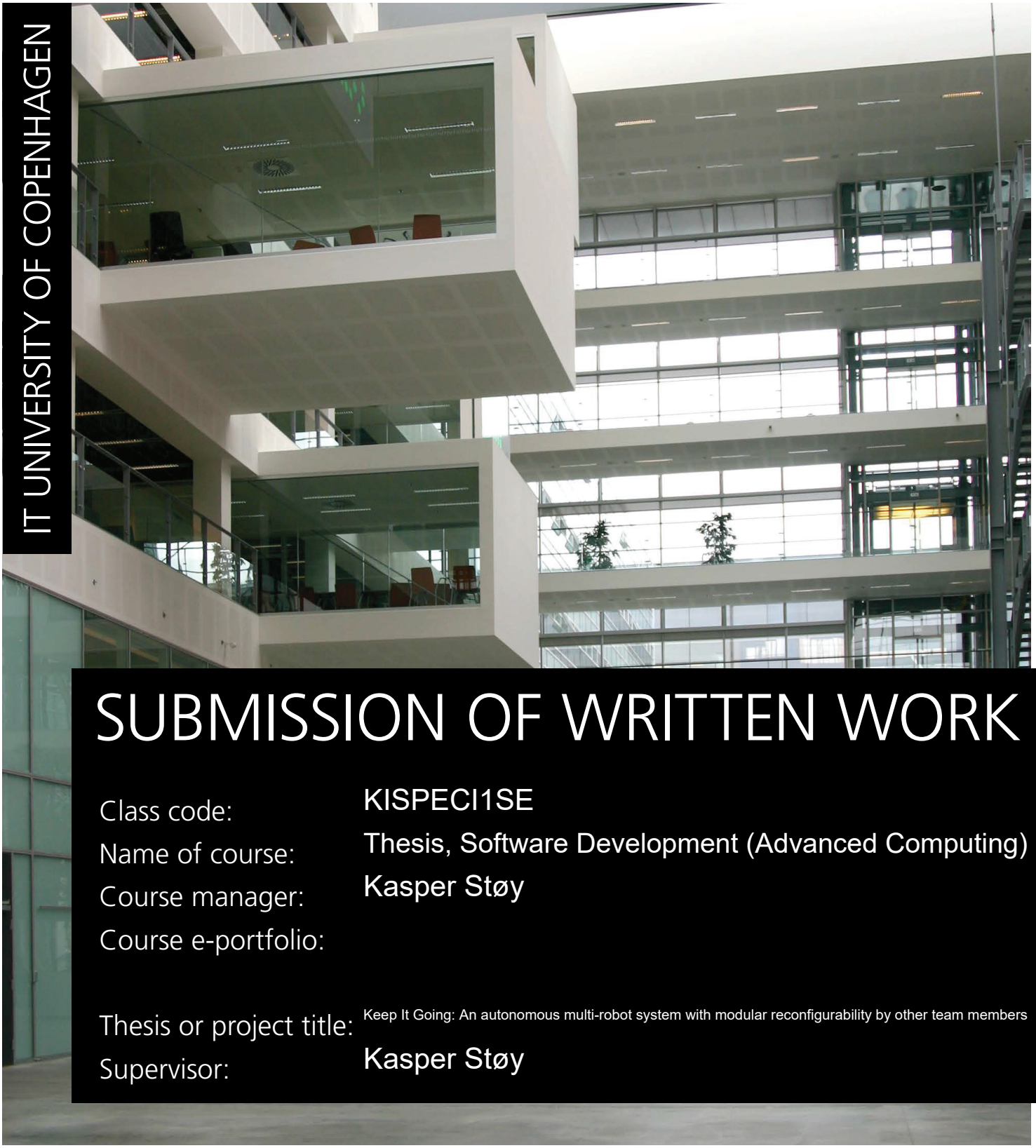
IT UNIVERSITY OF COPENHAGEN



# SUBMISSION OF WRITTEN WORK

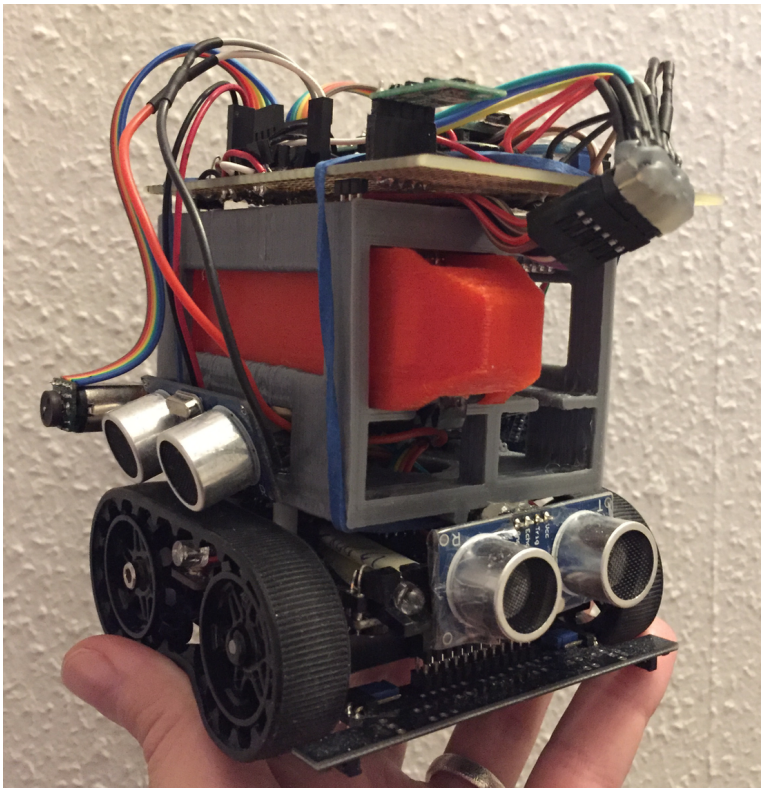
| | |
|---|---|
| Class code: | KISPECI1SE |
| Name of course: | Thesis, Software Development (Advanced Computing) |
| Course manager: | Kasper Støy |
| Course e-portfolio: | |
| Thesis or project title: | Keep It Going: An autonomous multi-robot system with modular reconfigurability by other team members |
| Supervisor: | Kasper Støy |

| | Full Name: | Birthdate (dd/mm-yyyy): | E-mail: |
|---|---|---|---|
| 1. | Lars Yndal Sørensen | 29/02-1984 | lynd @itu.dk |
| 2. | | | @itu.dk |
| 3. | | | @itu.dk |
| 4. | | | @itu.dk |
| 5. | | | @itu.dk |
| 6. | | | @itu.dk |
| 7. | | | @itu.dk |

# Keep It Going: An autonomous multi-robot system with modular reconfigurability by other team members

Lars Yndal Sørensen, M.Sc. Stud., lynd@itu.dk
IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Cph S

January, 2nd - 2017



# Contents

# 1 Abstract - English

An average of 2,978 fatalities and 7,604 injuries are experiences every year in just American mines and Chinese coals mines. The reason for this is first of all the tough working conditions, but also the use of heavy equipment to tear down walls and the afterwards processes.

Attempts to address these high numbers have been done by researching in remote controlled mining vehicles. But even though a vehicle was added a lots of sensors, which could provide valuable informations to the driver, it would still be very difficult to control it. Many of these mining vehicles may also just serve a single purpose in terms of transportation, hamming, drilling, etc. and in case they broke down, a technician would still have to enter the mine.

To address the core issues of these situations, this project will describe how a reconfigurable system for autonomous, mobile robots is developed, which includes 4 types of modules, a mobile robot, and a service station. The project will range all the way from the electrical circuits and the geometrical design, to the behavior of the robots and the service station. It will first address all subtasks, which can be considered as standalone projects which are thoroughly tested after each iteration, before they are combined and tested again in a full reconfiguration experiment. All of the standalone projects turns out to be successful, just like the full reconfiguration experiment, which also shows that the proposed system is absolutely capable of performing an autonomous reconfiguration of mobile robots.

# 2 Abstract - Dansk

Et gennemsnit på 2,978 omkomne og 7,604 skade sker hvert år i blot amerikanske miner og kinesiske kulminer. Grunden til dette er først og fremmest de hårde arbejdsvilkår, men også brugen af svært udstyr til at nedrivning af vægge og de efterfølgende processer.

Forsøg på at løse disse højre tal er blevet gjort ved at forske i fjernstyrede minekøretøjer. Men selvom et køretøj fik tilføjet en masse sensorer, som ville give værdifulde informationer til føreren, ville det stadig være svært at styre det. Mange af disse minekøretøjer løser også kun et enkelt formål i form af transport, hamring, boring, osv. and i tilfælde af at de brød sammen, ville en tekniker stadig skulle ned i minen.

For at adressere kerneproblemerne i disse situationer, vil dette projekt beskrive hvordan et rekonfigurerbart system for autonome, mobile robotter er udviklet, hvilket inkluderer 4 typer moduler, en mobil robot og en servicestation. Projektet vil spænde hele vejen fra de elektriske kredsløb og geometriske designs, til robotternes og servicestationens adfærd. Den vil først adressere alle underopgaver, som kan opfattes som selvstående projekter der er grundigt testet efter hver iteration, før de er kombineret og testet igen i et fuldt rekonfigureringseksperiment. Alle af de selvstående projekter viser sig at være succesfulde, ligesom det fulde rekonfigureringseksperiment, som også viser at det foreslåede system absolut er i stand til at udføre en autonom rekonfigurering af mobile robotter.

# 3 Dictionary

This chapter will describe words and phrases that the reader may not be familiar with. Some of these may also have been made up during this project for convenience and to prevent misunderstandings.

## 3.1 Zumo

Zumo is referred to as the lower part of the robot and is explained in detail in chapter 5.2. This part is manufactured by Pololu[1] and sold as an end product (see figure 1). The full name of this part is Zumo 32U4[75] and is capable of moving, near proximity detection (by IR), line sensor detecting (by IR), has a gyro and a (*poor*) magnetometer. The motors are also equipped with encodes, which may be used for odometry. Besides this, it also has multiple wholes for mounting additional equipment on top of it.

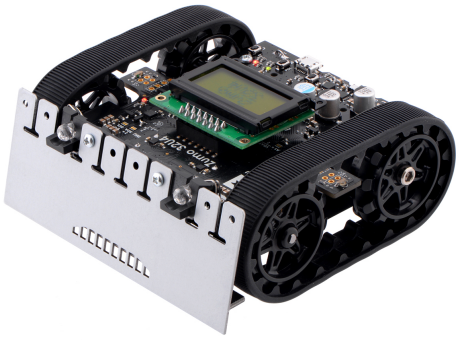


Figure 1: Pololu Zumo 32U4, which is used as the lower part of the robot in this project.

## 3.2 Teensy

The Teensy comes in a low cost or normal version, which respectively are called Teensy LC and Teensy 3.2[2]. They are very similar to the Arduinos and can programmed to interface with hardware. In this project, the Teensy LC was first used as the primary controller for the robot, but was replaced by the Teensy 3.2 after the initial designs because of space limitations in the flash. They are both showed in figure 7 in chapter 5.3, where they also are explained in details.

[1] Pololu's website: `www.pololu.com`

[2] Teensy 3.2 is the current version, but this number will increase as later versions are developed.

## 3.3 Module

In this project, a module is referred to as the device that may be inserted, ejected, or exchanged into/from a robot. The modules are used as examples and four types have been developed in this project: Motor, battery, laser, and LDR. The are explained in detail in 9 and the electrical schematics can be found in Appendix D. See figure 2 for a SolidWorks drawing and a physical laser module.

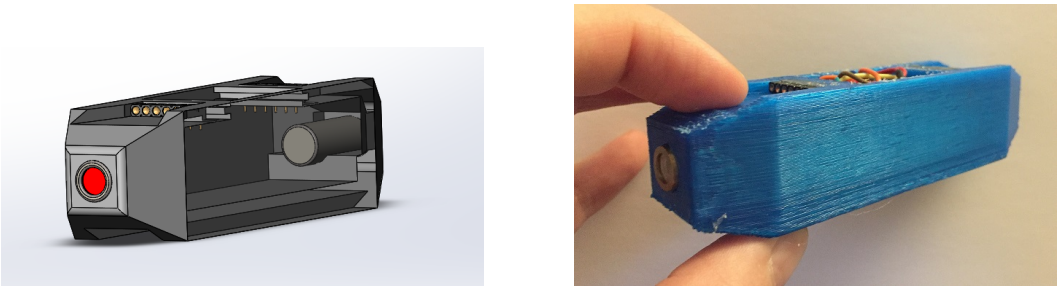


Figure 2: Examples of modules: To the left is the drawing from SolidWorks (with transparent side) and to the right is a physical module. Both modules are laser modules.

## 3.4 Platform

The platform is the upper part of the robot (see figure 3) and is only controlled by the Teensy. The platform is where the modules are inserted and ejected to/from by two small, geared micro motors. The platform is connected to the Zumo (lower part of the robot) with a total of 4 screws and some spacers. On top of the platform is the PCB, which has all the electrical circuits that are controlled by the Teensy: The Bluetooth communication, magnetometer, laser, laser receiver, back-up battery, exchange motors, etc..

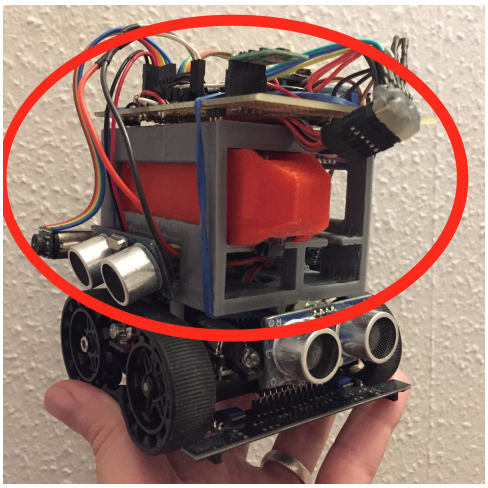


Figure 3: The entire robot at its final stage. The platform is the upper part (red circle).

## 3.5 Service station

The service station is the central unit in the system, which is responsible for assigning tasks to all available robots. It should always be running and will hold the required information about each robot, which include the robot's ID, MAC address of the HM-10 bluetooth module, and modules that it is carrying. The service station will respond to incoming requests about reconfiguration and chose an appropriate robot to carry out the reconfiguration task. Naturally, the service station will only pick a robot, that carries a module of the requested type.

## 3.6 Name convention for design goals

In order to give the reader a better overview, a name convention for the design goals has been established. This will be used in the chapters 6.5, 7.5, 8.5, 9.5, and 10.5, where it has the following format:

| <**a letter referring to the chapter**><**X if it is an extreme goal**>-<**goal number**> |
|---|

The letter referring to the chapter will the first letter in the chapter title: **C**(oarse), **A**(lignment), **D**(ocking), **R**(econfiguration), or **E**(rror detection). The optional X will be used for extreme design goals that are not a necessity for the scenario, but are though of as a bonus or an extra feature, which may increase the functional value of the system.

An example of a goal is **EX-3**, which will be the third extreme goal for the Error detection chapter.

# 4 Introduction

More than 2,978 people are killed in average every year from working in just American[67] mines and Chinese[62] coal mines, while an additional 7,604 people are being injured. On top of that come all of those who live in agony and die many years later from dust-clogged lungs.

For centuries humans have been working in open or underground mines in the attempt to extract resources like precious metals, iron, coal, etc. As the top layers were exhausted, the mines would simply go deeper into the ground, which gradually increased the risk for the workers being trapped, finding pockets with gases, etc. As the mines would keep going even further into the ground, the working conditions would also get worse, not only because of the increased heat when getting closer to the center of Earth, but also as fresh air, electricity, water supplies, potentially special working suits, and much more needed to be provided to make the workers able to do their jobs. These installations are not only costly, but they also prevent the workers from moving freely, which again contribute to an increased risk factor.

Attempts to address these working condition have been made, which mainly have been focusing on creating remote controlled mining vehicles[28]. These vehicle are full sized entrepreneur machines where equipment like LIDARs and cameras have been installed. The driver can then sit in front of a few monitors and a joystick, where he can control the large mining vehicle. Even though the driver is provided with a lot of information, it is still difficult for him be fully aware of the surroundings which makes it difficult to control the vehicle well. Regular miners can therefore not work next to these large vehicles for safety reasons.

As research projects[4, 57, 79] have showed that they are able to make detailed 3D visualizations from SLAM models based on, i.e. LIDAR reading, it should be possible to have robots working in the mines instead of humans, and thereby prevent the high number of fatalities and injuries every year. An extra bonus from this would be the elimination of the supplies of water, fresh air, special suits, etc.

The main issue, that currently needs to addressed before this can be fully achieved, is to make sure that the robots can be repaired without having a person entering the mine and allow for the robots to change tools when they need to attend another task - or if a tool breaks.

Modularity within mobile robots is not a subject that is overwhelmingly successful: A lot of research are attempting to create fully modular robots[2, 9, 13, 14, 15, 16, 19, 24, 27, 33, 34, 35, 42, 44, 51, 61, 64, 65, 77, 78, 82, 83, 92, 93, 94, 95, 97, 98, 99, 101, 102], but even though the intension and the theoretically results are very impressive, the practical results often need to be matured, before they can be used in real life. The reason for this is often the high complexity that comes with not only determining how a structure should be built and organizing the many modules, but also the issues of the modules not connecting properly into each other.

But modularity within fixed robots are very successful, though, as i.e. robot arms or CNC milling machines are able to change tool heads automatically, which is frequently used in the industry. The project that comes closest to have both the modularity and the mobility is the Tesla Battery Swap stations[43], where a Tesla car can drive into a platform and automatically have its battery replaced with a fully charged one. But as it can only replace the battery with a similar one, it is not truly modular.

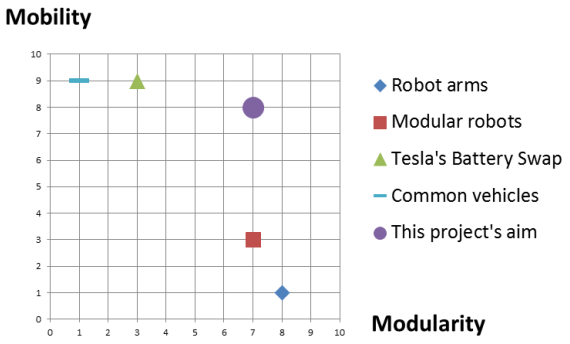



Figure 4: Chart of the available robot systems mobility and modularity. None of them addresses both challenges, except for the hypothesis of this project.

Please see figure 4 for current robots and their ability to be modular and/or mobile.

In general almost every produced items today, are made in a non-modular way, which prevents the user from replacing parts in a item, and therefore must either buy a new one or have a professional technician to perform the replacement, like with cars: The only modular part on a car is the wheels, as everything else must be replaced by an auto mechanic.

The hypothesis of this project is that it is possible to create a system with robots, where the robots are able to locate and repair/reconfigure each other. These robots should be able to work not only in the mines, but also in lot of other environments like industrial halls, for i.e. transportation related tasks, and on the road, for on-site reparation of cars.

All data will be public available from GitHub[84].

# 5 The multi-robot system with modular reconfigurability

This chapter will describe the intended usage scenario of this project and why it has been split into minor parts, that each focus on a single task. It will also provide an overview of the report structure and describe two of the main parts of the robot in details.

## 5.1 Overview

This project consists of not only being able to transfer a module from one robot to another, but also the events leading up to this situation (see figure 5 for the story in pictures): First a robot will request a reconfiguration, which can be caused from foreseeing the need of a missing tool or because an *error detection* system has detected a failure in one of the modules. When the robot has requested a reconfiguration from a coordinating unit, another robot will come and assist with the request, which will require for the two robots to meet/*rendezvous*. But before they can exchange the module(s) they need to be in a position that allows for this to happen with as little risk of failing as possible. Therefore the robot, which brings the working module, will be *docking* into the other robot, that requested the reconfiguration. But this should be done from a position where it can dock safely, which first will require an *alignment* relative to the other robot. When the docking has happened, the robots can perform the *reconfiguration*, where they exchange the module(s), before they leave each other and continue to do their work.

The entire mentioned process would become too overwhelming if it wasn't split up into smaller parts. Therefore this report has been structured with the following main chapters:

- **Positioning I - Rendezvous:** How a robot will meet with another robot
- **Positioning II - Alignment:** How the robot will position itself for the docking
- **Positioning III - Docking:** How the robot will dock with another robot
- **Reconfiguration:** How modules and a platform for these can be designed (hardware, software, and geometrical)
- **Error detection:** How an error within an module can be detected

Each of these chapter may be read out of context from the other chapters, as they all contain sub-chapters with related work, design goals, description of each iteration with evaluation, but also a discussion and conclusion.

After all of these, a complete reconfiguration experiment will be performed i chapter 11 where all of the above is combined and tested, to see if they can make it for a mobile reconfiguration system. The rest of this chapter will be used to describe some of the key components in this project, which are the bottom part of the robot, that is based on the Zumo 32U4 from Pololu, and the main controller in terms of a Teensy.

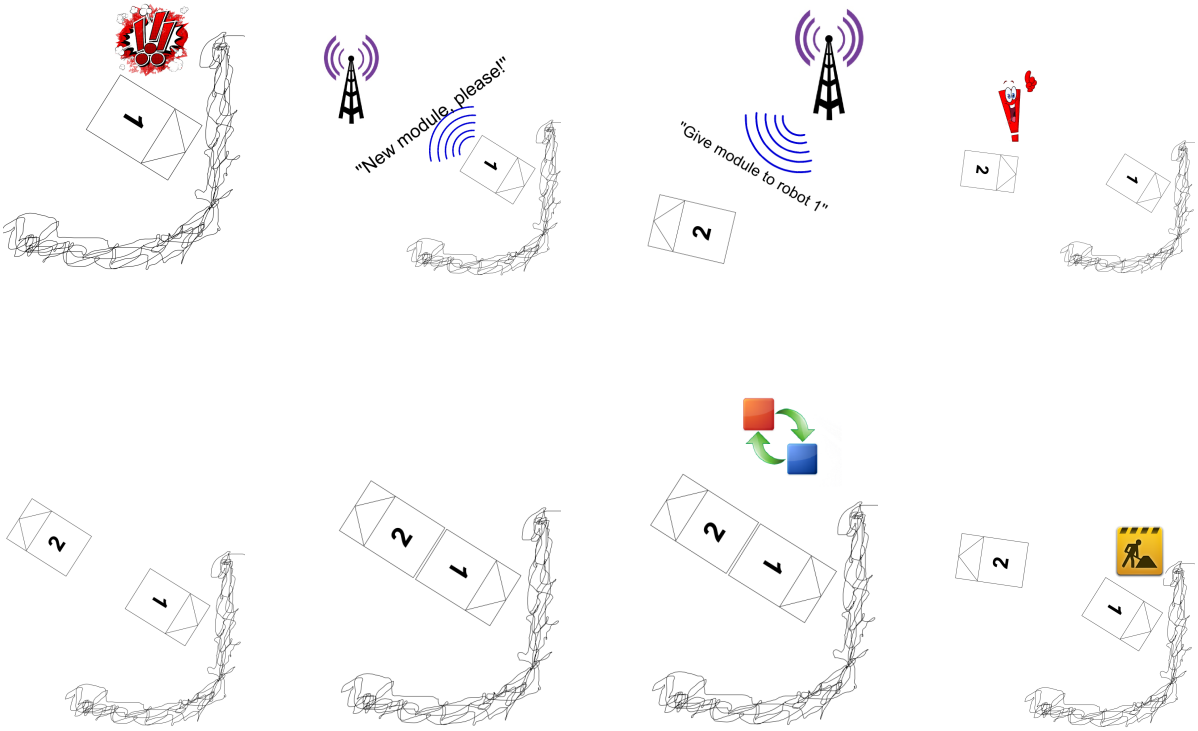



Figure 5: The process of a reconfiguration in 8 steps (starting from top left image): 1) Detecting an error or foreseeing the need of a new module while it it working (*error detection*), 2) Requesting new module from coordinating unit, 3) Coordinating unit choosing other robot to assist with the reconfiguration, 4) Other robot finds the robot (*rendezvous*), 5) Other robots aligns with the robot (*alignment*), 6) Other robot docks with the robot (*docking*), 7) Robots exchanges modules (*reconfiguration*), and 8) The robot can go back to work as the other robot is leaving.

