes it suitable for

[1] http://zeromq.org/bindings:_start

the Eurobot competition. In the project different programming languages has been used. The artificial intelligence (AI) is written in Java, while the control system and positioning system is written in C++. The request-respond (client-server) model was the model selected for our project. It is an unidirectional communication between client and server. The simple flow of this model is shown in Figure 4.1.

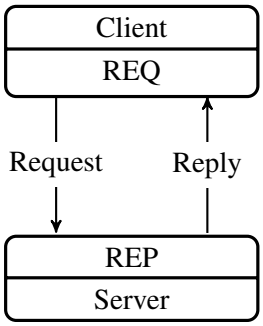


Figure 4.1: Request - reply. The client sends a request to the server and waits for the server to reply

ZMQ has a large open source community, with many contributors which makes the further development of the ZMQ library fast. The ZMQ project is licensed under LGPL. Listing 4.1 and Listing 4.2 illustrates how a connection easily can be established. The socket.send() and socket.recv() methods are synchronized which ensures that no messages will be lost. In appendix B a more complete example is shown.

```
1 zmq::context_t context(1);
2 zmq::socket_t socket(context, ZMQ_REP);
3 socket.bind("tcp://*5555");
```

Listing 4.1: Setup for the server to listen on incoming tcp stream. The initialized socket is a reply socket. We bind the socket to listen for incoming requests on a given address

```
1 zmq::context_t context(1);
2 zmq::socket_t socket(context, ZMQ_REQ);
3 socket.connect("tcp://localhost:5555");
```

Listing 4.2: Client request to connect to server. The initialzed socket is a request socket. For the client we use a connect rather than bind, since the client in our case only need to send message to the server

Another advantage with ZMQ is that it does not require you to start up in any particular order. If we first start the client, it will try to connect until it succeeds. This makes working with ZMQ quite easy, but in cases where the client should do other things if no server is running, a timer should be implemented since ZMQ does not detect disconnections.

ZMQ has different types of communication, such as publish-subscribe. This structure can be seen in Figure 4.2. This shows how easy applications can be built with ZMQ. For a control system where several of the systems connected need the

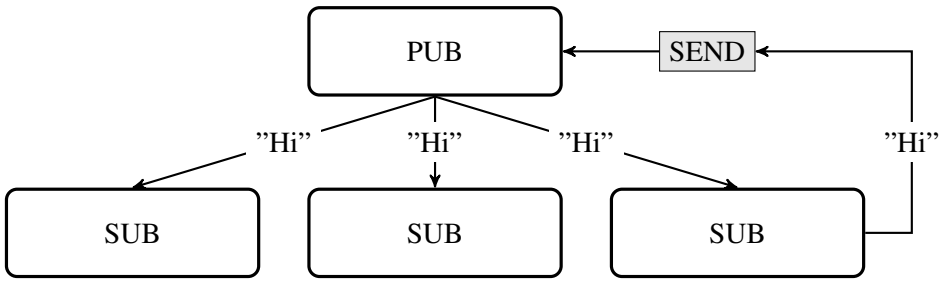


Figure 4.2: Publish - Subscribe. The client sends a request, to the server, and waits for the server to reply

same information sent between each other, would this be an easy and good way to implement the communication.

4.1.2 Computer

The computer acts like the robots brain. It handles all the heavy computation, and runs both the planning algorithm as well as reading the serial data from both the position system and the sensors. The planning algorithm decides for each step what action to perform. It then sends the goal position, and the action to be performed to the motor control system. The robot then applies the force to the motors that is needed to move to the goal position. Meanwhile the robot sends the values from the encoders to the positioning system, where we apply sensor fusion between the encoders and the beacon system. The new estimate of the position is sent back to the robot control system. Then after an action is performed, the AI asks the sensors for new information.

The reason why a computer was used as the robots brain was to be able to program in different programming language, and to make the programs relatively separate from each other. With the use of a computer and communication with ZMQ it is easy to make different programs to send information between them. The computer used on the robot was a Toshiba NB550D-105. To make the computer fit on the robot we needed to do some modifications to it. The screen was removed and hard drive moved to make more room. A twisted-pair cable was used to be able to log into the robots computer through another computer. A start script could then be run, starting all the programs needed. The AI then waited for the start signal from the start sensor. The robots control flow, can be seen in Figure 4.3.

4.2 Collision avoidance

The collision avoidance system is comprised of both ultrasound and infrared proximity sensors. An ultrasound sensor with a sensing area of approximate 1 meter, was placed in the center at the top of *Mario*. Two infrared proximity sensors were placed at the front sides of the robot. They were used to check for opponent robots coming from the sides. At the back of the robot another infrared sensor was placed for checking for opponents when reversing.

The stopping distance in front of the robot was set to be sure not to crash into the opponent robot. Since the stopping distance also need to take the other

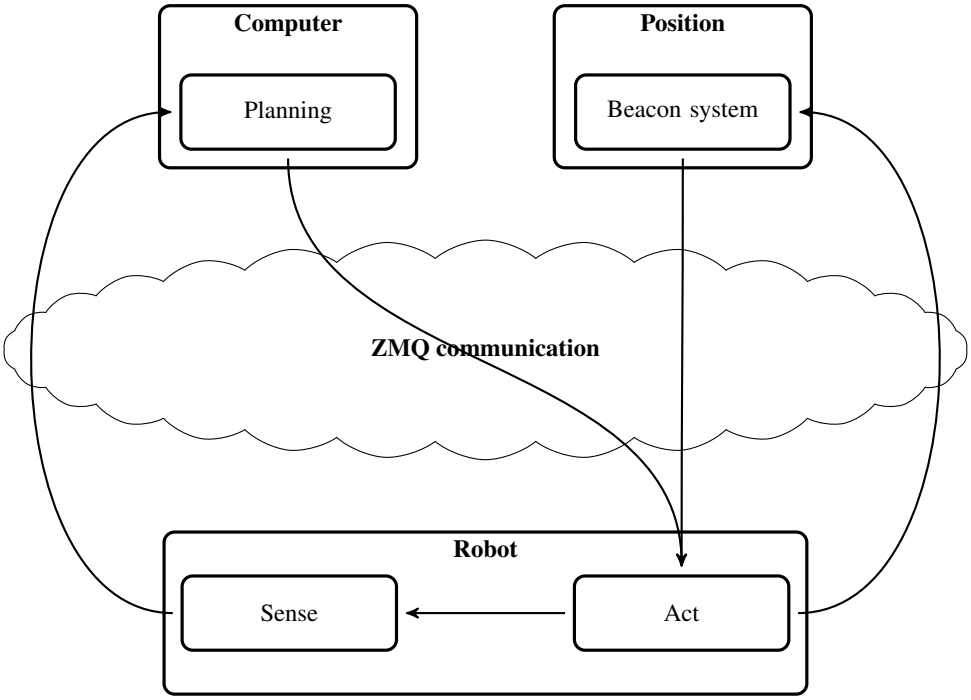


Figure 4.3: Open control architecture for Mario

robots stopping range into account, the distance should be set longer than *Mario's* stopping range. In section 6.2.4 velocity and stopping tests have been performed to show the performance of *Mario's* motors and the acceleration and deceleration of the robot. Based on these result a sensible stopping range can be set. For the rear end, a shorter distance may be set since the robot does not drive as fast when reversing. In Figure 4.4 the areas the different sensors covers are shaded. Since the robot stops before turning, no sensors at the sides where needed for the collision avoidance system. All the collision avoidance sensors were placed about 30cm high, so that only the opponent robot would be detected while non of the cups or stands placed on the game area would be detected by these sensors.

4.3 Design of the robot

Since this was the first year UiO participated in the Eurobot competition no robot skeleton from previous years had been made. We started by using a robot frame from Sonen at IFI[2] (Open zone for experimental informatics) that was available for us to use. Without any major modifications is the frame still the same. All tools and mounts for the robot have been design to fit this frame. At the end of the project, a laser cutter was available for us to use. The laser-cutter was used to make the upper floor of the robot, where the beacon tower is placed. If the laser cutter had been available in an earlier stage of the project, the design of the robot would be different than it is today considering shape and size. We recommend coming Eurobot projects to take this in use early to be able to design the entire frame of

[2]http://sonen.ifi.uio.no

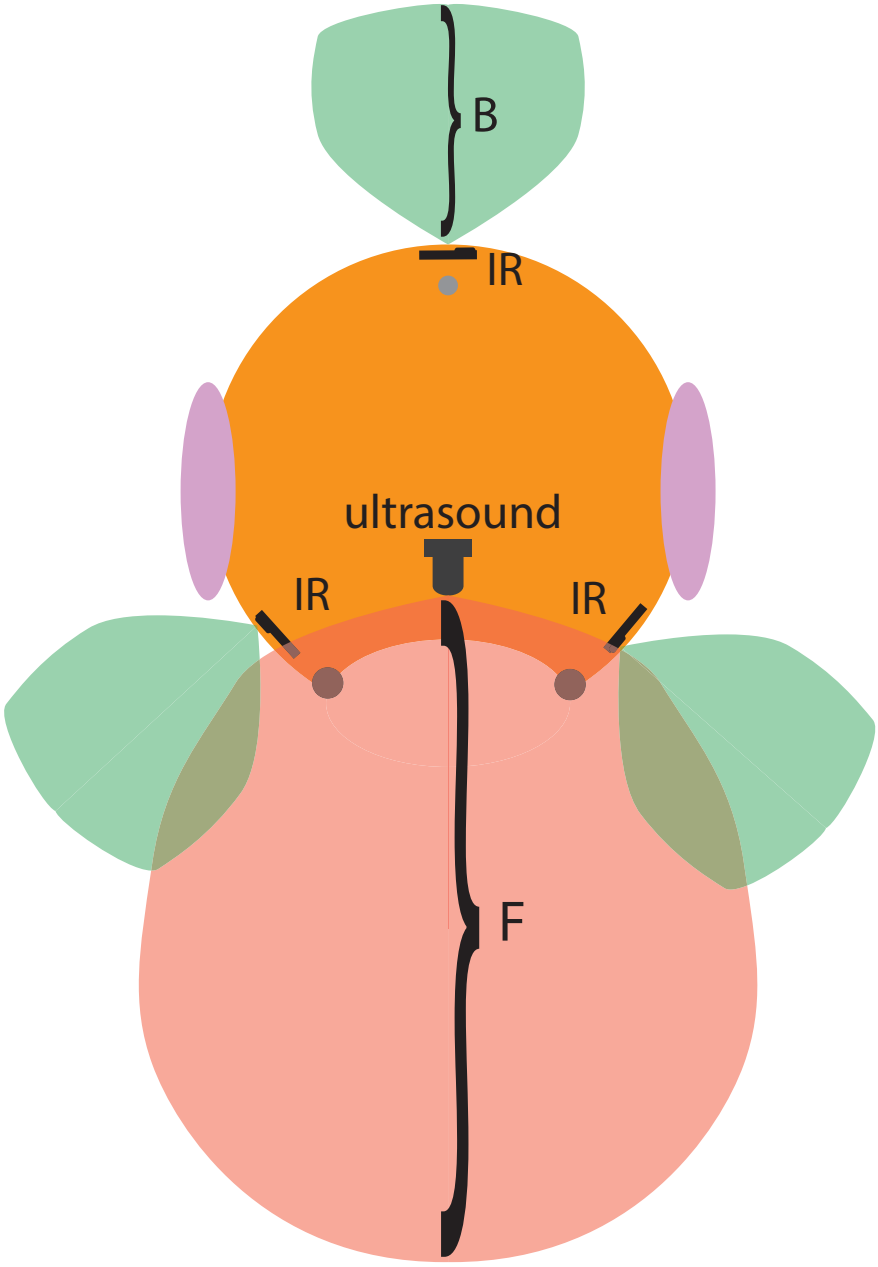


Figure 4.4: The location of the different proximity sensors are shown in the figure. Stop area is marked as a red shadow for the ultrasound sensor, and green shadow for the infrared sensors

the robot. A squared robot will be more suited for the square game areas in the Eurobot competition than the round robot that we used.

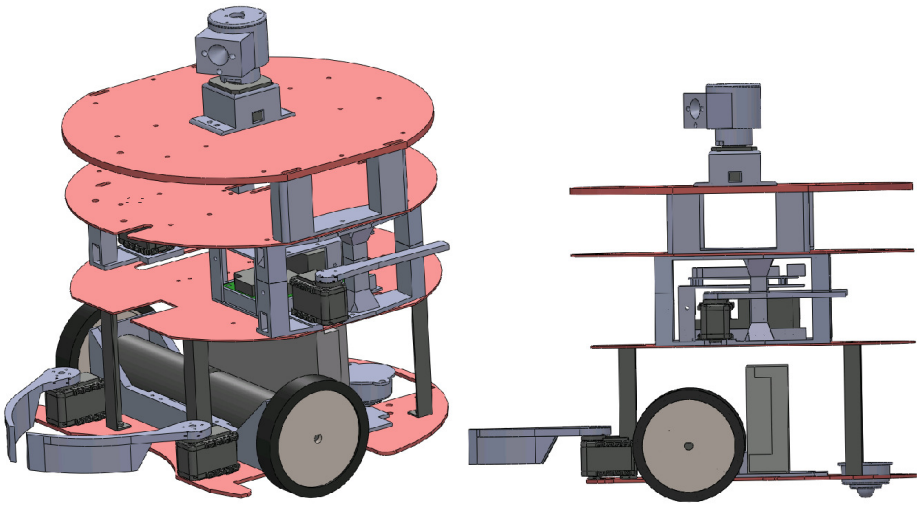
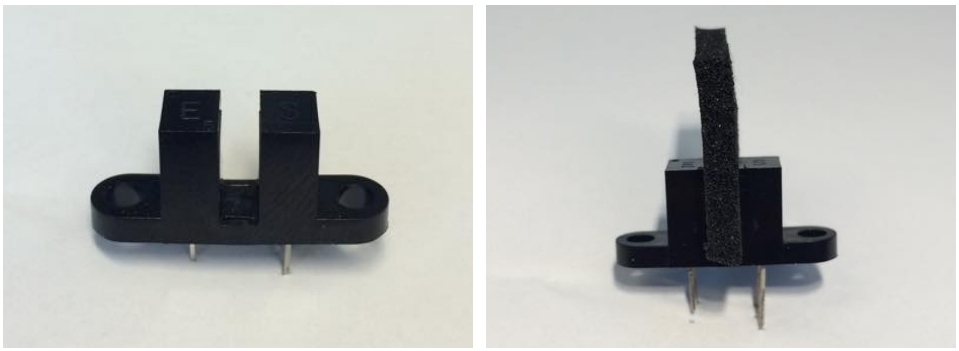


Figure 4.5: Robot design from solidWorks

The robot needed a start cord for remotely be able to start the robot. Without this the robot would not have been approved for the competition. This mechanism was made with an optical switch (OPB960T51). The sensor consists of an IR sender and an IR receiver. A piece of neoprene rubber, the same material used for clogging the beacon tower, was used to break the IR connection. When removed the receiver received the IR signal, and a signal was sent to the AI to start driving. The sensor can be seen in Figure 4.6 both with and without the neoprene rubber.



(a) Start sensor, without the neoprene rubber. (b) Start sensor, with the neoprene rubber.

Figure 4.6: The start sensor for the robot. In (b) a piece of neoprene rubber is blocking the IR signal. in (a) the neoprene rubber is removed, allowing IR signal to flow between the transmitter and receiver side, which indicates that the robot is allowed start.

Figure 4.7: The playing table we built for a more realistic testing environment. Built in 3 seperate modules to easily stow away.

4.3.1 Playing table

In order to test the robot in an environment similar to the competition, we built a replica of the game table. Both the table and the stairs were built using particleboard. The game table was built in three separate modules, which made it easy to move around, and stow away when not needed. The table was spray painted to make it as similar as possible to the competition table, with the start areas, and the red section in the middle. In Figure 4.7 our test table can be seen.

The stand objects were initially 3D printed, but they were switched to wood in order to make them as heavy as the objects in the competition. Both the wooden stand and the 3D printed stand can be seen in Figure 4.8. The clapperboards were not built since the task to flip these down was relatively straight forward. The positions of the clapperboards were marked on the edge of the table, so that we could see that the robot was able to hit the clapperboards, Figure 4.9b shows *Mario* flipping the clapperboard during one of the matches. In Figure 4.9a the stands, clapperboards and popcorn cups that were used in the competition is shown. For training we used glass as the popcorn cups. In the competition we saw that the popcorn cups were lighter than expected making the robot unable to deliver the popcorn cups with out them falling over.

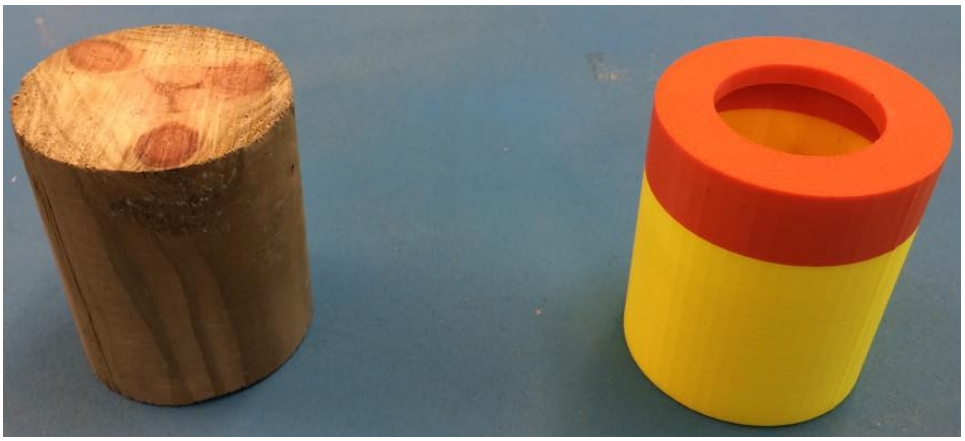


Figure 4.8: Both the wooden and 3D printed stand

(a) Stand objects, popcorn cups and the clapperboard are shown. The picture is taken from one of the official training table during the competition

(b) *Mario* about to flip one of the clapperboards during the second match

Figure 4.9: (a) show the objects used in the competition. (b) shows when Mario is about to flip one of the clapperboards. Both pictures are taken during the Competition in Switzerland.

4.3.2 Manipulators

To be able to move the object around, and to flip down the clapperboards, two different sets of manipulators have been made.

Arms to flip the clapboards

Arms at each side of the robot were designed to be able flip the clapperboards. The arms are attached at the same height as the clapperboards in order to be able to close them. They are controlled by a dynamixel servo for each arm. The arms were 3D printed, and different length of the arms were designed. The longest arms were the best suited in order to guarantee that the clapboards fell even though the position was not 100% accurate. Figure 4.10 show the robot with the extended arms at each side, both in SolidWorks and the complete robot.

Collectors

Grippers to collect the cups and stand objects were redesigned shortly before the competition. Initially a lift with capability to lift one and one cup and place them on top of each other had been designed. The lift mechanism was designed so that one dynamixel controlled the grippers with the use of opposing gears to synchronize the two arms. Another dynamixel servo controlled the elevation of the lift, by rotating a gear that stands up to a belt with tags fitting the gear. The belt was connected to the actual gripper plate, controlling its height. Right before the competition we found out that using larger arms to collect the objects would save a lot of time. We would then be able to collect the stand objects faster. the more objects the robot was able to collect per round would result in higher total score, even though

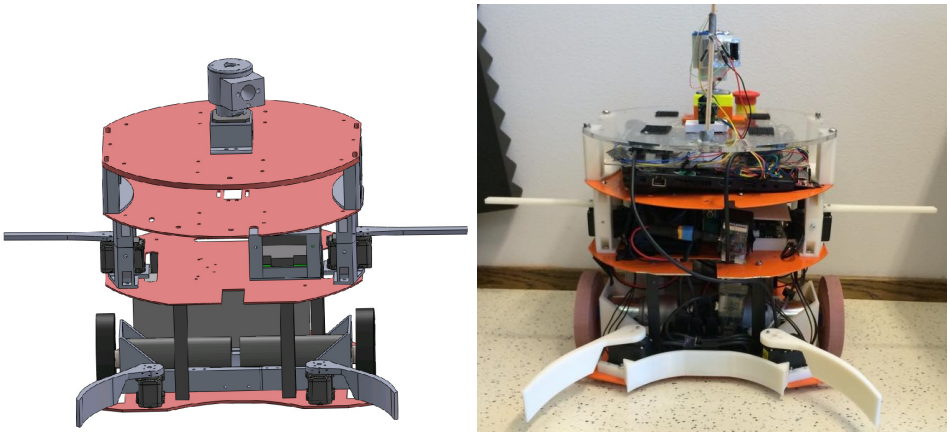
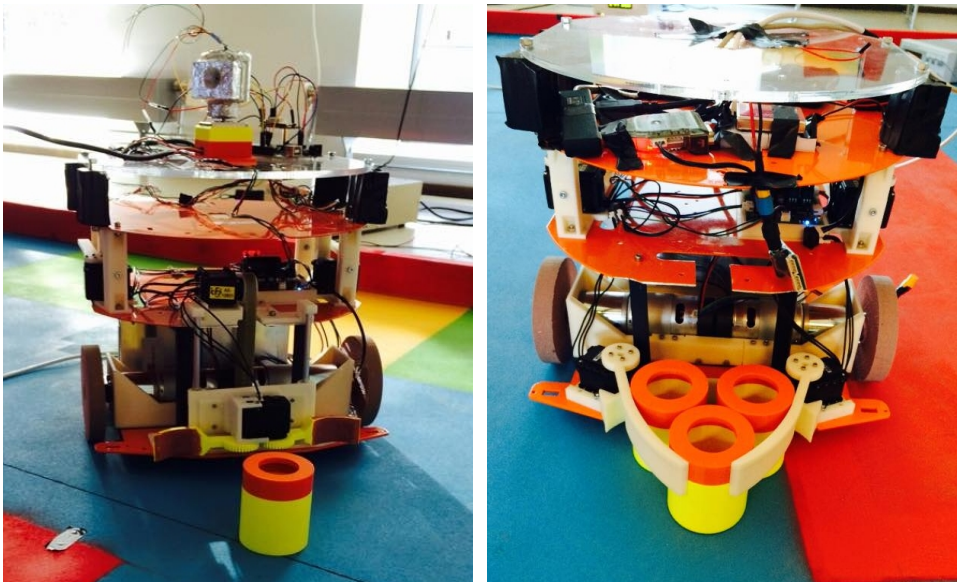


Figure 4.10: Robot with the arms and collectors fully extended. To the left is the SolidWorks model, and to the right is the actual robot

we dropped the task of building tower of the stands which would have resulted in bonus points. Figure 4.11a shows the lift which is restricted to lifting only one object at a time.



(a) Lift (b) Collector

Figure 4.11: In (a) the initial lift of the robot is shown. (b) shows the collector arms that were used on the final robot

The collectors are like the lift controlled by two dynamixels. The collector arms are also 3D printed. They are designed to be able to quickly collect several objects at once, as can be seen in Figure 4.11b. Three different positions were pre-programmed for the collectors:

- Closed - When the robot turns with objects, this position should be set

- Half open - When delivering objects, this position should be set.
- Open - When collecting objects, this position should be set for a wider collection area.

4.3.3 Motors

Devantech RB-Dev-38 motors were used on the robot. These motors were used mainly because they were relatively cheap with built in hall sensors to measure the speed and rotation of the motors used in the odometry. It also comes with 49:1 reduction gearbox. The motors had some backlash, that made it a bit inaccurate which became a problem with these motors. They were also a bit large, taking up much of the first floor of the robot. The motors did not have active brakes that made the stopping range a bit longer than those teams using motors with active brakes. Devantech RB-Dev-38 used electrical brakes, shorting the power and ground resulting in an inductance created by the motor that powered the motor in the opposite direction. The motors are driven by 24V from the DC-DC converter. Both the motors and the DC-DC converter can be seen in Figure 4.12. The complete implementation of the motor and control system is described in depth in Eivind Wikheims master thesis[55]

Power system

The robot is driven by three 14V Lithium polymer batteries (LIPO). Two batteries in serial goes in to the 24V DC-DC converter, that converts the input voltage to 24V on the output. The output from this converter is used to drive the motors. The other battery goes into a 12V DC-DC converter, and the output runs both the stepper motor for the beacon tower as well as the dynamixels controlling the different manipulators.