## 5.2 Robot base: Pololu Zumo 32U4

The Zumo 32U4[75] from Pololu is a small mobile robot, which comes with its own 16 MHz micro processor. It can be programmed in the Arduino environment and has 28 kb flash available for a sketch, while 2560 bytes for ram.

The Zumo comes with caterpillar tracks, a lot of sensors, and potential for easy expansion, which makes it a good choice when prototyping a mobile robot rapidly. The sensors available can be seen in figure 6, which include IR proximity and line detection, magnetometer, gyroscope, accelerometer, and encoders for the motors.

The IR proximity sensors are located on the low sensor bar, behind the metal bumper in the front, which helps protect these. It can detect proximity from the front and the sides, but not from the rear. The IR LEDs in the sides are quite small, and a test (see Appendix B) has shown that these are not as powerful as the larger ones in the front. The IR sensors have built-in 38 kHz filters, which means that background noise it decreased.

The Zumo has a low point of gravity and stands firmly on the ground, which party is caused by the low positioning of the 4 AA batteries, but also its low and wide design. The mounting holes and the available pin holes in the center of the Zumo, are great possibilities to add extra structure with sensors, manipulators, micro controllers, etc, on top of it.

The Zumo also comes with the normal variety of interfaces as seen from Arduino's and Teensy's, in terms of SPI, $I^2C$, and Serial. Even though it has a lot of interface capabilities and available pins, most of the pins are already occupied, which means that any addition of extra components must be considered carefully as current functionality most likely will have to be abandoned.

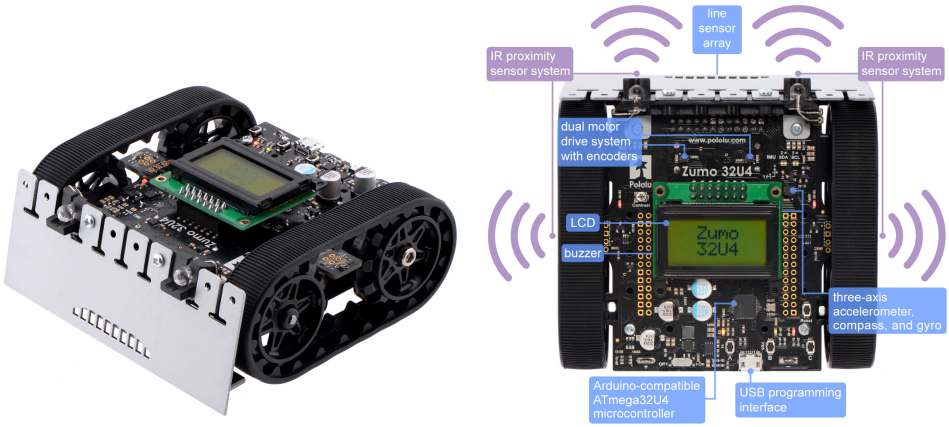



Figure 6: Pololu Zumo 32U4, which is used as the lower part of the robot in this project. The right image is a top view and shows some of the sensors and components available.

## 5.3 Primary micro controller: Teensy

The Teensy[3] comes as a low cost or normal version, which respectively are called Teensy LC and Teensy 3.2[4] and can be seen in figure 7. They can be acquired for 11.80/19.80 USD and are very similar to the Arduinos: They are programmed in the same integrated development environment (IDE) as the Arduinos, which simply has to have an add-on installed. During the initial design of the robot the Teensy LC was used, but had to be replaced with the 3.2 because of the space limitations of the flash (62 kb vs 256 kb).

The advantages of a Teensy, when compared to an Arduino, are in short everything: The only situation where an Arduino is better than an Teensy is when compared to the Arduino Mega which has more pins than the Teensy 3.2. But beside that, the Teensys are based on ARM processors, which runs at higher clock speeds, has more pins, more space, and ram than an Arduino. Even the price of a Teensy is lower than most genuine Arduinos.

The Teensy 3.2, which is the latest version[5] comes with 256 kb flash, 64 kb ram, 34 IO pins (where 21 can be use for analog input and all supports external interruption) and a clock speed of 72 MHz, but can be overclocked to 96 MHz. The 3.2 also supports several interfaces in terms of 3 Serials, 1 SPI, 2 $I^2C$, and 1 CAN Bus. By adding a small 32.768 kHz crystal, the Teensy supports real-time clock (RTC). A lot of other advantages like, i.e. a 12 bit DAC (digital-to-analog converter) can be mentioned, but as this is irrelevant for this project, it will not be elaborated on.


Figure 7: Teensy 3.2 to the left and Teensy LC (Low-Cost) to the right. Initially the Teensy LC was used, but has to be replaced by the Teensy 3.2 because of space limitations of the flash. Please note how the footprints are the same.

[3]Teensy web page: `https://www.pjrc.com/teensy/`

[4]Teensy 3.2 is the current version, but this number will increase as later versions are developed.

[5]Actually Paul Stoffregen, the inventor of the Teensy, is about to release a Teensy 3.5 and 3.6, which were funded on KickStarter just some months ago and runs at an amazing 120/180 MHz.

## 5.4 Experimental setup

In order to thoroughly evaluate each iteration of the positioning iterations, a scene was set up (see figure 8). This scene consisted of 4 x 5 white A4 papers that were taped together to form a 118.5 x 105.0 cm square. This test area was placed on the floor to ensure a horizontal space.

Well into this project, it was experiences that the white walls and the nearby window reflected the IR light very well, which cause the robot to misbehave. The window was then covered with A4 papers, which still gave some reflectance, though. This may be seen as a downside during the evaluations, but was considered to be an advantage as it had a negative effect on the results; meaning that the system would be even more successful than measured. It can also give more realistic results, when considering that the robot might work in a mine, where the mine tunnels are likely to have a reflecting surface too.

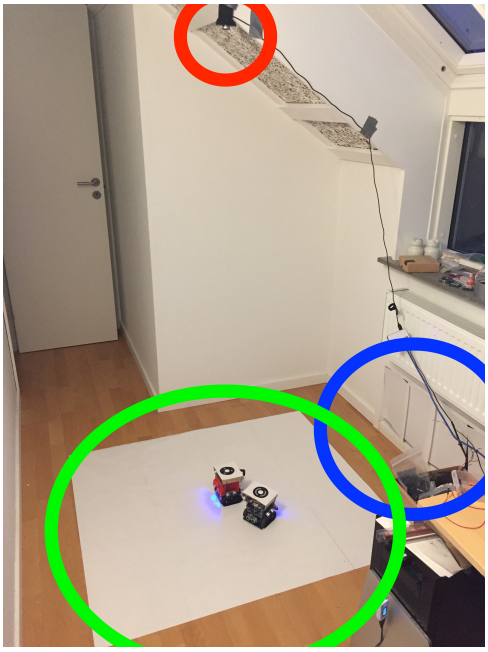


Figure 8: The experimental setup: A 118.5 x 105.0 cm flat surface from A4 papers. Please notice the web cam in top of the picture (red circle), the window covered with A4 papers (blue circle) and the test area (green circle) with two robots.

**Video tracking software**

Most software available for video tracking is made for commercial use[68, 70, 88], which cost a substantial amount of money and for that reason weren't available to this project. But there do exists some research project within video tracking too, which include SwisTrack[52] (see figure 9), EthoWatcher[31], idTracker[72] and OpenControl[1].

Attempts were made to get EthoWatcher running as this was one of the latest published and seemed well matured. But its was quite a struggle, trying to get it running and later the source code for SwisTrack was downloaded instead as SwisTrack was updated several times since the latest version was 4.0. To get SwisTrack up and running, a bunch of 3rd party software had to be installed before updating all the references in the source code. After a few days, it was running on Windows 10 - even though it occasionally crashed and the documentation in the software, but also on the SwisTrack WikiBooks web page [6], were barely useful. But after searching the Internet for general information of the filters, it could analyze the video footages and provide a

[6] SwisTrack WikiBooks web page: `https://en.wikibooks.org/wiki/SwisTrack`

txt file with the tracking data, which then could be added to an image by a program written for this project.

The process of creating the trails of the robots as showing the evaluations of the positioning iterations included several steps:

1. Add fiducial markers on top of the robots
2. Record video footage of the robots' actions
3. Convert the video footage to AVI files
4. Import one at a time into SwisTrack and modify the setup to fit this particular video
5. Let SwisTrack run and create the trails as txt files
6. Import a background image and the txt files into a java program written for this project, which adds lines to the background image

Please note that the compiled version of SwisTrack, the 3rd party software, and configuration file for this setup, can be found in the GitHub repository[84] with the rest of this project.

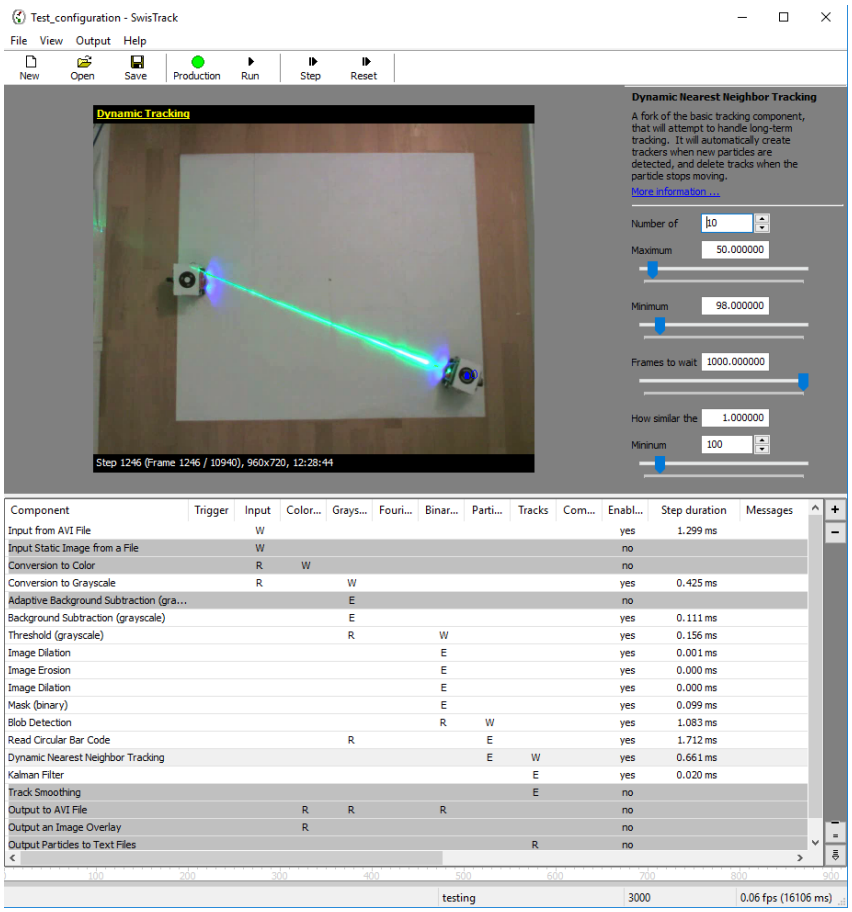



Figure 9: Screenshot of SwisTrack[52], which was used for documenting the trails of the robot in the positioning chapters: 6, 7, and 8. The lower part of the screen shows all the filters, that were used to get decent results.

# 6 Positioning I - Rendezvous

This chapter will describe how the first part of the positioning is evolved: How two robots can find each other and meet. This evolvement will be iteration based, and as this chapter is the first of the positioning chapters, it will also include the communication, while other chapter will refer to here about this. The geometrical design of the robot and its behavior will be based on a list of design goals, which in the end of the chapter, will be used to conclude if the design was successful or not. After the last iteration, the two robots are able to meet each other.

## 6.1 Scenario

When a robot needs to meet with another robot, it will most likely do this in following two steps: First it will try to get to the same room, hall, or field, as the other robot, then it will start moving to the actual position of the other robot.

How the robot achieves the first step really depends on available technologies for determining locations. If this was to be on ground level and under open air, a GPS could easily provide the robot with the required informations (absolute locations). But if it was to be under ground, like in a mine, or maybe indoor, the robot would have to use technologies that have been installed in the entire area where the robot may move - unless it could settle with relative location informations and preferably use technologies that are installed on the robot itself. These could then tell the robot about distances to nearby objects, light levels, signal strengths, etc. which could be interpreted and used to make estimations about the environment.

When the robot has come the same room, hall, or field, as the other robot, it needs to move in on the other robot. If an absolute position of the two robots are available, maybe even a detailed map, it simply has to follow these informations while making sure that no dynamic objects are in front of it. If no absolute informations are available, like the GPS or similar, the robot will still have to use its installed devices as guidance of nearby objects or walls. But at the same time it most also have a way to actually know if it is near the other robot: An quick example is if the other robot emits light in a single band, with a certain frequency, and in a specific pattern (like a TV remote). When the robot detects such a signal, it would know that the other robot is present and start measuring for directional information of the other robot. This doesn't necessarily has to be a light source, as just mentioned, but could also be a directional detection of strength in radio frequencies like bluetooth, RFID, NFC, etc. While listening for these and move in the direction of the signal, the robot would continue to measure for nearby objects and avoid this. No-matter if the robot use an absolute or relative location method, it would also know when to stop moving, so that it doesn't crash into the other robot. This can either be done based on the known/estimated distance to the other robot, provided by either the GPS, radio signal strength, or another proximity device.

## 6.2 Related work

Fortunately, the localization is one of the real big topics within current research describing several kinds of hardware and techniques to achieve this. Some of these provide an absolute position while others provide a relative position - all depending on the possibilities and needs. But common for most of these techniques are that some transmitters are sending *a* type of signal to some receivers. The transmitters are located on either some predefined positions or on the objects to track - the receivers will be positioned on the other. When a signal is received, the time differences or signal strengths (RSSI[7]) between the different receivers can be used to calculate an estimation of the position of the object to track.

Examples of such projects that use this technique may do this with infrared (IR) LEDs as these are very low cost and don't require much power[8]. Because of their low power consumption, they can easily be fitted onto mobile, battery-driven devices like a mobile robot.
Some of the projects[3, 23, 25, 40, 46, 69] that use IR LEDs, and the previously mentioned technique, have either the transmitter or the receiver/sensor mounted in a predefined grid in the ceiling. As the object to track, which has the other part (transmitter or receiver) attached to it, is moving, the detected signals will change and with a bit of math, an estimated position can be calculated.
One project[85] actually has the transmitter and receiver combined onto a single device, but are tracking humans instead of robots. By mounting the device in the ceiling, the system can detect when a person is passing beneath it. As this simply measured the distance to the nearest object, this idea can be applied to objects as well: A tall robot or car can just as well be detected as it will reflect the IR light, too.

Thinking about it, an IR transmitter and receiver setup is just like a camera, but with an extreme low resolution of just 1 pixel. A lot of research projects[18, 60, 63] have chosen to benefit from this and use a camera instead. By doing so, they will also require a lot more power, which shortens their time of performance drastically, if not plugged into the power grid.
When using a camera for detecting objects, this is done with computer vision (CV). CV can be used to simply detect the current position of a blob[9] or fiducial marker[45]. But it can also be used in the far more advanced scenario, where a SLAM[10] model is created[87]: By referencing certain points in a frame with the next, a 3D model can be slowly be created.
As creating a SLAM model obviously requires computational power similar a normal desktop PC, the system often transfers the video footage, by i.e. WiFi, to the computer, where the

[7]Recieved Signal Strength Indicator (RSSI): Indicator for how close the transmitter is to the receiver as the strength will weakens along with the distance

[8]A typical LED (infrared or other) often require 20-50 mA and 2-3 V.

[9]The term *blob* within computer vision is a figure of neighboring pixels.

[10]Simultaneous Localization And Mapping (SLAM): Reference key shapes from one frame to the next and build up a 3D model of the environment.

modeling will happen. If the camera is mounted on a drone, the computer may then control the drone to explore unknown territory or just create the model, while the drone is operated by a pilot.

But SLAM models may not just be created with cameras: Others[71] have done this with LIDAR[11] - some in a combination with camera though, as cameras may have a hard time determine the distance in a picture. LIDAR is a device that sends a lot of laser beams (often IR) out in a narrow grid one at a time. As the beams are reflected on objects, the distance can be determined be measuring the time-of-flight[12]. A project[86] that doesn't use SLAM, but solely uses a LIDAR has been able to track down a moving target. Some research, that also benefits from LIDAR, are focusing on mapping underground mines[57] or even controlling mining vehicles[4, 79] based on the created model.

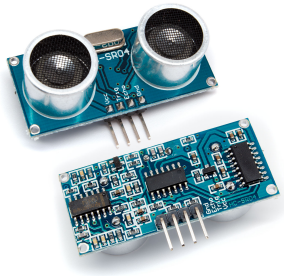



Figure 10: The ultrasonic distance sensor HC-SR04. Note that it has both a transmitter and a receiver, which increase the performance, especially at close ranges of 2 to 5 cm distances.

A popular device, within CV-based projects[17, 18], is the Microsoft© Kinect®, which is a camera device that also provides a very good estimate of the distances. One of the benefits of using the Kinect® is its price[13] and that the computations are being processed by the Kinect itself and thereby not requiring a normal desktop computer to do this.

Having the distance available is a great help for determine how far away an object is. More simple sensors as the common ultrasonic sensor (see figure 10) can do just this, but opposite the LIDAR it will emit a wide wave and provide the distance to the *closest* object that reflects this wave. Even though these type of sensors may be manufactured to emit a wave with a certain angle, it will never be as precise as the laser beam. But this can be turned around to an advantage, if the exact direction of the object to track is unknown or isn't important[49, 59, 60, 79, 91]. In general this device provides an easy and low cost solution when the bare distance is needed.

Another technology, which has just recently been able to provide not only distances but also directions, is RFID: Some times[37] two RFID antennas are angled with 45°, but decent results[36, 90] has been achieved with just a single antenna. The technique is build around the fact that the strength of the signal will vary depending on the source's angle from the antenna. Combining this with the facts, that the strength also is depending on the distance, robots have been able to track down stationary, but also moving, targets.

[11]Light Detection And Ranging (LIDAR): A device sending a dense grid of laser beams out and measures the distance of each as they return after reflecting on an object.

[12]Time-of-flight: Measuring the time difference from when a light beam is emitted to it returns, allows to easily determine the distance to the object it was reflected on.

[13]A Kinect® can be purchased for $50-120 from Amazon, depending on model.

Similar to this, others[54] has been using sound: By mounting four microphones with an angle of 90°, they have used the strength of the sound to determine in which way a docking station was positioned.

WiFi can also be used in a similar way by measuring the signal strength from the access points and compare this with the known positions of these. That is actually how most wireless positioning system works: Either by measuring the signal strength or the time-of-flight (TOD). An example of this is how a ZigBee[14]-based system has been able to do this[12], which also applies some algorithms to increase the accuracy.

The same has been done with regular WiFi, but is sometimes combined with other methods, like odometry[6] or inertial navigation systems[20].

A company that has specialized itself within the wireless technology is Estimote©[15]. They focus on using bluetooth and are capable of determine the position of a smartphone within 1-4 m [7] depending on the size of the location it is tracked in.

This precision is about the same, that GPS[16] can offer, but GPS requires a lot more power on the mobile device and open air. The computations regarding the use of the GPS is done in the GPS receiver, which then offers the position to the connected computer/micro processor.

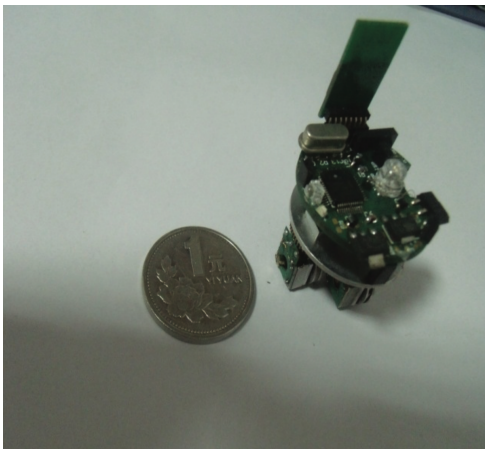


Figure 11: Mao et al.[58] was able to have two of such small robots find each other. Coin next to robot has a diameter of 22 mm. Sensors used are magnetometer and IR transmitter + receiver.

Instead of using devices which often require quite some effort to get working, it might be faster and easier to go more simple sensors. For instance, can gyroscopes or magnetometers reveal the relative (gyroscope) or absolute (magnetometer) headings. The last decade has really matured such devices to be very user-friendly and often comes with well-written libraries to use. Before that it could take weeks or month to establish a good connection to a magnetometer - which then turned out to not provide a high precision. Today a module with both a magnetometer and a gyroscope can be purchases for only 10-20 USD.

Mao et al.[58] has been able to have a small robot (see figure 11) find a similar robot by only using a magnetometer and an IR transmitter + receiver. As these kind of sensors only have a very sparse power consumption, they fit well into small and mobile robots.

[14]ZigBee's are radio device of 25x28 mm that works on a 2.4GHz band. They often come with their own protocol, which offers extra protection against loss of data packages.

[15]Estimote website: `http://estimote.com/`

[16]GPS is short for Global Positioning System and is founded in having 32 satellites orbiting around Earth.

Chen et al.[11] has demonstrated their theory about being able to navigate through a field with obstacles using only odometry and an IR range sensor. The robot, built in Lego Mindstorms, traveled in a corridor with several plastic bottles.

Odometry allows the robot to detect how much each wheel has turned. When knowing the diameter of the wheel, one can easily calculate how far the robot was *meant* to have moved. Because the odometry from the wheel only shows how much the wheel has moved, there is no guarantee that the wheel had a good grip to the surface or something was blocking it was - in order words: There are no feedback based on the *environment*, but only on the robot itself, which decreases the reliability significantly.

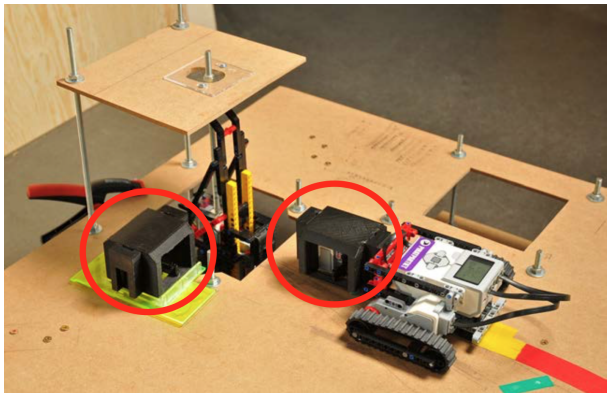


Figure 12: Bachelor project by Hojmark et al. in Lego. The robot can follow the line to a service station of have a sensor module (red circles) replaced.

There are other examples of Lego Mindstorms robots navigating in the wild: One[73] is from a course held at the University of Porto, Portugal. The students were to build a robot with two IR proximity sensors (front and back, but pointing to the same side) which would run in a squared shape: When the two IR sensors detected a white wall, it could 1) compare the distances to the wall and align itself according to this or 2) know the it should turn 90° if the end of the wall was reached.

The bachelor project by Hojmark et al.[26] was also based on Lego Mindstorms (see figure 12). They made a robot capable of following a line on the floor and have the robot go to a service station to change a module. After the module was changed, it could use the new module to follow the line.

As they mention in their conclusion, their *worker robot* was only able to exchange one module. The rest of the robot consisted on the Lego parts and was assembled in a non-modular way, which was a limitation.

In order to provide the reader with an better overview of all the available technologies, the following table has been made based on the previously mentioned techniques and hardware, but also from results from three fairly recent surveys[21, 41, 100]:

| Technology | Power | Average precision | Indoor use | Price |
|---|---|---|---|---|
| GPS | Very high | 5 m | No | High |
| Ultrasonic[12, 79] | Low | 2 cm | Yes | Medium |
| IR[3, 17, 23, 25, 40, 46, 53, 58, 69, 73] | Low | 0.5 cm | Yes | Low |
| RFID[36, 37, 85, 90] | Medium | 10 cm | Yes | High |
| Bluetooth[7] | Medium | 5 m | Yes | Medium |
| WiFi[6, 20] | High | 3 m | Yes | High |
| CV[17, 18, 45, 50] | Very high | 0.6 m | Yes | High |
| LIDAR[4, 50, 57, 63, 71, 79, 86] | High | 0.5 cm | Yes | High |
| SLAM[50] | Very high | 0.6 m | Yes | High |
| Magnetometer[4, 20, 58, 79] | Low | 1° | Yes | Medium |
| Gyroscope[4, 20, 79] | Low | 1.5° | Yes | Medium |
| Odometry[4, 11, 50, 79] | Very low | 22.5° / 10 cm | Yes | Low |
| Sound[54] | Low | 3° | Yes | Medium |

Table 1: Table of discussed technologies for localization with their power requirements, precision, price, and if they are suitable for indoor use.