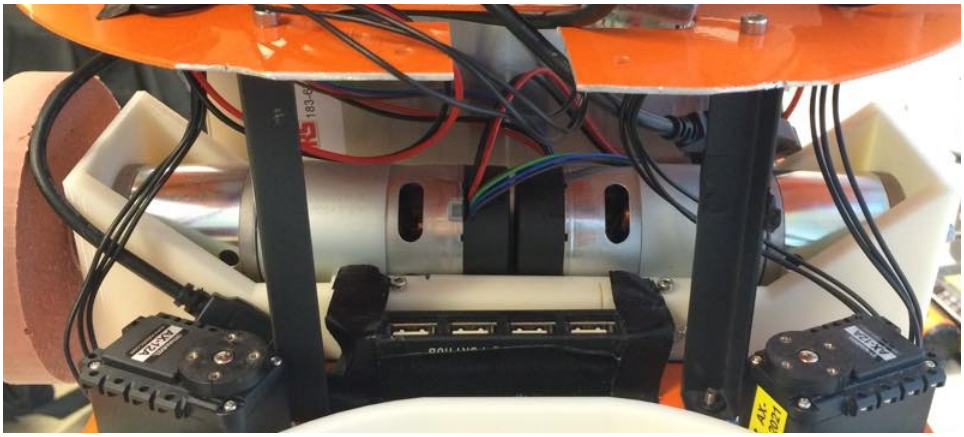
Wheels

As described in section 3.3.1 good wheels are crucial for a good position estimate. After a lot of testing with different wheels, which can be seen in section 6.2.4, we found that there were few wheels that met our expectations when it came to both accuracy and grip. Our solution, was a 3D printed wheel cast in silicone, shown in Figure 4.13. The wheels are a bit softer than the wheels with hard plastic. This results in a larger surface area for the silicone wheels than for the plastic wheels which helps improve the grip.

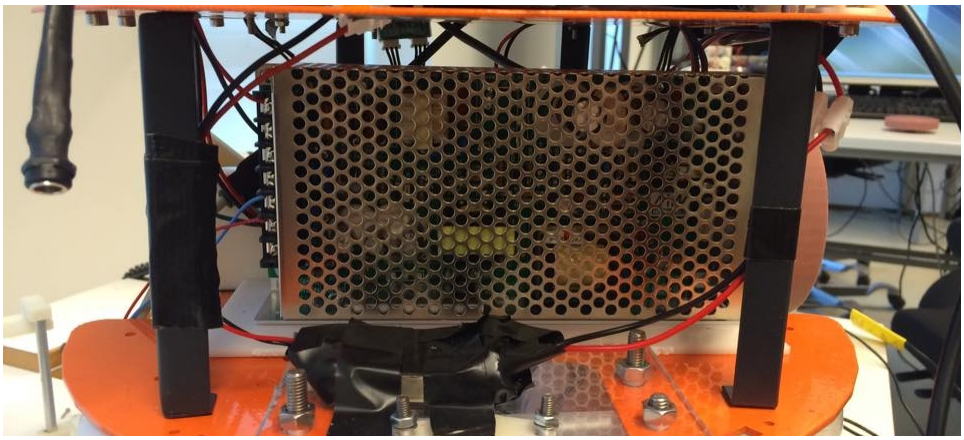
Two different support wheels were used, and are both shown in Figure 4.14. They both produced a lot of errors. We had ordered a different rear support wheel to use in the competition, but this did not arrive and we had to use use the ball wheel seen in Figure 4.14b.

The wheel seen in Figure 4.14a was initially on the robot frame we used. The wheel in Figure 4.14b was an old wheel borrowed from Sonen[3]. The wheel worked well for a while, but some issues with this wheel occurred during testing right before the competition. The results can be seen in section 6.2.2. During the

[3] http://www.sonen.ifi.uio.no

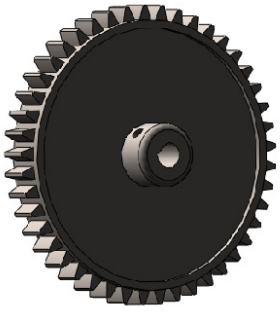


(a) Devantech RB-Dev-38 mounted on the robot

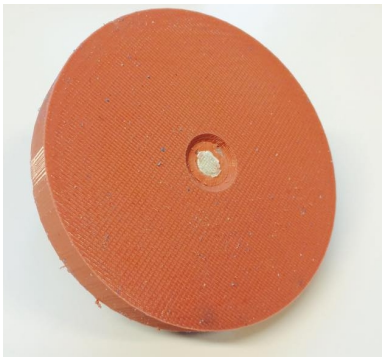


(b) The 24V DC-DC converter mounted behind the motors on the robot

Figure 4.12: (a) shows the motors used on the robot. (b) shows the DC-DC converter, converts the 28V from the two LIPO betteries, down to 24V which is used to drive the motors



(a) SolidWorks sketch of the wheels insides



(b) 3D print wheel molded with silicone

Figure 4.13: The wheel from the SolidWorks sketch in (a) works as the core of the silicone wheel, in order to make it harder and more precise, while the silicone gives the wheel a good grip. In (b), the core (a) is casted in silicone

(a) First rear support wheel used on the robot (b) The rear support wheel used during the competition

Figure 4.14: (a) is the first rear support wheel used. (b) is the wheel used during the competition

competition this was solved by lubricating the wheel to make it slide with an even friction throughout the competition.

4.4 Artificial Intelligence

The robots Artificial Intelligence (AI) use Goal Oriented Action Planning (GOAP), and is implemented by Bendik Kvamstad. The details of this system is described in his thesis[38]. In GOAP a sequence of action is planned in order to achieve the goal. The sequence of action also depends on what state the robot is in. This makes it suitable for autonomous robots, where the environment might change and new routes might be better suited. The same goal might have different sequence of action depending on the state. If the goal for the robot is to score a point, the different sequences of action might be:

- Needs object -> *Drive forward to object* -> *pick up* -> *Drive to base* = point
- Needs object -> *Rotate and drive to object* -> *pick up* -> *drive to base* = point

For the Eurobot competition the sequence of action might change due to obstacles or the other robots are blocking the path. Compared to a Finite State Machine (FSM), GOAP do not have connection between the different actions, and it is therefore possible to remove actions without needing to change other things in the code. Instead GOAP uses a cost for each action, and a high cost action will not be chosen over a low cost action. Using simulation and reinforcement learning the robot has been trained to evaluate the feature weights, and choosing the highest ranked plan. If we use the same example as previously, we can assign these different costs:

- *Drive forward to object*: Cost 2

- *Rotate and drive to object*: Cost 4
- *pick up*: Cost 1
- *drive to base*: Cost 2

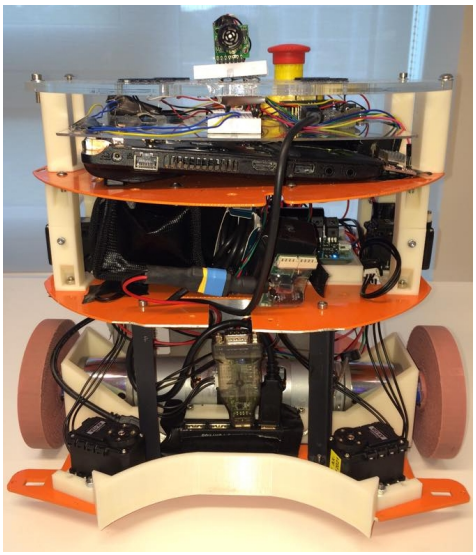
When looking at the set of actions above, possible action sequences can be

Needs object -> *Drive forward to object (2) -> pick up (1) -> drive to base (2)* = **points (5)**
Needs object -> *Rotate and drive to object (4) -> pick up (1) -> drive to base (2)* = **points (7)**

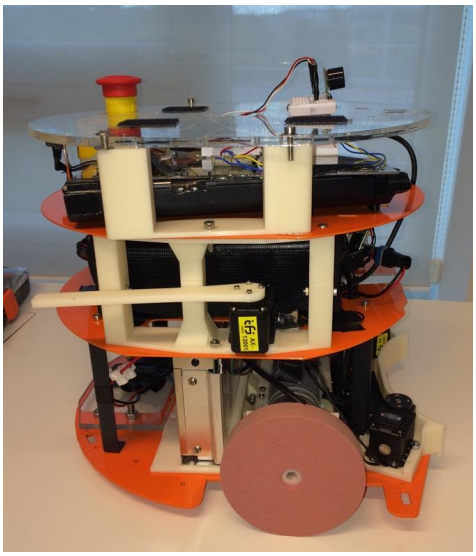
These are very overall actions, but should give a good view of how the GOAP is implemented. All actions have preconditions and effects. The precondition is the state that is required to run the action, and the effect is the change to a state when the action is complete. Due to the redesign of the robot shortly before the competition, the AI was hardcoded under the competition to achieve the most points, since the GOAP was implemented with the use of lift at that time, and not for the collectors that was used in the competition.

4.4.1 Robot

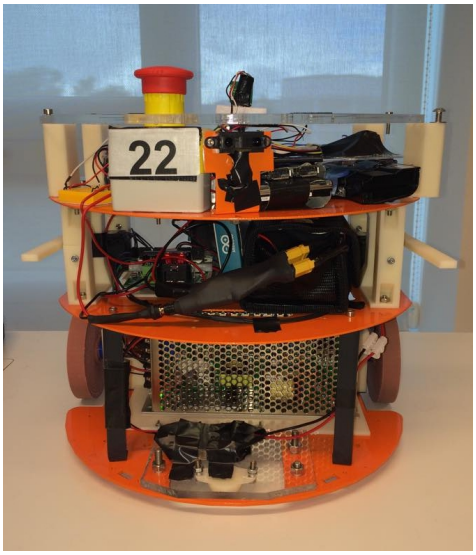
The robot without the collector arms and beacon tower is shown in Figure 4.15. On the ground floor of the robot, the motors and converter can be seen. On the first floor, the batteries, motor controller, and the arms to flip the clapperboards are found. The PC can be seen in the second floor, this is where the two IR distance sensors were placed as well. A plastic plate with the Arduinos for the sensors and beacon tower is placed on top of the PC. At the top floor the beacon tower was placed, with the opponent beacon stand and the ultrasound distance sensor. The stop button is easily accessible in order to cut all the power of the robot is needed.



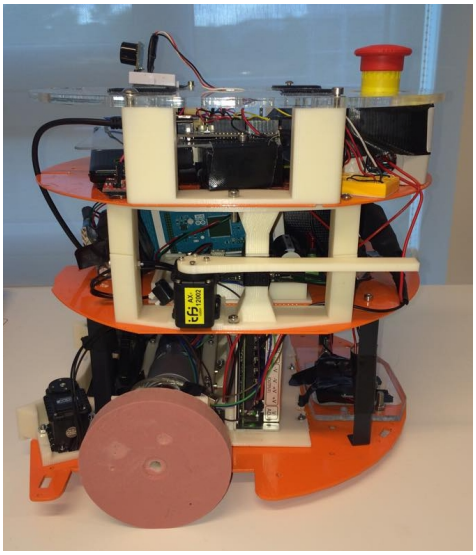
(a) Robot front



(b) Robot side1



(c) Robot back



(d) Robot side2

Figure 4.15: The robot with some of its components

Chapter 5

Design of the positioning system

In this chapter the implementation of the beacon system is presented. And the procedure of the tower is described.

5.1 Requirements for the beacon system

The minimum requirements for the beacon system are listed bellow:

Beacons:

- Max size 80x80x160mm
- Emit IR signals to the entire game area

Beacon tower

- Max size 80x80x80mm
- Should be able to receive signals over the entire game area

5.2 Low cost beacon system

The competition rules allow for each team to place five beacons, three beacons can be placed on fixed location at the side of the playing table, while the two others can be placed on the opponent robots. With the use of the three fixed beacons we can build a positioning system. Since these beacons never moves they will be used as our global positioning system (not to be mixed with GPS). One way to find the position of our robot is by looking at the angles between the three beacons. In land surveying this has been an extensively used technique. We can then apply Tienstra formula as described in section 3.2.1 to calculate the position of the robot. Our positioning system is a cheap and simple solution. It consists of IR emitters in each of the beacons, and an IR receiver in a rotating tower placed on top of the robot shown in Figure 5.4. All the parts that are used in the system are cheap and easy acquire, which was an important aspect in the design process. In [45] a system based on the similar principle with IR beacons is described.

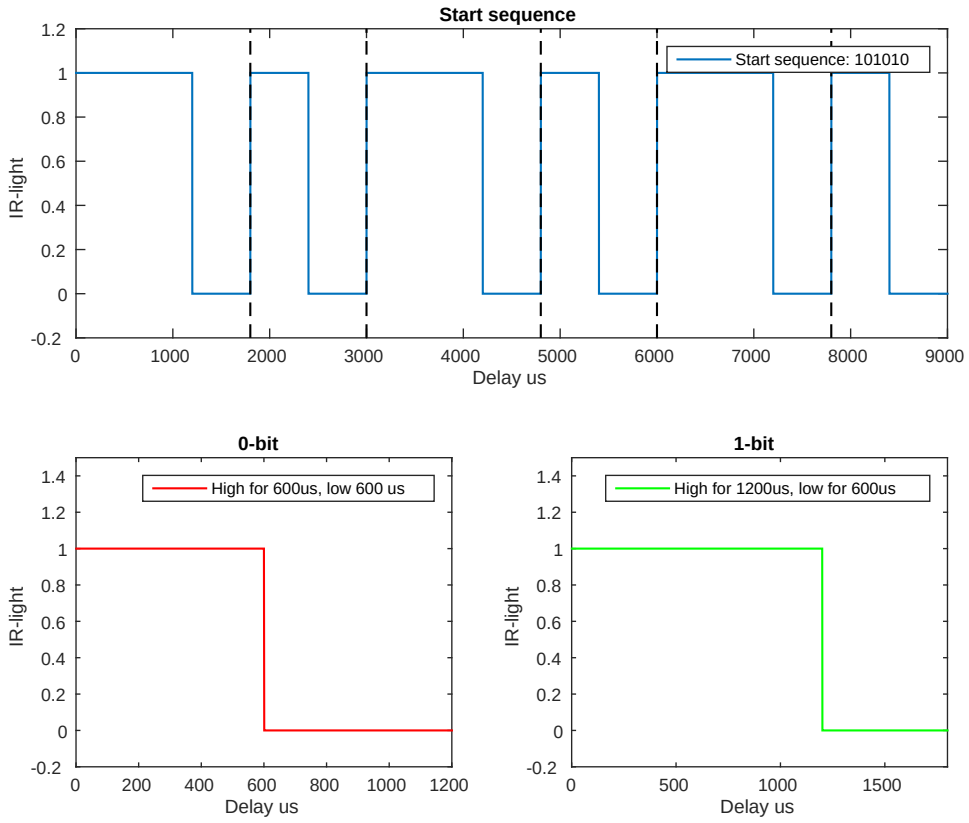


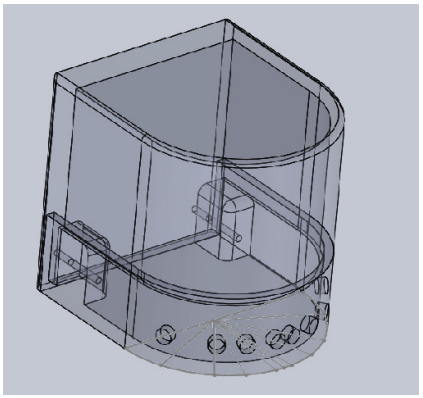
Figure 5.1: Definition of high and low IR signal, and the start sequence for a signal

5.2.1 Beacons and infrared codes

The beacon system consists of 3 beacon senders, and one receiver tower. The beacons are composed of multiple IR-LEDs (Infrared 940nm) to cover the hole game area. In section 6.2.1 the placement of the IR-LEDs are described, and the illumination of each IR-LED is illustrated. An Arduino nano is used for controlling the beacons and emitting an unique code that is different for each of the tower. This code is used by the receiver tower to identify which of the beacons that sent the signal. The infrared codes uses On-Off Keying (OOK). This means that the signal can be represented as either a one or a zero. A zero is an IR-signal modulated at 38KHz (high) for 600μs followed by 600μs of no modulation of the signal (low). A one is defined when the IR-light is held high for 1200μs followed by a 600μs low. In Figure 5.1 both the 0 and 1 are shown, as well as the start sequence. The shorter the code is the more signals per second we are able to send, but the code might then bee less unique.

Beacon B and C are each powered by a 9V battery, while beacon A is powered by two lithium polymer batteries (LIPO). This is because the Arduino don not manage to give enough current to drive all eight LEDs of beacon A. Therefore one LIPO battery drives the Arduino which controls a transistor for modulating the LEDs, while the other LIPO is used as the source voltage for the transistor, and drives the beacon. The LIPO batteries outputs a nominal 3.7V at 900mAh which is sufficient for our purpose. The three beacon towers are designed differently in

order to emit signal over the entire game area. Section 6.2.1 explains why the beacons have been designed differently, and why they are designed this way. In Figure 5.2 and 5.3 the final beacons are shown.



(a) Design of beacon A in SolidWorks

(b) Beacon A printed, with the IR lights

Figure 5.2: Design of beacon A. Emits IR light with an 180° *degree coverage of the game area*

(a) Beacon B

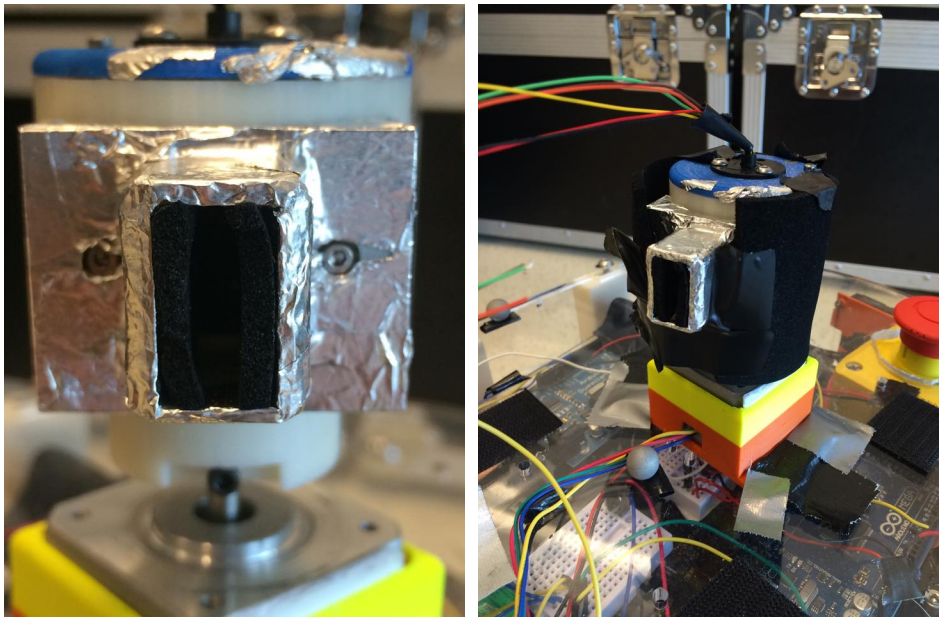
(b) Beacon C

Figure 5.3: The corner beacons. They are mirror image of each other for best proliferation of the IR signal

5.2.2 Receiver tower

The tower is design to be placed on the shaft of a stepper motor. The stepper motor is controlled by a motor controller (EasyDriver). The motor controller TB6612FNG was at first used, but since this driver did not support micro stepping the resolution of each step was 1.8° degrees. Therefore the EasyDriver driver was chosen to meet the requirement of micro stepping, and to get a higher resolution. The micro stepper adds an additional 8 steps, per original step. The stepper

motor we used (Mercury stepper motor SM-42BYG011-25) provides 200 steps for one rotation. This give us initially a resolution of $360/200 = 1.8^\circ$ which is not sufficient. With the use of the micro stepping, we increased the resolution to $360/200 * 8 = 0.225$ which is acceptable.



(a) (b)

Figure 5.4: The beacon tower. It uses a stepper motor to drive the tower. The tower seen (b) is clothed with neoprene rubber in order to absorb the IR light. The inside of the tower is also clothed with neoprene rubber

The IR-receiver is placed inside of the tower. The IR-receiver contains a band-pass filter that filters the IR signal which is modulated at 38kHz and interpret the signal as 1 or 0. Since the receiver only see IR-signal at 38KHz the infrared radiation from the sun and other sources should not interfere with the signal. The tower is clothed inside with noeprene rubber, and a lens formed as a protruding rectangle is attached to the tower as seen in Figure 5.4. This narrows the receiving width of the tower to make it more precise.

5.2.3 Procedure of the tower

In Figure 5.5 the procedure of the tower is shown. The motor controller drives the stepper motor with a 12V power supply. An Arduino runs on 5V and controls both the motor controller, as well as the angle calculation between the receiving infrared codes. The tower checks for a new IR signal for each step (0.225° degree). If a beacon signal is registered, and if it is the first signal received from that beacon, the angle to the beacon is registered as first received beacon signal for that round. When the last signal from that beacon is received, the angle is averaged as seen in

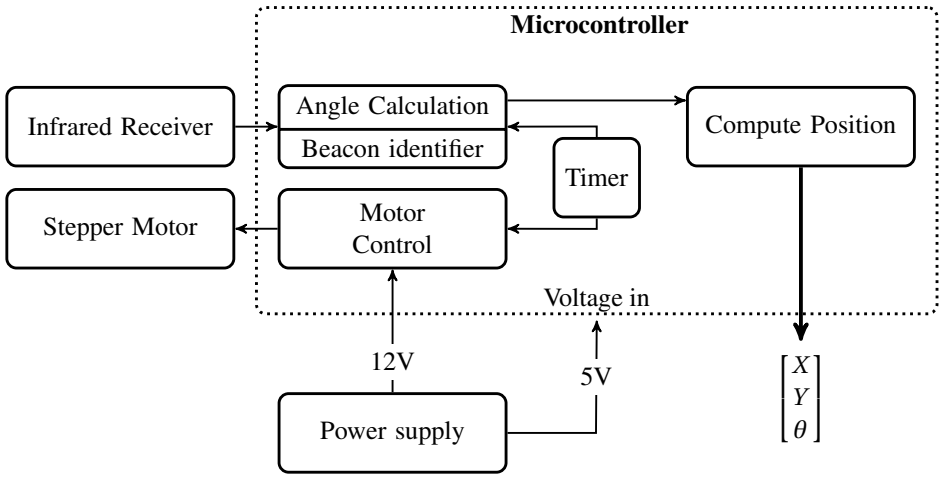


Figure 5.5: The architecture of the position system. The timer drives the motor at a given speed. In every time step it drives the motor one step, and checks for incoming infrared signals.

eq. (5.1). This gives us an estimate of the true angle to the beacon.

$$\theta_i = \frac{\theta_{i,first} + \theta_{i,last}}{2} \tag{5.1}$$

This angle is used in the position calculation where we apply Tienstra formula as described in section 3.2.1. In Tienstra formula the angle between A and B is denoted γ, the angle between B and C is denoted α and the angle between C and A is denoted β.

5.2.4 Source of the errors

The strength of the infrared signal received depends on the distance to each of the beacons as well as the angle between the tower and the beacon[46]. This results in different amount of samples received from each beacon, depending where on the table the robot is located. One of the requirement is that we at least get one signal from each beacon all over the game area. To achieve the most optimal system the average of the first and last angle received should be as close to the real angle as possible. This implies that we in Figure 5.6 want to narrow the peak as much and at the same time ensure we are able to read the hole signal.

If the opponent robot blocks one of the beacons, it will not get a valid reading. An estimation is given that for 20%-30% of the match it is likely that the opponent will block one of the beacons, resulting in not be able to calculate the position. There might also be some unwanted noise as other infrared light sources sending on the same frequency might corrupt some of the IR signal also resulting in invalid reading. Geometric constraints that makes the calculation impossible, described in section 3.2 will also impact the result.

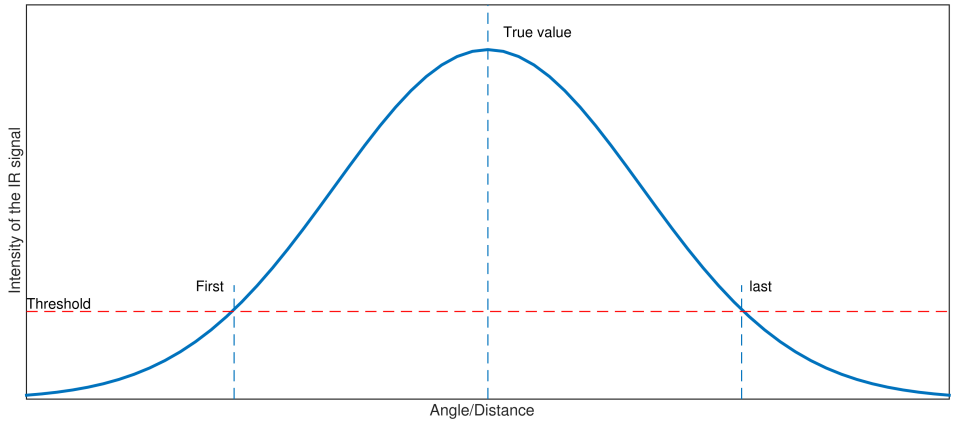


Figure 5.6: The intensity of the IR signal as the beacon tower passes the view of a beacon

5.2.5 Characteristics

The system can be built with off the shelf products, with an approximate cost of: 400NOK. The system is designed to be cheap and easy to replicate, and develop further. The approximate cost of the components in the beacon system are described in Table 5.1.

Component	**Quantity**	**Cost (NOK)**
Arduino Nano	3	100,-
Arduino Duemilanove	1	50,-
IR LED 940nm	16	100,-
Transistor	1	5,-
LIPO battery	3	50,-
EasyDrive	1	100,-
Complete beacon system		405,-

Table 5.1: The components of the beacon system with an approximate cost

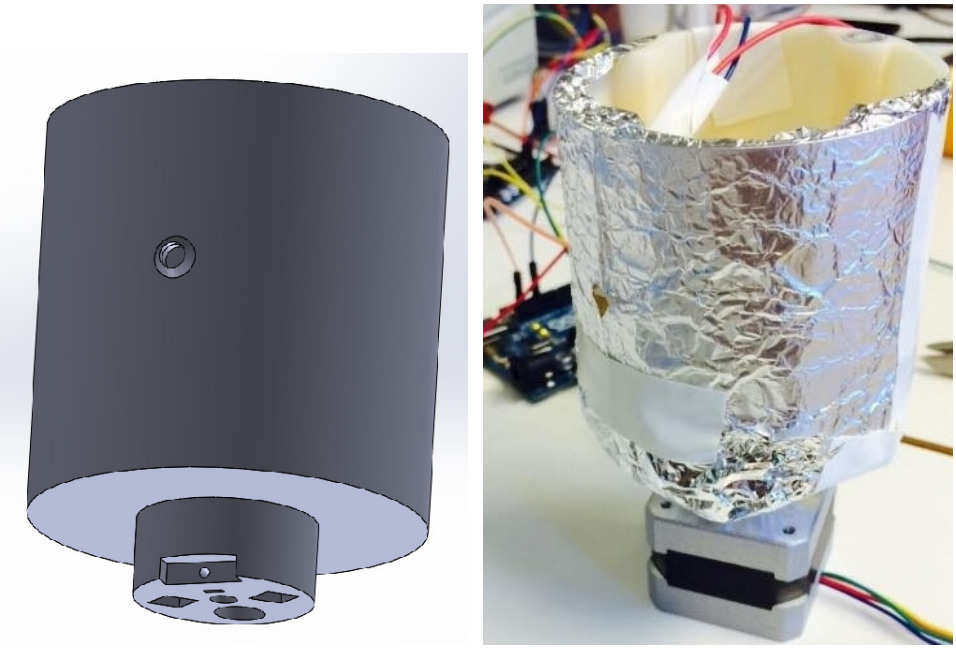
5.3 Prototypes

Several prototypes were developed during the process. The different designs were done in SolidWorks and the parts have been 3D printed.

5.3.1 First prototype

The first model of the beacon tower can be seen in Figure 5.7. This tower was designed to spin 1.5 round, and then turn around. This would assure that it registered all of the beacon signals, and the wires did not wrap around the shaft of the stepper motor. A hole, where the wires to the infrared receiver module placed inside of the tower went in, can be seen underneath the tower in Figure 5.7a. This

prototype was designed a bit bigger than needed in order to make it easier placing the different components inside the tower. The tower was printed on the Fortus 250 3D printer. Since the printer adds material layer by layer, the structure of the printed object was not completely lightproof and the IR light that were emitted from the beacons went through this material. This made the IR-receiver, receive signals all of the time.



(a) SolidWorks model of first beacon tower

(b) First printed beacon tower

Figure 5.7: The first beacon tower designed. (a) shows the SolidWorks model, and (b) shows the printed tower. Aluminum foil has been wrapped around (b) in order to reflect away unwanted IR signals, since signals went through the printed plastic

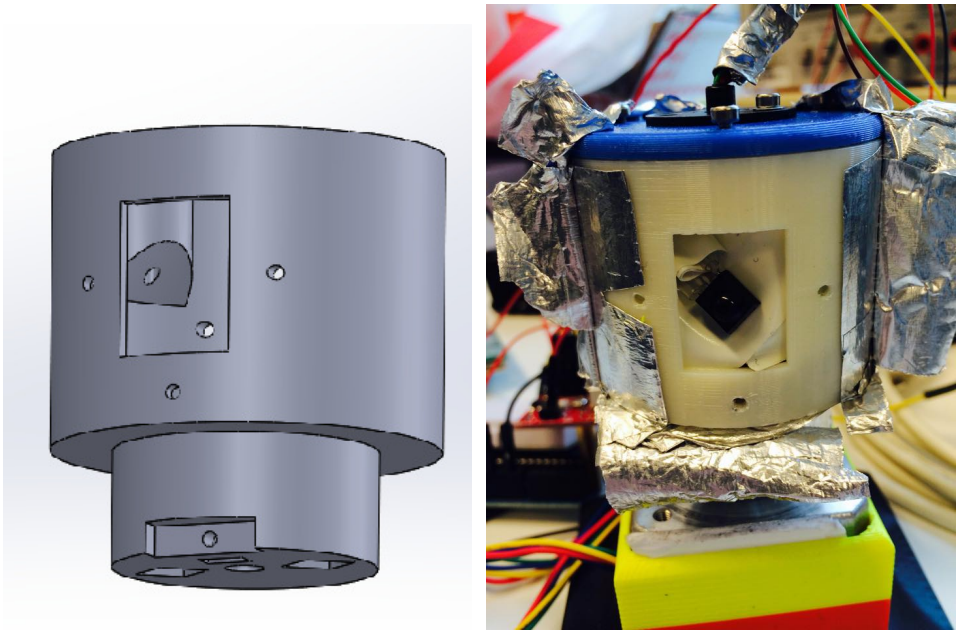
Modification

Different materials was used to try to clog the tower from any IR light. We found that few materials were able to shield the tower from the IR-light. The resulting material was a tin-foil tape that blocked and reflected the signals. The main problem with this prototype was that the tower had a quite wide area where it received signals. Due to signals easily being reflected in the small view tunnel of the tower. This prototype could then end up be able to see several beacons at the same time, resulting in invalid calculation, and it was a rather unstable prototype.

5.3.2 Second prototype

The second prototype was designed to solve the problems with the first prototype that had a to wide view. The idea was to reflect away IR-signals that came from steep angles, and only let the IR signals that were directed in front of the tower in to the receiver. This prototype was inspired by the principle of a parabola. The

tower was designed smaller, and made so that different ”lenses” could easily be 3D printed, and changed on the tower for easier testing. In Figure 5.8 the new tower is shown. The lenses were designed to easily be screwed in to three holes in the tower.



(a) New Design of the beacon tower in SolidWorks (b) The new designed tower printed

Figure 5.8: The new designed tower. Made so that different lenses could be screwed on for faster testing. (a) is the SolidWorks model, and (b) is the printed model. Aluminum tape has been taped to the tower in order to reflect away unwanted IR signals, since signals went through the printed plastic

Two lenses were initially designed as a parabola, one with 45 degree tilt seen in Figure 5.9a and the other with 55 degree tilt seen in Figure 5.9b, and the last was designed as a tunnel, seen in Figure 5.9c. The parabola lenses were made so that the IR signal would be reflected away if not emitted from the front of the beacon. They were similar to the tower, taped with aluminum tape to be able to reflect the signals.

Modification

A slip-ring was used to be able to achieve a constant angular velocity for the tower. With the slip-ring we get an electrical connection through a rotating assembly. This increased the frequency of the position calculation, since we with the slip-ring were able to make the tower rotate continuously without needing to turn around, which made it possible to calculate the position each time a new beacon signal was received. While we previously had to wait for the tower to spin one and a half round.

The size of the tower and lenses were changed several times in the design process. The tower was initially 150mm in diameter. This was so that it was

(a) 45 Degree tilt (b) 55 Degree tilt (c) Tunnel lens

Figure 5.9: The three first lenses used tested and used.

easier to work with it, and taking the different components in and out of the tower. The final tower was 70mm in diameter to achieve the requirement of max size 80x80x160mm.

Figure 5.10: Slipring that is used on the tower

5.3.3 Final prototype

For the final prototype we found that neoprene rubber could be used to absorbed the IR signal. With this material we could change the design of the lens from a parabola, back to a lens formed more as a tunnel. This helped narrow the sight of the tower further. With the use of the new lens, the width of the area that the beacon tower see can be set by changing x and h according to eq. (5.2), where x and h is respectively the width of the aperture of the lens, and h is the length of the tunnel.

Figure 5.12 illustrates how the IR signal from steep angles will be absorbed while only signals from the front is let in to the IR receiver, using the new lens.

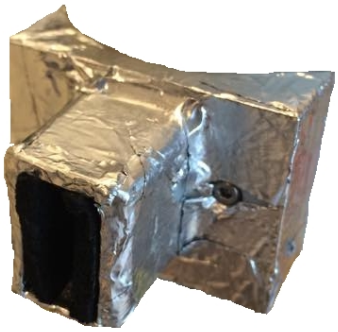


Figure 5.11: The final lens. It has a protruding rectangle filled with IR absorbing neoprene rubber at the sides in order to absorb the IR signals hitting the tunnel.

$$\theta = atan2(x, h) \tag{5.2}$$

In order to decrease θ, we can either make x smaller or h longer. For the final lens seen in Figure 5.11, the x length was $4mm$ and h length was $24mm$, resulting in an angle of $\theta = 9.5°$. This might sound a lot, but the IR receiver do not see the signal before its intensity is over the threshold as see in Figure 5.6, meaning that the angle when the tower sees the beacon is in reality smaller than 9.5°.

Heading

From the Tienstra formula we achieve the position of the robot, but we still need to express the heading. This is fairly easy to compute when we have the position of the robot as well as the angle from the robot to either of the beacons. The heading of the robot θ_R can then be expressed as seen in eq. (5.3), where θ_p is shown in Figure 5.13. In Figure 5.13 θ_A is used as θ_i.

$$\theta_R = \theta_i - \theta_p, i \in A, B, C \tag{5.3}$$

θ_i is read from the beacon system, while θ_p can be found by looking at the angle that θ_i forms with the robot. θ_p is then calculated as described in eq. (5.4), which can be put into eq. (5.3) in order to describe the heading θ_R.

$$\theta_p = 180 + atan2(Y_i - Y_R, X_i - X_R), i \in A, B, C \tag{5.4}$$

Because we use a stepper motor, we only know how many steps we have stepped from the starting position, so in order to achieve θ_i with respect to the robot direction the beacon tower needs to start pointing in the same direction as the robot.

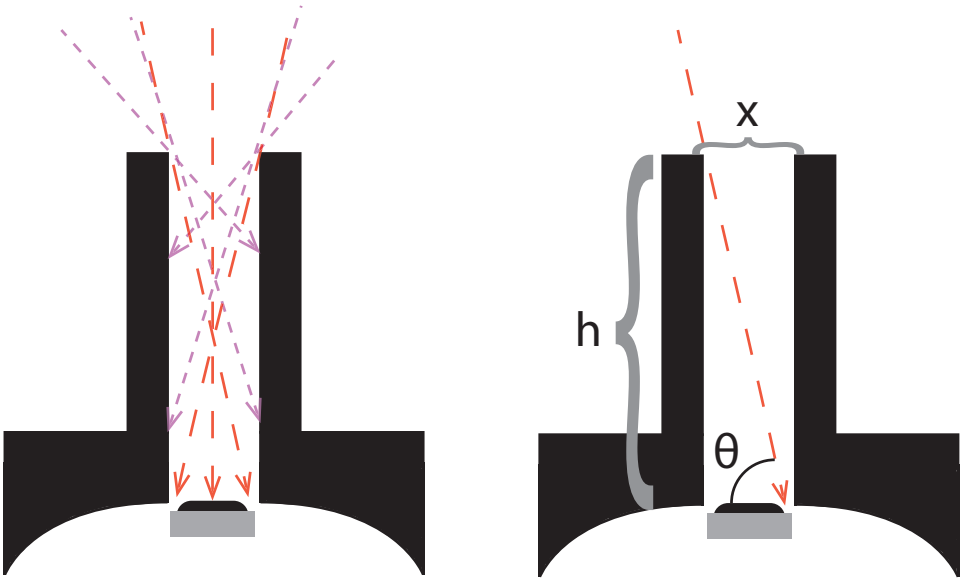


Figure 5.12: Overview of the final lens used with neoprene rubber on the tunnel side to absorb the IR-signals hitting the tunnel walls. The purple beams at the left figure are IR-signals that are absorbed by the neoprene rubber. The red beams reaches the the IR-receiver and are the signals are logged. x is the width of the lens, while h is the length of the tunnel.

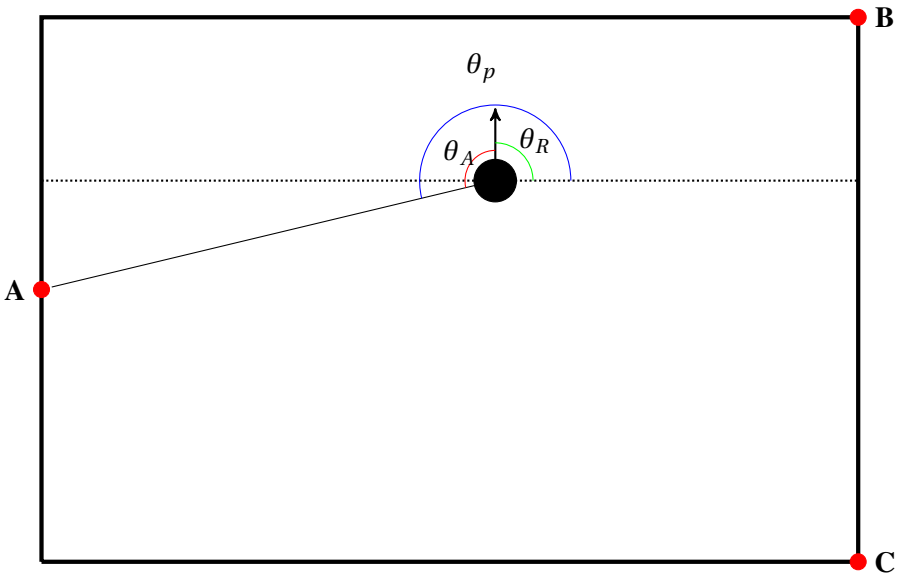


Figure 5.13: The heading of the robot is shown as the arrow and θ_R is the angle relative to the game table. The angle to beacon A is given by θ_A and the angle from 0 degree for the robot to towards beacon A is given as θ_p

5.4 Position estimation

Neither the beacon system, nor the positioning calculation from the encoders are perfect. They both have some source of error. In the section 6.2.1 the errors from the beacon system is presented. The odometry also contains errors, which may come if the parameters like the radius of the wheels, and the displacement between the wheels are not perfectly tuned. Over large areas this type inaccuracy as well as the backlash in the motors, will result in large errors. The greatest source of error is wheelspin or if the robot crashes, and since there is no way to know when this errors occur with only use of odometry, we need to correct the data with the use of other positioning systems. This is why some sort of sensor fusion between the beacon system and odometry should be applied. For the sensor fusion the Kalman filter technique has been used. The Kalman filter is optimal, in terms that it minimizes the mean square error. Though to be optimal it requires a zero mean wide sense stationary error for the measurement data. In the Kalman filter also different other sensors might be incorporated to stabilize the system further. For the heading of the robot, compass could be used. The Kalman filter was implemented in C++, and the c++ linear algebra library Armadillo was used for all the matrix multiplications. This library has a good trade off between speed and ease of use, and the syntax is rather similar to Matlabs.

Chapter 6

Evaluation and experimental results

The different systems developed and sensors used in this project all have their constraints and weaknesses. It is of great value to identify and study these constraints. This can give a more thoroughly and deeper understanding of how to improve the systems.

6.1 Purpose of the experiments

The main purpose of the thesis is research of positioning and sensor systems. The Eurobot competition was a valuable arena to develop such systems. The purpose of the experiments is to show the different constraints for each systems, and to show why different design choices have been made. The conducted experiments can be divided into five separate parts:

1. **Beacon system**
   Modification of the beacon tower, speed of the tower, placement of the infrared LEDs and comparison between lenses

2. **Positioning system**
   Simulation of the Kalman filter, Driving tests, rear wheel tests

3. **Sensors**
   Ultrasound, infrared

4. **Motor performance**
   Stop tests, velocity tests, wheels

5. **Competition**
   Competition

The data collected from both the positioning tests, and the velocity and stopping tests have been performed together with Eivind Wikheim, whom was responsible for the motor control system. The visualization and presentation of the data might be different.

Beacon system

In order to get the most accurate positioning system, and tune parameters such as the rotational speed of the tower, as well as what type of lens to use the tests presented is performed. We want to find constraints and weaknesses with the systems. The results can also be useful in order to develop the system further.

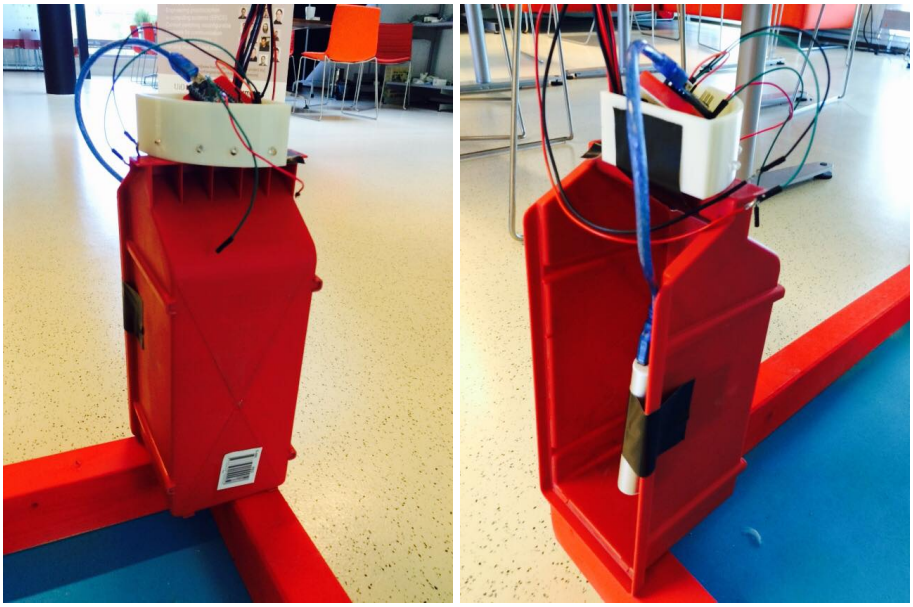
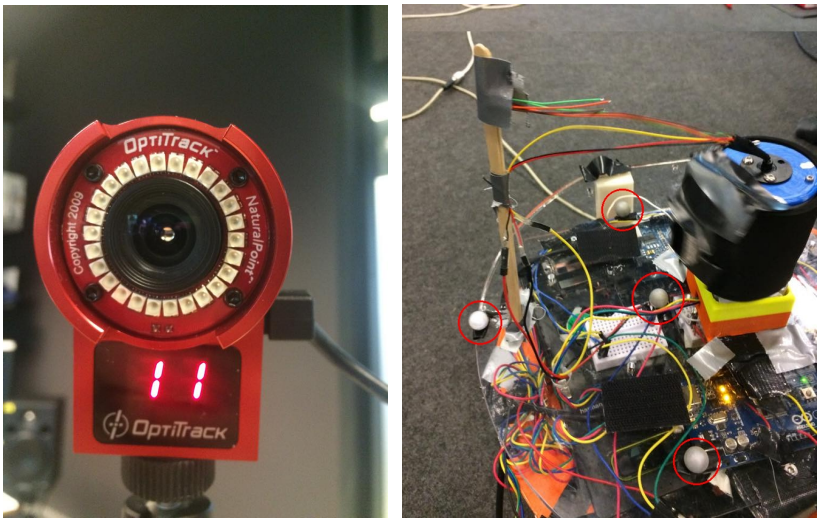


Figure 6.1: Beacon C during testing of the beacon system

Most of the tests have been performed with the beacons placed on the fixed positions around the edge of the table. Figure 6.1 shows the placement of beacon C during testing. It is placed in the same height as the beacon tower. It was during the tests powered by a 5V chargeable battery, but in the final prototype a 9V battery was used in order to fit inside the beacon.

Positioning system

Different test runs with the robot were performed in order to show the inaccuracy of all parts in the positioning systems, and to show what may cause these. To get the ground truth these runs were performed in a motion capture lab. The system used is the OptiTrack, and the setup consists of twelve high-speed OptiTrack FLEX:V100 cameras, one of the cameras can be seen in Figure 6.2a. The cameras capture up to 100 frames per second, producing a path with high accuracy. The cameras are tracking the movement of 4 reflecting balls placed on the robot as seen in Figure 6.2b. The cameras covers a large area, and can show if the robot drives off the game area. Different runs were made, to show constraints in the different systems. In all of the tests, the odometry was precisely calibrated, resulting in quite accurate positioning for small distances. The result from these tests will be valuable in order to improve the system further, and to decide what systems are worth develop further.



(a) Motion capture camera (b) 4 Reflecting balls attached to the robot

Figure 6.2: (a) shows one of the motion capture cameras used to capture the movement of the robot. (b) shows four reflecting balls attached on top of the robot. They are used by the motion capture system to track the movement

Sensor systems

The results from the sensor systems have been produced by pointing the proximity sensors against an object and logging the output data. This has been done for different distances, and the data was then plotted to show the characteristics of the sensors. The results were used to find the optimal scaling factors, for the different methods, and find what method was the most suitable to use in the competition. In the tests, the input voltage for both ultrasound and infrared proximity sensors was 5V. The object used to measure the distance to, was a piece of metal. This was because most of the robots in the competition was milled, and would therefore produce the most correct result. It is worth mention that the result will not vary too much with the use of different materials as long as it has about the same density, and are not absorbing the signals like neoprene rubber. In the tests only short distances were tested, 0-60cm for ultrasound and 0-100cm for infrared. This was because it is most important to get the short distances as correct as possible since we mainly will be using it for collision avoidance in immediate vicinity.

Motor performance

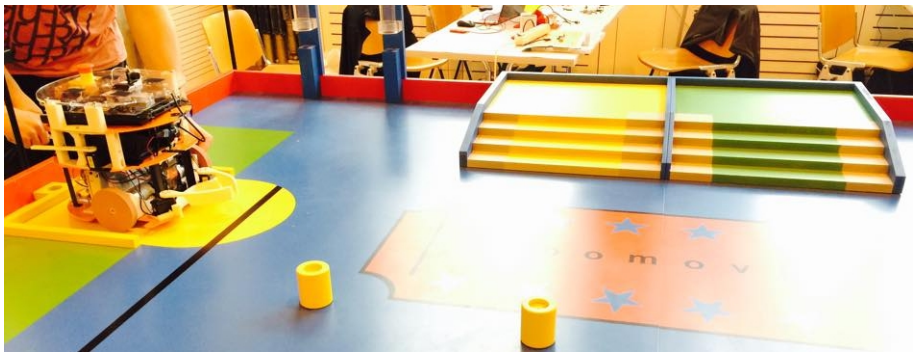
These tests have been performed by driving the robot from 0 to max speed, then after 2 seconds break as fast as possible. The tests have been performed several times in order to filter out irregularities that may occur. The tests were used for among other things such as set the stopping distance for the robots collision avoidance system. Some extra distance needs to be taken into account for the collision system, like for instance the opponent robot moving towards the robot as well as the delay from when the sensor data is read, and the AI reads it, and then sends the stop signal to the encoders. This latency is minimal, but is a factor

to take into account. The motors that we used has a regulator that can either be turned on or off, it is therefore hard to tell how the regulator is tuned, but from the results we will try to give a brief analysis of it. One important part for accurate positioning were the wheels of the robot. They should have as much friction as possible in order to avoid wheel slip. Custom made wheels were designed to meet this requirement. Comparison between the custom wheels and the other wheels have been done in order to see the advantages and disadvantages with them. The static friction test have been performed by placing the robot on the table, and hang a bag attached to the robot outside the table. Then weight were added to the bag until the robot started to slip. Then the weight of the bag was measured, and then multiplied by the gravitational force.

Competition

As mentioned in section 1.4 the competition was this year held in Yverdon-Les Bains in Switzerland from 22-25 of May. Due to the lack of sponsors and tight budget the UiO team traveled down May 21th, and stayed until May 24th, since these where the cheapest day to travel. The robot needed to be approved by Friday 22 before taking part in the competition on Saturday. Valuable knowledge from other teams, and experience from the competition will be presented in this section, as well as a short overview of how the UiO team performed in the competition. In Figure 6.3 game table where the matches were held can be seen. Three identical tables could be used for testing and tuning the robot in between the matches.

(a) The game table during a match



(b) Tuning of the robot between the matches on one of the test tables

Figure 6.3: Match and training during the competition in Yverdon-Les-Bains

6.2 Experimentations

In this section the result from the tests of several systems are presented. The tests are presented along with an evaluation of the test, as well as an analysis of the complete system at the end of each subsection.