## 6.3 Design considerations

This part will describe the considerations for a general rendezvous system, while more project specific considerations are to be mentioned later.

Some of the big game-changers regarding the previously mentioned technologies are whether the location is under open air and if it is a mobile system. Both of these can eliminating a substantial amount of the mentioned technologies used in previous research projects, as mobility will imply that it runs on a limited power source. CV, LIDAR, and SLAM all require a fairly good computer, which often is followed by the need of a lot of power to do the mathematically computations. In case the system isn't mobile, these technologies offers very powerful data from just a single sensor.

A technology as WiFi doesn't offer the same amount of details, but also requires a substantial amount of power, which doesn't make it ideal for a mobile robot even though the approach is possible. The use of WiFi would also require for a lot of access point to be installed, which is convenient in city environments as a lot of households already have this, while environments outside of the city might not have this. These locations would therefore need to install a lot of access points in the environments along with a constant power source, which doesn't seem beneficial in a cost/benefit perspective.

In such environments a GPS receiver might be used instead as all external equipment, in terms of 32 orbiting satellites, already are installed - even though the GPS receiver might be a bit power hungry. But GPS receiver can only used in more or less open air environments as the reception from the satellites needs to be good.

As with the WiFi, RFID and bluetooth (in terms of iBeacons) also require plastering the environment with devices. Often this will work in a purely research related scenario to prove the concept of the technology, but seldom reaches the real world, as the end users rarely enjoy spending days or month installing external devices, before they can use the system. On the other hand, bluetooth may also be used for communicating, which would address another issue at the same time.

The system could also aim at using simple sensors, as these often are less expensive. The downside of these simple sensors is all the time that has to be spend for not only getting correct signals and values from the sensors, but also interpreting these. Even though this might take extra time to reach the same level as with other sensors, the end product will be more appealing for end users as the price will be lower.

All of these technologies have their advantages and disadvantages, but as the project should aim at working in *as many* environment as possible, the technologies that are chosen shouldn't be depending on i.e. large power sources, or external equipment.

## 6.4 Implementation considerations

This part is a follow-up on the previously part *Design considerations*. It will describe the considerations specifically relatable to the implementation of this chapter's topic.

**Communication strategy**

An important aspect is the communication, as each robot will undoubtedly have to communicate with a central unit and/or other robots. This will not include references to previous work as the methods are already known. Instead this will give a description of the common communication strategies.

From an overall perspective there are four types of strategies: 1) One-to-one, 2) One-to-many, 3) Many-to-one, and 4) Many-to-many. Occasionally, these types are merged into an in-between result, but these options will not be discussed until later, if they become relevant.

All of the four mentioned strategies types have advantages and disadvantages. In general the One-to types suffer from centralization as a central breakdown is extremely critical. On the other hand, they scale better as they benefit from not having to ”discuss” any options between central units, which introduces a lot of overhead.

For the Many-to types, they have the benefits of not being centralized, which means redundant data persistence and distributed computational power. The downside of this is that it scales poorly because of the overhead of ”discussing” options. To have an overview of the different advantages and disadvantages, please refer to table 2.

| **Strategy** | **Advantages** | **Disadvantages** |
|---|---|---|
| One-to-one | Single recipient<br>Easy to overview | Isolation<br>Only a party for two |
| One-to-many | One place for decision makings<br>One place for all data<br>Good scalability | One place for decision makings<br>One place for all data |
| Many-to-one | Redundant places for data<br>Distributed computational power | Complex decision making<br>Decision overhead<br>Scales poorly |
| Many-to-many | Redundant places for data<br>Distributed computational power | Complex decision making<br>Decision overhead<br>Complex communication<br>Scales very poorly |

Table 2: Table listing the common advantages and disadvantages of the four general communication strategies.

**Means of communication**

There are several options for transferring data signal between the units in this system. The first point to be made is that it should be wireless. A cabled robot is not ”free” and it decreases its mobility. The current available and inexpensive devices for wireless data transferal falls under bluetooth, WiFi, IR, Xbee, and general RF[17] devices. As mentioned earlier in this section, the WiFi tend to have high demands for power, which decreases the mobility of the robot. It was also mentioned that access points may not be preinstalled in the working environment of the robots as these are to work in i.e. mines, which speaks against a WiFi solution.

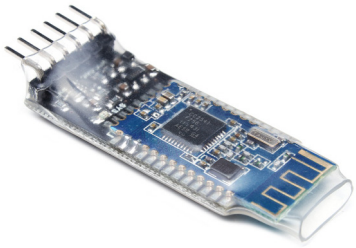


Figure 13: HM-10 modules: Low-cost BLE modules that can detect but also emit iBeacon signals.

Instead bluetooth devices like HM-10 (see figure 13) are getting popular, because of their small size (38 x 16 x 4 mm), low cost (approx. 15 USD), and easy interfacing. They use a serial protocol and, depending on the manufacturer, they support baud rates that ranges from 4800 to 38400. Despite their low power consumption, which is common for BLE[18] devices, they have a range of 10-20 m. The interfacing is easy as the user simply sends AT commands[19] telling the module how to act. A downside is that AT commands will also be broadcasted to a connected unit in the other end and potentially create confusion.

The XBees comes in a large variety and the least expensive cost more or less the same as a HM-10 module. The XBees use a 2.4 GHz network protocol, which may be encrypted if this is desired. The XBees have to be set up beforehand to act either as a Coordinator (center role), Router (distributive role) or End node, but when this is done they will forward all data to each other acting as *one* large mesh network. The downside of this approach is the overhead of data as everything will be transmitted to everybody. The interface with XBees is also serial communication, but they often support baud rates of +57600.

Another wireless possibility is the use of IR. This approach is extremely low cost, but is sensitive to background illumination - especially daylight. The communication could be modulated to distinct background noise from the actual data signals, which often is done by 38 kHz as frequencies near this ir rarely seen in the wild. But even when background noise is attempted to be filtered out, the effective distance is very limited - besides the fact, that it will need a direct line-of-sight!

Low-cost RF devices has earlier proven to be very unreliable when the distance gets more than just a few meters. This could be a coincidence, but given that other more promising devices can be acquired for the same cost, these will not be used.

[17] RF is short for *radio frequencies*.

[18] BLE is short for Bluetooth Low-Energy and is the standard after bluetooth 4.0.

[19] AT commands are serial commands that starts with *AT+* followed the actual command. These may be used to configure a HM-10 module.

## 6.5 Design goals

Founded in the previously parts of this chapter, a list of design goals has been made. This list will show all the needed functionality to be achieved in order for a robot to perform the rendezvous. As this project has a very tight time frame, the goals will only concern a robot that makes a rendezvous with another robot in the same room or field - how the robot will get from its initial position into the room or field, will not be addressed.

At the end, a set of extreme goals has been listed. These are not required, but will be nice to have (note the X in name of the goal).

**Design goals**

**C-1** Must be able to establish a communication link to the other robot
**C-2** Must be able to request actions from the other robot
**C-3** Must be able to respond to requests from the other robot
**C-4** Must be able to notice if the other robot is present
**C-5** Must be able to determine the direction to the other robot
**C-6** Must be able to move towards other robot, when direction is found
**C-7** Must stop at a predefined distance of other robot
**C-8** Must not collide with other robot during the entire procedure

**CX-1** Be able to find its way through a maze and end up in the same tunnel, room, or hall as the other robot

As any of these goals get addressed or improved, they will be mentioned under *Evaluation* for each iteration.

## 6.6 Initial design - IR LEDs for direction

The Zumo platform comes with a lot of sensors preinstalled, which can be useful when addressing the design goals from chapter 6.5. As far as it makes sense these sensor on the Zumo will be used, but otherwise additional sensors will be installed. One of the big advantages of the Zumo is its use of IR for distance estimation: By having one robot turn on its IR emitters, another robot should be able to estimate the direction to the first robot. But as the Zumo only has IR emitters installed in the front and sides (see figure 6), two additional IR LEDs were installed in the back (see figure 14). These were installed to the existing circuit after verifying that they wouldn't harm this, by consulting the data sheet of the Zumo.

The Zumo is able to turn on of each of the sides for emitting IR signals, which then may be detected by the three IR receivers installed in the front sensor bar: One in each side and one in the middle. How well a Zumo is able to detect the IR coming from another Zumo has been documented in Appendix B. This shows the borders between stable, unstable, and no-detectable distances the emitted IR from around the Zumo. The shortest distance for a stable reception was approx. 50 cm, which was when the Zumo was directly to the side of the emitting Zumo. Considering the scale of the system, it seemed like a fair distance as it could be scaled up for real mining machines, automobiles, etc. The emitted IR light was set to have a modulation of 38 kHz to allow the receivers to detect this, which had 38 kHz filters.

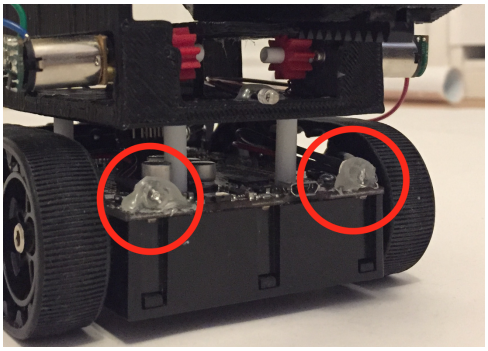


Figure 14: Extra IR LEDs (red circles) were mounted in the rear of the Zumo. These would compliment the existing IR LEDs and allow the Zumo to emit IR light in 360°.

The main controller will be a Teensy LC, which is described in 5.3. The Teensy LC costs 11.80 USD and provides 27 IO pins with interfaces like UART, $I^2C$, SPI, etc, while also running 48 MHz which is 6 times faster than a normal Arduino Uno. It also holds twice as much flash for sketches compare to the common Arduino Uno.

Tests of using a HM-10 modules for estimating the direction and distance was also tried out[20]. The idea was to have one module emit an iBeacon signal, while one to three other HM-10 modules were estimating the distance. The idea was to estimate the distance by using triangulation or have the robot move around to get more values. Unfortunately, the collected values were far to affected by the surrounding environment to be useful.

An approach for detecting the direction instead was also tested: Having covered half of the HM-10 module in a distance of 2 cm with 20 layers of tinfoil, should have given a rough es-

[20]The tests with HM-10 for direction and distance estimation is undocumented.

timation of another modules relative direction. The tinfoil barely suppressed the BLE signals, which made it impossible to estimate the direction. BLE should also be affected by elements containing water, but building a water reservoir half the way around a module seemed dangerous because of the nearby electronics and was rejected as well.

Ultrasonic distance sensors were also tested to see if they could estimate a direction of another ultrasonic sensor: The HC-SR04 ultrasonic distance sensors (see figure 10) are equipped with a transmitter, but also receiver. The interface has a pin for triggering an ultrasonic wave and a pin for detecting the echo, suggesting that the transmitter and receiver could be controlled individually. When setting a robot to constantly listen for ultrasonic signals and another for transmitting, a direction could be determined 80-90 % of the time. But when implementing it into robots that occasionally had to attend other stuff, it didn't work: It turned out that the listening for a signal should be triggered at the correct moment if a positive value should be returned. Another problem was when the robots were pointing in the same direction, as the transmitter would constantly emit a small signal, but increase the power of such a signal, when the trigger pin was raised. This also introduced false positives when a signal bounced off a wall, as it was impossible to distinguish between an "idle" emitted wave or an actual intended wave, because the detection doesn't rely on the strength of the signal, but on the time-of-flight.

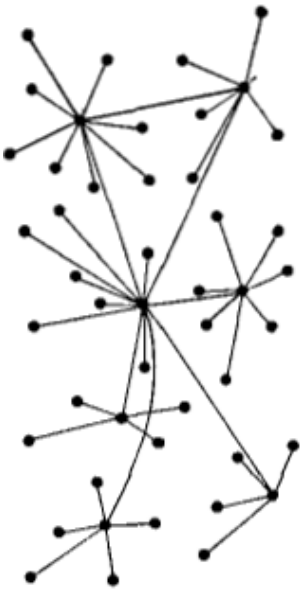


Figure 15: Visualization of how a distributed network is. This is suggested if the system is scaled up drastically.

For determining the distance to another robot, the IR proximity sensors of the Zumo appeared to be an obvious choice. During initial testing it showed that it was mainly able to divide the proximity of an nearby object into six levels in the range of approximately 4-20 cm. These levels was also a bit sensitive to the light, but other then that they worked quite well. The reasons for them to work that well, could be the result of the IR detectors having built-in modulation filters - this couldn't be tested as the filters couldn't be disabled. But given the fact, that rather precise distance estimations would be needed and some of these could be more than 20 cm away, it was chosen to have an ultrasonic sensor installed. In case later iterations implies that the ultrasonic sensor could be avoided, it will be removed from the design, as the design will try to use as low an amount and types of sensors as possible.

**Communication strategy**

Because of the small size of the system, a centralized communication strategy has been chosen (one-to-many). If the project is going to be scaled up drastically, a decentralized strategy is encouraged (see figure 15) as this is a in-between solution of the one-to-many and many-to-many strategy: A few nodes are defined as the highest hierarchically ones, which make the decisions and forward these to lower hierarchically ones. These will then be in charge of having the decisions executed on a robot in that particular group.

Even though a centralized communication strategy has been chosen, tests will often be conducted by directly sending the command to the robot itself, skipping the central unit for more clean test results.

**Means of communication**

The best options for communication are using bluetooth or XBees: Both of these are relatively low-cost, more or less easy to interface with, both have a small design and at least the bluetooth doesn't give a communication overhead. They are both very good options, but to keep the cost as low as possible and avoid potentially end users to struggle with setting of the XBee configuration, the initial design will be based on bluetooth.

This has been chosen by also keeping in mind the possibilities of the iBeacon features, as a robot may need to broadcast a simple value later in the design of the system. Such a value could be the ID of the robot, a value describing its state, or something else. The iBeacon functionality comes with the HM-10 modules at a total cost of 15 USD and is set up with AT commands. Compared to the XBee, AT commands will introduce some extra work, but saving the end user from setting up the XBees, for which reason it seems as the best choice.

The design of the robot after the initial design can be seen in figure 16.

### 6.6.1 Evaluation

The first big surprise was that the HM-10 bluetooth modules were fake replica of the real HM-10 modules: They only supported *very* few AT commands and a search on the Internet showed a lot of users being extreme dissatisfied with these! In order to get them to work with more (but not all) AT commands, the firmware had to the flashed, which was quite a hassle. Unfortunately, these replica also missed some hardware components[21], which implies a big potential for bad communication reception. At the tested distances of within $1\,\mathrm{m}$, any bad reception wasn't

[21] One of the hardware components, which were missing from the fake HM-10 modules was the crystal in the upper left corner. This is most likely used for timing the readings/writings and could result in a bad communication reception.

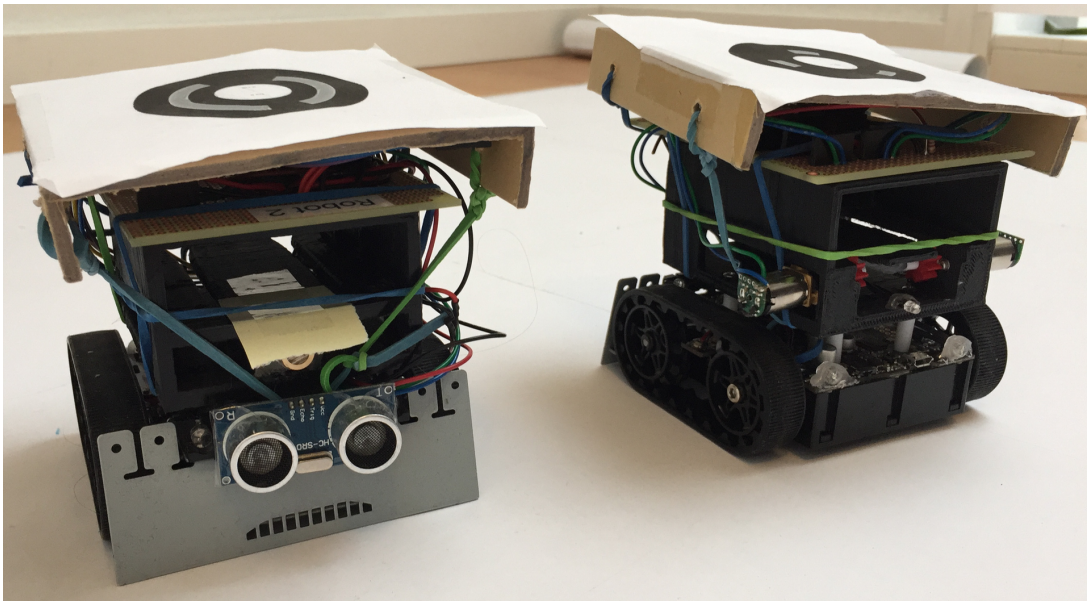



Figure 16: Initial design of the robot. The robots carry the fiducial markers on the top, which were used to track the robots during the evaluations.

noted, but the establishing of the connection could sometimes be difficult: About half of the times it would connect very shortly and the disconnect. This could also be because they had to put into a horizontal position, when a fiducial had to be added on top of the robot for tracking.

But the HC-SR04 ultrasonic sensor, which was installed in the front occasionally gave wrong results in the range of one out of every 100-150 readings. To minimize this, the sensor defaulted to average 5 measurings before returning the result, which didn't eliminated the issue but made it insignificant enough to use the HC-SR04.

Besides that, the initial design worked quite well in most cases, which can be seen on figure 17: When position 45-60 cm away from the robot in the center it was able to establish a communication to the other robot and ask the other robot to turn off all unnecessary equipment, but leave the IR LEDs on with a modulation of 38k Hz. The robot could then scan for the IR light by first turning 90° to the left, return to 0 and then turn 90° the right. The robot would return to the direction of the strongest light signal and move forward while the light level would increase and not fall more than two out of six levels of the maximum detected. The IR level is the sum of the two sensors used: The front sensor will always be used, while the left sensor will be used when turning left, and the right sensor will be used when turning right. This routine would continue until the ultrasonic sensor, in front of the robot, detected a distance, that was equal of less than the required.

On figure 17 the 8 tests can be seen as colored lines, where every "curl" on a line is where the robot has stopped and scanned for a better IR signal. The two lines in the lower, left corner of the figure show how the robot appears to be affected by the reflections of a window a bit

further to the left than the scene, which is 118.5 x 105.0 cm. As the front IR LEDs on the robot are the strongest and the robot, which is attempted located, is pointing left on the picture, the reflections in the nearby window could very well be the reason why one of the tests failed (yellow line) and one took a long detour (sky-blue line). But in total, 7 out of 8 attempts to make a rendezvous with another robot succeeded - even those from behind the other robot, where the added IR LEDs were used.

Following is a list of the goals from 6.5 that has been addressed in this initial design:

- **C-1** Was be able to establish a communication link to the other robot
- **C-2** Was be able to request actions from the other robot
- **C-3** Was be able to respond to requests from the other robot
- **C-4** Was be able to notice if the other robot is present
- **C-5** Was be able to determine the direction to the other robot
- **C-6** Was be able to move towards other robot, when direction is found
- **C-7** Was stop at a predefined distance of other robot

Even though all of the required goals are addressed in this iteration, there is still room for further improvements. Some of these is to have *all* 8 attempts to succeed and avoid any detours.

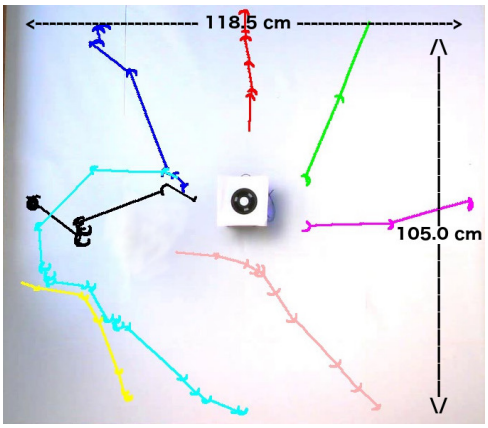



Figure 17: SwisTrack data showing the trails (colored lines) of one robot locating another (marker in the center, pointing left). Depending on the start position the distance between the robots would be 45-60 cm. 8 trails were done from positions, that appeared the most difficult ones, where 1 failed and 2 were significantly disturbed by a nearby white wall and/or window in the left of the picture.

## 6.7 1st iteration - Increasing IR reception

This iteration will focus on decreasing/eliminating the dead angles of the IR field, which had the robot to often stop and do a scan (the ”curls” on the lines in figure 17). For the same reason the iteration will also try to make the emitting of the IR more stable and allow for better IR detection.

But first it must be noted, that the Teensy LC, which acted as the primary controller on the robot, had to be exchanged because of lack of space. About 95 % of the available 62 kb flash was used, even when the code had been thoroughly examined for ways to free up space. A Teensy 3.2 was inserted instead as it had the same footprint, but a lot more flash and ram available for the sketch to run: 256 kb flash and 64 kb of ram (the Teensy LC only had 8 kb of ram). An extra feature is the increased clock speed of 96 MHz, which means that i.e. the IR routine can be called more often as the rest of the routines are executed in shorter time. A side-note is that the Teensy 3.2 is 5v tolerant opposite the Teensy LC.

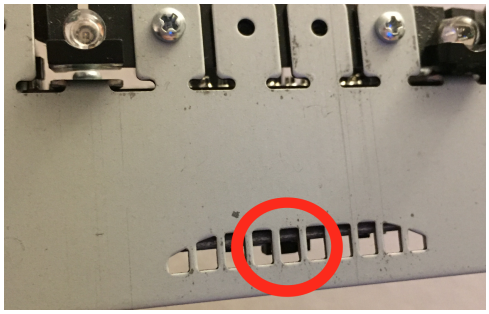


Figure 18: Front bumper of the Zumo, which covers some of the sensors. The red circle shows where the front IR sensor is.

When the faster micro controller was installed and IR emitting routine was called in shorter intervals, the number of IR signals was also increased from 6 to 17: The first half of the IR signals were broadcasted with intensities close to each other, which means that they increased by 4-6 out of 255 for every signal that was broadcasted. The remaining half would exponentially increase in strength, to allow of detections at longer distances, too. To increase better reception, the entire front metal bumper was removed from the Zumo, as this appeared to be covering parts of the sensors - especially the IR sensor, which can be seen in figure 18.

It was also attempted to have the robot measure the distances based on the IR levels, but as described in details in chapter 7.7, they could only be use to very poor distance estimates - too poor, for this purpose. The attempt was made to avoid having the ultrasonic sensor in the front, which now will be left in place.

Measuring the distance with the ultrasonic sensor while scanning for IR, was also implemented in this iteration: If an object is detected closely in front of the robot, while it is performing the scan, it will be interpreted as the other robot. This will prevent the robot from bumping into the other robot, when they are very close to each other, but the scanning robot hasn't yet detected the other one,yet.

It was also implemented that the robot should start by doing a 360° scan, to determine the first direction of the other robot. This seemed more realistic, as the robot seldomly would be pointing in the direction of the other robot from the very beginning.

The address the trouble of connecting to the fake HM-10 module, a delay of 200 ms was inserted after having established the connection. Hopefully this would give the module some time to settle, before transmitting messages.

### 6.7.1 Evaluation

The change from Teensy LC to Teensy 3.2 didn't seem to have affected the robots in any negative way. How much it actually increased with the more stable emitting of IR signals is difficult to conclude as this isn't the only improvement for the IR in this iteration. But because of the same footprint, the Teensy 3.2 fitted right into the socket on the robot.

But increasing the amount of transmitted IR signal per broadcasting combined with the faster micro controller and removing the front metal bumper, did have a very good effect on the tests (see figure 19): The trails of the robot are now going more directly towards the other robot. Only a single trail (the green) took a small detour and left the tracked area - after 20-30 seconds it re-entered the area and went straight toward the other robot. To prevent IR reflections from the windows a close row of white A4 papers were put in front of this, but didn't seem to prevent all of the reflections as the white paper also seemed to reflect the light. In all of the tests the robot was placed 45-60 cm away from the one in the center, like in the previous evaluation, but this time they pointed in the opposite direction of the robot in the center. The 360° scan, in the beginning of each procedure was a success: All, except for one (green line), got a very good initial heading of the other robot.