6.2.1 Beacon system

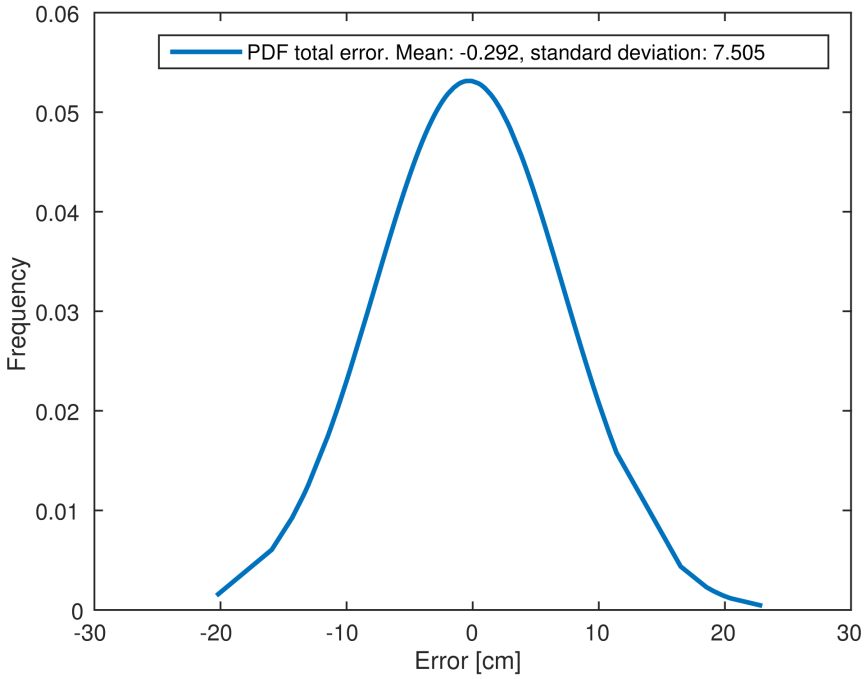


Figure 6.4: Measurement distribution of the tower, when spinning with a delay of 400μs

In order to get a measure of how good the beacon tower performs the beacon tower has been placed on a known location on the board, and the data is logged. From the data, a measurement distribution is plotted and shown in Figure 6.4. This is the measurement distribution of the error for the final setup of the beacon system using the protruding lens and tower spinning with optimal speed. This was used for finding the uncertainty of the beacon system, and to get a measure of how good the tower works. Figure 6.6 shows that these measurements are fairly close to being Gaussian. Based on this result using them in the Kalman filter should produce a good approximation of the position. In Figure 6.5 the measurement distribution of the error in x and y direction is shown. These will be used for setting the uncertainty matrix of the beacon system for the Kalman filter.

Speed of the tower

After the modification with the slip-ring described in section 5.3.2, the position from the beacon tower was able to be updated three times for each rotation of the tower. This means that if we have 400μs delay between each step, we achieve a

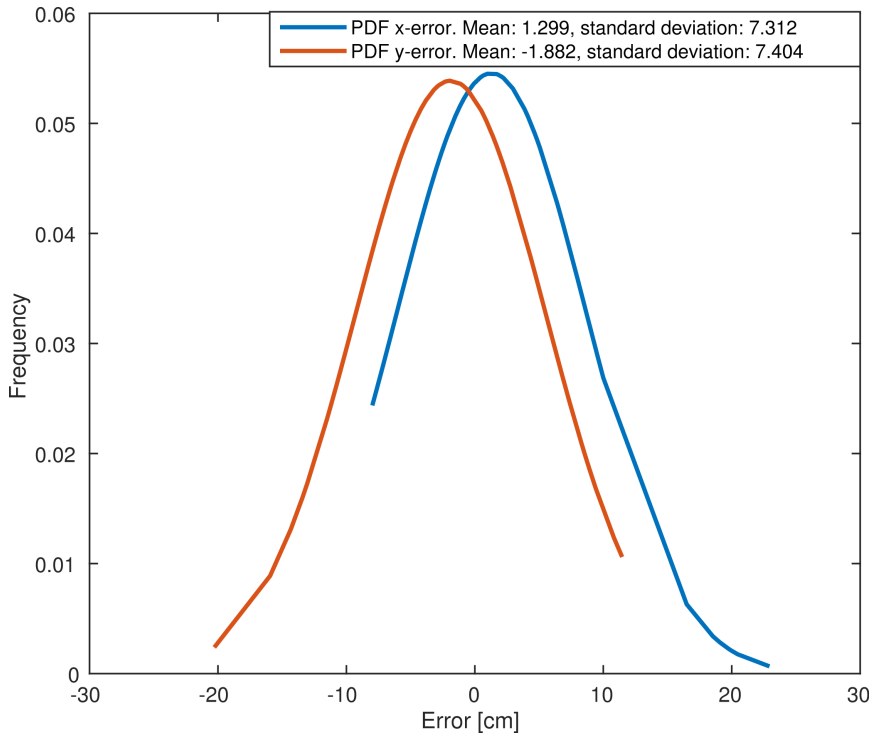


Figure 6.5: Measurement distribution, when spinning with a delay of 400μs

frequency of the position estimate as: $10^6/(400 * 1600) * 3 = 4.69Hz$. This is the frequency of the position update if all beacon signals are captured for each round. The position is not updated if one or more of the beacons are not in the towers line of sight. As mentioned the angular velocity of the tower is controlled by the delay between each step. In Table 6.1 different delays have been tested, and the accuracy of each was calculated in order to pick the most suited tower speed. With delays less than 400μs, the stepper became unstable in terms that it struggled to drive the stepper in a constant speed. This is because the EasyDriver motor controller did not provide the current needed fast enough to the stepper motor.

Delay (μs)	**Mean**	**Standard deviation (σ)**	**Variance (σ^2)**
400	-0.292	7.505	56.325
450	0.977	10.681	114.083
500	0.028	8.154	66.488
600	0.863	13.774	189.723
650	3.37	10.038	100.761
700	0.89	7.654	58.583

Table 6.1: The speed results for different tower speed.

Evaluation From Table 6.1, we clearly see that the smallest delay, resulting in fastest speed gives the least errors. Therefore this delay has been used for the stepper controlling the beacon tower. However with the use of 500μs the mean

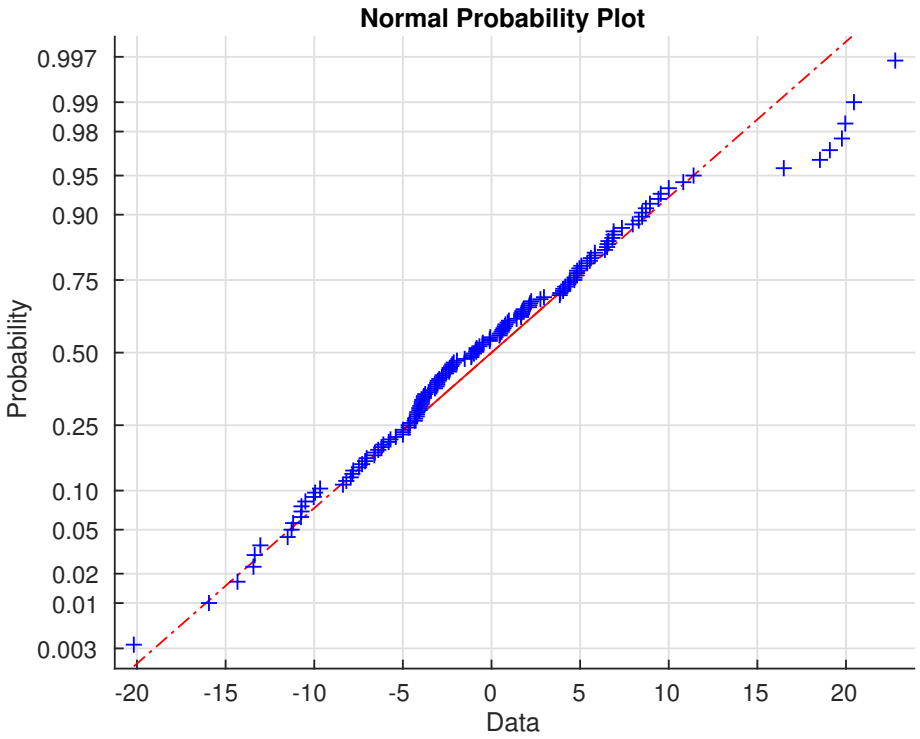


Figure 6.6: A test has been made to see if the measures follows the normal distribution. We see that the measures shown as blue +, are fairly close to the red line, implying a normal distribution

is closer to zero. But it results in a larger standard deviation, as well as it has a slower position update rate. From the modification with the slip-ring the position calculation frequency was over 3 times faster, which should make it more stable when driving, since the robot does not move as far between each time it sees a new beacon. In theory the average position updates per round for the tower with slip-ring was 3, while for the tower with out slip-ring, the average position calculations per round was ≈ 0.666, which is a great improvement

Design of the different beacons

Since the IR senders (beacons) are placed differently on the edge of the playing table, they need to be designed differently. Beacon A needs to cover an area spanning over 180° degree, while beacon B and C only need 90° degree coverage. Table 6.2 and Table 6.3 lists the angles of each IR LEDs for beacon A and beacon B.

The angles of each IR-LED were found by using 1 IR-LED placed at the beacon position, and then place the IR receiver in the corner of the table at the same side as the IR-LED. Then the IR-LED was rotated until the receiver lost the signal. Then the IR-LED was rotated back until the receiver again received the IR signal. Now the angle of the IR-LED is saved as θ_1 as can be seen in Table 6.2. Then the IR-receiver is moved along the edge of the table until no signal is received. It is then moved back until it again receives the signal. Now the area covered by the first IR-LED is known, and the same procedure can be performed to find the angle for

Beacon A		
LED	**Calculation**	**≈ Angle**
1	$atan2(45,100)$	24°
2	$atan2(131,100)$	52°
3	$atan2(270,100)$	69°
4	$atan2(300,23)$	85°
5	$180-\theta_4$	95°
6	$180-\theta_3$	111°
7	$180-\theta_2$	128°
8	$180-\theta_1$	156°

Table 6.2: The angle for each of the IR senders for beacon A with 180 degree coverage. Needs at least 8 IR senders for a full coverage of the table

Beacon B		
LED	**Calculation**	**≈ Angle**
1	$atan2(91,200)$	24°
2	$atan2(223,200)$	48°
3	$atan2(300,135)$	65°
4	$atan2(300,30)$	84°

Table 6.3: The angle for each of the IR senders for beacon B with 90 degree coverage. Needs at least 4 IR senders for a full coverage of the table. Beacon C is just a mirrored version

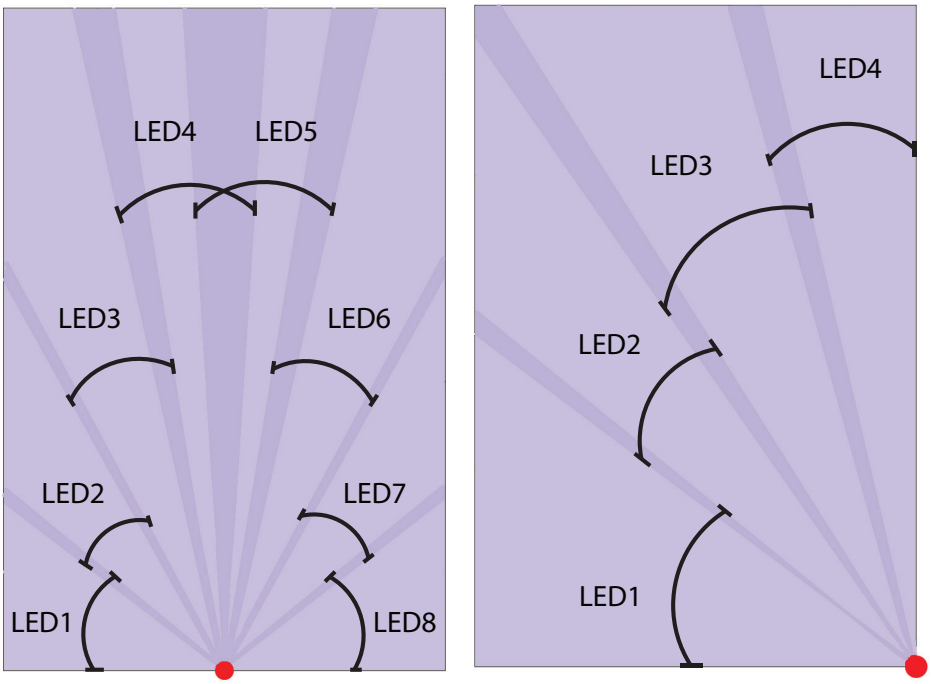
the rest of the IR-LEDs in order to cover the entire game table. Since beacon B and C are corner beacons they do only need to cover an area of 90° degree. Beacon C is placed on the other side of the table of beacon B. The design of beacon C is therefore only a mirror of beacon B. Figure 6.7 show the areas the different IR-LEDs cover, and where the IR-LEDs are overlapping each other.

Evaluation The beacons were able to emit the IR-signal to the entire game table. The design of the beacons worked good, and had room for all components, but they got a bit messy due to all wires connecting all the components. This can be solved by making the circuit as a printed circuit board (PCB).

Comparison between parabolic lenses

The second prototype of the tower described in section 5.3.2 made it possible to test different lenses in order to find the optimal design to reflect away IR-signals from the side of the tower. In Figure 6.8 the results from the two parabola lenses can be seen. They both have errors all around the table, but the lens with 55 degree tilt is a bit more stable and accurate than the 45 degree tilt. In the plots, the placement of the three beacon are shown with red dots, and the position of the tower is shown with a black dot. The calculated positions are marked with green x'es. The test shown in Figure 6.8 is a small scale of the table. In Figure 6.9 the position test with the 55 degree over the full game table can be seen. The same test with the final

lens is presented in Figure 6.10 in order to compare the lenses.



(a) Illumination of the IR-LEDs for beacon A (b) Illumination of the IR-LEDs for beacon B

Figure 6.7: The light areas are where there is only one IR light. The darker areas are the places where we have overlapping signals from two IR LEDs. The area each IR led is covering is marked with an arc. (a) shows the illumination of beacon A, while (b) shows the illumination of beacon B

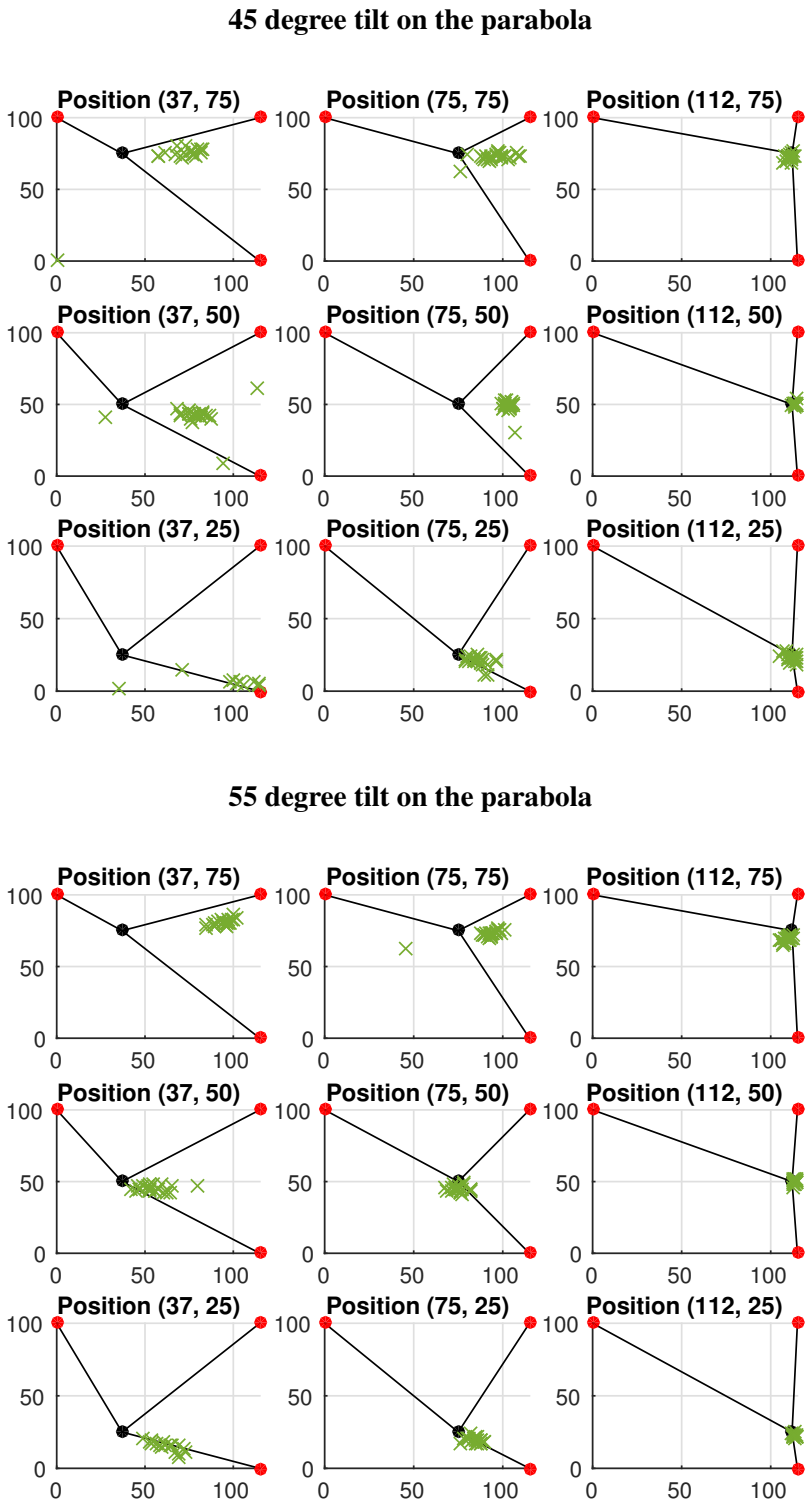


Figure 6.8: Position test of a parabola lens with a 45 degree tilt and with 55 degree tilt. Test was performed on a 1x1.5m table

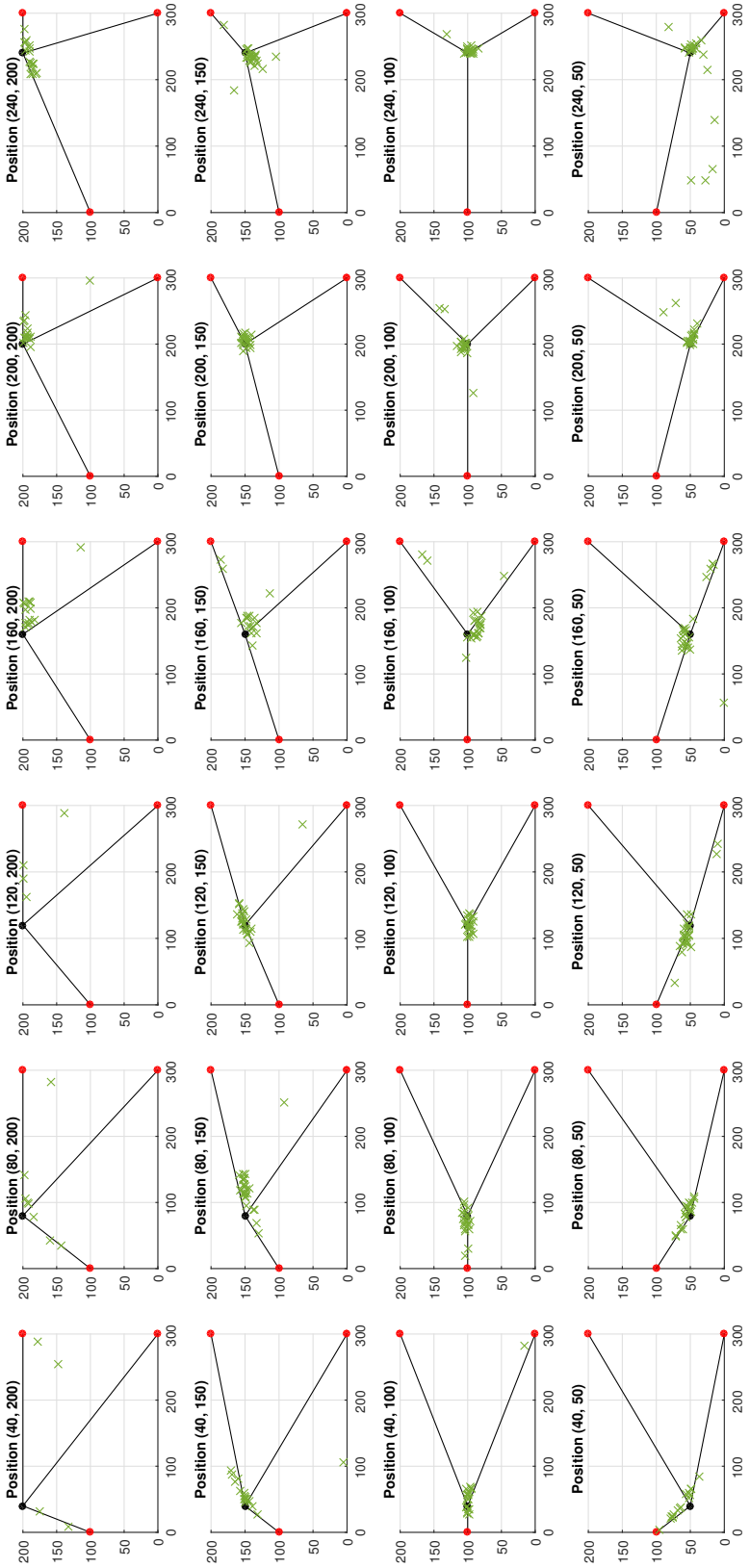


Figure 6.9: Beacon tower test with parabola lens of 55 degree over the full size playing table.

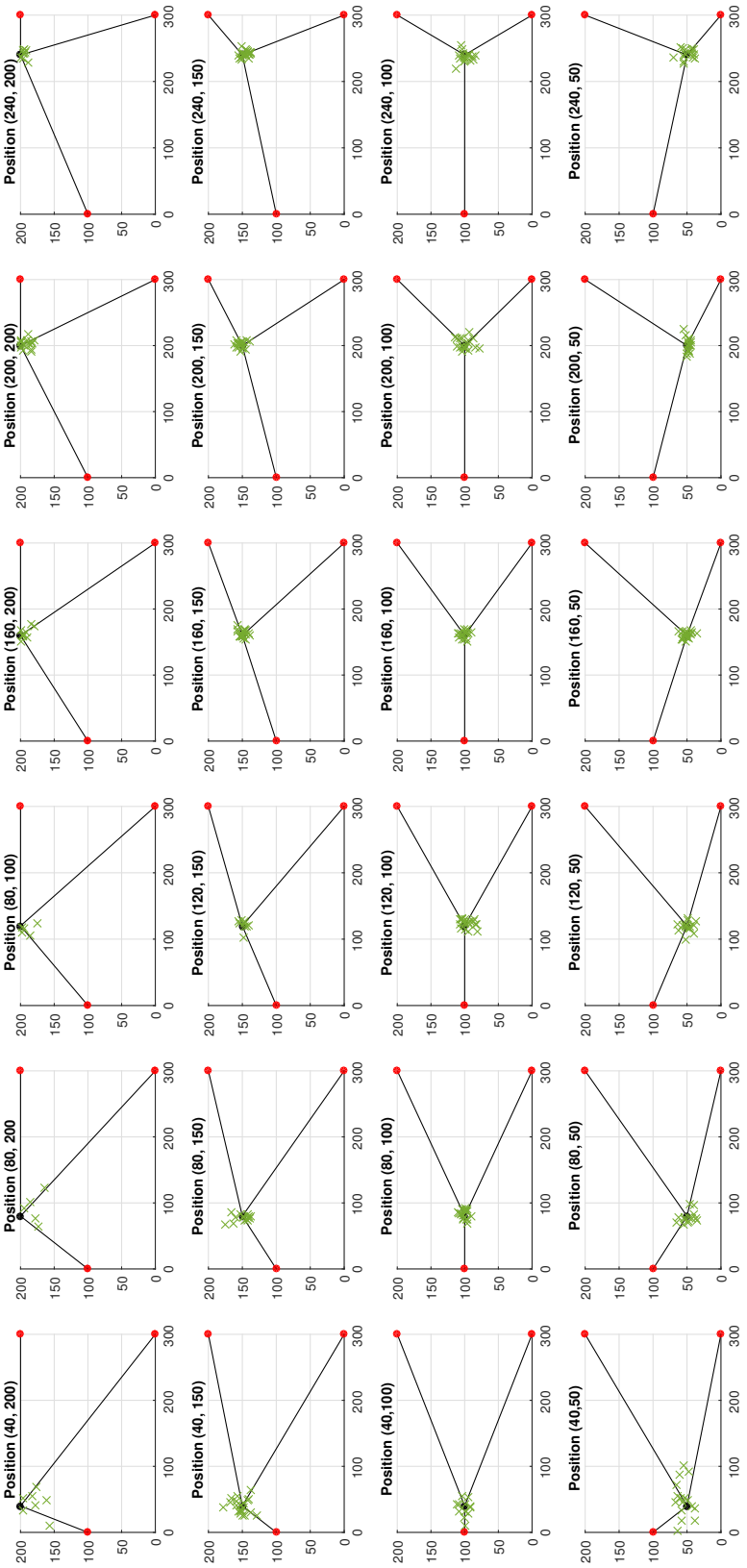


Figure 6.10: Beacon tower test with the final lens over the full size playing table.

Evaluation With the use of the parabolic lens, the calculation became a lot more stable than with the first tower, with just a hole. It also worked a lot better than the tunnel lens backing up the theory that the signals were reflected in the tunnel resulting in a wide tower view. The results shows that the 55° degree lens worked a bit better than the 45° lens, by narrowing its view. Figure 6.9 clearly shows the weak spots of the parabolic lens. Especially in the corners and along the edge where the calculation of the position is quite poor. This is because it is hard for the tower to differentiate between beacon B and beacon C in these spots, and it was not before the final lens, that the beacon system became fairly stable.

Analysis of the beacon system

A quick test in order to check whether the beacon system was Gaussian or not was made. Figure 6.6 shows that the measures follows the linear line, and we can conclude that these measures are relatively Gaussian. From all tests performed here, the beacon tower was not moving, and as we can see did the complete system work quite well. With the final lens the stability was improved further. But it has some shortcomings when placed on a moving robot, as will be revealed in section 6.2.2. We feel that it is possibilities for further improvement of the beacon system, by adjusting the width and height of the final lens further. A gear could also be used in order to achieve a higher resolution for each step of the stepper motor.

6.2.2 Positioning system

As mentioned the robot measures an angles between all three beacons and then calculates the position. With the use of the Extended kalman filter technique (EKF) described in section 2.4.3 the beacon data and odometry data is combined. The Kalman filter receives a series of measurement data and encoder data, and apply these together in order to produce the optimal position. The idea here is that the error from several sensors can give a more accurate prediction then error from just one sensor. It is worth noting that data from the beacon system is not needed. By estimation only 60 percent of the time, the beacon system gets a valid reading. When no beacon data is available only the odometry is being used. Both the measurement data and control data is assumed to be zero-mean wide-sense stationary with some variance. This is shown not to be correct as see Figure 6.4, which shows a deviation from the zero mean requirement in the beacon system test, when the tower is not moving. By using these measurements to represent the measurement noise will make us end up with a non-optimalw solution. However, the measures are not to far from the requirements, and can be used to produce a good approximation of the position.

Simulation of the Kalman filter

A couple of simulation have been made in order to test the Kalman filter, and different covariance matrices. The simulations can be seen in Figure 6.11 and 6.12. The measurements from the beacon system has been simulated with some amount of noise, which should simulate as close to how the beacon system behave in order to create a realistic simulation.

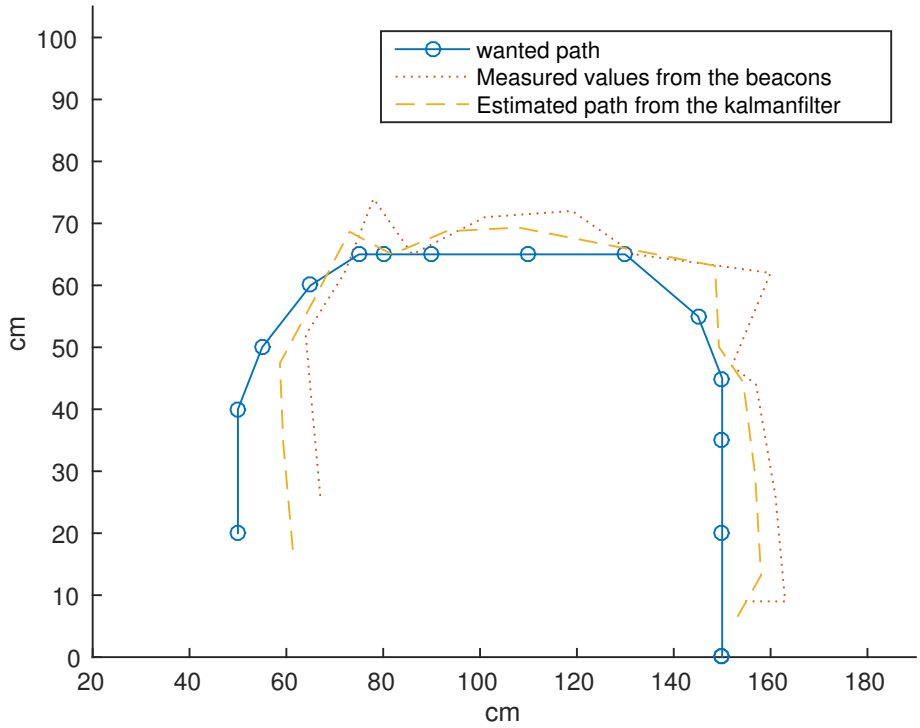


Figure 6.11: Simulation of the Kalman filter. Uncertainty matrix with low uncertainty for the beacon system has been used

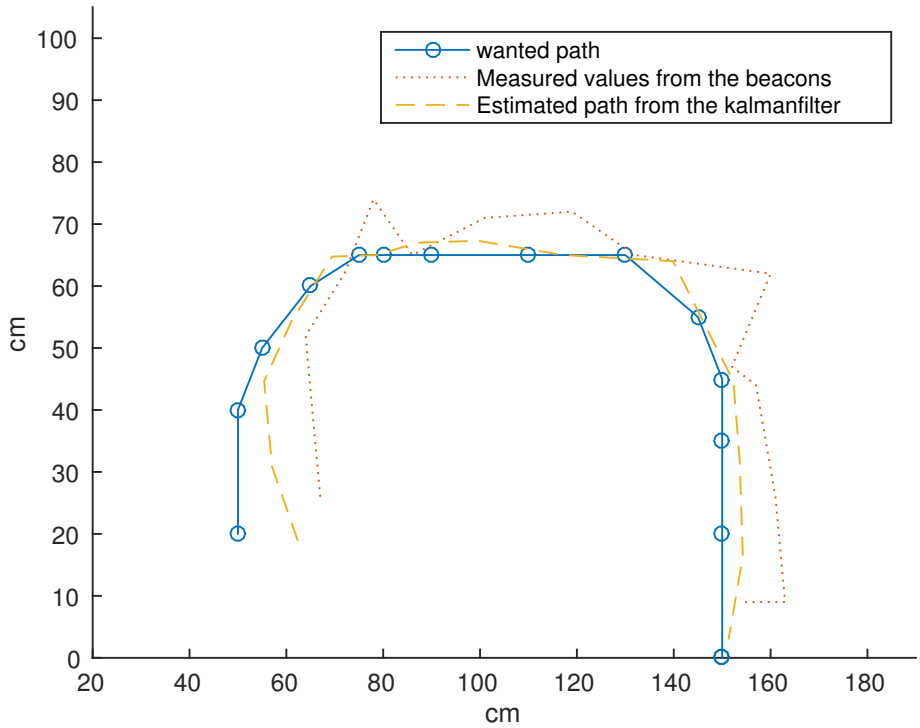


Figure 6.12: Simulation of the Kalman filter. Uncertainty matrix with higher uncertainty for the beacon system has been used

For Figure 6.11 the measure covariance matrix was a bit to small. In Figure 6.12 the measure covariance matrix is tuned as the variance of the error, resulting in a more accurate positioning.

Driving tests

In the Kalman filter, the measurement uncertainty matrix was described by looking at the measurement distribution of x and y, as seen in Figure 6.5. Resulting in a standard deviation for x as $\sigma_x = 7.312cm$ giving a variance of $\sigma_x^2 \approx 52cm$. For y, the standard deviation is $\sigma_y \approx 7.404cm$ giving a variance of $\sigma_y^2 \approx 54cm$. The standard deviation of the heading was measured to be $\sigma_\theta = 4.8^\circ$ resulting in a variance of $\sigma_\theta^2 \approx 23$.

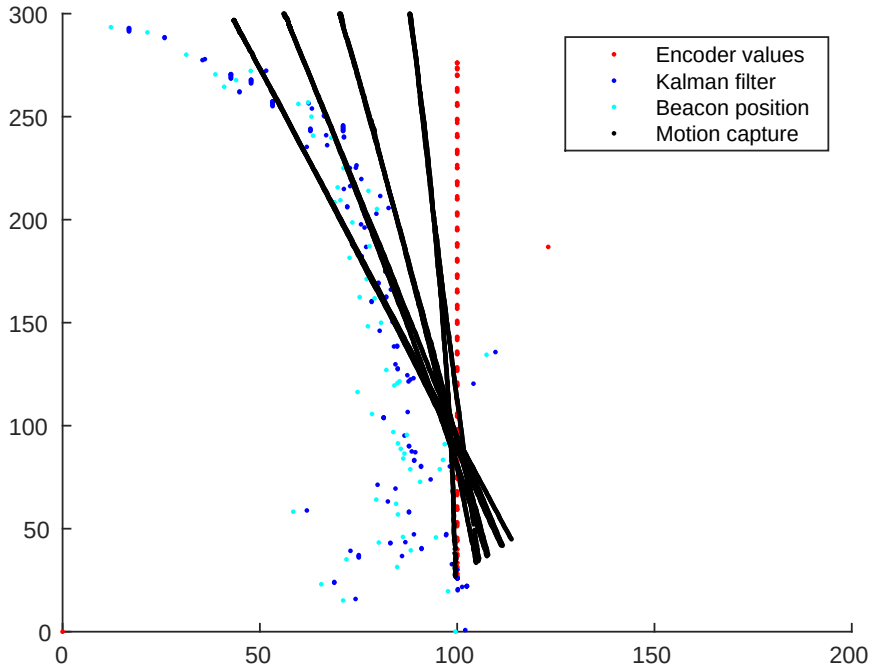


Figure 6.13: Test run with the initial wheel

One of the major problems that occurred, was due to the use of a lousy rear support wheel. On the initial frame that we used, the rear support wheel shown in Figure 4.14 was supplied. This wheel introduced errors when the robot turned due to its design that made it jerk when turning, and especially going from forward to reverse and vice versa. In Figure 6.13 the robot drives forward and backward several times. The plot clearly shows that the odometry deviates from the wanted path over time. The positive part about this wheel is that it gives a quite concise error when it goes from forward to reverse and from reverse to forward, making it possible to predict the error. The transition from reverse to forward, and forward to reverse was an edge case, and introduced a lot more error than when driving and turning normally. The same test has been done with the wheel used in the competition Figure 6.14. This wheel has a bit less error, but the accuracy of this wheel is still a bit poor. One thing to mention is that this wheel became worse over time, and the large error seen in the figure appears because the wheel locks, and

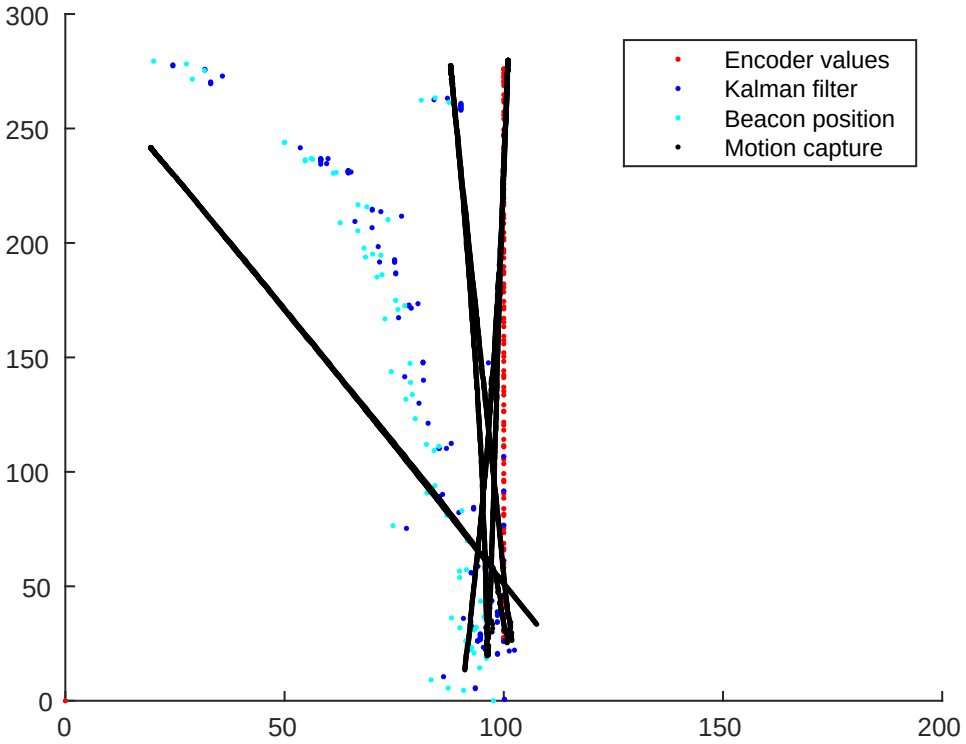


Figure 6.14: Test run with the round wheel

stop spinning. Under the competition we lubricated the wheel. This made it slide, rather than spin, resulting in an even friction throughout the run, which lead to minimal amount of errors. In both tests with the support rear wheel we see that the beacon position and the Kalman filter is more accurate over time than the encoders. However they do not provide very accurate position, which is a result of the beacon towers unstable calculations when the tower is moving.

In Figure 6.15 the robot starts driving from position (26,100) and drives up along the y-axis on the graph (which is the x-axis on the game table). Then it drives in a square, and should end up in the same spot as the start position. In this test the rare support wheel makes the robot divert from its path in the second turn, making the robot to end up in a wrong position. It is worth to notice the points from the beacon system, that are quite off in the top right of the path from the first turn. They are a result of the robot rotating, and the angle between the different beacons that the tower sees are shifted resulting in wrong calculations.

In Figure 6.16 the robot starts driving from the same start position as previous (26,100), but this time a wheelspin has occurred. This wheelspin was made by holding one of the sides of the robot in the first turn. We see in this run that the belief of the encoders position from this point and through out the run is quite off, and the robot ends up driving out off the track. Again we notice the points calculated from the beacon system that has been shifted when the robot turns, resulting in inaccuracies in the Kalman filter.

In Figure 6.17 the Root mean square error (RMSE) from the odometry, the beacon system and the Kalman filter is shown. We see that from sample 15 the wheel slip occurs, as seen in Figure 6.16 at the top right of the robot path. The

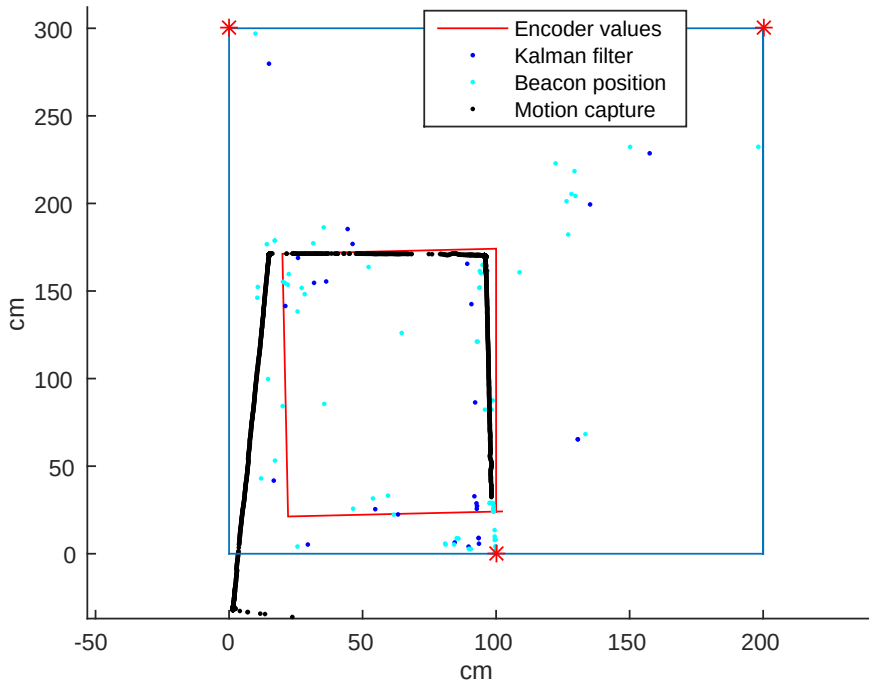


Figure 6.15: The robot does not slip, but due to the poor rare wheel some errors from the encoders occur when the robot rotates. The beacon position thats located towards the right corner is due to the rotation of the robot in the second turn. The Kalman filter is not optimal when the beacon system gets this unstable, and not uniformly random.

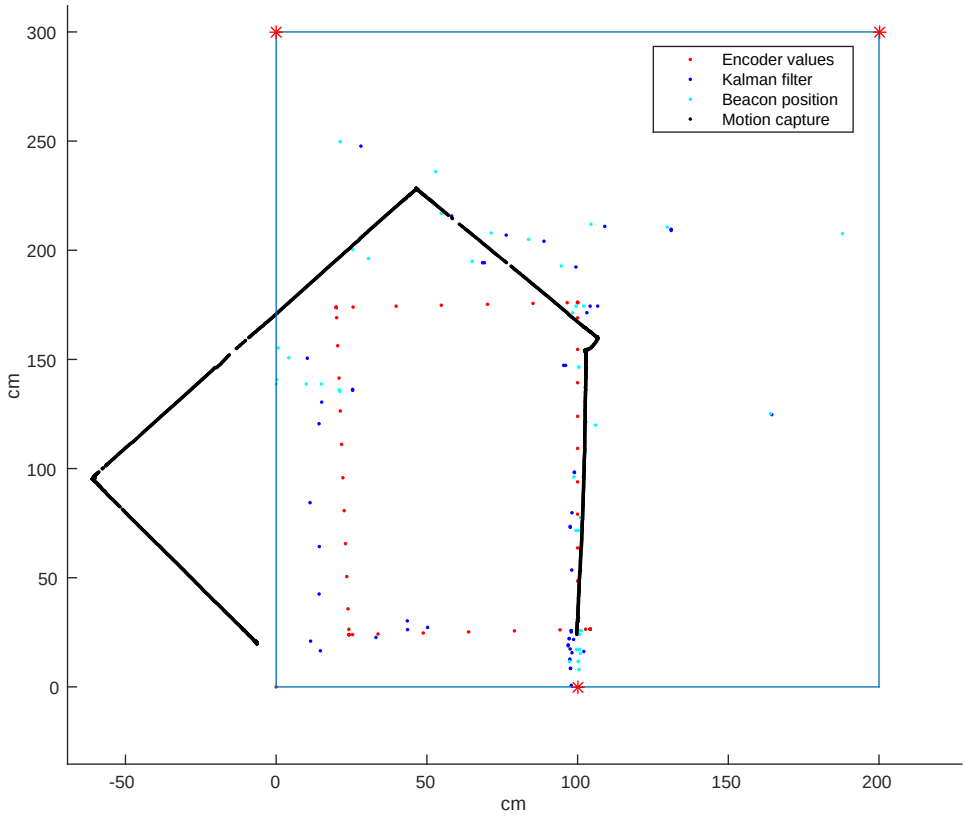


Figure 6.16: A run where the wheels have slipped, and the position gets quite off. The position thats outside of the game area is not calculated by the beacon system, and therefore the Kalman filter uses only the encoder values from that point.

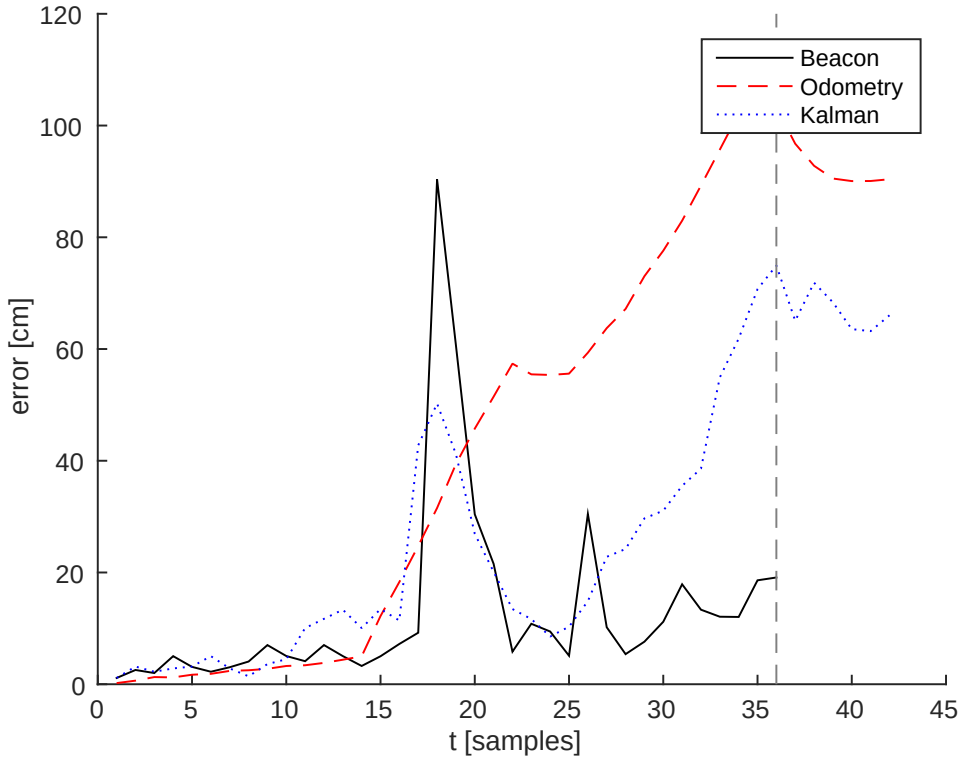


Figure 6.17: The figure show the error of both beacon system, Kalman filter and the odometry. The error from the odometry comes from the wheelspin in Figure 6.16. The peak in the beacon, is when the robot turns, this introduces error since the robot is rotating as well as the tower. Beacon points after the gray vertical line is invalide, since this points are located outside the game area, and are not logged. From this point, only the encoder value is used.

errors that occurs for the beacon system at sample 17 is due to the rotation of the robot. When the robot rotates the angle between the beacons are shifted resulting in large errors, which easily can be seen in the Figure 6.17. From sample 36 the robot has driven off the game area, and is not tracked by the beacon system.

Analysis of the position system

The uncertainty matrices applied in the following tests were not correctly tuned for the tests performed. Resulting in the Kalman filter trusting the beacons a lot more than the encoder data. This was because in the worst cases where the robot have wheel slip, the errors is getting quite large for the odometry, as well that when finding the error variance of the beacon system, the beacon tower was placed on a known position, and did not move. We saw during the tests that the movement of the robot introduced a lot of extra error. We also found that when the robot rotated the beacon system was quite off. But since the beacon system is unstable when moving and rotating, the randomness was not uniform, and the variance changed over time depending on the movement of the robot. The estimated position from the Kalman filter could be furthered improved by setting a higher uncertainty matrix for the beacon system, trusting the odometry more.

Since the beacon system is this unstable when moving, another strategy than using a Kalman filter might be better. One solution is for example to use the system in order to calibrate the position once in a while, by stopping and take the mean of the measures from the beacon system.

The rear support wheel created a lot of errors in the turns for the robot. As mentioned the rear support wheel was lubricated during the competition in order to make it slide resulting in almost no errors.

Using a gear to speed up the rotation of the tower, should help to decrease the error and could resulting in higher stability when moving.

6.2.3 Sensors

Ultrasound and infrared proximity sensors were used for the distance measurement. There characteristics are presented in the following subsection. The constraints of the sensors are clearly shown, especially for the infrared sensor where the resulting graph has a distinct characteristic due to the triangulation method used by the sensor, as described in section 3.4.2.

Ultrasound

The LV-MaxSonar-EZ3 has three possible methods to get data using different pins of the sensor. We have looked at two of the methods one using the analog pin, the other using the pulse width (PW) pin. The analog method is very simple. For this method a high frequency signal is sent out and the strength of the reflected signal is measured. The PW method is also pretty easy. This approach is based on the flight time of the signal. By sending out a signal we can measure the time before the reflected signal returned. Different material will affect the returned signal for both methods, but this will not vary to much to be a problem. The ultrasound sensor produces a relatively linear relation between distance and output/time as seen in Figure 6.18 and 6.20. Therefore a scaling factor can easily be extracted from the slope of the graph from the two methods, The scaling factor is shown in Figure 6.19 and 6.21.

Infrared

The measured output values are shown on the y-axis, and the distance is shown on the x-axis. All the measures we made for each point are averaged, and values between measured points are found by interpolation. A moving average filter with a filter size of 3 has been used to smooth the linearity of the graph that occurred as a result of the interpolation.