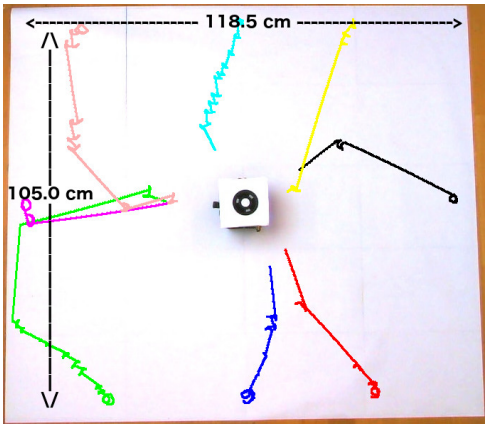



Figure 19: Trails of the robots path towards the robot in the center (with a marker on top and points left). The green line took a detour from the tracked area but re-entered after 20-30 seconds. The "curls" on the lines are where the robot performed an IR scan to determine the direction of the other robot.

During the few scans where the robot was close enough to the other robot that the added detection of objects in front the robot would be triggered was also a success. A couple of the times, it seemed like the robot was about to rotate into the other robot, but stopped just before the collision and told that it had found the other robot with a beep.

About the added delay when the connection between the fake HM-10 modules was established, seemed to have helped significantly. Now the connection is only broken every 15th time, which can be a result of the missing components, as mentioned previously. It may also be the horizontal position because of the fiducial marker, as a vertical direction would be preferred.

Following is a list of the goals from 6.5 that have been improved in this iteration:

**C-1** Better chance for establishing a communication link to the other robot
**C-4** Can notice if the other robot is present
**C-5** Can now determine the initial direction of the other robot
**C-7** Would stop, when the distance to the other was close enough during the IR scan
**C-8** Didn't collide with other robot during the entire procedure

Up to this point all of the required goals have been addressed and some even improved. Further improvements should be towards avoiding any detours.

## 6.8 2nd iteration - Increase IR emission

This iteration will focus on improving the determination of the direction to the other robot, which should decrease the amount of detours and time consuming IR scans.

To improve the determination of the direction to the other robot, the code was changed to only use the front IR receiver. To avoid the robot from ending a scan too soon, the range it would perform a scan was then increased slightly.

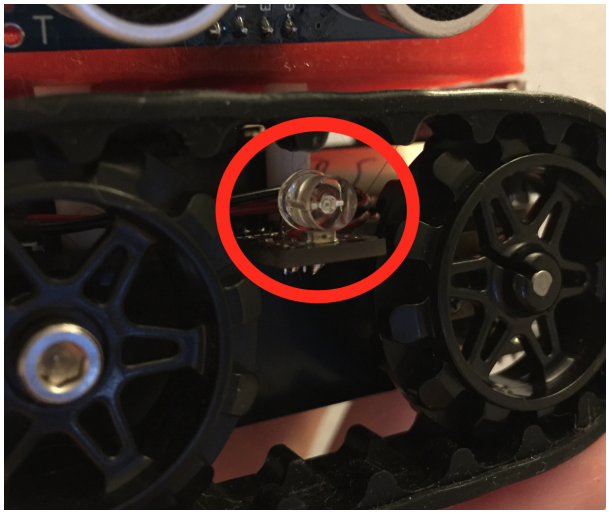


Figure 20: Zumo from the side, which each had an extra IR LED installed (red circle).

As the position of the robots who would go for the detour, was located to the left of the other robot, a strong 90° IR LED was added in between the tracks on each side of the robot (see figure 20). Each LED was soldered the circuit for each side, but in parallel with the added LED in the back. This was make sure that it would get enough power, but not fry anything.

Under the optimization of when adding the extra LEDs at the sides of the Zumo, a better way for reading IR was discovered: Instead of setting the power level to zero on the receiving robot, it was possible to simply get a single reading from one of the sensors. This wouldn't emit any IR signal, but only make a reading and return the value in terms of 0 or 1. By having the robot performing this for 10 ms and the other robot emitting the signals for 100 ms, the chances are that a scan period isn't interrupted and result in better IR scans.

### 6.8.1 Evaluation

This iteration was quite successful as all of the tests ended with having the robot being right next to the robot to rendezvous with (see figure 21). The robot was again positioned 45-60 cm away, but in angles where it appeared most difficult to detect the IR signals because of the borders between the emitters. The robot was also pointing away from the center in all tests, but only one test showed a minor detour (blue trail). The detour can be perceived as negative, but compared to the previous iteration it is an improvement. The robot is still going straight for the other robot, but seems to prefer the stronger light from the front IR emitters, even though an extra IR LED was added to each side of the robot. Even if the IR from the front is preferred, it still seems as they are worth keeping as, i.e. the green trail was attracted to the center of the robot and not the back as in the previous iterations.

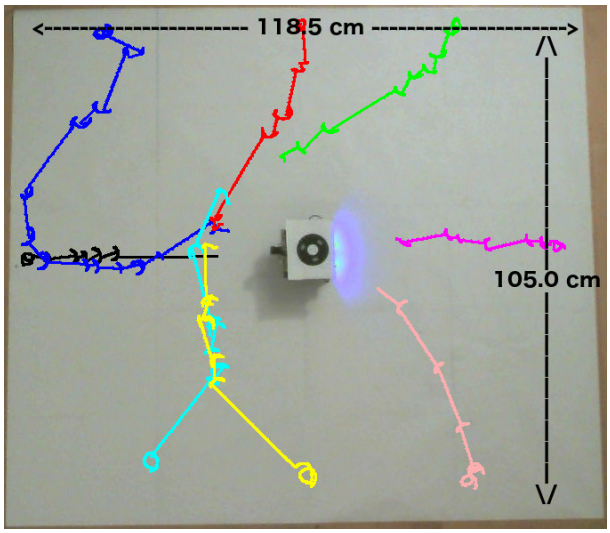



Figure 21: Trails (colored lines) of the robot when makes a rendezvous with the robot in the center (has a fiducial marker on top and is pointing left). One test showed a small detour, but all ended up nicely next to the other robot. All tests had the robot initially pointing away from the center. (The ”curls” are where the robot did an IR scan to determine the direction of the other robot.)

Using only the front IR receiver, could also be why most of the tests showed the robot going rather straight towards robot in the center, but also why the detour was decreased in size.

The goals from 6.5 that have been improved in this iteration are:

**C-4** Can notice if the other robot is present

**C-5** Can better determine the direction of the other robot

## 6.9 Discussion

So how well can the robot make a rendezvous with another robot? And can it even be related when the robot only has the size of 9.4 x 10.0 x 13.0 cm (length x width x height)?

From the very first design of the robot, and its behavior, it was able to find another robot based on simple sensors like IR light and ultrasonic distance sensors - but it wouldn't do this *every* time. For this initial design two of the tests showed that it would react to reflections from a nearby window instead of going to directly to the other robot. One of these two tests actually ended up failing.

As the robot matured by having a few IR LEDs added, the front bumper removed, and updated its behavior, it was now able to find the other robot every time for the two other iterations. It would still be drawn to the reflections from white paper, that covered the previous mentioned window, which caused one of out eight tests take a detour. The remaining seven tests showed that the robot would head fairly straight for the other robot - even though some of these tests also showed how the robot would go for the stronger LEDs on the other robot: Because the robot had preinstalled IR LEDs, it wasn't possible to get an evenly distributed IR field, which can be seen in Appendix B that was measured as a part of the initial design, which was before the two LEDs in the sides were added. It would have been possible to ignore the already installed LEDs, but as these were good enough for proving the concept, they weren't replaced. The fact that the front bumper also had to be removed to get a better reception, tells that the Zumo32U4 from Pololu isn't ideal for this purpose. But such a small robot will doubtfully be of any real use in an underground mine or industry hall, as described in the scenario of this chapter. So if anyone is to use this in the real world, they would most likely have to use a larger robot or built one themselves. This would also allow them to install more LEDs, which would be required since the distances between the robots also would be greater.

One of the issues that also should be addressed is to replace the caterpillar tracks as these hindered the robot from performing its IR scan well: For obvious reasons they are made for great friction, but the IR scan will turn the robot from side to side and try to force the tracks to move in the orthogonal direction of what they are meant to!.The tracks didn't come of during any of the tests or unrecorded trails between the tests, but given that the robot would run for years or the robot was scaled up, say, 10 times, the tracks might come off, leaving the robot great difficulties when attempting to move. A solution to this could be to replace the caterpillar tracks with omni-wheel, which are shown in figure 22. These types of wheels allow the robot to move in any direction while keeping the same heading, but can also rotate the robot without risking to come off. Clearly, this would require for two extra motors to be installed, as the caterpillar tracks were driven from the front wheels, while the back wheels were simple idlers for the tracks. One downside of the omni wheels, is the larger amount of moving parts: Given

that the robot would work in a non-clean environment, these are all prone for dust and dirt that might get stuck, hindering the minor wheels in the omni wheel from rotating smoothly.

Figure 22: Omni wheel, which would allow the robot to rotate in the same position without experiencing friction or risking the current caterpillar tracks to come off.

The robot was tested in a flat, horizontal arena, but the only thing speaking against using it in a rough terrain with a lot of stones and rocks, which would be found in a mine, is its low ground clearance (beside its scale): The battery compartment and the front sensor are the only things preventing this, which means that if they were raised or moved to the top of the Zumo (which also should provide better IR reception), it could be used in this terrain. Removing the front bumper also exposed the sensor bar to potential damage, which wouldn't be an issues anymore, when it was lifted.

It is worth noting that all of the used sensors only use a small amount of power compared to, i.e. LIDAR, CV, or similar, which is one of the crucial topics, when dealing with mobile robots. The precise power consumption wasn't measured, but there can be no doubt that the motors for the caterpillar tracks were the most power demanding components on the robot. Despite these, the robot could still run for hours on 4 AA batteries[22].

But the HM-10 modules, which turned out to be a fake replica, should definitely be replaced by another radio units. These unit may used more power, but it was experiences that these types of modules take 8-10 seconds just for establishing a connection and the use of an AT command interface had its drawbacks as the commands often were forwarded to a connected unit. Instead a technology that either could be connected constantly, without using too much power, and send information to *one* specific recipient, should be used. What comes to mind is a technology *similar* to WiFi, but without the same power demands.

[22]The 4 AA batteries, which powered the robot, was non-rechargeable, which means that they had a total of 6 volts compared to rechargeable one with a total of 5 volts.

Through the latest years, the LoRa technology[23] has been well matured a lot and seems to be grow fairly popular. LoRa is a low-range radio technology developed for IoT devices[24], which can be powered by batteries and has a bandwidth from 0.3 - 50 kbps, depending on how good the reception is. LoRa also uses encryption schemes, which is crucial when communicating with large robots working in mines or industry halls, as an evil-minded person could try to take over the control of these large vehicles. Currently the prices of LoRa units starts at 20 USD, but these will most likely drop a bit as the technology gets even more popular.

One might also use XBees instead, but even though these have the benefit of creating an entire mesh network, they also produce a large overhead as every node will get every message.

[23]LoRa website: `https://www.lora-alliance.org/`

[24]IoT devices is short for Internet of things, which is a common term for battery-powered sensor units that often er placed in the wild.

## 6.10 Conclusion

Through an iterative approach, this chapter has proven how a rendezvous can achieved for two mobile robots. The robots have been tested for all three iterations[25] and progress has been documented by using a video tracking software in a test area of 118.5 x 105.0 cm covered with white paper. The test conditions wasn't optimal as nearby white walls and window seemed to have a strong reflectance of the emitted IR light, which distracted the robots. This only supports the results, as these then reflect a real-life scenario and not a scenario made in a protected environment.

The test results for the final iteration were all successful, but one of the tests showed how the robot was drawn to the IR reflections from the white papers, covering the window. But it was able to correct its path before leaving the relative small test area. In the other seven tests of the last iteration, the robot went straight to the other robot, even though the stronger LEDs naturally seemed more appealing compared to the less IR intensive areas around the robot.

Even thought the Pololu Zumo 32U4 had some disadvantages like low ground clearance and caterpillar tracks, which limited the robot when rotating around itself, it has clearly been showed what is required to achieve two robots performing a successful rendezvous.

The goals of this chapter were stated in 6.5 in terms of seven goals and a single extreme goal. All of the required goals have been addressed, while the extreme goal hasn't'. The achieved goals are showed in the following list:

**C-1** Robot is able to establish a communication link to the other robot
**C-2** Robot is able to request actions from the other robot
**C-3** Robot is able to respond to requests from the other robot
**C-4** Robot is able to notice if the other robot is present
**C-5** Robot is able to determine the direction to the other robot
**C-6** Robot is able to move towards other robot, when direction is found
**C-7** Robot is able to stop at a predefined distance of other robot
**C-8** Robot doesn't collide with other robot during the entire procedure

This makes the robot capable of positioning itself correctly before starting the alignment procedure in the next chapter. After which the robot can dock and exchange any of the modules.

[25] Three iterations as the initial design is also considered an iteration.

# 7 Positioning II - Alignment

This chapter will address the second part of the positioning: The alignment, where one robot will align itself in a certain position with relation to the other robot. This will be evolved in several iterations, and after drastic changes are made in the 2nd iteration, the robot is able to move around the other robot with only little clearance and position itself correctly.

## 7.1 Scenario

When a robot has meet with another robot and it needs to position itself relative to the other, it often requires for the robot to move around the other one, which can be done in different ways: One way is to use absolute locations, which means that the robot knowns that the other robot is located in position X and itself is in position Y. Based on this information it can calculate an estimated distance/path to move around the other robot before it ends up in the wanted position. As this approach implies, it will require some global awareness of where the robots are, which can come from external equipments like GPS, indoor position systems, etc. One thing is that it might not always be feasible to have a lot of external equipment installed, but the fact that the positions might comes from another external unit, introduces a security risk as the provided information doesn't come directly from the counterpart and therefore might have been tampered with. But when going around the robot solely based on the absolute locations, the robot should also keep a certain safety distance as a collision is not detectable based just the absolute locations. Apart from that, the transmitter on one device could also be more obstructed than the other, resulting in the actual positions differs from those in the system. A small note is that there do exist devices which provide an absolute location, but don't directly need external equipment: A magnetometer relies on the magnetic field around the Earth and provides the information directly to the unit that it is connected to.

Another, and in general preferred, way is to use relative locations, which relates the location of an object to where the measurements are done: If an ultrasonic sensor, that emits in a very small angle, detects an objects to be 32.7 cm away, the robot with the ultrasonic sensor knows that an object is exactly 32.7 cm away and in the direction of where the ultrasonic sensor is pointing. Based on this information, the system has the opportunity to move around it, without risking any collisions, and can do this without relying on any external sensors. By using this method, the robot may con-

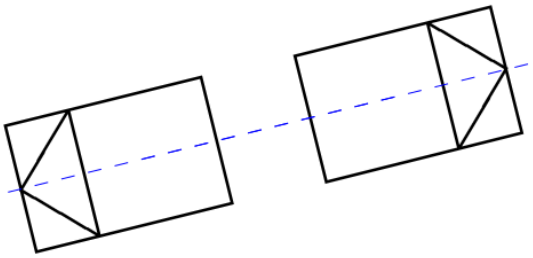


Figure 23: Ending of the alignment scenario: The two robots are standing on the same fictive line (dotted blue) pointing away from each other.

stantly detect the distance to the object while it moves into the correct position. When the position is reached, the robot should have some way of know this, which i.e. could done by having a light source be detectable *only* from a certain angle.

If a lever arm was used to keep the distance to the other robot, it may come in contact with a certain area of the other robot that emits a signal, like a small voltage, which tells the robot where to stop. This could introduce other concerns in terms of avoiding the lever arm to grab any loose wires or similar, that could damage either of the robots.

In order to prepare for the next step in the overall positioning, which is the docking, the robot should turn away when it has reached the center behind the other robot. It should then move away for a certain distance, while keeping the opposite direction that the other robot and end up in the position which can be seen in figure 23. When this is done, it is ready for the docking.

## 7.2 Related work

The types of localization that may be used for the described scenario have been covered and well described in chapter 6.2. To avoid any redundant chapters the reader is kindly referred to this, even though a few interesting and extra relevant projects will be described here:

The one by Chen et al.[11] where they managed to have Lego Mindstorms robot steer through a corridor of plastic bottles by only using odometry and IR proximity sensors. This is somewhat similar to the part of the project that this section tries to solve, as it may have to go around the other robot.

Also the project by Mao et al.[58] where they use a robot (see figure 11) only carrying a magnetometer and an IR transmitter + receiver. Having two of these robot makes it possible for them to find each other and even dock. But they also describe as being sensitive to the environment in terms of IR light.

## 7.3 Design considerations

Most of the design considerations from the chapter 6.3 also applies for this chapter: Examples of this is that CV, LIDAR and SLAM will be too power hungry and/or computational demanding to fit onto a small and mobile robot. The GPS will neither work in this scenario, as the robot may still be in an underground mine or an industrial hall. The communication parts will completely rely on the previously discussed and evolved throughout chapter 6 as nothing new is added during this scenario.

For a more detailed walk-through of the design considerations, the reader is kindly referred to chapter 6.3.

## 7.4 Implementation considerations

As described in the scenario, the robot is to move around another robot and position itself directly in the back of it. Then the robot has to drive straight away from the other robot for a certain distance in order to prepare for the docking in chapter 8. The part where the robot has to drive straight away from the other robot, calls for the headings of the two robots to be known. Fortunately, the Zumo comes with a built-in magnetometer, which could provide these headings. Knowing the headings will not only help when driving away from the other robot, but also as the robot moves around the other: If the ultrasonic distance sensor in front of the robot is used to the estimate the gap between the two robots, the offset from the robot's current position and where it should position itself, before driving away, can be calculated as described in figure 24.

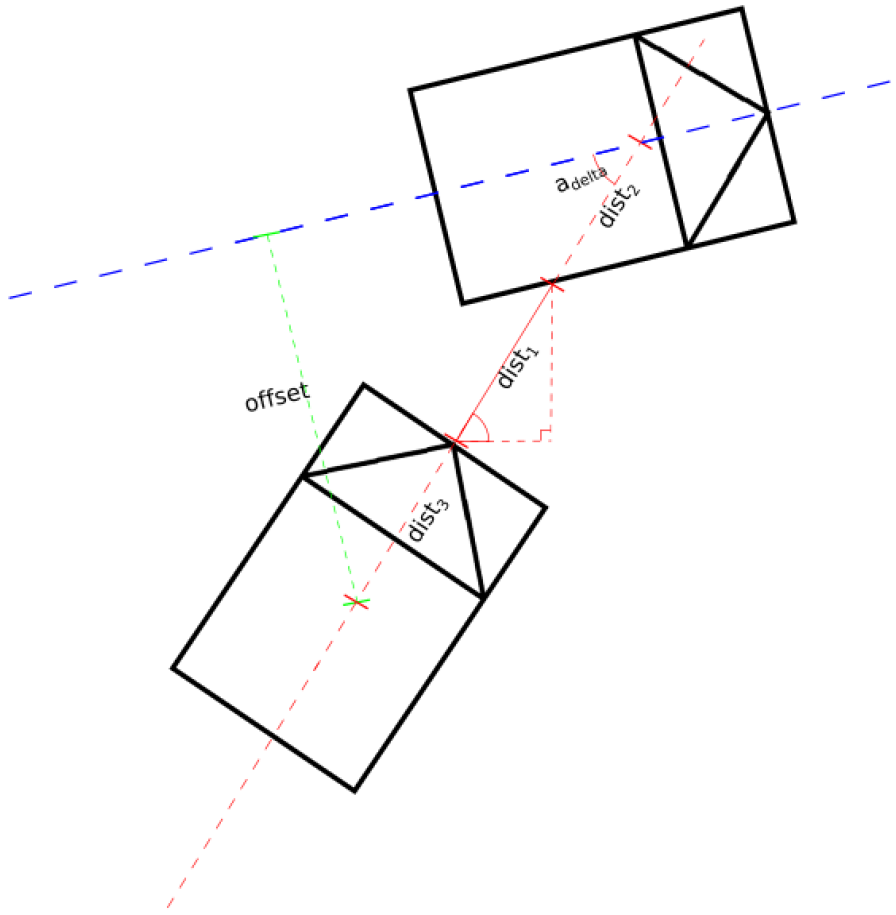



Figure 24: How the offset from the collinear position can be calculated based on the robots' headings and the distance between them (the lower left robot is the one to move around the robot in the upper right corner): $dist_1$ is measured by the ultrasonic sensor, $dist_2$ can be calculated based a robot's width and $a_{delta}$, which is the difference of the two robots' headings. $dist_3$ is half of a robots' length and the $offset$ can be calculated based on the sum of $dist_1$, $dist_2$, $dist_3$ and the angle of $a_{delta}$.

The robot could then move around the other robot based on the calculated offset and the robot's heading. As this approach is very sensitive to the estimated distance to the other robot, it would be preferable to have a system to estimate the side-distance to the other robot. The built-in IR LEDs and detectors might help with this, as precision of this safety distance isn't of high importance in *this* situation: Al it needs to do, is not to crash into the other robot - whether the distance is 1 or 10 cm doesn't matter that much for the initial design. Another and more precise solution could be to install ultrasonic sensors in the sides of the robot. Adding more equipment to the robot is not something this project strives for, but if the alternative is crashing into to the robot, they will be added! Otherwise the ultrasound sensor attached to the front of the robot in chapter 6 could be used: The robot could occasionally turn 90°, pointing towards the other robot, and estimate the distance. Even though it would be more time consuming, it would most likely work.

The Zumo also comes with built-in gyroscope and accelerometer. Some online examples from Pololu's website, shows how the Zumo attempts to drive in a square based on the gyroscope and accelerometer. The attempts were not bad, but should be tested before implementation as the examples showed that the Zumo often deviated with approx. 20° when trying to turn 90°.

## 7.5 Design goals

Based on the previously mentioned scenario, a list of design goals has been made. This list will show all the needed tasks to be done before the alignment procedure can be considered to be absolutely successful. Goals that relates to the communication are not listed below as these already are covered in chapter 6. At the end of the list, has been listed an extreme goal. This is not required for the alignment procedure to be successful, but would be nice to have (note the X in name of the goal).

**Design goals**

**A-1** Must be able to estimate the distance to the other robot
**A-2** Must be able to estimate the direction of the other robot
**A-3** Must be able to move around the other robot without collisions
**A-4** Must be able to get to the middle of the other robot's back
**A-5** Must be able to position itself 25 cm away from the other robot $\pm$ 10 cm
**A-6** Must be able to position itself collinear to the other robot with a sideways distance of $\pm$ 15 cm

**AX-1** Be able to move around the robot in a distance of 5 $\pm$ 4 cm

As any of these goals are addressed or improved upon, they will be mentioned under *Evaluation* for each iteration.

## 7.6 Initial design - Alignment based on magnetometer

In the previous chapter an ultrasonic distance sensor was installed in front of the robot. This could conveniently be used to estimate the distance to the other robot, when it should back a bit away, just before moving to the back of the other robot. But estimating a precise direction to the other robot was a rather difficult task, and thoughts of mounting a LED on top of the other robot as a sort of lighthouse were considered. Unfortunately, LEDs aren't able to emit in a complete half sphere, which would have been necessary to cover all 360 ° around the robot. Otherwise a lot of LEDs should be put together as a kind of disco ball on top of the robot.

Instead it was decided to use the ultrasonic sensor in front of the robot to estimate the direction: Having the robot scan to each side until a long distance was detected should provide the borders of the other robot. The center of these two borders, should give a good estimate of the direction towards the other robot. Now that the robot would be able to estimate the direction to the other robot and back a bit away, all it needed was to go around the other robot.

When the robot was to go around the other robot, it would really benefit from the Zumo's built-in magnetometer. Unfortunately, early tests showed that this magnetometer is pretty close to useless for the Z-axis, which is the vertical axis of the Zumo. The manufacturer of the Zumo, didn't describe this on the product page, but while searching heavily for a way to solve this, a thread in their forum revealed that they have had this issue too - even though they claimed to have been able to get *"decent result"*[26].

Another approach that was thoroughly tested was the use of the built-in gyroscope and accelerometer. Naturally this would require all of the robots to start with a specific heading as this estimation is relative to the start heading. With code examples from the Internet, the Zumo was intended to drive in a square with sides of approx. 50 cm. The Zumo often performed a turn of $90° \pm 20°$, but occasionally it would be $\pm 40°$, which seemed too imprecise and unreliable to be used.

Instead the search for a magnetometers was conducted: The best hobby magnetometer costs 150 USD and was a breakout of Honeywells HMC6343 magnetometer[27], which comes with built-it algorithms to filter out both hard and soft magnetic interference[28]. The HMC6343's algorithms also included tilt-compensation, which works out-of-the-box. The only reason for not using this unit was the high price, which would speak against the low-cost concept of this system. So other options were investigated and lead to the Honeywell HCM5883L breakout with

[26]Pololu claiming to have achieved decent result when estimating the heading: `https://www.pololu.com/docs/0J63/3.7`

[27]Breakout of the Honeywell HMC6343: `https://www.sparkfun.com/products/12916`

[28]Hard magnetic interference is from permanent magnets, while soft magnetic interference is from steel, i.e. metal frames from furniture or building.

a price of 15 USD[29] and Pololu's MinIMU-9 v5[30], which held not only a magnetometer, but also gyroscope and accelerometer for the price of 16 USD. Knowing that magnetometers tend to drift, the robot was designed with the MinIMU-9 using a drift compensating algorithm[74] and naturally placed in the corner of the PCB, where it would experience as little electrical interferences as possible. Concurrently, the HCM5883L breakout was tested and showed good results, but a minor drifting from the HCM5883L was noted for which reasons further use of this breakout was stopped.

While the MinIMU-9 was implemented into the robot's firmware, more tests were often conducted. Some of these showed very strange results, but got a bit better when the unit was calibrated before use[31]. After testing the MinIMU-9 for a very long time at different locations, i.e. the desktop, kitchen table, 3 locations in a flat, etc. it was discovered that the unit is fairly sensitive to magnetic interferences: The desktop had a metal frame and 2 of the 3 locations in the flat had pipes for water utilities running right below the floor, which effectively manipulated the MinIMU-9. When a test location without these interferences was picked, it provided a reliable heading. Comparing this heading between two MinIMU-9s pointing in the same direction, showed a misalignment of approximately 15°. The drifting of the unit was well handled with the algorithm, but showed that the MinIMU-9 should have a second or two for settling when it had been rotated.

Now that the robot should be able to know which direction to head in, it just needed to avoid collisions with the other robot when moving around it. The idea of having a very stable and reliable ultrasonic sensors estimating the distances to each side was very appealing, but starting to mount one sensor to each side seemed a bit drastic and would take up a lot of space on the small robot. Using the IR which came with the Zumo was considered, but for the initial design, the robot was instructed to simple have an extra good distance to the other robot based on odometry. An third way of estimating the distance was to use a lever at each side: When a lever was triggered the robot was too close to the other. This seemed as a nice and simple way of keeping the distance, but the robot could potentially hang on to nearby objects or wires from the other robot, for which reason the levers weren't implemented.

[29]Honeywell HCM5883L breakout: `https://www.sparkfun.com/products/10530`

[30]Pololu's MinIMU-9 v5: `https://www.pololu.com/product/2738`

[31]Calibration of the MinIMU-9 was simply to spin it for some time, before use and set the measured min and max values.

### 7.6.1 Evaluation

All of the communication issues was already addressed in chapter 6, which allows this chapter to focus on the navigation parts. In general the robot handled it good, but there are definitely some of the parts that may be improved: One of these are the safety distance based on odometry, which was very varying with a closest distance of 1 cm, while the longest distance was +20 cm. These distances also seemed to affect the end position, but was also very varying in terms of the orthogonal offset.

But as it can be seen in figure 25, the robot was able to move around the other robot from all eight test positions. The initial estimation of the distance worked well when using the ultrasonic distance sensor, which was added to the robot in chapter 6, while the estimation of the other robot's borders was less successful: It can't be seen in the figure, but the video footages[32] shows that the estimated direction to the other robot, based on the measured borders, was significantly off. This direction was used in two situations of the alignment procedure, where the first one was to estimate a safe way when backing away, while the second one was to estimate it relative position and therefore the path it should drive, when moving around the other robot. For the first part, it wasn't of great importance that the direction to the other robot was significantly off, as long as it wasn't in danger of colliding with the other robot, which it *never* was[33]. But as it is revealed in figure 25, the path around the other robot was not nearly consistent. In this test there is a lot of space around the other robot, but in a real-life scenario this might not be the case. Therefore this issue is something that needs to be addressed as it, for the last step in the procedure, also positioned itself with an orthogonal distance -6 to 27 cm behind the robot, which is more than stated in design goal **A-6**. The collinear distance to the other robot was between 40 and 58 cm, when the robot had aligned itself to the other robot.

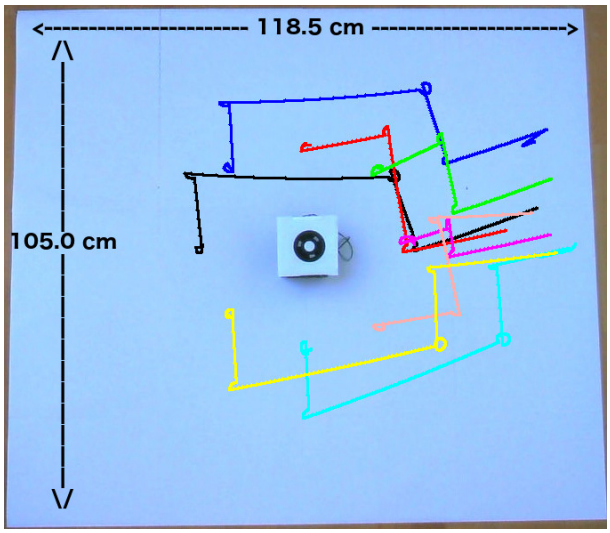



Figure 25: Trails (colored lines) of the robot as it would move the position behind the other robot (marker in the center, heading left). The end position was to the right in the image, while the start position was with a center-to-center distance to other robot of 20 cm in 0, 45, 90, 135, 180, 225, 270 and 315°.

[32]The video footages can be access by contacting the author.
[33]This would have required an estimation of the direction to the other robot to be off with at least 60°

Another thing to note about the trails is that the direction, in which the robot moved, are only very little in parallel, which suggests that the magnetometer on the moving robot also could be inconsistent. This was to be expected, when using low-cost components, but should be taken into account for later iterations. A reason for this inconsistency might be a little affected by how the code is implemented: The robot will turn as slow as possible until the correct heading is detected or passed. As this isn't implemented with interrupts, which isn't possible with this magnetometer, the heading will almost always be a bit off - but not as much as it can be seen in the figure.

The distances that the robot had to move was based on the encoders, which are installed to the Zumo's motors. This worked very well as simply math[34] solved this issue.

The following list of goals from chapter 7.5 have been addressed in the initial design:

**A-1** The robot was able to estimate the distance to the other robot
**A-2** The robot was able to estimate the direction of the other robot, but not precise enough to use this for estimating a good path around the other robot
**A-3** The robot was able to move around the other robot without collisions
**A-4** The robot was able to get to the middle of the other robot's back

Even though **A-2** was addressed, it should really be improved before it can be used in real-life scenarios.

[34]The math used to calculate the distances to drive was based on the encoders having 12 ticks per rotation, the gearing on the motor to be 150.58:1 and the wheel/track diameter of 39 mm. Which leads to: $\frac{12 \times 150.58}{39 \times \pi} = 14.748$ ticks per mm.

## 7.7 1st iteration - Magnetometer calibration

One of the most important issues to address in this iteration was the imprecision when estimating the direction to the other robot. Therefore it was considered to mount four LDRs in top of the other robot and let it notify this robot when they were triggered by a vertical laser - similar to the one used for docking in chapter 8. Theoretically this should be possibly, but only in combination with the robot knowing its heading when the LDRs are triggered. The heading of the robot could be estimated with the already installed magnetometer or based on odometry from the encoders in the motors. But as the magnetometer seems to be unreliable, which is also proven later in this iteration, and odometry is very sensitive to lost friction between the robot and the ground, it was rejected. Lost friction is a significant problem for caterpillar tracks, because when the robot is turning around its own vertical axis, the tracks will slide a lot and most likely move the robot away from its original position.

Another idea could be to remove the entire estimation of the direction to the other robot: If the robot used its ultrasonic sensor in the front and moved around the other robot, by stopping every few cm, turn 90°, measure the distance and coordinate according to this, it should be able to find its way around the other robot. Unfortunately, this would take a lot of time the path around the other robot is a lot more than a few cm in the worst case.

An alternative to this is to use the IR LEDs around the robot and try to estimate the distance based on how much of the reflection that is detected be the IR sensors. This is known to be a good way of estimating distances within approx. 15 cm, which should be just fine for this project. Therefore some prior test was done, which included to have IR LEDs be triggered at a series of different power levels for each cycle. This should give a better estimation of how close the robots would be: The more times the sensors were triggered, the closer they would be. A total of 17 power levels were added to each cycle, where half of them was set to emit low signals in order to better estimate the closer distances. Tests were done not only to the sides but also for the front IR sensor. The side sensors turned out to peak at 3 - 9 cm (same amount of detected power levels) and the next distance level would be at 15.5 cm. The peak for the front sensor was between 5.5 and 11.5 cm (also same amount of detected power levels), but would have the next distance level at 15.0

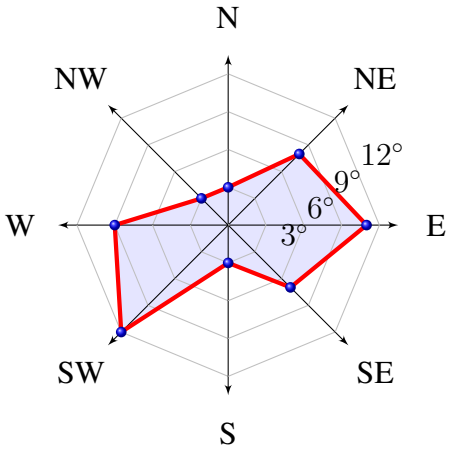



Figure 26: The deviation between the magnetometers as the were pointing in the same direction: N (3°), NE (8°), E (11°), SE (7°), S (3°), SW (12°), W (9°) and NW (3°).

cm. Both of these result are far too imprecise to be used in this project.

The reason for them to *that* bad, compared to IR distance sensor that are built for this purpose, is considered to be a mixture of the variating IR field (see Appendix B) and the fact that the emitter and receiver are not mounted next to each other.

During this iteration the magnetometer was also tested to be sure of how much the experiences inconsistencies were. The two robots were fixed to each other and made sure that they were heading in the exact same direction. Then they were manually rotated, which revealed that the offset of the magnetometers deviated depending on their heading. In figure 26 it can be seen how much the magnetometers deviated at different headings. It is peculiar how one should ”prefer” to compare the headings when pointing along the magnetic field of the Earth and try to avoid this when heading orthogonal. The maximum recorded deviation was 12°, when pointing SW. Even when the magnetometers were calibrated for longer time (30 seconds) on every start-up, it didn't improve the results.

Therefore the safety distance around the other robot was increased, as the robots were only 1 cm from each other in a previous test. In order to increase the precision of the ultrasonic sensor, 5 measurements would now be averaged before they were returned as a single result.

### 7.7.1 Evaluation

A lot of this iteration went with testing different ways to address the issue about a poor estimation of the direction to the other robot. The tested methods were either not feasible or would require too much time for the robot to move around the other. Time is not a real critical issue in this project, but common sense of what seems reasonable rejected the idea of having the robot turn 90° for every few centimeters to measure the distance to the other robot.

It was also experiences that the magnetometers are a lot worse than the initial impression as they are currently considered to be on the very edge of whether they can be used or not.

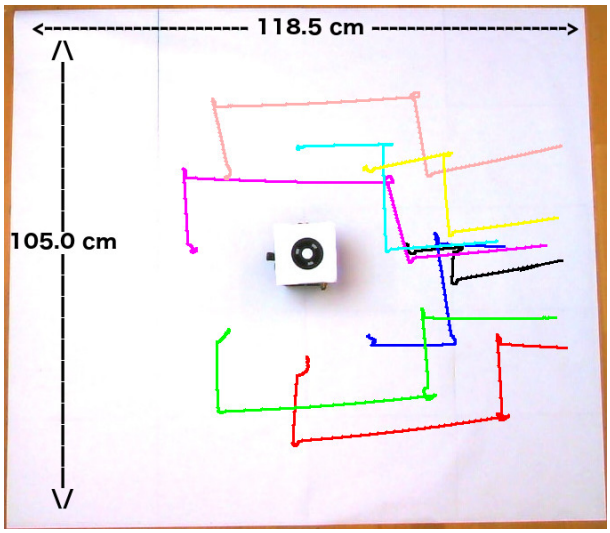



Figure 27: Trails (colored lines) of the robot as it moved around the other robot in the center (black marker, heading left) and position itself behind the other robot. The end position was to the right in the image, while the start position was with a center-to-center distance to other robot of 20 cm in 0, 45, 90, 135, 180, 225, 270 and 315°.

But a few improvements *were* made in this iteration, which results can be seen in figure 27. The added calibration time appeared to have helped on the magnetometers headings', even though the deviating still is poor, when comparing with the results on the initial design in figure 25. This can be concluded as the directions of the last part of the path, where the robot moves away from the other, still isn't close to parallel. The extra readings of the ultrasonic distance sensor also seems to have improved the positioning slightly, but not by much as the robot sometimes drives too far (see red trail in the figure).

In this iteration none of the stated goals from chapter 7.3 was improved enough to be listed.

## 7.8 2nd iteration - Distance sensor instead of magnetometer

This iteration will attempt to solve the issues from the previous iteration by eliminating the use of the ultrasonic sensor for estimating the direction of the other robot when planing the path around the other robot. This will be done by installing a ultrasonic distance sensor[35], similar to the one in front of the robot, on each side pointing in a 90° angle away from the robot. The sensors will also be pointing down a few degrees to better detect the objects that will be closest when passing another robot: The motors for the exchange of modules, the caterpillar tracks, and the sensors themselves (see figure 28). To avoid any unnecessary cables on the outside of the robot, a small whole was added to the bottom of the platform, which could guide the wires from the sensors to the expansion area of the Zumo. Certain functionality had to be removed from the Zumo in order to free up pins for the newly added sensors. Some of the functionalities that were removed were the line sensor in the front and the magnetometer on the Zumo (which couldn't be used for estimating the heading around the vertical axis anyway).

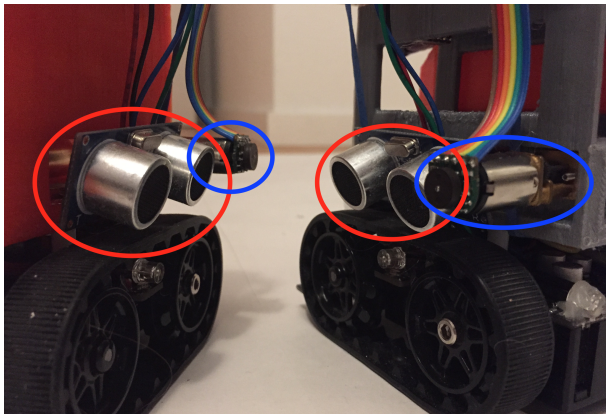


Figure 28: The ultrasonic sensors (red circles) on the side of the robot. Note how they point a few degrees down to better detect the closest objects from another robot: The motors (blue circles), the ultrasonic sensors, and the caterpillar tracks.

The ultrasonic sensors will be used to keep a predefined distance to the other robot, while slowly moving around it. When it is performing this procedure it will also constantly measure for it current heading and expect to be behind the other robot when its heading it the same at the other robot + or - 90° depending on its start position.

Estimating whether the robot could be heading towards or away from the other robot was a bit tricky: If the robot had moved 1 cm closer to the other robot but also turned 30°, the distance measured would have increased, while the actual distance to the other robot would be a lot closer. But as the distance sensor is estimated to detect the distance more than five times each second, this wasn't considered to be a realistic problem as the robot would have had time to react before getting in such a position. If the robot should start to have drastically more actions to attend to, while performing this procedure, it would be wise to install an ultrasonic sensor that emits the waves in a wide angle and use a PID controller for calculating the speed of each track.

Even though the added code for keeping the distance to the other robot was fairly simply,

[35]The used ultrasonic distance sensors are HC-SR04.

the additional informations made the code rather complex as it had to handle if the robot was oscillating in the direction of $\pm 180^\circ$ when it was driving South and trying to keep the correct distance, while also making sure that it hadn't passed the end heading of the $\pm 90^\circ$ from the other robot's heading. The solution to the oscillation around $\pm 180^\circ$ was to count how many times it passed South: Only when it had passed South an odd number of times it would be counted as valid.

When the robot detected that it had passed the end-heading, it would stop and turn 90 towards the other robot. Then it would estimate the center of the other robot and drive forward until there only was 5 cm of distance between them. It would then turn to the left and move back and forth, while scanning the distances on the right side, so that it could estimate the center of the other robot and drive to that point. Then it would turn -90°, so that it would have the opposite heading than the other robot and move 25 cm forward.

In this setup the dependency on the magnetometer had been reduced significantly but also eliminated the calculations of the path around the other robot based on the estimated direction with the ultrasonic sensor in the front. This removed a lot of the weak links in the alignment procedure!

### 7.8.1 Evaluation

The results of this iteration's improvements can clearly be seen in figure 29: The clumsy squared path around the other robot is gone and instead a consistent distance is kept. In the initial test the safety distance was set to 15 cm (left image), but was quickly reduced to only 8 cm (right image). The results for tests with the large safety distance are really good, but those with the small safety distance are beyond all expectations! Even the end positions fall almost right on the on top of each other with an orthogonal variation of only 3 cm, ranging from 8 to 11 cm, while the collinear distances are within 2 cm as it range from 31 to 33 cm.

What seemed the most difficult for the robot, during the tests, was when it was right behind the other robot and should stop based on its current heading and the heading of the other robot: Because of the large deviations of the magnetometers heading, the robot occasionally seemed to move beyond the back of the other, but stopped in time and was able to continue without any problems.

The center-to-center distances of the robot's start positions were reduced to 15 cm as this seemed more realistic, when looking at the results from chapter 6. As the robot was driving around the other robot, the distances to the other robot [36] was noted: For the tests with the close safety distance the actual distance ranged from 1 to 8 cm.

[36] Because some parts on the robots was pointing out, the measured distances were based on as if thread was drawn from the outer corners of the robot.

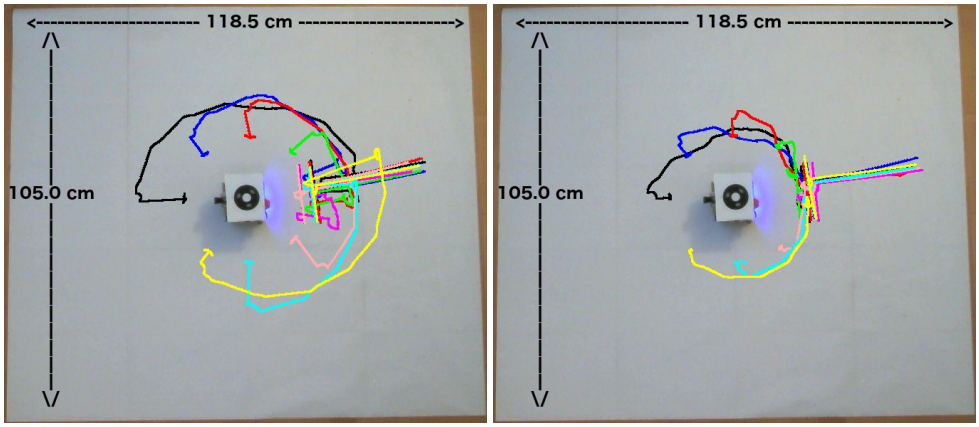



Figure 29: Trails of the tests where the robot must drive to the back of the robot in the center (with a black marker and pointing left). The left image is with a safety distance of 15 cm, while the right images is with a safety distance of just 8 cm. In both images the center-to-center distances of the start positions were 15 cm and evenly distributed with a 45° interval.

A potential improvement to this design is to have a distance sensor only on one of the sides: Then the robot could move forwards or backwards to get around the other robot - or choose to only move clockwise around the other robot if sensor was to the right side, so that it still has the front sensor to prevent potential collisions with other objects.

The following design goals from chapter 7.5 have been addressed in this iteration:

**A-3** The robot was able to move around the other robot without collisions

**A-4** The robot was able to get to the middle of the other robot's back

**A-5** The robot was able to position itself 25 cm away from the other robot $\pm$ 10 cm

**A-6** The robot was able to position itself collinear to the other robot with a sideways distance of $\pm$ 15 cm

**AX-1** The robot was able to move around the robot in a distance of 5 $\pm$ 4 cm

All of the design goals have now been well addressed and even the extreme goal **AX-1** has also been achieved.

## 7.9 Discussion

The latest iteration in this chapter shows that the robot was well capable of performing the required operations to achieve the alignment. The road to get to this point had been a bit bumpy, because of poor components, but this is to be expected, when aiming at low-cost sensors. An example of this is the magnetometers, where the attempt to use the one that came with the Zumo failed because of the nearby motors and encoders, that carried magnets. When other magnetometers, with or without gyroscope and accelerometer, were tested some turned to be drifting while others simply had a deviation between the exact same products of the heading for up till 12°.

After this was discovered, the use of magnetometers was reduced to the absolute minimum and by using ultrasonic distance sensors to estimate the distance to the other robot, while moving around it. A benefit of this is also that magnetometers might experience a lot of interference, if they are used in, i.e. an iron mine or an industrial hall, where there might be a lot of soft and hard magnetic interferences in terms of metal from the builds or machinery, but also wireless access points. In the last iteration the compass was only used to estimate when the robot was behind the other robot based on their headings. If the headings are significantly off, a potential solution could be to have a vertical laser mounted on the side of the robot: In chapter 8 an LDR[37] will end up being mounted in the back of the robots, which also could detect, when the robot in the alignment process passes the center of its back. In this way the need for a magnetometer is removed and the sensors used are ultrasonic, laser and LDRs, which all often are only vaguely interfered by the environment (the LDR should only react to a narrow optical band and measure the light intensity before setting a threshold).

The downside of using the two ultrasonic sensors is that they occupy extra space on the robot and will also add to the total weight and power consumption. The location of the ultrasonic distance sensors are a few millimeters beyond the wheels of the robot, which also puts them in a small risk of getting damages during a collision.

If one still prefers the approach of using a magnetometer, it might be worth acquiring one that is not in the low-cost category and has built-in algorithms for tilt-compensation along with hard and soft magnetic interference like the HMC6343 from Honeywell. The chip comes in a break version from SparkFun[38] for 149,95 USD. Because this device already has tilt-compensation it can also be used if the robot is not working in a completely flat area. The used magnetometer had tilt-compensation, which is important to remember as lifting the just a few degrees might result in big variations of the estimated heading of the robot - even though that didn't happen in this project.

[37] The LDR in chapter 8 is Vishay's TEPT4400.

[38] Breakout board with Honeywell HMC6343: `https://www.sparkfun.com/products/12916`

As the robot moves around the other, a certain amount of power is given to each motor in a certain amount of time. It was experiences that as the batteries run low, the power to the motors will drop even though code request for the same power level as before. This may result in a bit different paths for the robot and even though it should be able to make the path without collisions it should be changed to be based on the motor rotations: The encoders can be used to detect how fast the motors *actually* are running and from this regulate the supplied power to the motors. Then the motors would always run at the same speed, even as the battery gets drained. Using timed movements is also not to prefer, for the same reason, but this would also be solved in the previous mentioned solution.

## 7.10 Conclusion

As shown in the evaluation of the last iteration the robot is very capable of moving from the start position, which is 15 cm from the other robot (center-to-center distance), around the robot while keeping a predefined safety distance of 8 or 15 cm and notice when it is at the other robot's back. Then it can re-align itself to be at the center of the other robot's back before moving away from the robot. As shown in figure 29 of the short safety distance, the robot is very capable of position itself in almost the same spot for every test.

The only concern there might be with the robot is magnetic interferences of the magnetometer, which may occur in i.e. an iron mine, but a valid solution to this has been mention in the discussion.