Analysis of the sensors

For the ultrasound sensor the analog method was shown to be more suited for our purpose. As seen in Figure 6.18 we are able to measure distances as close as 6cm. Compared to the pulse width method seen in Figure 6.20 where the closest distance possible is 19cm. By placing the sensor closer to the center of the robot, we minimized the dead zone in front of the robot. Based on these results the scaling factor for the analog method was 0,6699. For the pulse width method the scaling

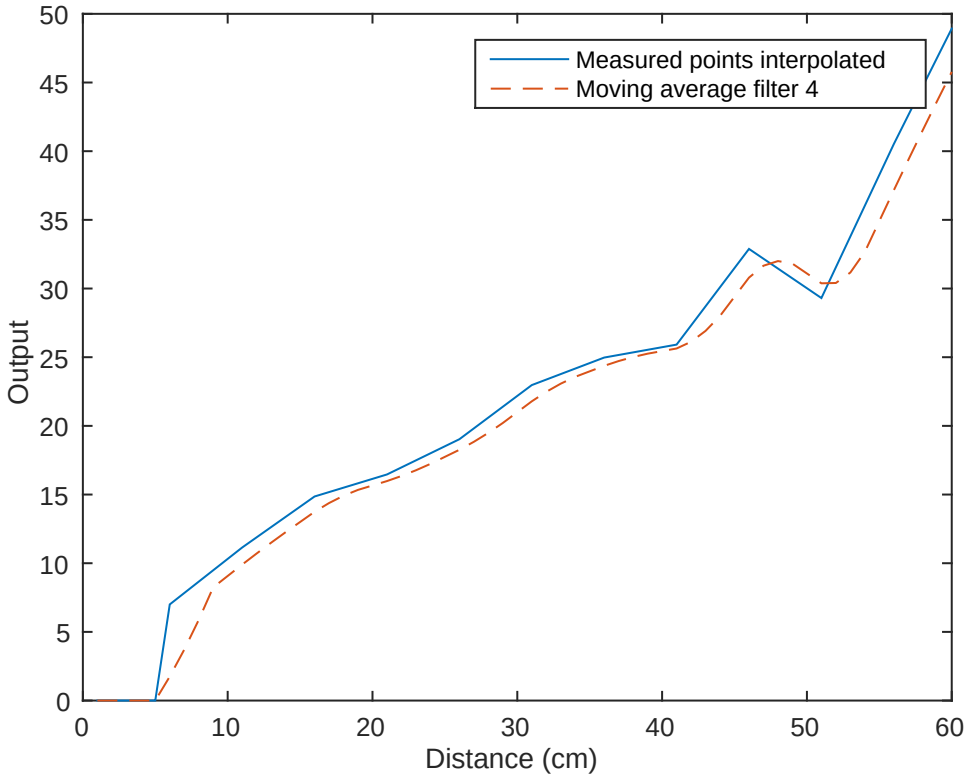


Figure 6.18: Plot over the output measures with the maxSonar sensor using the analog pin

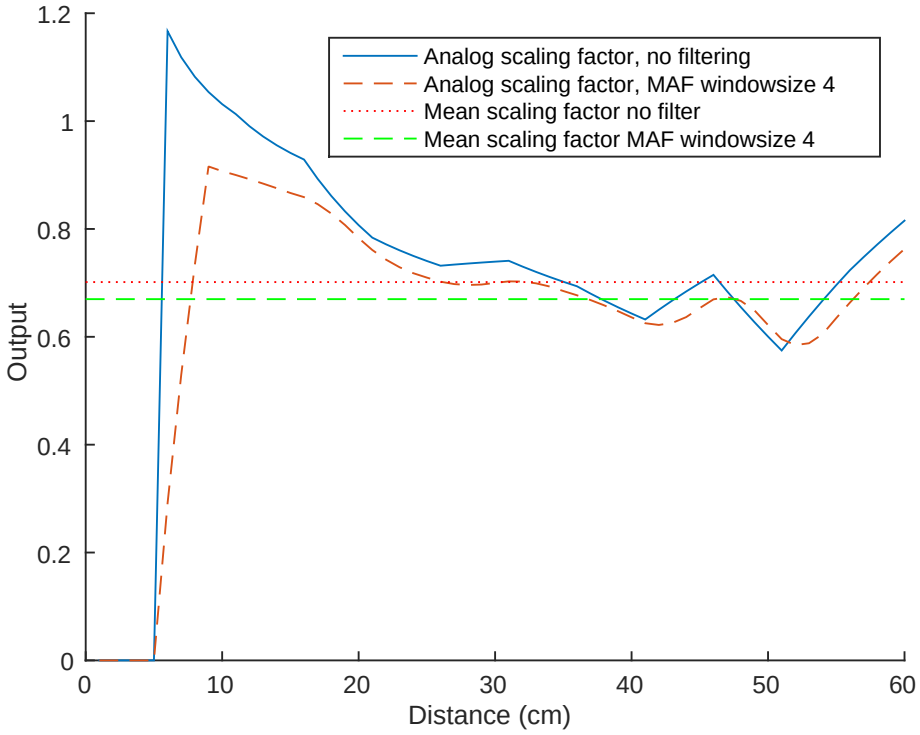


Figure 6.19: Scaling factor for the analog output when using the analog pin for the maxSonar

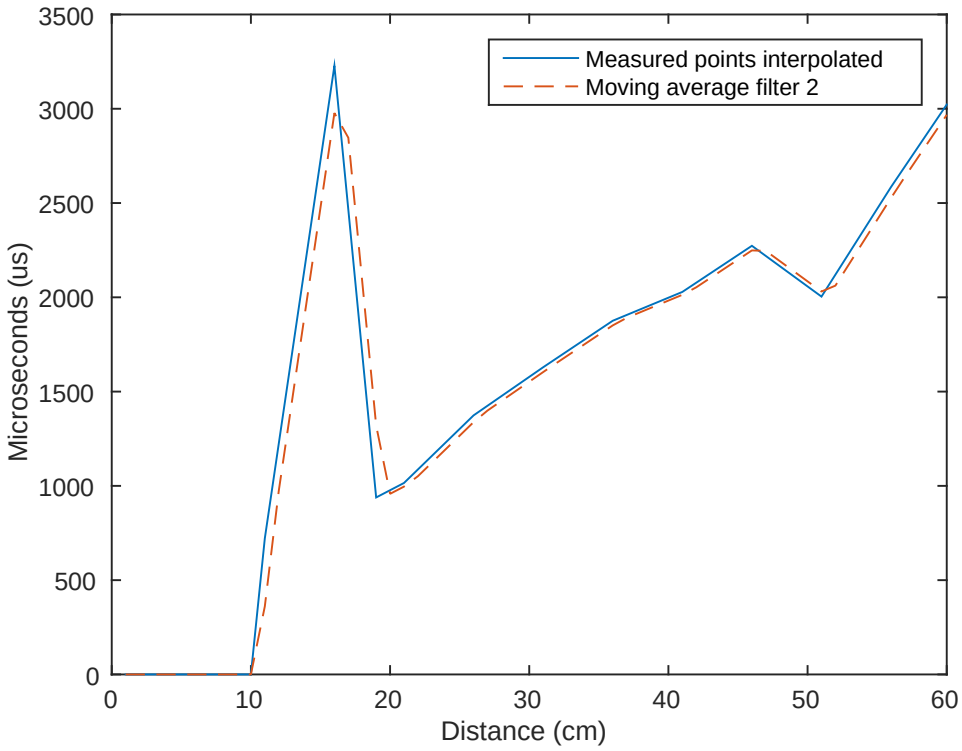


Figure 6.20: Plot over the output measures with the maxSonar sensor using the PW pin

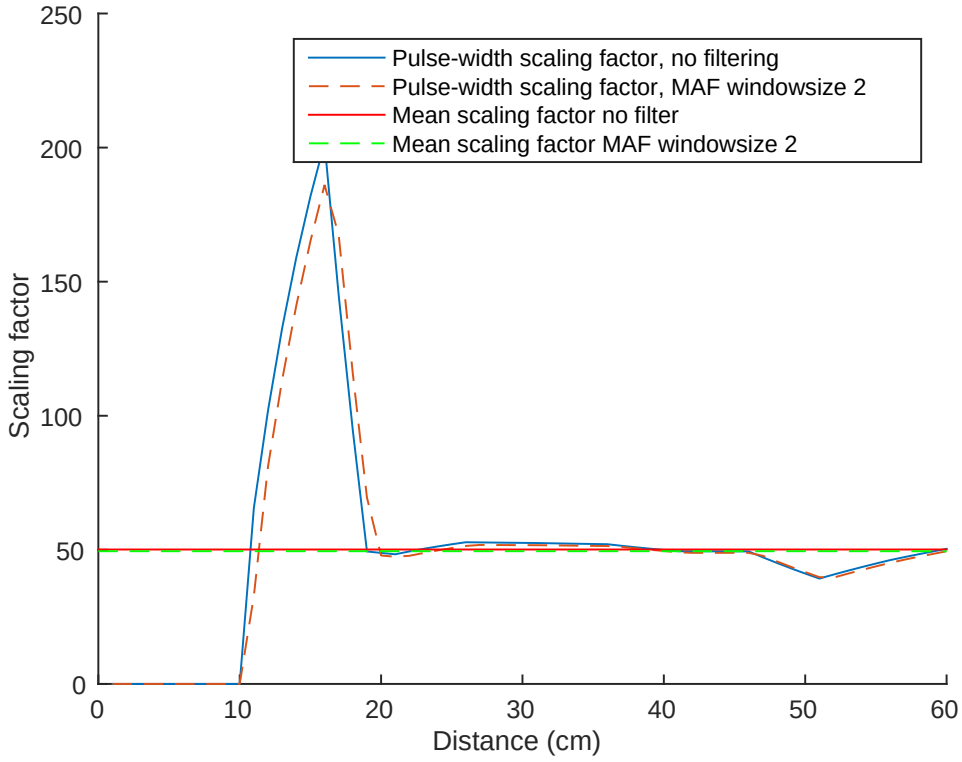


Figure 6.21: Scaling factor for the analog output using the pw pin for the maxSonar

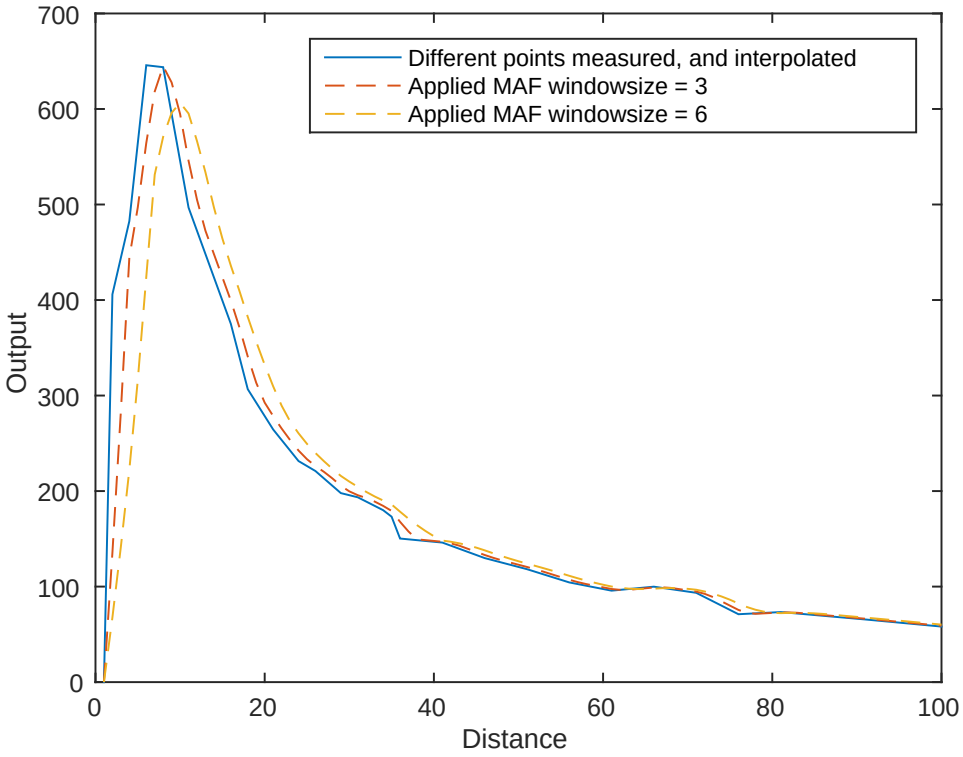


Figure 6.22: Output from the sharp IR sensor

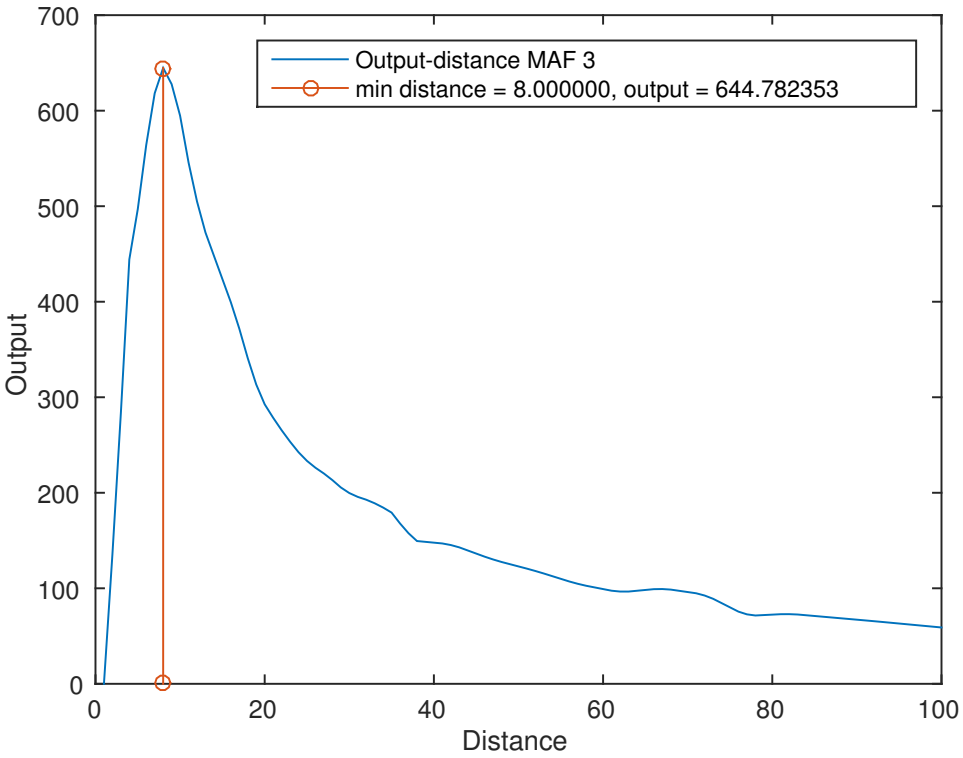


Figure 6.23: Filtered output from sharp IR sensor

factor was 49,48. The characteristics of both the ultrasound methods are relatively linear, which makes the distance calculation based on the output fairly easy. For the infrared sensor the distance is based on how much of the CCD array that is illuminated. Since the ratio of how much of the CCD array that is illuminated is not linear, we end up with a non linear output graph, as seen in Figure 6.22. We see that in Figure 6.23 distances closer than 8cm, do not have a unique output voltage, making the closest distance of the IR sensor 8cm, which is acceptable.

During the qualification for the compeititon, we had problems that the beacon tower stands, located at the edge of the game areas were detected by the sensors making the robot stop at unwanted places believing the opponent was in front of the robot. This could be fixed using som dynamical collision avoidance. During the competition this was fixed by tuning the distance sensors further.

6.2.4 Motor performance

In the competition the robot should drive as fast as possible in order to quickly reach the different tasks and objects, thus all the tests are carried out with maximum speed on the motors.

Three different accelerations have been tested. All tests shows the behavior of the motors both with and without the regulator. The solid lines in Figure 6.25, 6.26 and 6.27 shows the mean of several runs with the same acceleration. The shaded area is the 99% confidence area. We see that for all the different acceleration we have spikes right before braking. Normally the encoder values are logged each $4\mu s$, but at the spikes a stop signal is sent to the motor between two of the samples, resulting in a longer time between these two samples. The time the stop signal takes might vary, and therefore not taken into the calculation. The acceleration can be set from 1-10, where 1 is the slowest, and 10 is the fastest. The default configuration of the motors is with an acceleration of 5 with the regulator turned on. This was the configuration used during the competition, and it produced less errors than when the acceleration is set to max, while still having a usable stopping range as can be seen Table 6.4. In the tests, 3 is low acceleration, 5 is medium and 10 is high. It is not possible to compare the stopping range of the motors we used, with more expensive motors using active brakes. These types of motors are able to stop a lot faster by using a part that produce high friction and wear resistance. Then push it strongly to the wheels or axle. Often a strong magnet is used to create this friction, and lock the wheels completely.

From the tests we can see how fast the robot is able to stop, and use this information for the anti collision system. It is important to notice that the stopping distance of the robot is not the only factor when setting the stopping range of the collision avoidance system. There is also some latency from the distance is read out from the sensor, until the Artificial Intelligence request it, and then sends the stop signal to the motors if needed to stop. The other robots movement must also be taken into account. In Figure 6.24 the stopping range with different acceleration is visualized. This graph gives us a good view of using regulator compared to not using the regulator.

Evaluation From Figure 6.25, 6.26 and 6.27, we see that with the regulator the robot is not able to reach as high velocity as without the regulator. But it makes

Acceleration	Distance (mm)	Time (ms)	Regulator
10	66	208	Yes
	52	172	No
5	115	348	Yes
	160	364	No
3	197	568	Yes
	241	612	No

Table 6.4: Breaking distances with different acceleration and with and without regulator

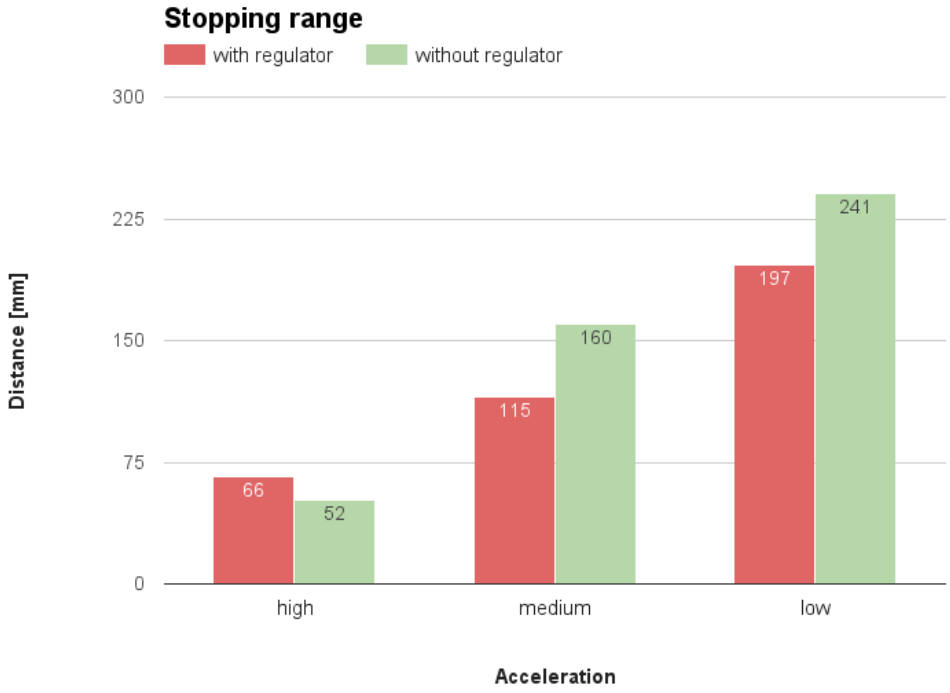


Figure 6.24: The bars shows the stopping distances with different acceleration setting on the motors. The red bars shows the distances when the regulator is on, compared to the green bars, showing when the regulator is turned off. The default value for the motors is the medium acceleration with regulator turned on

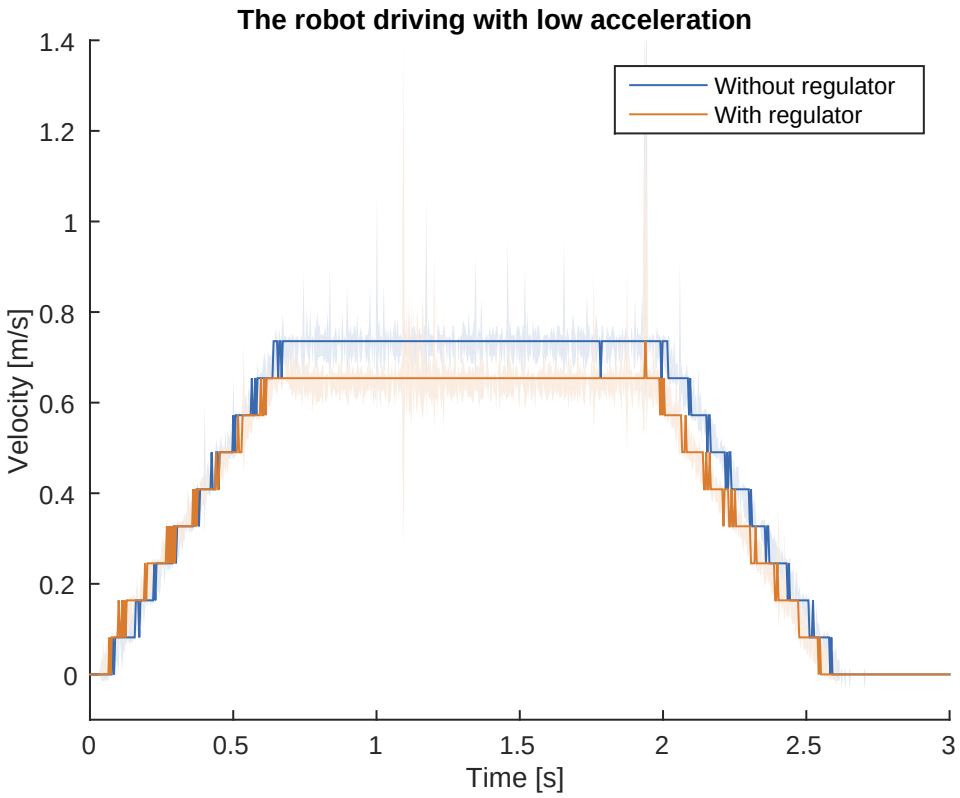


Figure 6.25: The robot driving with full speed. The acceleration is set to 3

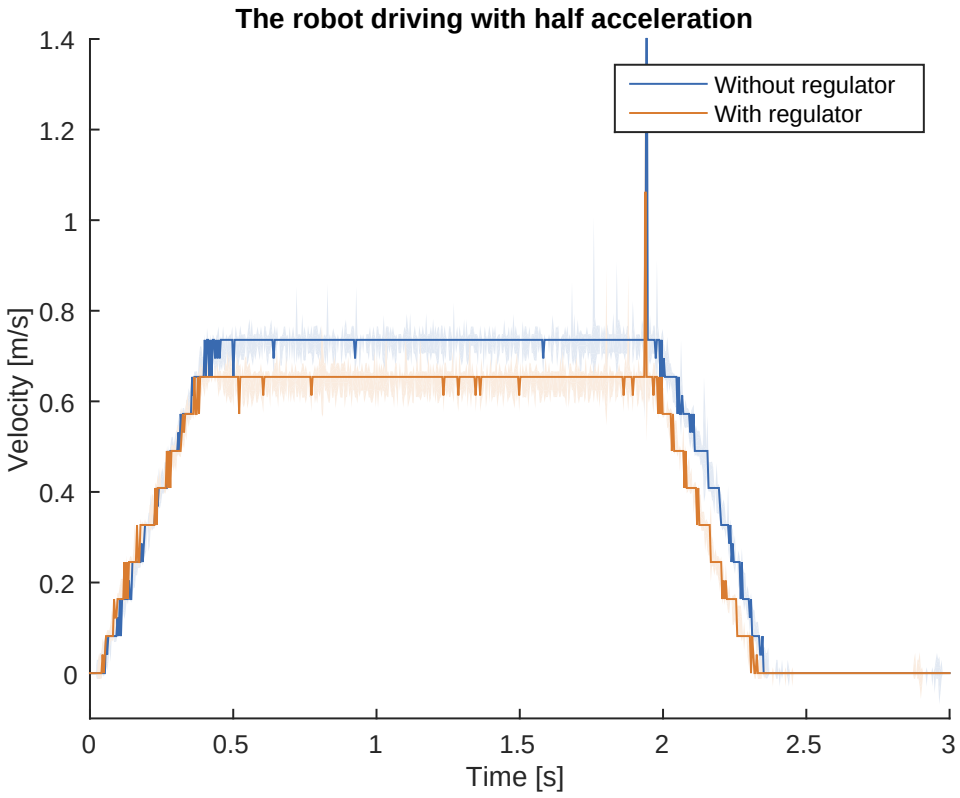


Figure 6.26: The robot driving with full speed. The acceleration is set to 5

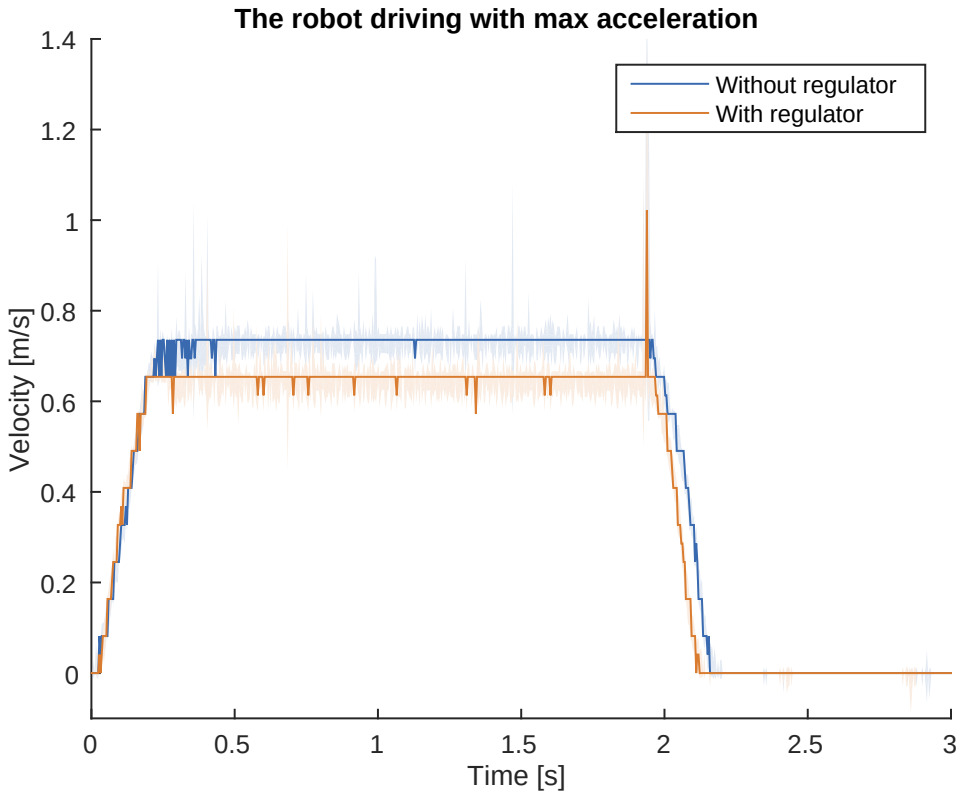


Figure 6.27: The robot driving with full speed. The acceleration is set to 10

the robot drive more smooth, resulting in less errors for the odometry data. The maximum velocity without regulator is $0.735 m/s$ while maximum velocity with the regulator is $0.654 m/s$.