The goals of this chapter were stated in 7.5 in terms of six goals and one extreme goal. All of the required goals and the extreme goal have been addressed, which are listed here:

**A-1** Robot is able to estimate the distance to the other robot
**A-2** Robot is able to estimate the direction of the other robot
**A-3** Robot is able to move around the other robot without collisions
**A-4** Robot is able to get to the middle of the other robot's back
**A-5** Robot is able to position itself 25 cm away from the other robot $\pm$ 10 cm
**A-6** Robot is able to position itself collinear to the other robot with a sideways distance of $\pm$ 15 cm

**AX-1** Robot is able to move around the robot in a distance of 5 $\pm$ 4 cm

Goals that concern communication where addressed and achieved in chapter 6.

The robot can therefore align itself with respect of another robot. This makes in ready for docking to the other robot, which will be achieved in the next chapter.

# 8 Positioning III - Docking

This chapter will describe the evolving of the third and last part of the positioning: The docking. Here the robot will attempt to dock into the other robot, which in later chapters will be used for transferring modules between the two robots.

Several iterations are made to mature the robot's design and behavior, which all are tested at the end of each iteration. After the last iteration the robot will be able to dock from distances greater than those available in the test area of 118.5 x 105.0 cm.

## 8.1 Scenario

When a robot needs to dock with another robot it will often need to known that it is right in front of the other robot's docking mechanism, so that it won't start the docking procedure and fails because of an initial wrong angle. Potentially the robot can coordinate well enough that the start position for the docking may be more or less irrelevant, which implies the use of not only robust equipment but also that the robot's behavior is very robust. In both cases the robot must continually be able to relate its own speed, heading, and distance to the other robot's - and coordinate according to these values. In theory all of this could be done with an external system, that can provide absolute values of the robots, but as the docking procedure needs to be precise down to the millimeter, it will be preferable to use a relative approach.

Depending on the mechanical principles of the docking, the robot may continue to move forward, while adjusting its speed and heading, until a button is pressed or another signal notifies the robot, that the docking procedure is completed. Many robots carry their tool, or similar, in their front, for which reason it may be easier to have robots dock so they will be side-by-side or back-to-back. How the position to each robot should be, will depend on the purpose of the docking, which won't be described here, but could most likely be related to recharging, tools exchange, or technical inspection scenarios.

As docking often is a sensitive procedure, it may occasionally fail. It these situations, the robot should be able to re-attempt the docking.

## 8.2 Related work

The related work regarding docking for the purpose of reconfiguring the robot is extremely limited - not to say non-existing. But there exists work, which should be relevant by arguing that the *purpose* of docking is irrelevant: Whether a robot wants to have its battery recharged or needs to be reconfigured, shouldn't affect the process of docking. That being said, there exists two types of docking: To a stationary or a mobile unit.

Figure 30: The Roomba® (red circle) vacuum cleaner from iRobot, while it is docking into the stationary docking station (blue circle) to recharge.

A lot of work has been done, where the purpose were to have a robot dock to a station[29, 38, 39, 47, 54, 55, 56, 66, 76, 80, 81, 89, 96]. Often this is done to recharge[29, 55, 56, 66, 76, 80, 96] the batteries of the robot and there even exists an algorithm for where to ideally position a mobile recharging station[89]. The most commonly known robot, that docks to recharge its battery would be the Roomba® vacuum cleaner by iRobot[29] (see figure 30). The other types of work are focusing on docking to another mobile unit for recharging[10, 59, 60], to enhance the robot's possible actions, or assist the other robot, like with the JL-1[101] and JL-2[91, 92], which can be related to modular robots as described in chapter 9.2.

Whether a robot is docking to a station or another robot, doesn't really change the way it determines the position of the other unit. This can be done with ultrasound[49, 76, 91] or IR[76, 81], where there is a transmitter and a receiver. Both of these use the signal strength to mathematically estimate where the robot is, relative to the docking station/other robot, and how is should continue the approach.

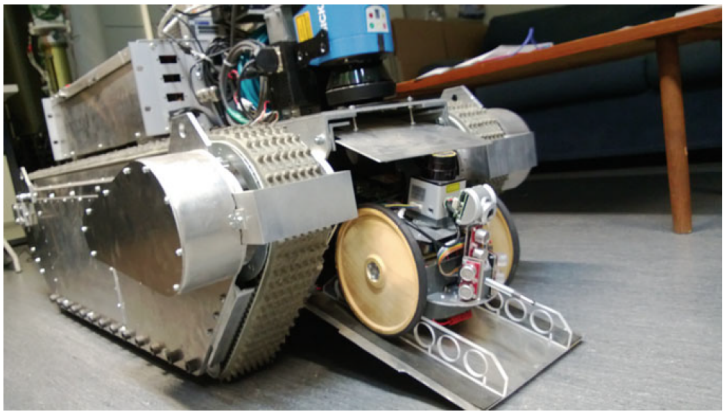


Figure 31: Marsubot Society: The mother robot is able to carry smaller robots in the belly, which may be deployed to explore the nearby area.

The system by Lou, et al.[56] actually uses a laser and an LDR[39], where the laser is fitted into the docking station, guiding the robot directly towards a stick, that moves the recharging pins in the robot's direction. The moving of the recharge pins is for when the robot is not approaching the station directly in front of it, but still allows the robot to dock successfully.

Other projects have looked into, how a small swarm of robots could be deployed and recharged by a "mother ship"[10, 59, 60], where Carlson, et at., have focused on a lifting mechanism of the small robots to delicately deploy these and lift them back into the mother when needed. Martinez, et al. and Matusiak, et al., have been researching in the Marsubot Society (see figure 31), which i.e. can very rapidly explore the nearby area of the mother ship with the smaller robots or have them recharged in the belly of the mother ship.

When the main part of the docking has been accomplished, some projects have used geometrical shapes[56, 76, 81] or physical strength[80, 96] to achieve the final precision.

[39]LDR is short for Light Dependent Resistor, which is a resistor that changes its resistance based on the amount of light it exposed to.

## 8.3 Design considerations

As mention shortly in the chapter of the scenario, it will in general be preferable to use a relative positing system compared to an absolute positioning system. This is for several reasons, where some of them are that the absolute system are designed to provide a good *global* estimation of where the units are, while the relative system provides a good local estimation of where the unit are: The global/absolute system covers large areas, but not down to a cm precision. The relative systems are often able to achieve such kinds of positions, but can only cover local areas. A point is also, as mentioned in chapter 6 and 7, that the absolute positioning system, like GPS or similar, only works on the surface of the ground with a more or less unobstructed line-of-sight to the skies - if the system is to work under the ground, such systems can't be used there unless a lot of equipment is installed in the environment.

Because these kinds of robots are mobile, it will most likely be unfeasible to use power hungry equipment or sensors that needs strong computational computers, which also is mentioned during the two other chapters for positioning in the project: Chapter 6 and 7.

## 8.4 Implementation considerations

The project by Mao, et al. and the Roomba® vacuum cleaner from iRobot[40] both make sure to be pointing directly towards the docking station, before approaching the dock - while also constantly adjusting their heading during the procedure. Mao, et al.'s project pointed towards the other robot before moving closer, while iRobot's vacuum cleaner would detect the docking station and then move in front of it, while keeping the same distance. First then it would turn and move towards the stations in sinusoidal-like motions. This seems like a very nice to start as the firsts step is to position the robot correctly, which should increase the chance of a successful docking.

The Zumo comes with IR LEDs and allows for controlling the LED circuit for each side of the robot. As the Zumo also has an IR detector in the front, which allows to read left and right measured beams, it provides an opportunity to guide two robot together based solely on this[41], almost like Roh, et al.[76] or Wang, et al.[91] with their ultrasonic sensors. If one of the robots emitted an IR light with a, say, 13 kHz modulation on the left side and a 42 kHz modulation on the right, the other robot should be able to distinct each side from the other. If the robot also emitted each signal in sequences of different power levels, the other robot could interpret how well each side was detected. This could then be used for determining the heading towards the robot. In order to achieve this, the IR emitters should emit the light in a broad angle so that the other robot can detect this, even though it is considerably misaligned.

Another way is to use a laser: Lasers are light sources emitting only a single band of light, contrary to the IR light bands which are starting from approx. 750 nm and goes well beyond 1100 nm. As the laser detector only should observe for *one* band, it reduces potential background noise and if the laser even would be modulated it could further reduce the background noise. The disadvantage is that many low-cost laser are made to emit a dot, which are likely to not hit the detector when the ground is not leveled to perfection or a bit of dust/dirt lifts one side of the robot a fraction of a millimeter[42]. A few lasers that emit a line instead of a dot *can* be acquired for a low price, and if the line was emitted as a vertical line, the robot could be tilting a lot and still be detected by the other robot, wearing the detector.

But even if the laser on the robot could be detected, the heading between the two robots could still be way off. One way to solve this could be to have a second detector mounted, but at a another distance to the laser: One detector placed in front of the robot and another placed at the back - preferable at the top to ensure good line-of-sight. If both detectors were triggered at the same time, the two robots would be collinear - otherwise the robot would have to reposition

[40] iRobot's website: `www.irobot.com`

[41] While being aware that the detectors are created with non-disableable 38 kHz filters, the idea is still considered as simple filter-less IR detectors could be installed.

[42] When the robot is lifted a fraction of a millimeter, the laser beam would be hitting a lot higher because of the distance to the detector.

itself (what side it should move to, could be known based on in which order the two detectors are triggered). For several reasons, this approach is quite fragile: First of all it would need to increase the total height of the robot a lot, as the detector had to work, not only at far distances but also at very near distances. When the robot would be close to the other, the other robot would very likely hinder the laser from triggering the detector unless this it placed in at high position, as just mentioned. The extra height is not something to strive for if the robot is to work, i.e. tunnels for mining. Another reason for this being fragile is that, having one small laser hit two small points at the same time, can be very difficult. A solution to this could use it for determining how much the heading is wrong: Assuming the robot is moving with the same paste, the longer time it takes for the laser to hit each detector, the more off it will be. An based on which one is trigger the last, it could be determined which side the robot is positioned to.

The magnetometer, installed in chapter 7, could be replaced with one of the detectors: If the laser can trigger a point on the other robot and both of the robots have the same heading, they should be collinear.

Whether the docking should be front-to-front, back-to-back, or front-to-back, really depends on the actions to happen. As this project is focusing on modules with tools for the robots to use and the tools assumingly would be used in front of the robot, the docking would most likely be front-to-front or back-to-back. In case it is front-to-front, the robots simply should keep driving, while adjusting the headings, etc., until the other robot is detected based on distance or maybe a collision detection, i.e. a touch button. In case the docking is back-to-back, the robot would need to turn around in a far enough distance to avoid collisions before backing up to the other robot, or simply drive backwards all the way.

## 8.5 Design goals

Based on the described scenario, a list of design goals has been made. The list will describe all the needed tasks that have to be done, before the docking procedure can be considered successful. Goals concerning communication needs will not be addressed here as they are already achieved in chapter 6. In the end of the list are a few extreme goals, which are nice to have, but not thought of as vital (note the X in name of the goal).

**Design goals**

**D-1** Must be able to determine the exact direction of the other robot
**D-2** Must be able to reposition itself to a better position if its start position isn't in front of the docking mechanism
**D-3** Must be able to dock from a distance of more than 40 cm and with an orthogonal offset of more than 30 cm
**D-4** Must not be more than 1 cm orthogonal misaligned with other robot, when docking is complete
**D-5** Must not be more than 1 cm away from the other robot in a collinear distance, when docking is complete
**D-6** Must not crash into the other robot during the docking procedure
**D-7** Must be able to detect, when the docking has completed
**D-8** Must be able to successfully dock more than 80% of the times
**D-9** Must be in full contact with the other robot when docking procedure is completed

**DX-1** Be able dock without touching the other robot
**DX-2** Be able to reattempt the docking procedure if it fails

As any of these goals get addressed or improved, they will be mentioned under *Evaluation* of each iteration.

## 8.6 Initial design - Laser and magnetometer

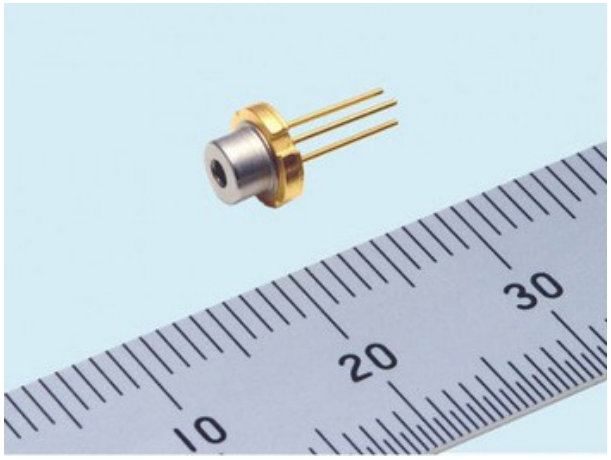



Figure 32: A TO18 laser diode component. A similar was used in the attempt to increase the strength of a laser, but the legs broke off during assembling.

Chapter 7 describes how a magnetometer was installed to provide an absolute heading of the robots and how this is used for the for navigation issues. Because the magnetometer is already installed, it appeared as a good choice to use this in combination with a line laser with a vertical beam: Having the robot pointing directly towards the other robot and they both have the same heading, will mean that they are collinear. Lasers are also low-cost, emits only a single band and are not too power consuming. Unfortunately, it wasn't possible to acquire a good, strong infrared laser, which would have ensured good reception, not only because of the watts emitted but also because of the IR light, which there exists less of in the daylight compared to i.e. red. Therefore it was attempted to change a 5 mW TO18 red laser component from a low-cost laser module with a 300 mW TO18 IR laser diode (see figure 32). The new TO18 was successfully replaced, under carefully considering power requirements, heat dissipation, etc, but under the final assembling one of the legs broke off. This might actually have been very lucky as 300 mW is a *very* powerful laser and the IR emittance would have prevented eye reflexes from closing the eye in case of an exposure - in order words; the leg breaking of the TO18 could possibly have saved a person's vision.

Having realized this, the project continued using lasers, but in much more modest versions: A 5 mW line laser emitting 650 nm (red) waves was used instead. Having chosen the red laser, it was experienced that it was very difficult to come up with a low-cost detecting device for this. Most light dependent resistors (LDRs) were either responding to all light waves or only IR waves. After some time or searching, it was discovered that Vishay's TEPT4400 should be suitable for the red laser as it responds approximately to 90 % for the 650 nm wave. It responds to 100 % of the 532 nm waves (green), which was kept in mind for later iterations of docking procedure. Now that the laser and

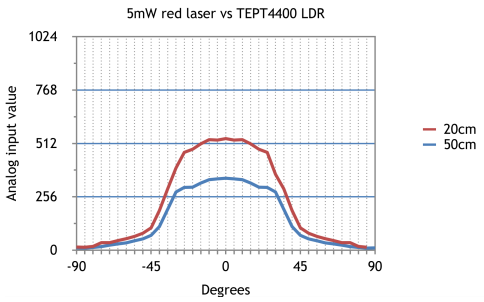



Figure 33: The analog value detected when a Vishay TEPT4400 light-dependent resistor was hit by a 5 mW 650 nm (red) laser at different angles. The TEPT4400 had a $2.7\,\mathrm{k}\Omega$ pull-down resistor attached.

the laser detector were chosen, these were installed on
the robot, where a standard BC547B transistor[43] controlled the laser and a $2.7\,\mathrm{k}\Omega$ pull-down resistor was installed for the TEPT4400 to avoid floating. The reception of the laser was tested from difference angles (see figure 33) and at a distance of 20 cm and 50 cm. According to the datasheet of the Vishay TEPT4400, the reception angle should be weaker in the center while the strongest respondance would be from angles of 10-15° - this is not what the test showed though: A quite good reception between -30° to 30° was experienced at a distances of 20 cm, while the detected values would be approximately halved at a distance of 50 cm. Having the laser more than 50 cm away, provided a barely detectable signal.

The threshold for the determining if the laser was detected on the other robot, was based on having the other robot measure the ambient light for 1 second. This value would then be added by 100 (from a 1024 scale) and set as the threshold.

Now that the robot was able to locate the direction another robot and already had the magnetometers implemented, it should be able to do the docking procedure. It was coded so that robot would move orthogonal 2 cm towards a collinear position of the other robot, as long as the two robots' heading had a difference of more than 5°, while the laser was detected. When the difference was less than 5° the robot would estimate itself to be close to collinear with the other robot and therefore slowly move towards the other robot, while oscillating around the TEPT4400. When a predefined distance was detected in front of the robot, it would stop, turn 180°, and back up to the other robot, while the shape of an inserted module would compensate for minor misalignments (please note that the modules are developed in chapter 9).

### 8.6.1 Evaluation

All communication issues were already dealt with in chapter 6. In general the robot handled the docking procedure quite well, while still having some significant shortcomings. The biggest shortcoming was the ability of detecting the laser as any distances over 50 cm wouldn't be detected, which was problematic when the start position was to the far left or far right, as this would increase the direct distance to the other robot. For these positions, the TEPT4400 should have a better reception according to the datasheet, contrary to the initial tests for this iteration. The start positions for the robot, were 40 cm right behind the other robot with orthogonal intervals of $\pm 10$ cm (see figure 34). Because of the increased distances for some of the trails, the robot was only able to approach the other robot 7 out of 9 times. It could partly be a result of the defined threshold that was based on the ambient light, but having a threshold of approx. 10% more than the ambient light seems reasonable to to avoid false positives and shouldn't be lowered.

[43]The standard transistor was BC547Bs, which can handle currents as high as 100 mA.

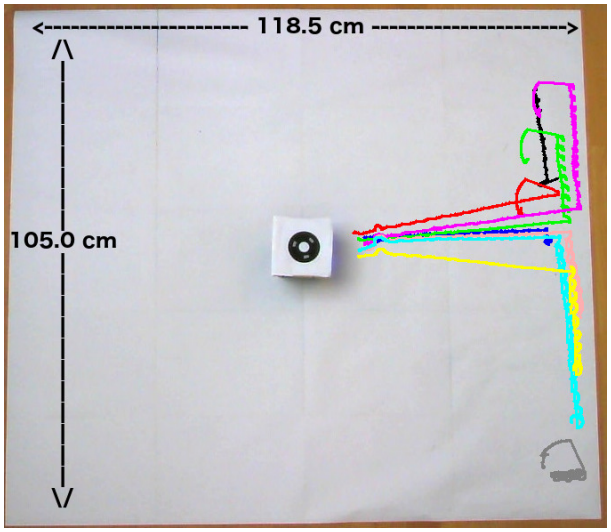



Figure 34: Trails from the robot (colored lines) where it attempted to dock with the other robot (in the center pointing left with a black marker). The first robot was positioned 40 cm behind the other and at 10 cm intervals to each side (North and South in picture). 7 out 9 attempts were able to dock, but it was noted that the laser was too weak to be detected over 50 cm.

For this initial evaluation, it was chosen not to have any modules inserted in either of the robots, as it was preferred to measure the sensors' and the laser's performance for the entire tests. Later iterations might evaluate with modules inserted to benefit from the modules' geometrical shapes.

One of the noted issues, was also the magnetometer from chapter 7 as this wasn't always consist in its readings. As it can be seen in figure 34, the estimated collinear position deviated 17 cm at a distance of 40 cm, which corresponds to $\pm 12.27°$[44] even though it was programmed to be considered collinear when the absolute difference of the robots' headings was less than 5°.

This inconsistency of the magnetometer is also one of the reasons why the robot didn't align complete after it had reached the other robot, turned 180° and back up: When the robot should turn 180°, the inconsistency between the magnetometers would have the robot not be collinear with the other robot, for which reasons it would often end up in an angle that doubtfully could allow a module to be transferred.

The ultrasonic distance sensor from chapter 6, worked very well and allowed the robot when it had reached the correct distance to the other robot, before turning the 180°.

The following list of goals from chapter 8.5 have been addressed in the initial design:

**D-1** Must be able to determine the exact direction of the other robot

**D-2** Must be able to reposition itself to a better position if its start position isn't in front of the docking mechanism

**D-5** Must not be more than 1 cm away from the other robot in a collinear distance, when docking is complete

**D-6** Must not crash into the other robot during the docking procedure

[44] The 12.27° is given from this equation: $\sin^{-1}(\frac{17\,\text{cm}}{2\times 40\,\text{cm}})$

## 8.7 1st iteration - Stronger laser

This iteration will focus on increasing the detectability of the laser, even for longer ranges, and if the inconsistencies of the magnetometers can be avoided.

First it was considered if the TEPT4400 and the laser could change places and have the robot back up to the other robot. The laser could then be used *all* the way to the other robot, instead of relying on the magnetometer for the last few centimeters, which causes the angled end position of the robot. The reason for not implementing this, was that the robot wouldn't know when to stop as the ultrasonic distance sensor was mounted in the front. Unfortunately, there wasn't enough room for an additional sensor in the back as most of the space already was used for the motors that would transfer the modules between the robots (described in chapter 9). Another point is that this project is trying to limit the amount of sensors that is used to not only decrease the complexity, but also make it easier for others to continue this project by keeping as much pins and space around the robot free. Instead the shape of the modules' ends were changed to be angled and benefit from this geometrical shape, which is described in details in chapter 9.8.

To address the distance issues that were experiences with the detection of the laser at "long" distances, multiple solutions were considered: First of all the laser was replaced by a much stronger one, that was able to emit a 50 mA laser beam which was 10 times stronger than the previous one. The new laser was also emitting waves in 532 nm (green), which is the ideal wavelength for the TEPT4400. The TEPT4400 only reacted 90% to the previous 650 nm (red) laser, while 100% to the new 532 nm waves. The stronger laser was measured to use 317 mA to generate the 50 mA beams, for which reason a total of 5 BC547B transistor were used to control it. The BC547B transistor can handle as much as 100 mA each, why 4 should seem to be enough, but if a single transistor fails, the rest would get burned, so an extra was added as a precaution. It was also considered to replace the Vishay TEPT4400 with the TEPT5600 as the specifications are the same with the exception of the TEPT5600 had a diameter of 5mm compared to the TEPT4400 which only had a diameter of 3 mm. If the 5mm TEPT5600 was used instead, the horizontal area where the laser could be detected might have been larger. It was also considered to install an additional two TEPT4400s nexx to the current one, which could either be connected in parallel to simply increase the area where the laser could be detected or they could be installed to different pins. If they were installed to different pins, the robot could know not only *if* the laser was detected but also if it was detected in the center or to one of the sides, and thereby know how much it should turn when approaching.

The author was also aware that the increased strength of the laser could easily cause eye damage for which reason a decent pair of safety goggles was used at *all* times! An extra 3D printed piece (see figure 35) was also added to limit the risk of the laser beam reaching eye height.

### 8.7.1 Evaluation

This iteration was on some points a bit more successful that the previous as the laser was detected a bit better but not nearly as well as expected. For this test, one of the robots would always carry a module as the shape of a module's end is used to not only guide a module from one robot to another, but also to guide the robots while docking.

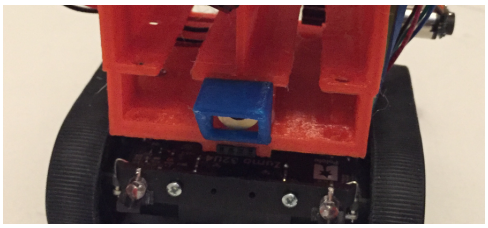


Figure 35: The strong laser gave risk of eye damage to persons without the proper eye protection. To decrease this risk, an extra piece (blue object) was added, so the height of the laser's vertical line was limited.

In order address the design goals D-3, the distance between robots were increased from 40 to 50 cm, while keeping the orthogonal intervals of $\pm 10$ cm (see figure 36). This seemed reasonable because of the 10 times stronger laser, but surprisingly the result was that the other robot occasionally didn't detect the laser, for the first part of the procedure where it would scan left and right to have the laser detected. When the laser wasn't detected, it had to try a few time until the other robot reported the a detection, and the robot could move a bit left or right to correct its position. The robot only moved 2 cm each time, which made the initial process take a few minutes. Potentially the 2 cm could be increased to speed it up, but this would be on the cost of the precision and the risk of the robot being trapped in an endless routine of moving xx cm to one side, only to discover that it was to far and it had to move back again.

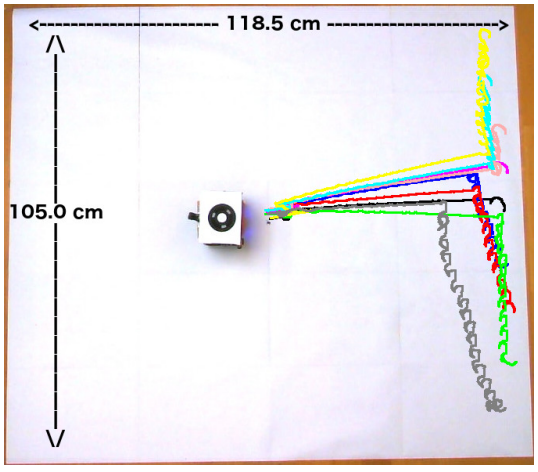



Figure 36: Trails (colored line) of the robot docking the other robot (to the left, pointing left, and with a black marker). The start positions was increased from 40 to 50 cm and the orthogonal distances was $\pm$ 10 cm.

When it was doing the part where it scanned left and right to have the laser detected, it would always scan to left first. This was even if the robot already had move a bit to the right, and it therefore knew that the other robot would most likely still be to the right. This seemed as a waste of time and could be addressed to decrease the overall time, for situations where the robot would start to the left of the other robot.

Surprisingly the laser wasn't detecting during oscillation part either - even half the way towards the other robot! This would result in the robot turning 360° in the search the TEPT4400 on the other robot. When it had done a full rotation the laser was often detected and the robot would continue towards the other robot and finish the docking.

Because of the strong laser there was also noted a drastic drop of potential runtime for the system: The stronger lasers seemed to drain the 4*1.5v AA batteries in about 45 minutes - so fast that the batteries would actually increase in temperature.

There seemed to be a lot of trouble around the new laser, since it didn't work as well as expected and also drained the batteries *that* fast. In the next iteration it was noted that the new laser wasn't installed correctly and only emitted a fraction of its capability.

Regarding improved shapes of the modules ends (described in chapter 9.8), these should have helped against having the robot end up in an angled position, which could lead to failures when transferring modules between the robots. During the tests of this iteration, the robot would dock correctly 5 out 9 times, while the remaining 4 would have a minor misalignment of about 3 mm and 15-20°. The 3 mm isn't much but the 15-20° could cause trouble for the reconfiguration the upcoming chapter 9.

Following is the list of design goals from chapter 8.5, which have been addressed or improved in this iteration:

**D-1** Must be able to determine the exact direction of the other robot

**D-2** Must be able to reposition itself to a better position if its start position isn't in front of the docking mechanism

**D-3** Must be able to dock from a distance of more than 40 cm and with an orthogonal offset of more than 30 cm

**D-4** Must not be more than 1 cm orthogonal misaligned with other robot, when docking is complete

**D-5** Must not be more than 1 cm away from the other robot in a collinear distance, when docking is complete

The mentioned design goals was only improved slightly, but most of these will be further improved in the next iteration, where it is discovered that the laser wasn't installed correctly and therefore barely emitted a beam.

## 8.8 2nd iteration - Minimize dependency of inconsistent magnetometers

This iteration will focus on increasing the detection of the laser and avoid relying on the magnetometer for the last few centimeters of the docking procedure. By avoiding the use of the magnetometer for the last part, the robots will hopefully not be angled when the docking is completed, but in full contact with each other.

First of all it was discovered that laser was incorrect installed and therefore only emitted a very weak beam. After the mistake was corrected there was an obvious improvement in the laser line!

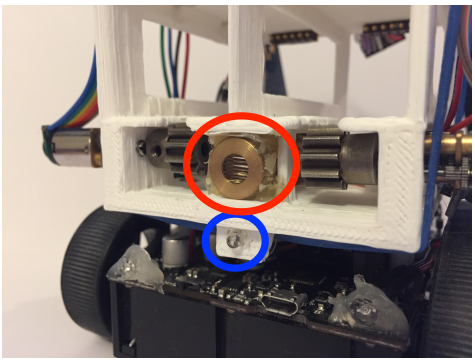


Figure 37: The rear of the robot, where the laser (red circle) and the TEPT4400 (blue circle) now are located.

As an attempt to avoid the use of the magnetometer when the robot is backing into the other robot and finishing the docking procedure, the laser was moved to the rear of the robot. The TEPT4400 was also kept in the rear of the robot, but was positioned under the platform, while the laser was mounted in between the motors to eject the modules (see figure 37). The first design of this didn't leave enough room for having a screwdriver to reach the motors for holding the motors in place, which was corrected as a part of this iteration.

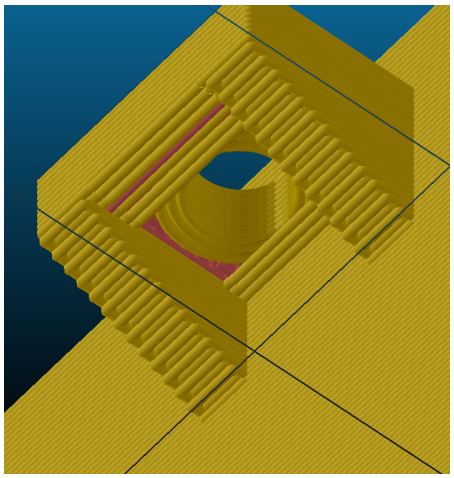


Figure 38: Screenshot from Slicer of the TEPT4400 mount seen from the bottom: Notice how a single layer is removed, so the circle's three circles can be extruded in the lowest layer. This minimizes the amount of hanging filament significantly!

With both having the laser and the TEPT4400 in the rear, meant that the robot now would have to back into the other robot. This idea was rejected in the previous iteration as the robot wouldn't be able to know when it had reached the other robot. To address this, it would take advantage of the vertical offset between the laser and the TEPT4400: The laser line was emitted horizontally with an angle of approx. $\pm 45°$ and because the TEPT4400 is 14 mm lower than the laser it wouldn't be detected at ranges lower than approx. 14 mm, which is less than the length of the end of the modules - the modules would therefore already be a bit inserted when the laser wouldn't be detectable. The code for the docking procedure would therefore have a timeout for the laser to be detected which was set to 2 seconds. Upon this timeout, the robot would be considered to be docked. The wires for the TEPT4400 would go through the hole in the bottom of the platform which was made in chapter 7.8 for the wires of the ultrasonic distance sensors in the sides.

The platform, at which the TEPT4400 was attached, was designed in SolidWorks and printed on a homemade 3D printer. A good tip for using 3D printers is to avoid support structures and, if possible, use minor ”hacks” to prevent some of the printed filament from hanging down and interfere with, i.e. an LDR as the TEPT4400: By removing 0.3 mm from the back of the mount for the TEPT440, the printer could print overhangs[45] first, before adding the 3 circles around the hole (see figure 38).

The mount for the TEPT4400 has also been moved 3 mm into the robot to prevent the TEPT4400 on each robot to collide and ”slide off”.

To address the issue about the angles as a result from the inconsistent magnetometers, there was considered more than to move the laser to the rear with the TEPT4400: Sensors always have a limitation to their precision and when it comes to docking, the sensors are often abandoned for the final steps and force or geometrical shapes are used instead. In this iteration it was very strongly considered to use the ”beak” design as shown in figure 39, which should help guiding the two robots to be collinear, even with a significant initial misalignment. From the side, the design has a lot in common with bird’s beak because of the slope. The slope was to allow the robots to dock while being in a rough terrain and the robots had different lateral angles. This design was rejected because it would interfere with the TEPT4400 which now was mounted below the platform.

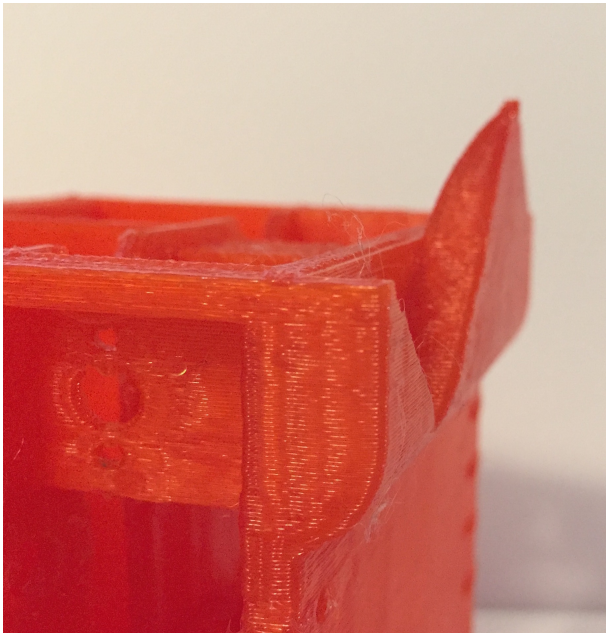


Figure 39: Image of the ”beak design” in an upright position. The tip in the back would guide the robot into the correct position by gliding into the counterpart (closest in image) on the other robot. Notice how the ”beak” is curved to prevent getting blocked because of a non-flat surface.

Another attempt get the robots to be fully aligned was far more basic as it simply would have the robot to ”wiggle” as the final step: When the robot was considered to be docked according to the previous iterations, it would now move left and right at high speeds to force the robots to be fully docked.

[45] Overhangs are when the 3D prints a thread of filament from one point of the structure to another, but with no previous layers to support this thread.

### 8.8.1 Evaluation

The 2nd iteration was a huge success as the laser was detected every time and all 9 tests ended with full docking that had no misalignments.

Now that the laser was implemented correctly it was detected every time. For this reason the tests, which can seen in figure 40 was conducted with the longest distances possible: The robots were now 1 meter from each other from center-to-center, still with the orthogonal interval of $\pm$ 10 cm, and the laser was *still* detected every time. It is likely that a weaker laser could have performed the same results in this scale, and thereby save some of the power required by the strong laser. Actually the laser was so strong that it interfered with the camera which was used to record the trails: About every 10th - 15th second, the camera would attempt to refocus, but without luck as it blurred image instead.

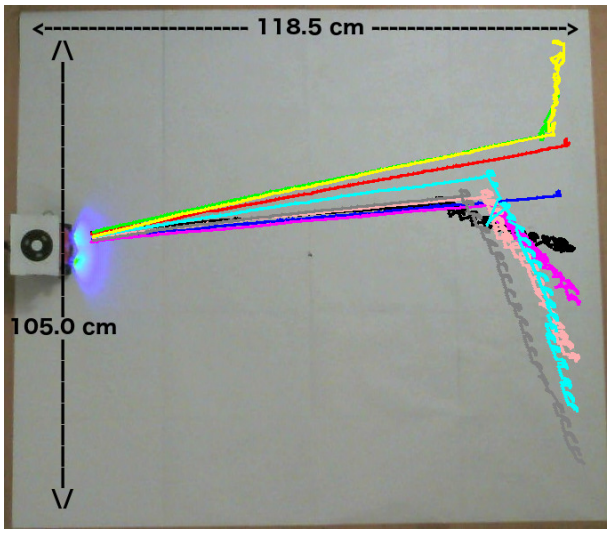



Figure 40: Trails (colored line) of the robot docking the other robot (to the left, pointing left and with a black marker). The start positions had a collinear distance of an entire 100 cm and the orthogonal distances was $\pm$ 10 cm. All attempts to dock succeeded very well!

When the part where the robot should oscillate around the TEPT4400, the backing introduced no new issues and the robot would stop after 2 seconds of inactivity from laser detector. This approach might still be prone to errors, but now the robots are able to dock without being misaligned - which also could be the result of ”wiggle” the robots into positions.

The ”wiggling part” worked very well, but seemed a bit brutal and lowering the amount of power for this could be advisable. Even though this part wouldn't start before the robots were almost docked and at least one side was in contact with the other robot, the docking robot would move the other a few centimeters.

The following list shows which of the design goals from chapter 7.5 that was addressed or improved in this iteration:

**D-1** Must be able to determine the exact direction of the other robot
**D-2** Must be able to reposition itself to a better position if its start position isn't in front of the docking mechanism
**D-3** Must be able to dock from a distance of more than 40 cm and with an orthogonal offset of more than 30 cm
**D-4** Must not be more than 1 cm orthogonal misaligned with other robot, when docking is complete
**D-5** Must not be more than 1 cm away from the other robot in a collinear distance, when docking is complete
**D-6** Must not crash into the other robot during the docking procedure
**D-7** Must be able to detect, when the docking has completed
**D-8** Must be able to successfully dock more than 80% of the times
**D-9** Must be in full contact with the other robot when docking procedure is completed

For this iteration all of the required goals were addressed or improve.

## 8.9 Discussion

Throughout the iterations in this chapter the robot's docking capability has been drastically improved from *often* being able to dock at a collinear distance 40 cm, but with an questionable end position, to docking from 100 cm and with an almost perfect alignment.

Some of the issues that were experience, were often related to the laser, which ended up being a very powerful laser - so powerful that the camera used for documentation attempted to refocus all the time. It hasn't been tested how far away the laser is able to trigger the receiver (TEPT4400) on the other robot, but the impression is that the power of the laser could be lowered to about half of the current power level without decreasing the performance. This would not only make the batteries last longer, which is highly preferable for mobile robots, but also lower the risk of eye damages. During the final iterations, it was noted that the batteries would actually increase in temperature, and only lasted for approx. 45 minutes. Because the laser is controlled with transistors and the pins used from the Teensy 3.2 support pulse width modulation (PWM), the laser's power level could be controlled by this, which should be one of the first things to address for any upcoming iterations. Technically, this will turn the laser on and off in very high cycles, which in theory could cause the laser beam to be turned off just as it passes the receiver, but for this to happen the robot would have to be very far away or move extremely fast. Another thing that could decrease the power consumption is to turn on the laser only when it is used: It is currently turned on for the entire docking procedure and will therefore waste a lot of precious energy, making fancy disco show, instead of conserving the power and make the most out of it.

Another aspect is that if this robot system is to work near people, like in an industrial hall, using a too strong laser could force the workers to wear eye protection, which some may find limiting when executing their work tasks. If the power of the laser was lowered (not with PWM, but actually lowered) and optics made the laser line only to be 10 cm tall at all distances, it is getting closer to could work in an environment with people. But if there is no people next to the robot, it doesn't matter: An underground mine could use this system and in the rare case that a human should enter, all robots could continue to work, but each stop if it was about to start the docking procedure.

The low-cost magnetometer installed on the robot was also experienced to generate some trouble, because inconsistencies between each of the magnetometers. Therefore the use of this was limited as much as possible: Initially it was used for the very last steps of the docking, where the robot would back up to the other robot, but now it is only used for reposition the robot to be collinear within 5° of the other robot. A way to further decrease the risk related to the magnetometer could be to let the robot reposition itself as long as it is only moving one of the sides. When it would detect that it should move in the opposite direction, it means that it has just reached the center.

The part where the robot is repositioning itself to be collinear with the other robot can also be very time consuming as it only moves 2 cm for every cycle. This doesn't seem as much but at a distance of 40 cm it is approx. 2.89°, which seems reasonable when searching for the center. But it is not just the small steps that increase the time for this part of the docking: Having to scan for the other robot for every step is what really takes up a lot of time. Not only does it always start with a scan to the left, even if the other robot was detected to the right the last time, but it also does this in a slow speed to increase the chance of the laser being detected on the other robot. A solution to this would require a laser detector like the TEPT4400 on one of the sides of the robot. The robot could then turn this side towards the other robot while following the orthogonal line until the laser from the other robot is detected, at which point it would know that it is in the correct for docking.

In the last iteration, the "wiggling" of the robot was added, which showed very good results even though the power level might be lowered and avoid the robot moving the other robot a few centimeters. But it helped significantly to have the robot be in full contact with each other and get them aligned within of 1 mm, which is far better than required from the design goals, which stated 1 cm.

The robot has addressed all but one of the design goals: In its current state, the robot cannot detect if it ac*actually* has docked, which can be quite problematic, if the robot is to perform a task that depends on this. A way to address this, could be to add a metal plate in each side of the robot's back: The robot could send a small signal in one of them, while listening for a similar signal from the other. If the robots were fully docked, they would both be able to detect the signal in one of the plates while providing the signal to the other plate.

If it was able to detect when a docking wasn't successful, it could drive 40-50 cm forward and attempt the docking procedure again. This would have increased the reliability of the system, knowing that it would attempt to dock until it was successfully done.

## 8.10 Conclusion

As it can be seen from the evaluation of the last iteration of this chapter, the robot is absolutely capable of docking with another similar robot. It does this with good precision and final sequence of motions, where it moves left and right, helps getting the robot fully docked well within the stated design goals

One of the required design goals hasn't been addressed, which states that the robot should be able to detect if the docking was succeeded. Even though wasn't achieved, the robot was able to dock in 100% of the tests.

The achieved design goals from chapter 8.5 have been listed here:

**D-1** Robot is able to determine the exact direction of the other robot
**D-2** Robot is able to reposition itself to a better position if its start position isn't in front of the docking mechanism
**D-3** Robot is able to dock from a distance of more than 40 cm and with an orthogonal offset of more than 30 cm
**D-4** Robot is not more than 1 cm orthogonal misaligned with other robot, when docking is complete
**D-5** Robot is not more than 1 cm away from the other robot in a collinear distance, when docking is complete
**D-6** Robot does not crash into the other robot during the docking procedure
**D-8** Robot is able to successfully dock more than 80% of the times
**D-9** Robot is in full contact with the other robot when docking procedure is completed

Now that the robot is able to dock with another robot, it is finally ready to start the reconfiguration, which is the core topic of this project and described in the next chapter.

# 9 Reconfiguration

This chapter will focus on the creation of modules for the robots and how these can be exchanged between robots. The modules may carry tools, sensors, batteries, etc. and is matured over multiple iterations, along with the robots and the platform which holds the modules. After each iteration the improvements are tested and documented.

As a part of the reconfiguration, the development of a service station to handle a robots' requests for reconfiguration will also take place. This service station will be capable of holding meta-data about the robots and choose which other robot that should assist a robot with a reconfiguration.

After the last iteration, a total of four modules, a service station, and a reconfiguration platform, have been developed and well tested. All of these are capable of working together and make it out for a reconfigurable system.

## 9.1 Scenario

When reconfiguration is being used in a system, like for the robots in this project, it is built up from multiple minor parts: First of all the robot needs a reason for starting the procedure of reconfiguration. Then the parts that are going to be used for the reconfiguration need to be at the same position as the robot, before the actual exchange of these can take place. Potentially, a part from the robot is then brought back for a central point in the overall system, which may be a service station, garage, or similar, where it can be stored/repaired.

The reason for the robot to a have the reconfiguration take place can come from three situations: 1) Either it foresees that it needs the reconfiguration in order to continue working on its current task, 2) it is being told to start a new task that requires for a reconfiguration, or 3) one of its current tools has failed and needs to be replaced (see chapter 10 for error detection). Any of this situations can trigger the need for a reconfiguration, which may be requested by the robot itself or a central part of the entire system, that coordinate everything. Which of the three mentioned situations and who that requests the reconfiguration all depends on how the system is put together and where the responsibility it placed: Some systems might be built where the central unit takes care of it and knows everything, while some see the advantage of having a decentralized system, that can to some degree continue to work without the central unit.

After the need of a reconfiguration has been triggered, the part which will be inserted to the robot and the robot itself need to be in the same position. If the robot is very expensive when it is not running, the best solution would most likely be to have the part brought to the robot, and the robot might keep on working if the reconfiguration is not triggered by a failed tool. Another approach is to have the robot to pick up the tool itself at the parts location, that makes the robot more independent, which in general is preferable.

When the robot and the part are at the same location, and the robot is about to have the tool added, they will first need to be positioned correctly according to each other, since a robot seldom is able to pick and install a tool on itself - unless it is in a very precise position according to the robot itself. When the robot and the part is getting in the right positions, they will first need to known exactly where one or the other is, before one of them can approach and dock into the other. All these situations about positioning, where the tool is carried on another robot, is discussed and addressed in chapter 6, 7, and 8.

Now that the robot is ready to have the tool installed, it can have the module transferred in an unlimited amount of ways, but some of the concepts of these could be to base this on friction instead of using gears: Be using friction, the robot decreases the risk of breaking anything during the transfer to almost non-existing. On the other hand, it might be difficult to tell if the tool is stuck half-way into the transferral.

No-matter, which of these methods that are used, the robot and the tool have a design that allow for minor misalignments. The positioning from the sensors, can only be precise down to a certain degree. Therefore the geometrical designs of the robot and the tool must be able to adjust for these misalignments - A typical example would be a cone which is inserted into a cylinder, where the shape of the cone helps it get into a centered position as it is fully inserted.

When the tools are transferred and the robot has checked that it is working, an potential failed module can be transferred the other way or the robot can simply go back to work on it job task. If a failed tool is given to the robot that just brought the working module, this robot could now return the tool to a service station, where it can be repaired and used for a future tasks.

## 9.2 Related work

The amount of related literature from this area is very limited: Most research projects go all the way into modularity and creates system where multiple small modules can be transformed into a single organism. The area between such modularity and non-modularity is rarely touched; topics where a robot is build as a case for +1 modules.

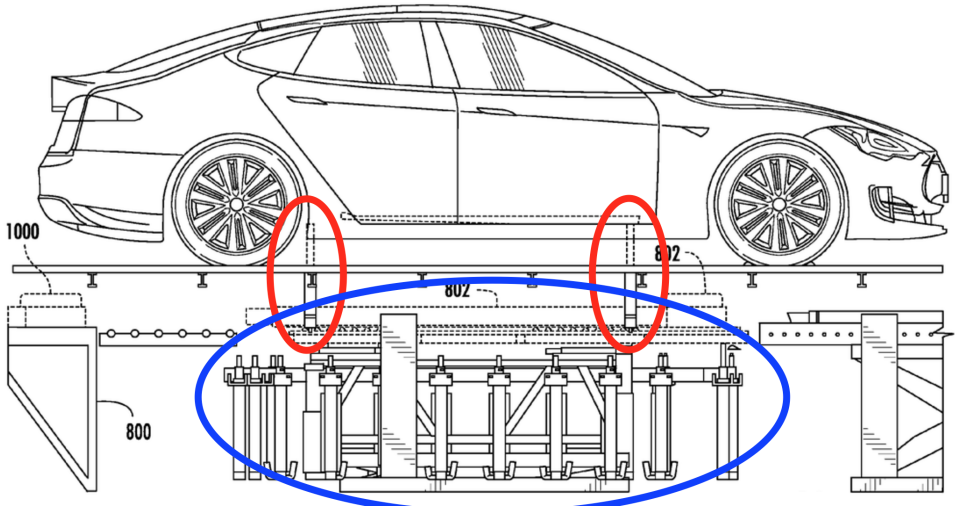



Figure 41: Patent drawing of Tesla's battery swap system. The lift (red ellipses) will raise the car, before the tools (blue ellipse) will be raised to swap the batteries.

One of the best relatable projects, which are being used today, is when the car company Tesla Motors© allows its customers to have the battery in their car replaced by one of the autonomous battery swap stations[43]: The car owner simply drives onto a small platform, where the car is lifted from the ground, before tools appear from the floor of the platform (see figure 41). These tools will then remove the battery of the car and replace it with a new, recharged one within minutes.

Another project[5] (see figure 42) has attempted to create heterogeneous robots for Mars expeditions, which basically was a very small fork lift that could add/remove modules on another robot. This was done with the computer vision (CV), where the robot would approach the other robot from the side based on the fiducial marker the other robot had on its side.

A bachelor project[26] from the IT University of Copenhagen, used a Lego robot for a reconfigurable system: The robot was set to follow a line in the floor to a service station, where the sensor for line detection could be replaced. After the replacement, the robot was still able to follow the line, which implies that the replacement was successful.

A robot that partly can be related to the topic of this chapter, is a robot arm that are used, i.e. advanced CNC milling machines. The robot arm in these CNC machines are able to change the tool head by itself t o optimize the time it takes for the machine to mill an object: The CNC machine often starts with the largest drill heads and mills all the large parts, before it

automatically replaces the drill head with a smaller one (see figure 43), which can mill the smaller parts. In this way, the time is not only minimized, but all the fine details of the object is kept.

Besides the Tesla's battery swap system and the tool-changing robot arms, most products that are built today are *one non-modular* unit: All the items within a product are firmly put into place in a way that makes it difficult to change or repair these by the customers - often the consumer will have to chose between buying a completely new product or spending a lot of time/money having the part changed by the production company.