It is hard to say much about how the regulator is tuned, since the motor controller only allows us to set it either on or off. We can however by looking at the results see how the different parameters might be tuned, and what is missing. The default settings for the motor controller is half acceleration with regulator turned on. We can only assume that the controller therefore is optimized with these values. The two positive things with the regulator turned on, is that it moves faster to the wanted velocity in the beginning. This can be a result of a good calibrated proportional term increasing the gain. As it gets closer to the wanted velocity it starts to decrease the acceleration, in order to avoid overshoot, and to achieve a smoother movement. This is probably due to a derivation part. So far the regulator has given the wanted behavior, but if we now compare it with the graph without the regulator, we see that it do not reach the max velocity for either of the tested acceleration. This might be to a missing integral term adding or remove gain if we are not at our goal position.

Wheels

High requirements to the wheels were important, in order for an accurate position estimate using the encoders. Therefore different wheels were tested. The first pair of wheels we had available were not usable due to unequal size of their diameter.

They were made out of rubber, and there form was not perfectly round, as can be seen in Figure 6.29a. These wheels were supplied with the robot frame. We also had access to a set of precision wheels. They were on the other hand to small for the robot and the friction test was not possible to do on these wheels. A set of plastic wheels, seen in Figure 6.29b were supplied together with the Devantech motors and were used a while. But the wheels grip was a bit poor. Therefore a set of silicone wheels, seen in Figure 6.29c were made. The different properties of the wheels can be seen in Table 6.5. Figure 6.28 describes the different properties.

Wheels:	*Silicone*	*Plastic*	*Precision*	*Rubber*
Connectivity area [mm]	17	11	16	16
Width [mm]	16	29	5.8	23
Diameter [mm]	101.6	124.8	76.2	135
Static friction [N]	44.4	20.44	-	-

Table 6.5: Dimensions of the wheels, and the surface connectivity area, the measured grip for the silicone wheel and plastic wheel was done. We see that the silicone wheels grip was superior to the grip of the plastic wheel.

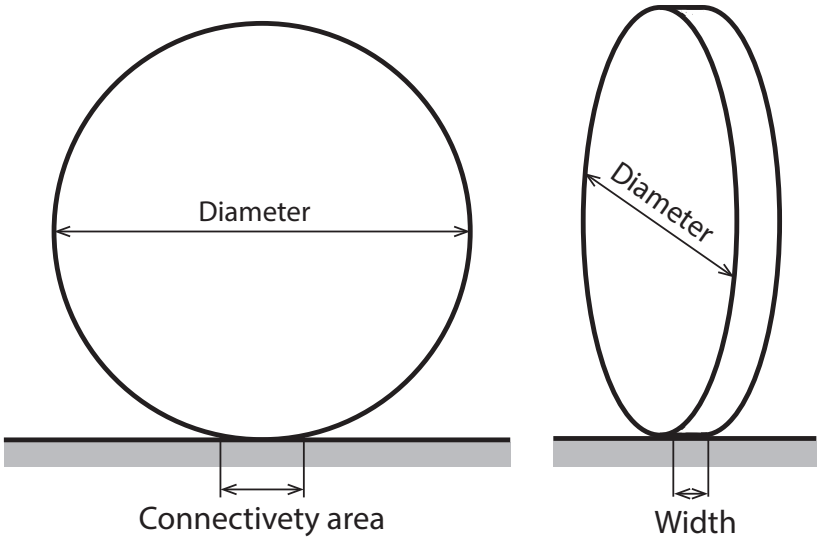


Figure 6.28: Describes the wheel properties that have been measured

Evaluation The wheels casted in silicone was chosen due to its good grip and size. Since these wheels were designed by us, the size was made to fit perfectly on the robot. The wheels are a bit softer than the plastic wheels resulting in a larger connectivity area for the wheels. This again resulted in a much better grip and higher static friction than the other wheels, as can be seen in Table 6.5. One problem that occurred using the 3D printed wheel, was that it loosened from the motor shaft since the mount is 3D printed plastic. This was solved by using super glue. in order to glue it to the motor shaft.

(a) First wheel (b) Plastic wheel (c) Silicone wheel

Figure 6.29: (a) was the first wheel used, we clearly see that the wheel is far from being perfectly round, and would therefore produce quite wrong calculations for the odometry. (b) was used a while, but the grip of the wheel was not as good as wanted, (c) is the silicone wheel used on the final robot

Analysis of the motor performance

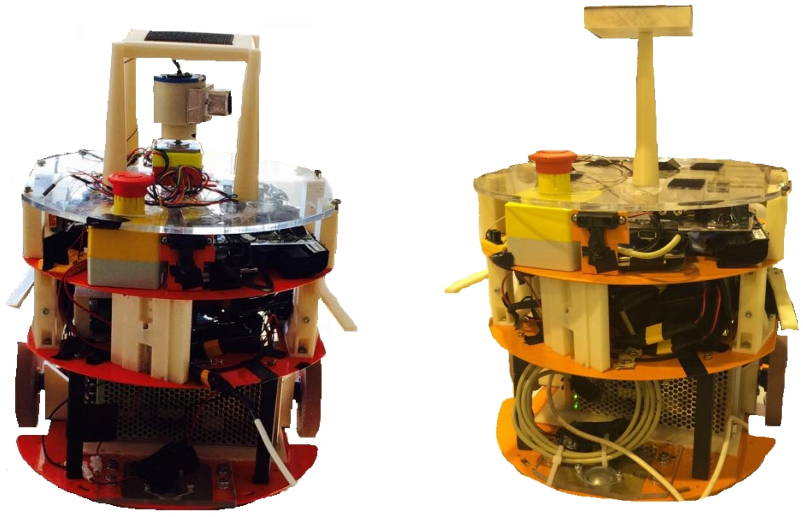
As well as being a bit slower than many other teams motors, the motor had a bit of backlash. We recommend for coming year participation to look into other motors. Maxon EC-45 were one of the candidates we looked into, but they were discarded due to the cost of these motors. The Maxon motors also need a motor controller, and an encoder. Maxon delivers all of these components, but the price of the total system is a bit higher than we could afford. However are these motors quite accurate and many of the teams in the competition used these motors. The silicone wheels worked really good with good grip and high accuracy, and we recommend to use these, or cast a set of new wheels in silicone for coming years.

6.2.5 Competition

For the robot to be approved for the competition, it needed to pass several tests and requirements to size, laser, height and opponent beacon stand. In the approval the collision avoidance system was tested and needed to work before going through to the qualification round.

Due to misunderstandings in the rules, the opponent stand was designed a bit to wide. This resulted in not be able to use the beacon tower and positioning system, because the beacon stand had to be shortened inn, and placed where the beacon tower was supposed to stand. Figure 6.30 shows this modification.

If a beacon system is to be used in future projects the opponent stand needs to be redesigned. However as we saw in the competition no other team used a global



(a) The original opponent beacon stand (b) The modified opponent beacon stand

Figure 6.30: (a) shows the original opponent stand, while (b) shows the modified opponent stand, in order to get the robot approved

positioning system. They used more accurate encoders. One of the most advanced robots in the competition used expensive laser ranging system to keep track of the opponent, and choose the path so that minimal contact with the opponent were made. The robot can be seen in Figure 6.33.

6.2.6 Matches

After the robot had been qualified, matches were set up 1 team vs 1 team. As mentioned in section 6.2.2 the rear support wheel started locking and stop rolling under the game. After the wheel was lubricated the accuracy was quite good, even with out the positioning system. The robot was accurate enough to fulfill the tasks with out missing any objects. One of the main problem was the speed, and the lack of a second robot to climb the stairs, which would have resulted in a lot of extra points. The total result was 2 wins, and 2 losses. We also had to forfeit 2 matches due to an early flight on Sunday, and we ended up with a total of 100 points.

1. Match The first match was rather hectic. The matches started at 9 am and we were playing in the second match. We got this information 5 minutes before the competition started. We then needed to log in to the robot computer, starting the start script before rushing up to the game area. In this first round, we met the Canadian team with their robot Druinobot. We lost this match, scoring 25 points, while the opponent team managed to score 33 points in total.

2. Match In the second match we met the robot ESEO - ANGERS from France. They accomplished a really high score of 97 seven points against our 25 points, resulting in another loss for the UiO team.

3. Match In the third match we met the robot Natural Born Killers from United Kingdom. This was the first victory for the UiO team, where our robot again scored 25 points to the 11 points the opponent robot.

4. Match The fourth match also led to a victory. In this match we met the Tunisian team and their robot OUCHI-TIME. Again our robot managed 25 points, while the opponent robot scored 19 points.

Evaluation In the competition the robot performed better than expected despite some problems that emerged. The robot achieved a stable amount of points each round. Almost every team made use of a second robot. This was often designed really simple and its only purpose was to climb the stairs and put down the carpet. By doing this task it was possible to achieve a lot of points. We had one robot we were going to use for this task. However, since it were required to have a stop button, opponent beacon stand, start mechanism and a functionally collision avoidance system we chose to focus on the main robot. We saw that in the competition the requirement for the smaller robot were not as hard as for the main robot. With this in mind the little time to make the second robot work would probably have resulted in a lot more points. In Figure 6.31 the robot we had planned to use for climbing the stairs is shown. Tests with the robot shows that it easily could climb the stairs due to its long belts.

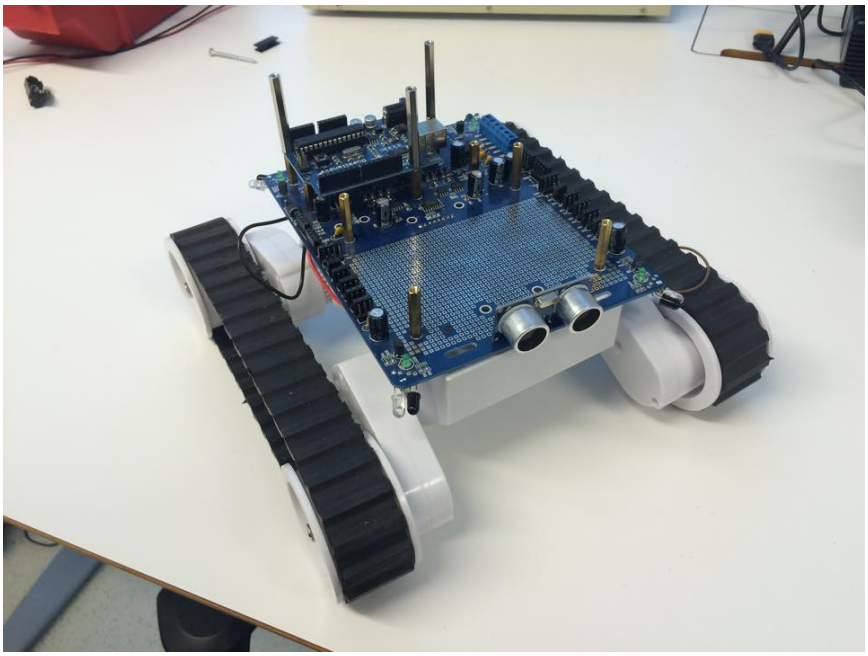


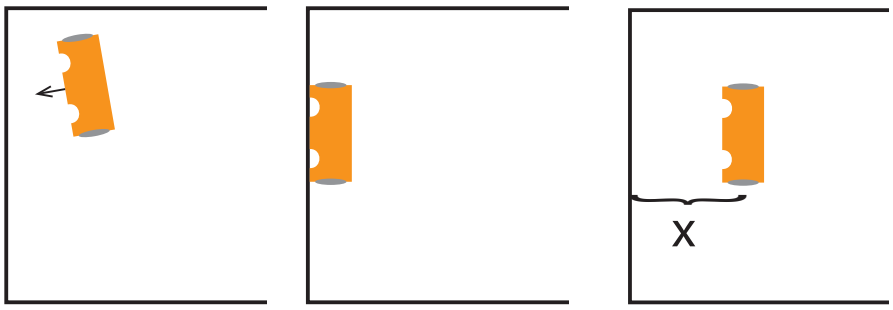
Figure 6.31: The second robot, intended to use in the competition in order to climb the stairs

Analysis of the competition

The competition showed us that no other team used a global positioning system. Most of the team participating have participated in the competition several years,

and are mostly teams of 10-15 persons. The focus for many of the teams were to have good motors and precise encoders. By keeping track of the opponent, collision could be avoided, and minimal of wheel slip would occur.

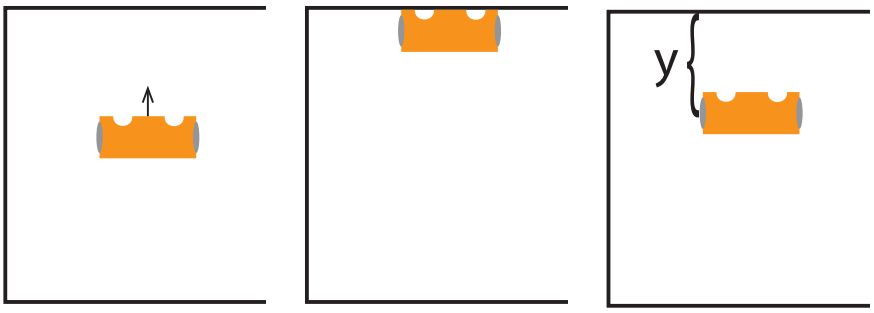
Even with expensive and advanced sensors, collision might occur as can be seen in Figure 6.34 which shows a crash between some of the most advanced robots in the competition. Since high accuracy (less than 1 cm) beacon system is hard to build with in a strict budget, more accurate motors and encoders is a better way to ensure accurate driving and positioning. By using a square robot, different calibration techniques could be used. For instance when picking up objects near a wall, the robot could drive into it to make it parallel with the wall. Now when driving, the distance from the wall is quite accurate. By turning 90 degrees, and driving into the wall in the other direction, the robot is than parallel to this wall as well, and the accurate position of the robot is known. Figure 6.32 shows how a calibration could be performed using a square robot. This idea came from one of the other teams, using this technique in order to get the correct position in the start area.



(a) The robot has lost its heading and position

(b) Drives into the first wall in order to correct the heading

(c) When driving away from the wall, the x-position is accurate



(d) Rotates 90 degree, towards the other wall

(e) Drives until it hits the wall

(f) When driving away from the wall, the y-position is also quite accurate

Figure 6.32: A possible way to calibrate/reset the position and heading during a match

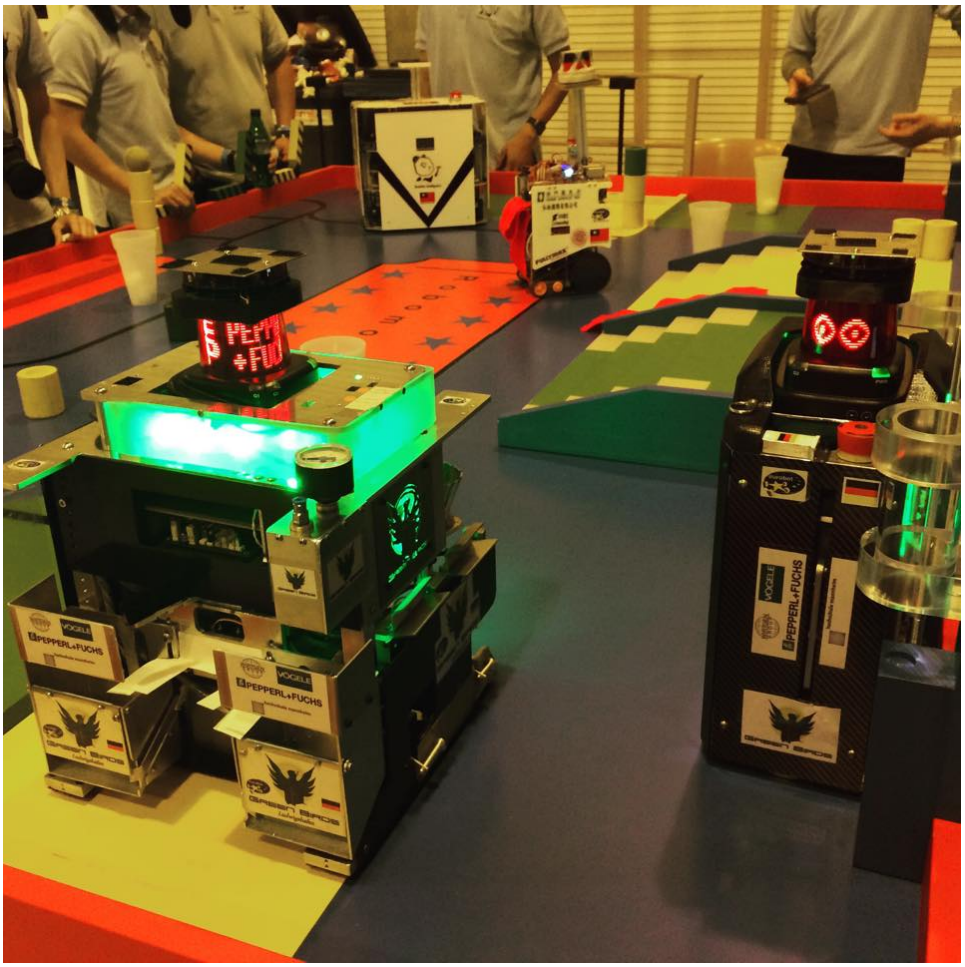


Figure 6.33: The German team's (Green bird) robots. Using lidar on the top of the robot to keep track of the closest robot at all time.

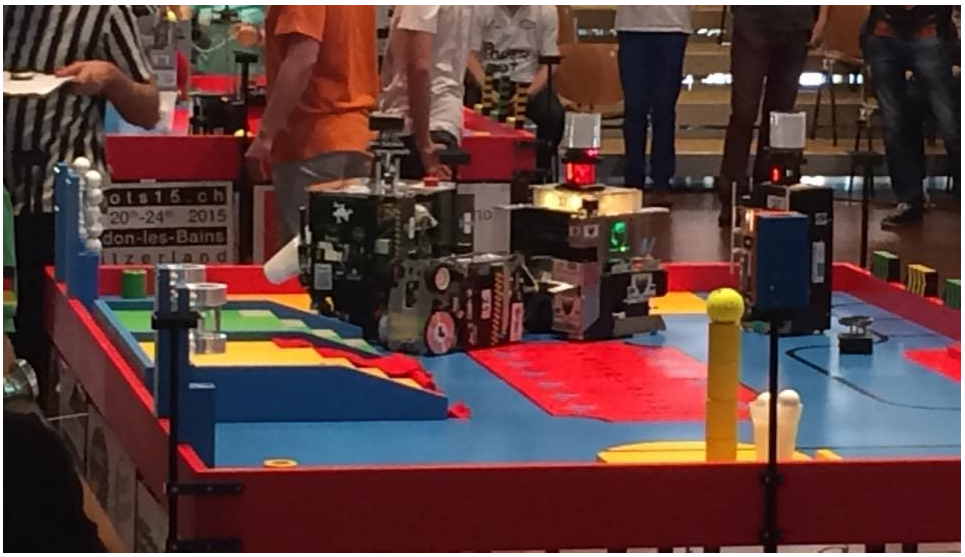


Figure 6.34: One of the two leading robots in a crash under one of the matches

Chapter 7

Discussion

7.1 General Discussion

The complete robot system performed well in the competition it was also one of the cheapest robots in the competition. The position, based only on the odometry, which was used during the competition, worked well as long as the rear support wheel slid and did not jerk. This was solved by lubricating the wheel. The beacon system worked good when not moving, and the improvement from parabolic lens to the final lens with the protruding tunnel made it a lot more stable. However, this low cost system did not work adequately when moving, and especially when the robot rotated. When the robot was moving the noise of the system was not random, making it hard to set a uncertainty matrix describing the measurement noise in the Kalman filter. A dynamic uncertainty matrix could have been made based on the movement of the robot, in order to make the filtered output more stable.

The communication using ZMQ made the integration of each systems fast and simple. It has worked flawless, and no problems have encountered using ZMQ.

7.2 Conclusion

In this project we have developed an autonomous vehicle that participated in Eurobot 2015, winning 2, loosing 2 and forfeiting 2 matches. The robot is easily re-configurable and re-programmable for future work on the robot.

The mechanical design of the robot is quite simple. Where the arms designed to flip the clapperboards worked perfectly while on the other hand a fast lift would have resulted in a much higher score, but the the servo-mechanism of the lift was to slow resulting in to little time to stack the stand objects making collector arms more suited.

A personal computer (PC) was used in order to allow for more freedom when it come to implementation choices. With the use of a PC and the ZMQ message library all team members could chose the most suited programming language for their system.

In order to avoid collision the robot is equiped with proximity sensors both in front and in the back. Based on the knowledge from these sets of sensors the robot can avoid collisions during the competition. However, the system did not provide information of where the opponent robot was located. A system such as this would

have been valuable in order to take smart path choices, creating more advanced game tactics by avoiding the opponent robot before it was in the immediate vicinity.

A low cost beacon system, based on infrared beacons and a rotating beacon tower has been designed and built. It uses a triangulation algorithm to calculate its position. The beacon tower seemed to work well for all the tests, but problems emerged when putting it on top of the moving robot, which introduced errors to the system. In order to use the system for accurate positioning for mobile robots it still needs a bit tuning and testing. But as seen from tests, it is capable of accurate positioning when not moving. Making it useful for application like correcting the position of the robot when still.

An extended Kalman filter has been implemented in order to combine motor encoder data with the measurement data from the beacon system. The filter worked as supposed, but due to large error from the beacon system when turning and moving, the error were far from the wide-sense stationary requirements of the Kalman filter, since the probability distribution changed a lot over time. This gave a very unstable state prediction from the filter. Driving only on the encoders worked sufficiently in such a small area.

We can conclude that a low cost robot can be developed possessing the same features as a more expensive robot. However, the performance of such a system is worse than a robot with more advanced equipment and systems. Especially with the use of better motors and a suited rear support wheel with low rolling resistance. The main wheel of the robot, made out of silicone exceeded all expectations, with the extremely good grip as well as being really accurate. A low cost positioning system is possible to develop, but the stability of such a system can not be compared to more advanced and expensive positioning systems. Therefore positioning with accurate encoders and clever calibration during the matches are recommended methods in order to keep the cost down.

7.3 Future work

Since the objectives of the competition changes from each year, is it likely that the mechanics and tools of the robot needs to be changed. Some shortcomings have emerged during the process, that have not been fixed due to short time.

7.3.1 Collision avoidance

The collision avoidance was rather simple this year, and the action the robot took was hard programmed. However is a possibility to use dynamic-avoidance and base the action on what sensor detected the opponent. With the use of neural network, the robot could be trained to take good choices based on the sensory data. In [39] a dynamic path planing based on the observations from sensor is described. Four actions could be used for the collision avoidance.

- **Halt:** A fully stop with no translational movement.
- **Turn left:** Rotates to the left, with no translation movement
- **Turn right:** Rotates to the right, with no translation movement

- **Ignore:** No action is performed based on the sensory data

7.3.2 Proposed positioning system

Experience from the competition showed what parts were important to spend time developing in order to achieve a high score. As mentioned previously a global positioning system is hard to implement accurate enough in order to be useful. This was mainly why no other teams had developed such a system. The encoders are really accurate, and since the game area only is 2x3 meters, they will not drift much. *However, this may result in a to simple robot which would be less suitable for a masters thesis*. Below is a proposed positioning system based on two parallel lasers described. An illustration of this is shown in fig. 7.1. This could be worth taking a better look into if a global positioning system is to be used in coming years.

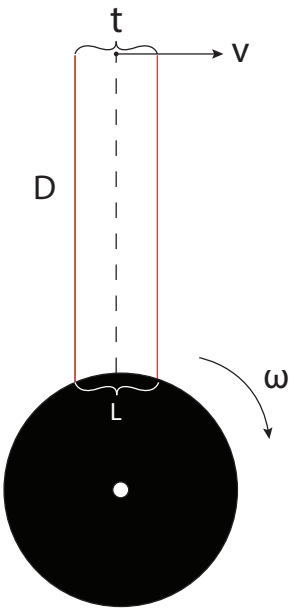


Figure 7.1: Proposed beacon tower

The proposed method is based on trilateration, and the distance is found by using two parallel lasers and a tower with constant angular velocity. The beacon is composed of a laser detector that detects the laser beam from the tower, a timer that takes the time between the two parallel laser beams, as shown in fig. 7.2, and a radio sender that transmits the times back to the beacon tower.

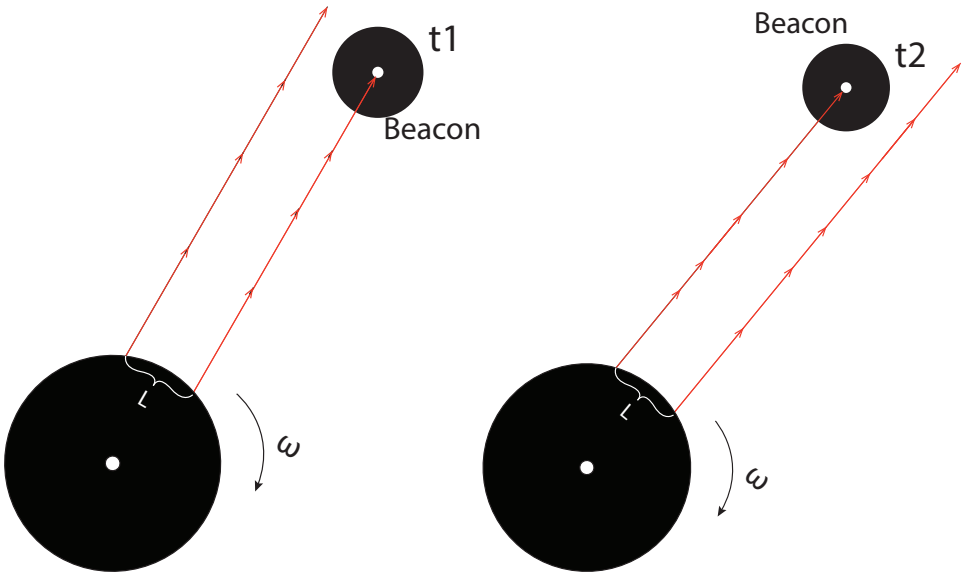


Figure 7.2: Caption of the two laser beams. The time between the laser is t2-t1

$$D = \frac{v}{\omega} \tag{7.1}$$

$$\Delta t = t2 - t1 \tag{7.2}$$

$$v = \frac{L}{\Delta t} \tag{7.3}$$

The advantage of using laser, is that it has a strong light, with a very narrow wave spectrum, making it quite reliable. However this requires the beacons to be placed in the exact same height as the beacon tower in order for the laser to hit it.

This system is not yet tested, but in theory it should give a much higher precision, than with the IR positioning system. Especially since the tower may rotate faster than the IR-beacon tower was able to.

7.4 Recommendation for future participants

Recommendation to future participants is to implement a opponent detection system in order to keep track of the robot. With such a system the robot may take smart path choices in order to avoid the opponent robot

7.4.1 Proposed opponent detection system

To know the distance and angle to the opponent was shown to be an effective method to avoid collisions, and to optimize the efficiency of the route the robot takes. One of the german teams used a laser ranging system to keep track of the robot closest to itself. The system can be seen on top of the robot in fig. 6.33. The laser system that was used on that robot had a cost of 5000€ Euros for 1 module, but the basic of this system can be built with less expensive sensors.

A possible solution for collision detection, and to keep track of the robot for coming years is the Infrared Control Freak (IRCF360) sensor. This sensor contains 8 IR emitting LEDs. By putting the sensor high on the robot, as the robots ”beacon tower” will it give 360° degree coverage in order to keep distance and angle to objects in the same height as the sensor. IRCF360 support ambient light sensing. This can be used for keeping track of the opponent robot by placing a beacon on top of the robot which is emitting trackable light. There are no moving parts which helps keeping the design quite simple and makes the power consumption of the device quite low. The price of the IRCF360 sensor is not yet been revealed since its not yet for sale, but it will be in the lower scale for this type of sensors. The sensor uses RS232 asynchronous serial communication for easily communication with most microcontrollers.

7.4.2 Motor

The possibility to acquire better motors and motor controller should be considered. Using motor controller with a controller where it is possible to change the different gains is a plus. This allow for changing the behavior and response of the motors after needs. And the values can be tuned for optimal speed and accuracy. By using the Maxon EC-45 motors a lot of space can be freed since these motors are a lot smaller than the Devantech RB-Dev-38 that was used on the robot. The Maxon motors does not have backlash such as the Devantech motors, and there performance are a lot better than the Devantech. The no load speed for the Maxon motors is 6710rpm, while for Devantech: 143rpm.

7.4.3 Design

A complete redesign of the robot would be a smart investment of the time. The laser cutter could be used in order to make the robot design fast. A square robot is the most common design in the competition. The reason is because it is easier to drive close to the edges and corners of the game table with a square robot. By using a robot that is square, the calibration method mentioned earlier, and seen in Figure 6.32 can be used in order to ”update” the position occasionally.

7.4.4 Sponsors

It is hard to know exactly how much the different teams spend on their robots, but based on the equipment many of the teams used, it is quite a lot. We know that NTNU have had Kongsberg Gruppen ASA as sponsor several years. There total spending varies, but we have seen that their budget is on approximately 100 000,- NOK each year. In [37] they have revealed numbers in their budget. This year they got 65 000,- from Kongsberg Gruppen ASA, 15 000,- NOK from previous year, and 15 000,- NOK from the faculty. The German team (TURAG) have a total of 18 sponsors which are listed on TURAGs webpage[1].

This year we did not have any sponsors. In order to make a research of what components should be prioritized to use money on, the project this year has been on a tight budget, using cheap components, and using what tools were available on

[1] https://www.turag.de/partner/

the lab. In order to be able to acquire more advanced and expensive components, a sponsor is important. In the master we have spent a lot of our own money. The travel to Yverdon-Les-Bains and the stay during the competition is something we have had to pay our self, as well as some of the equipment that has been used. However most of the sensors used in the project have been supplied by the research group Robotics and Intelligent Systems (ROBIN) at the Department of informatics. An application for a sponsorship should be made as early as possible to be able to acquire the components needed in a early stage.

Bibliography

[1] Acroname. *Sharp Infrared Ranger Comparison*. 2008. URL: http://acroname.com/articles/sharp-infrared-ranger-comparison (visited on 03/19/2015).

[2] Sung-Hoon Ahn et al. 'Anisotropic material properties of fused deposition modeling ABS'. In: *Rapid Prototyping Journal* 8.4 (2002), pp. 248–257.

[3] David Anisi et al. 'Robot automation in oil and gas facilities: Indoor and onsite demonstrations'. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 4729–4734.

[4] Arduino. *Arduino Arduino home page*. 2015. URL: https://www.arduino.cc/en/Main/FAQ (visited on 20/07/2015).

[5] Paramvir Bahl and Venkata N Padmanabhan. 'RADAR: An in-building RF-based user location and tracking system'. In: *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 2. Ieee. 2000, pp. 775–784.

[6] Jan Kenneth Bakkeng. 'Time and synchronization, GPS and INS'. Lecture in FYS3240 at the Department of Physics, at University of Oslo. 2013.

[7] Beaconsandwich. *Beaconsandwich What is iBeacon*. 2014. URL: http://www.beaconsandwich.com/what-is-ibeacon.html (visited on 26/05/2015).

[8] Anja Bekkelien, Michel Deriaz and Stéphane Marchand-Maillet. 'Bluetooth indoor positioning'. In: *Master's thesis, University of Geneva* (2012).

[9] Margrit Betke and Leonid Gurvits. 'Mobile robot localization using landmarks'. In: *Robotics and Automation, IEEE Transactions on* 13.2 (1997), pp. 251–263.

[10] Ph Bonnifait and Gaetan Garcia. 'A multisensor localization algorithm for mobile robots and its real-time experimental validation'. In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. Vol. 2. IEEE. 1996, pp. 1395–1400.

[11] J. Borenstein and Liqiang Feng. 'Measurement and correction of systematic odometry errors in mobile robots'. In: *Robotics and Automation, IEEE Transactions on* 12.6 (Dec. 1996), pp. 869–880. ISSN: 1042-296X. DOI: 10.1109/70.544770.

[12] Johann Borenstein et al. *Mobile robot positioning-sensors and techniques*. Tech. rep. DTIC Document, 1997.

[13] Sven Böttcher. 'Principles of robot locomotion'. In: *Proceedings of human robot interaction seminar*. 2006.

[14] Mathieu Bouet and Aldri L Dos Santos. 'RFID tags: Positioning principles and localization techniques'. In: *Wireless Days, 2008. WD'08. 1st IFIP*. IEEE. 2008, pp. 1–5.

[15] Kai Briechle and Uwe D Hanebeck. 'Localization of a mobile robot using relative bearing measurements'. In: *Robotics and Automation, IEEE Transactions on* 20.1 (2004), pp. 36–44.

[16] Natural Resources Canada. *Passive vs. Active Sensing*. http://www.nrcan.gc.ca/earth-sciences/geomatics/satellite-imagery-air-photos/satellite-imagery-products/educational-resources/14639. Accessed on 2014-9-21. 2014.

[17] Rafael Capurro et al. *Ethics and robotics*. IOS Press, 2009.

[18] B Cook et al. 'Indoor location using trilateration characteristics'. In: *Proc. London Communications Symposium*. 2005, pp. 147–150.

[19] Malcolm N Cooke et al. 'Use of stereolithography to manufacture critical-sized 3D biodegradable scaffolds for bone ingrowth'. In: *Journal of Biomedical Materials Research Part B: Applied Biomaterials* 64.2 (2003), pp. 65–69.

[20] Martin Dekan and František Duchoň. 'Navigation of autonomous mobile robot using ultrasonic and infrared sensors'. In: *Robotics in Education* (2010), pp. 16–17.

[21] Andrzej Dworak et al. 'Middleware trends and market leaders 2011'. In: *Conf. Proc.* Vol. 111010. CERN-ATS-2011-196. 2011, FRBHMULT05.

[22] João Sena Esteves, Adriano Carvalho and Carlos Couto. 'Generalized geometric triangulation algorithm for mobile robot absolute self-localization'. In: *Industrial Electronics, 2003. ISIE'03. 2003 IEEE International Symposium on*. Vol. 1. IEEE. 2003, pp. 346–351.

[23] Eurobot. *Robomovies*. http://www.eurobot.org/attachments/article/31/E2015_Rules_EU_EN_final.pdf. Accessed on 2015-1-23. 2015.

[24] John Fauvel, Robin J Wilson and Raymond Flood. *Möbius and his band: mathematics and astronomy in nineteenth-century Germany*. 1993.

[25] Josep Maria Font and Joaquim A Batlle. 'Mobile robot localization. Revisiting the triangulation methods'. In: *Proceedings of the IFAC Symposium on Robot Control, Bologna*. 2006.

[26] José Gaspar, Niall Winters and José Santos-Victor. 'Vision-based navigation and environmental representations with an omnidirectional camera'. In: *Robotics and Automation, IEEE Transactions on* 16.6 (2000), pp. 890–898.

[27] Steve Goldberg. 'An Introduction to Mobile Robotics'. Lecture in INF3480 at the Department of Informatics, at University of Oslo. 2013.

[28] M. Hebert. 'Active and passive range sensing for robotics'. In: *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*. Vol. 1. 2000, 102–110 vol.1. DOI: 10.1109/ROBOT.2000.844046.

[29] Martin Hellebrandt, Rudolf Mathar and Markus Scheibenbogen. 'Estimating position and velocity of mobiles in a cellular radio network'. In: *Vehicular Technology, IEEE Transactions on* 46.1 (1997), pp. 65–71.

[30] Jeffrey Hightower, Roy Want and Gaetano Borriello. 'SpotON: An indoor 3D location sensing technology based on RF signal strength'. In: *UW CSE 00-02-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA* 1 (2000).

[31] Pieter Hintjens. *ZeroMQ: Messaging for Many Applications*. " O'Reilly Media, Inc.", 2013.

[32] Mats Høvin. 'Additive manufacturing'. Lecture in INF4500 (Rappid prototyping) at the Department of Informatics, at University of Oslo. 2015. URL: http://heim.ifi.uio.no/matsh/inf4500/lec/htm9/L9.pdf.

[33] William J. Hughes. *Global Positioning System (GPS), Standard Positioning Service (SPS) Performance Analysis Report*. Tech. rep. Atlantic City International Airport, July 2014. URL: http://www.nstb.tc.faa.gov/reports/PAN86_0714.pdf.

[34] Larry Irving. *Automation in the Oil and Gas Industry: Past, Present, Future*. URL: http://www.spegcs.org/media/files/pages/7c03ff6a/larry_irving.pdf.

[35] Frédéric Jurie and Michel Dhome. 'A simple and efficient template matching algorithm'. In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. Vol. 2. IEEE. 2001, pp. 544–549.

[36] Evgeni Kiriy and Martin Buehler. 'Three-state extended kalman filter for mobile robot localization'. In: *McGill University., Montreal, Canada, Tech. Rep. TR-CIM* 5 (2002).

[37] Gunnar Kjemphol. 'Eurobot 2007'. In: (2007).

[38] Bendik Kvamstad. 'To appear'. MA thesis. Norway: University of Oslo, 2015.

[39] Rune Andreassen Magnussen. 'Path planning based on behavior observations'. MA thesis. Norway: University of Oslo, 2015.

[40] Mario Mata et al. 'A visual landmark recognition system for topological navigation of mobile robots'. In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. Vol. 2. IEEE. 2001, pp. 1124–1129.

[41] Jules G McNeff. 'The global positioning system'. In: *IEEE Transactions on Microwave Theory and Techniques* 50.3 (2002), pp. 645–652.

[42] Gregory Michael Nordstrom et al. *Interrupt and message batching apparatus and method*. US Patent 6,085,277. July 2000.

[43] Parallax. *PING Ultrasonic Distance Sensor*. http://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf. Accessed on 2014-9-24. 2014.

[44] Sunhong Park and S. Hashimoto. 'Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment'. In: *Industrial Electronics, IEEE Transactions on* 56.7 (July 2009), pp. 2366–2373. ISSN: 0278-0046. DOI: 10.1109/TIE.2009.2013690.

[45] Vincent Pierlot and Marc Van Droogenbroeck. 'A simple and low cost angle measurement system for mobile robot positioning'. In: *Workshop on Circuits, Systems and Signal Processing (ProRISC)*. 2009, pp. 251–254.

[46] Vincent Pierlot and Marc Van Droogenbroeck. 'Analysis of a robot positioning system based on a rotating receiver, beacons, and coded signals'. In: *European Signal Processing Conference (EUSIPCO)*. 2011, pp. 1766–1770.

[47] Josep Maria Porta Pleite, Federico Thomas et al. 'Concise proof of Tienstra's formula'. In: (2009).

[48] A.N. Raghavan et al. 'Accurate mobile robot localization in indoor environments using bluetooth'. In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. May 2010, pp. 4391–4396. DOI: 10.1109/ROBOT.2010.5509232.

[49] RobotWorx. *Proprioceptive sensors - Self-Monitoring*. URL: http://www.robots.com/education/proprioceptive-sensors (visited on 03/24/2015).

[50] Muriel Salvan. *A quick message queue benchmark: ActiveMQ, RabbitMQ, HornetQ, QPID, Apollo*. URL: http://blog.x-aeon.com/2013/04/10/a-quick-message-queue-benchmark-activemq-rabbitmq-hornetq-qpid-apollo/.

[51] Dmitry A Savochkin. 'Simple approach for passive RFID-based trilateration without offline training stage'. In: *RFID Technology and Applications Conference (RFID-TA), 2014 IEEE*. IEEE. 2014, pp. 159–164.

[52] Steven W Smith. *Digital signal processing: a practical guide for engineers and scientists*. Newnes, 2003.

[53] Mark W Spong, Seth Hutchinson and Mathukumalli Vidyasagar. *Robot modeling and control*. Vol. 3. Wiley New York, 2006.

[54] Andy Ward, Alan Jones and Andy Hopper. 'A new location technique for the active office'. In: *Personal Communications, IEEE* 4.5 (1997), pp. 42–47.

[55] Eivind Wikheim. 'To appear'. MA thesis. Norway: University of Oslo, 2015.

[56] Wikipedia. *Diogenes LaÃ«rtius*. [Online; accessed 8-February-2015]. 2011. URL: http://en.wikipedia.org/wiki/Diogenes_La%C3%83%C2%ABrtius.

[57] Pifu Zhang et al. 'Navigation with IMU/GPS/digital compass with unscented Kalman filter'. In: *Mechatronics and Automation, 2005 IEEE International Conference*. Vol. 3. IEEE. 2005, pp. 1497–1502.

[58] Yu Zhou. 'An efficient least-squares trilateration algorithm for mobile robot localization'. In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE. 2009, pp. 3474–3479.

Appendix A

Trigonometry

A.1 Atan2

Atan2 is a expansion of the trigonometric function arctan. The arctan function gives an angle int he range from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. To be able to express the angle in all the range, we use atan2(x, y) which is a two-argument arctan function. The arctan function is valid for all $(x, y) \neq (0, 0)$.

$$cos(\theta) = \frac{x}{\sqrt{(x^2 + y^2)}}, \qquad sin(\theta) = \frac{y}{\sqrt{(x^2 + + y^2)}} \tag{A.1}$$

With this equation we are sure that we get the correct quadrant for the angle.

Written in a more programmable matter we get Equation A.2

$$atan2(x, y) = \begin{cases} arctan\frac{y}{x} & \text{x > 0} \\ arctan\frac{y}{x} + \pi & \text{y≥0, x < 0} \\ arctan\frac{y}{x} - \pi & \text{y < 0, x < 0} \\ \pi/2 & \text{y > 0, x = 0} \\ -\pi/2 & \text{y < 0, x = 0} \\ \text{Undefined} & \text{y > 0, x = 0} \end{cases} \tag{A.2}$$

Appendix B

Source code

B.1 Server

```
#include <zmq.hpp>
#include <string>
#include <iostream>
#include <vector>

#define IP "tcp://*:5555"

void server() {
  zmq::context_t context(1);
  zmq::socket_t socket(context, ZMQ_REP);
  socket.bind(IP);

  while(true) {
    zmq::message_t request;
    socket.recv(&request);

    sleep(1);
    //Fetch the position from the client
    std::string rep = std::string(static_cast<char*>(
       request.data()), request.size());
    std::cout << "Pos:" << rep << std::endl;

    //Convert the position to string and send it back
    std::stringstream ss;
    ss << xPos << "," << yPos << heading;
    std::string result = ss.str();

    zmq::message_t reply(result.length());
    memcpy ((void *) reply.data(), result.c_str(),
       result.length());
    socket.send(reply);
  }
```

```
}
```

Listing B.1: Simple setup for the server with zmq

B.2 Client

```
#include <zmq.hpp>
#include <string>
#include <iostream>
#include <sstream>
#include <vector>

#define IP "tcp://localhost_5555"

int xPos = 50;
int yPos = 50;
int heading = 0;

int client() {
  //convert the position to string to send it
  std::stringstream ss;
  ss << xPos << "," << yPos << "," << heading;
  std::string result = ss.str();

  //Prepare the context and socket
  zmq::context_t context(1);
  zmq::socket_t socket(context, ZMQ_REQ);
  socket.connect(IP);

  //Make message ready to send
  zmq::message_t request(result.length());
  memcpy((void*) request.data(), result.c_str(),
     result.length());
  std::cout << "Sending␣Hello" << std::endl;
  socket.send(request);

  //Get the reply
  zmq::message_t reply;
  socket.recv (&reply);

  std::string rpl = std::string(static_cast<char*>(
     reply.data()), reply.size());
  std::cout << rpl << std::endl;
  return 0;
}
```

Listing B.2: Simple setup for the client with zmq

Appendix C

Poster

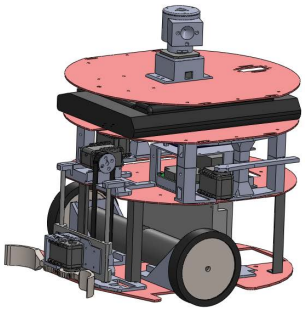
UiO : Department of Informatics
University of Oslo

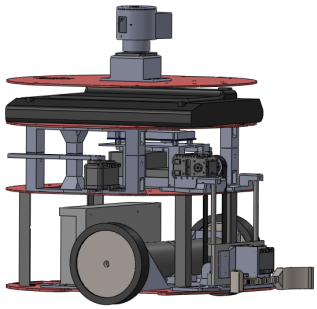
Balotelli's Support Group

Robot Design

Most of the parts for our robot are custom-made by 3D-printing.

The parts are modelled in Solidworks and printed on a Fortus 250mc. This printer uses a technique called Fused Deposition Modelling, and the parts are created by printing layers of molten plastic that fuses together.





Motor Systems

1. Devantech 24V 49:1 Motors

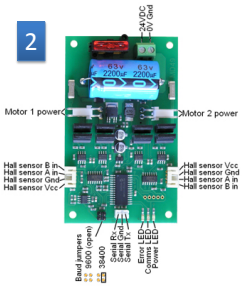
Voltage	24 V
Torque	1.6 N.m
Speed	122 rpm
Rated Current	2. 1A
Encoders	Hall Sensors

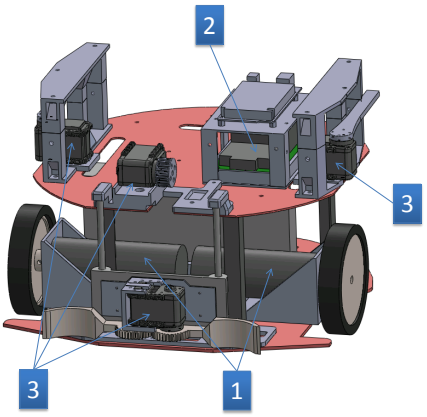
2. Devantech MD49 Motor Driver

Communication	Serial TTL
Voltage	24 V
Rated Current	5 A

3. Dynamixel AX-12A Servo

Voltage	12V
Torque	1.5 N.m
Speed	59 RPM

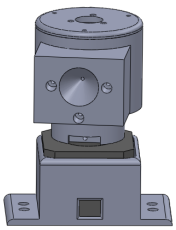


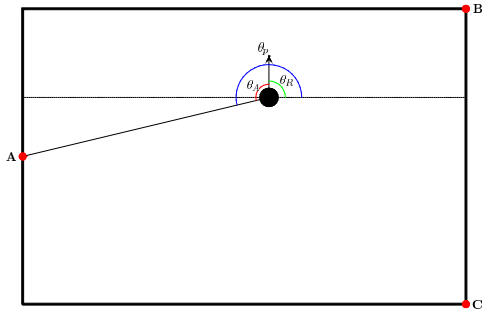


Navigation

The navigation is based on two separate systems; odometry based on encoder readings, and an infrared beacon system.

The robot uses odometry to find an approximate local position, while the beacon system is used to find the global position of the robot with respect to the playing field. An extended Kalman filter is used for the sensor fusion.

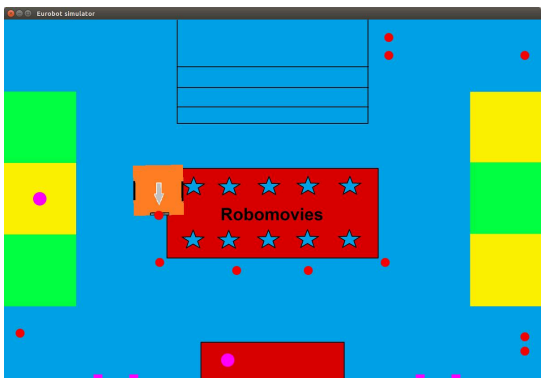




AI

The AI uses a dynamic Goal Oriented Action Planning system.

Goals and actions have feature weights calculated from position and world state. Using simulation and reinforcement learning the robot has been trained to evaluate the feature weights. The planner will dynamically choose the highest ranked plan rated by the trained evaluation of the features.



Team members:
Andre Kramer Orten
Bendik Kvamstad
Eivind Wikheim