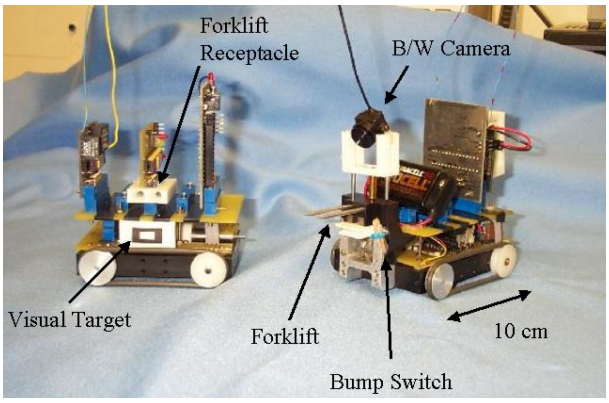



Figure 42: Heterogeneous robots by Bererton, et. al., which reminds of small forklifts. They were able to add/remove modules on each other based on primary computer vision.

The fully modular robots address this issue, but also introduce new issues: As the area is still new, a lot the research in this fields tries to find a good design for the modules, but also the software for these as complexity is a significant factor in this topic. So far some of the most successful are ATHLETE[95], ATRON[9, 13, 14, 15, 16], CKbot[98], CONRO[83], JL-I[101] and JL-2[92], MoleCube[102], M-TRAN[32], Sambot[93, 94], SuperBot[78], SYMBRION[51] and X-Cell[27].

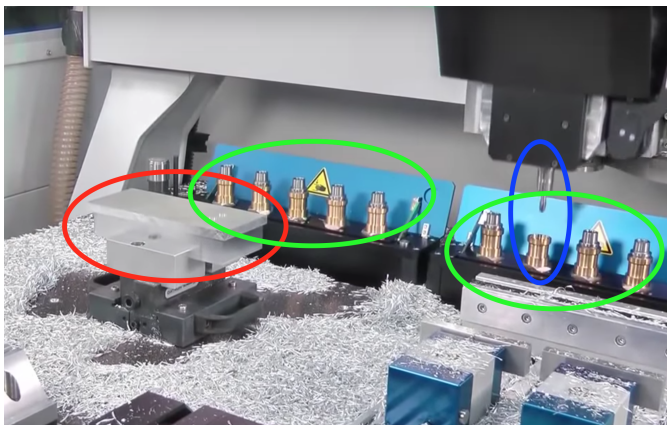


Figure 43: CNC milling machine, which can automatically change drill heads. Red circle is the material to milling into, the blue circle is the current drill head, which is being inserted into storage position, and the green circles are storage positions for the currently unused drill heads.

To describe one of these, please look at figure 45 which shows the ATRON module: This module is capable of connecting with other similar modules and can rotate the upper half part around its own axis. This module has shown to be very versatile, which can be seen in figure 44, where a group of modules have assembled themselves to function as a car (right) and others are almost done assembling themselves to function as a snake (left). The modules are able to connect to each other with a set of small claws that fits into the counterpart on another module.

Each type of the mentioned modules can physically connect to another by either using motor controlled connectors (see figure 45 of an ATRON module) or electromagnetism (see figure 46 of a MoleCube): First it gets in the correct position by aligning itself with the other module. This can i.e. be done with IR[27]. When the modules are aligned, it will activate the motor for the connectors or activate an electrical magnet which neutralizes/weakens the actual magnet's field[27, 102].

Figure 44: ATRON modules, where a group has assembled themselves to work as a car (right model) and others are about to have assembled themselves to work as a snake (left).

But most modules use connectors in terms of claws/arms, which ensures that the modules stay together - unless a strong force tears the modules apart and breaks the modules. Those using magnets won't suffer from such catastrophes as the magnets will simply get separated. As the system detects this, they will attempt to re-create the structure. A benefit of using magnets is also that they dock easier, because the magnets will take care of any imprecisions by automatically aligning the magnets.

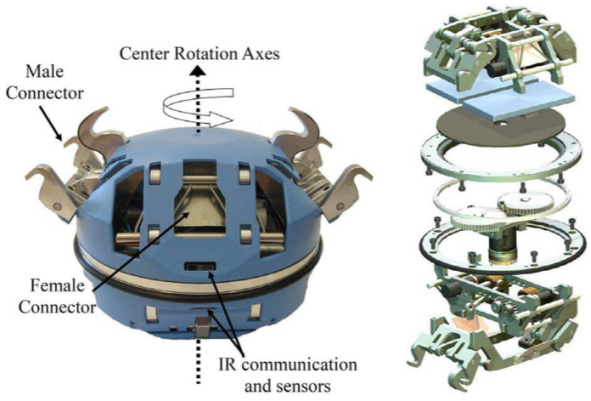



Figure 45: ATRON module: Note how the male connectors may create a fixed joint between two modules.

An important part of these fully modular systems is also how they construct themselves[19, 33, 44, 61, 82] or potentially repair the structure they belong to[2, 77, 98]. How a structure is made can be extremely complex, but Stoy[82] comes with a suggestion, where the path of each module is determined based on a CAD module of the structure and the position of the first module.

Many of the modules use IR, when they need to position themselves next to another module before interlocking. An example of this is the X-Cell modules[27], where each side has an IR emitter to the left and an IR receiver to the right. As the signal gets strong, a module will know it is facing more directly the other module and visa versa. When the physical connection is created, most of the mentioned

system use an I2C protocol for communicating between all of the connected modules. This protocol has the advantage of being able to address the intended receiver of the data and often supports $2^7$ or $2^8$ addresses on a single channel. The protocol is also so common that most micro controllers support this.

As the area is still maturing, several publications[24, 34, 35, 64, 65, 97, 99] try to compare the proposed modules in terms of max modules, price per module, heterogeneity/homogeneity. But as this type of modularity isn't exactly in the same category as the modularity which is proposed in this project, it isn't elaborated on.

Figure 46: MoleCube by Zykov et al. Note the magnet in the center, which may be neutralized/weakened by an electromagnet to when a joint must be broken.

## 9.3 Design considerations

This chapter will describe the design considerations going towards the general demands of a reconfiguration. To provide a better overview for the reader, it has been divided into sections focusing on a single area at a time.

**Modules**

The modules are one of the core parts of this project, as these are what makes the robot reconfigurable by having a generic shape and an electrical interface. They will contain important objects like tools, sensors, battery packs, etc. One could also imagine that the modules could carry supplies to people in need of these, if the robots had the size of a car. The types of supplies could vary a lot - just like those that are being transported by trucks in most countries today. Examples of such types of supplies are food, compressed gases, fuel, oil, water, etc The ideas are endless, as modules might have a specially designed container inside to store the supplies in a safe manner. For those modules that carry a device that can be activated, like a tool or sensors, the activation of the module should correspond to how it is inserted: As the module would have to be usable for all robots and these may transfer it between each other, the modules should be usable from both ends if the direction of which a module is inserted can be both front and back. But it shouldn't activate both ends when a robot is using it: Depending on the system design, only the front or the back of the modules should run, as i.e. it wouldn't make much sense and actually be dangerous to have a drilling module spin in both ends at the same time. A person or robot passing behind the robot could be hit by the driller and be severely damaged. Another point is that when these modules preferable should be hermetically sealed to prevent any dust, dirt, or bacteria to enter the module. Such contamination could compromise anything inside of the module whether it might be equipment and/or supplies.

**Transferring the modules**

How the modules should be transfered from one robot to another is likely to be the most difficult part in reconfiguration. Not only should the modules be able to get from one robot to another, but they must also do this in a way that tries to ensure a very specific end position so that the modules and the robot are properly connected to each other. If the modules aren't reliably connected to the robot, the modules can't be controlled and used by the robot, in which case they are nothing more than extra weight to the robot. To avoid such a situation, the use of carefully designed geometrical shapes may be used; just like a cone, that is turned upside-down, would self-center when it is being lowered into a hole. Such methods are often used as sensors only will be able to measure down to a certain precision. After this the shape of the modules could take over, but sometimes it is also possible to have a strong physical force that can ensure the correct positioning, which could come from a strong motor or servo.

But before the final precision is achieved, the modules needs to be moved from one robot to another. How this should be done, truly depends on the shape of a module and its required end orientation. Some approaches have the benefit of not only moving a module, but at the same time also lower the potential friction: An example of this is if the module is in an tube and air is blow into one end of the tube. The air pressure would not only push the module in the correct direction, but also get in between the module and the walls of the tube, acting as a kind of pillow and significantly decrease the friction. Another way is also to use manipulators without gears as i.e. a wheel with a rubber tire, which clearly could be used to transfer a module, while having a gently and non-damaging grip of the module. The disadvantage of these types of approaches is that the robot would need extra sensors to be aware of whether the module actually is moving or not, and how far the module has got. If manipulators with some kind of gear is used, this would not be an issue as the robot would have an extremely good grip. But on the other hand, gears would try to force the module forwards until one of two things happen: 1) The manipulator stalls or 2) something breaks - neither of these situations are worth striving for. In Appendix A are described, in text and drawings, five ways that potentially can be used to transfer the modules.

What kind of motors to use for the transferring can be hard to say, as these come in all kinds of types, shapes, sizes, and forces. Considering that the transferring might will be done in a straight line, and linear actuator might be suitable. This could be a motor that has a linear movable piston, which could push/pull the module back or forth. These are often quite expensive, even for smaller ones, but might be useful. Another commonly used motor is the servo, which basically is a geared DC motor with an potentiometer that tells the knob's orientation. Because of the gearing the servo can be very powerful even though the casing is small. Servos are semi-expensive and cost approx. +10 USD. One might also consider to use a simple geared motor, which, similar to the servo, can have a very small size, but because of the gearing it may have a lot of force. Sometimes it is possible to add an encoder to such a motor, which makes is possible to tell how much the motor has turned and if it is currently turning. No-matter which type of motor that is going to be used, it should preferable be driven by a DC power source, as the robot preferably will be mobile and the use of a battery seems like a good power source.

**Service station**

When a reconfiguration process is needed, there could be a central unit to hold the information about available robots and which modules each of these carry. Another possibility is to have each robot being able to broadcast a request to every other robot, where they would respond if they could be able to assist, but this would most likely produce a lot of overhead for the request, contrary the approach with the service station.

The responsibility of a service station should not only be to hold information about which robot that carries what modules, but it should also be able to choose the best candidate for a reconfiguration request and supply that robot with the needed information. Suggestions to such information would be: The best route to the other robot, the type of module to exchange, which slot the module is in and if the reconfiguration request is caused by a failure, which means that the failed module should be returned to a repair center.

## 9.4 Implementation considerations

This chapter will describe the implementation considerations of the reconfiguration ability with respect to the described scenario in chapter 9.1. The chapter has been split into smaller topics to give the reader a better overview.

**Construction materials**

Regarding the materials used for the platform, it really depends on a lot of criteria, i.e. which temperature and humidity it should work in, the production time, production costs, the smoothness and friction of the material, and a lot more. A lot of high end productions tools are available for shaping 3D objects from materials like metal, wood, plastic, etc. But as this project attempt to address people and organization with a modest budget, only solutions with a low production cost will be discussed. This is to focus on having the proposed system available to as many people would wide as possible, while highly competitive companies could look into further optimization of the materials by themselves.

Popular machinery for custom designs often include 3D printers and laser cutters. The 3D printers allow of a highly customized 3D object to be printed in a matter of hours, while the laser cutter can do this for a 2D object in a matter of minutes. The 3D printers are still being developed and constantly tweaked to allow for better final results. The 3D printers currently available are mostly designed for plastic or resin materials, but other exotic filament types have also been seen: Some restaurants use it for printing chocolate figures for their desserts (see figure 47), artists have used it as a clay printer, and some have even made a pancake printer. Any of the filament types work by laying a very thin line of the material in one layer, after which it will increase the distance and print the next layer. The Achilles heel of this approach is the balance between having the material soft enough to be printed, but also having it to harden before any deformations occur. To address this problem, some 3D printers in the industrial class have been designed to use powder instead: Laying out a very thin layer of powder and having a laser to melt or activate resins molecules within the powder for each layer, will have the object to constantly be supported and avoid any deformations. Examples of such printer have been used by the US Army for printing spare parts for the vehicles, but such printers are very costly and are therefore not included as a potential option. Using resin for 3D printers are also used for pure resin printers:

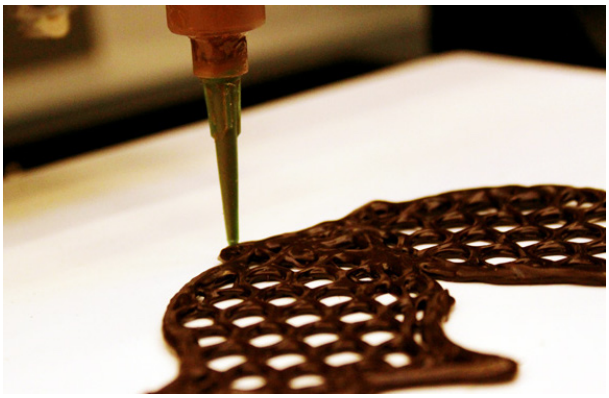


Figure 47: 3D chocolate printer in the middle of a printing two eatable bells for a dessert.

Liquid resin is put into a container and a laser is activating the resin closest to the platform and an object will slowly be pulled up from the resin container as the layers of hardened resin create the object. This, too, takes some hours to print, but the structure is very strong, though brittle.

The commonly used 3D printers are, as mentioned, most of the time used with plastic materials. These are often ABS, which creates toxic fumes when heated, or PLA, which is non-toxic[46]. ABS is what LEGO® bricks are made of and has the advantage of being very strong, but very rigid. The contours in an ABS printed object can be smoothened by exposing the object with acetone fumes for a short amount of time: The fumes will try to dissolve the ABS and slowly liquefies the surface, which then will smoothen the contours. PLA will not react to common chemicals in this way, but has the advantage of being a bit more flexible than ABS: Bending PLA will gradually change its color to white, as a sign of wear, before it finally breaks. The cost of ABS and PLA is almost the same, but PLA is the most wide used of these two because of not creating toxic fumes when heated.

Because the 3D printer operate in layers, a slope may consist of several small ”steps”, which reminds of a stair way. If two slopes are meant to slide against each other, one of them may be printed so these ”steps” occur. But then the other object should be printed on the side, which will prevent *both* of the slopes to have these ”steps” and make them slide very smoothly.

If a laser cutter is used, it will significantly decrease the product time, but add to the complexity of the model if it is more than just a flat sheet. This is a result of the laser cutter only being able to cut in 2D, which means that a design, with interlocking parts, might need shapes to look into, if spending a long time gluing doesn't sound tempting. Laser cutters can cut into materials like wood, carton, POM, acrylic, etc. As the robot most likely is going to have a design where material will rub against each other, wood, carton, and similar should be avoided because of the friction. Instead materials like acrylic, POM, etc. could be a way to go. These types of plastic comes with a lot of different characteristics, which should be considered thoroughly before starting to entire armies of reconfigurable robots. An example of this is the acrylic, which a low-cost material and can be acquired from most local hardware stores. But one of its characteristics is that it is very brittle, which can cause the model to break if, i.e. a screw is tightened too much. POM has a lot of acrylic's characteristics, but isn't brittle. It's still a low-cost material and is in general good for design of robots as its tolerance for minor design imperfections is good.

[46]PLA is made from sugar and *should* be eatable. The author hasn't tested this, as it is unknown how eatable the color in the PLA is!

A material often being used for robots and robot-like devices is metal. Metal is a good strong material, but requires the right tools (often professional) when shaping it. Depending on the type of metal, it can withstand most environments, whether it being the ocean, forest, or outer space. One of metal's downsides in the robotic world, is its ability to conduct electricity and cause short circuits. That being said it is also one of metals upsides as wires, PCB[47], and other technical equipment benefits from copper's high conductive characteristic.

**Communication protocol**

The interface is also worth considering, even though there might not be as many modules attached at one time as with some of the modular robotics described in the related work section. But it is highly likely that there will be several kinds of modules and there might also be, say, 1 to 10 modules attached at the same time, which requires a connection that supports multiple units at the same time.

Some of the most common communication protocols for micro controllers are $I^2C$, Serial, and SPI. But this also strongly implies that each module should hold a micro controller, which might not be the case. Instead a more direct way to control and could be used by simply having a generic interface in terms of having one pin for each action like, i.e. and on/off, speed, state, etc. This would save the trouble of using a micro controller as middleman, but potentially limit the interaction with the modules as the interface must be more or less generic. A way to address this could be to define the control pins depending on the type of module: When, i.e. a battery module is loaded, a pin would be for measuring the power left in the battery, while if it is a motor module it would control the speed of the motor. This would require for dynamically changing the pins between inputs or outputs, which is perfectly possible for many micro controllers - including the Teensy, which is used in this project.

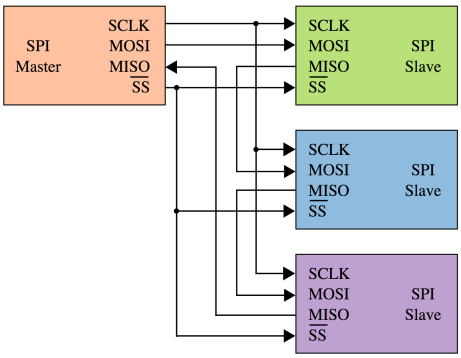



Figure 48: Units using the SPI protocol set up in a daisy chain: When a slave receives a message, it will automatically be cloned to the outgoing port, forwarding it to the next slave, and so on.

If the direct way would be used, and avoid the middleman micro controller, it must be decided which protocol to use. Among the common protocols the $I^2C$ seems best suitable as this can communicate with a large number of unit using only 2 + 2 wires[48]. It works by having one unit being the master, while the rest is slaves, where each slave has an address based on

[47] PCB is short for Printed Circuit Board, which can be found in most electrical devices.

[48] Wires needed for the $I^2C$ protocol is four: One for each of voltage, ground, signal clock and signal data.

7 or 8 bits, which allows a maximum of $2^7 = 128$ or $2^8 = 256$ slaves + one master. Another option is the SPI [49], which can be connected in multiple ways, where the one with fewest wires requires 2 + 4 for each device[50]. The less complex way to setup it up is by having one slave select pin for each slave, which would increase by the amount of slave devices. Otherwise the slaves can be daisy chained and forward the incoming message to each other, which only requires one slave select (see figure 48). The reason for this is that the SPI protocol is designed to send a copy of the incoming message to its outgoing port - by having all the slaves' slave select pin raised at the same time, they would forward the message to each other and thereby distribute the messages.

An alternative is Serial protocol, which requires 2 + 2 pins[51], but lacks the ability to address multiple units: The transmit and receive pins are connected to the other unit's corresponding pins and any communication/noise on any of these will be interpreted as data, regardless of how many that are listening. This means that one *could* have multiple receiving units, which could interpret incoming data, but the central unit wouldn't be able to known who sends any incoming data.

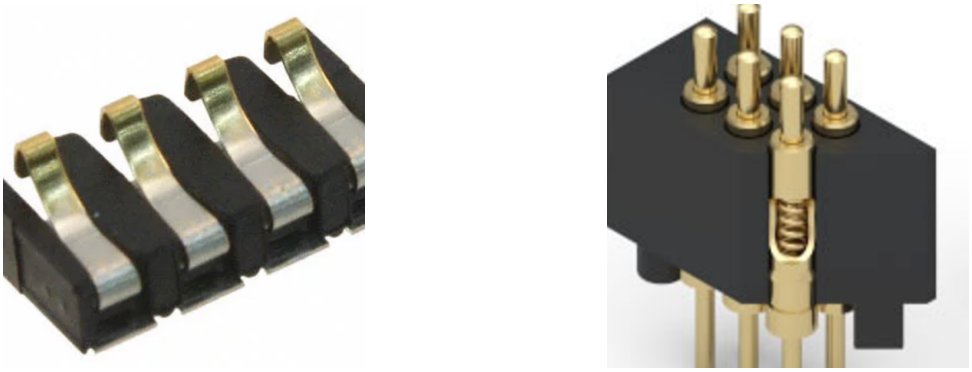


Figure 49: Two types of spring loaded connectors (SLC): To the left is the spring metal bent to allow for the spring function, while the one to the right has a built-in spring.

**Connection**

Depending on which communication protocol that is used, there will have to be established a physical connection between the modules and the robot. In case the protocol is going to be some of the previously mentioned, it would most likely be a physical connection contrary to wireless technologies as bluetooth, RFID, or NFC. The wireless technologies would for obvious reasons require at least an antenna in an most cases a micro controller for handling data. As this project most likely not will go for a wireless communication for the modules, this is only mentioned

[49] SPI is short for Serial Peripheral Interface and theoretically allows for endless of devices to be connected.

[50] SPI requires a total of 6 wires: Voltage, ground, master in/slave out, slave out/master in, clock and slave select.

[51] The Serial protocol needs a pin for each of: Voltage, ground, transmit and receive.

as an opportunity for others to pursue.

To establish a physical dynamic connection between two circuits, there are not that many options to choose from. A possibility is to use normal pin header and try to make a design, where the pin are guided carefully into the others, which can be quite a hassle because of only little tolerances. An other and more used option is to use spring loaded connectors (SLC) which also fall under the pin header group. There exists differing variations of these (see figure 49), but both are with the characteristic of having the males being able to apply a minor force towards the female pin header. One of them is made by using spring metal that are bent and the other is where the males have a spring pushing forth the pin a few millimeters. This can be beneficial to increase the tolerance of the distance between each of the pin headers. The female pin headers can also be acquired with a slightly concave design and thereby guide the male pin header to a better, centered position, increasing the likelihood of having a good connection for all of the pin headers.

## 9.5 Design goals

Following is a list of design goals, which should be achieved if the reconfiguration are to be a success. These design goals are based on the previously described scenario from chapter 9.1 and therefore also split into design goals for the robot and for the service station.

At the end of the lists are some extreme goals, which are not vital for the reconfiguration procedure, but are features that will improve the use of the system. These goals will have a *X* in the name.

**Design goals - Robot**

**R-1** Must be able to activate a module's function
**R-2** Must be able to deactivate a module's function
**R-3** Must be able to request new module from service station, specifying module type and slot for insertion
**R-4** Must be able to handle request about reconfiguration
**R-5** Must be able to inform other robot about start of module ejection
**R-6** Must be able to detect when module is fully inserted
**R-7** Must be able to notify other robot module is fully inserted
**R-8** Must be able to autonomously detect module type
**R-9** Must have a tool module
**R-10** Must have a sensor module
**R-11** Must have a battery module
**R-12** Must be able to receive a module even if the other robot has an orthogonal offset of up to 8 mm and an axial offset of up to 15 mm

**RX-1** Has several tool modules
**RX-2** Has several sensor modules
**RX-3** Can drive solely by using battery module
**RX-4** Can foresee when a tool is needed
**RX-5** Only the end in the front of the robot is activated, when module is used

**Design goals - Service station**

**RS-1** Must be able to accept incoming meta-information about robots (robot ID, carried modules, etc)

**RS-2** Must hold meta-data about all available robots

**RS-3** Must be able to choose robot, that can help other robot with reconfiguration

**RSX-1** Has a user interface that shows the location of each robot

**RSX-2** Has a user interface that shows the status of each robot

**RSX-3** Has data persistence that can be restored after i.e. power failure has occurred

As any of these goals get addressed or improved, they will be mentioned under *Evaluation* for each iteration.

## 9.6 Initial design - Designs and components for transferring modules

This chapter will describe how the first steps of the reconfiguration and have focus on primary the geometrical design of the modules and the platform, but also the main components for transferring the modules. As the chapter addressed a large number of issues, it has been split into areas of: Platform design and Modules.

**Platform design**

The design of the platform has required lots of thoughts as transferring an object between two robots can lead to a million problems. First of all the ability to transfer the module between the robots was to be addressed, but preferably the platform should also fit onto a Zumo32U4 as this appeared as a nice foundation for a mobile platform. In appendix A are showed some of the ways a module could be exchange between robots. The described ways include how a module could be grabbed by the end of a linear actuator and lifted to the other robot, which was rejected because of two things: First of all it would be a rather long actuator if it should be able to push the module all the way from the approx. 10 cm long robot onto the other, while also have enough space in the end of the robot for the mounting. Another problem would also be how it should grab and lift the module: A hook could be used for the lifting, but could get stuck when the module was transferred and the lifting would apply a vertical force to the actuator, which isn't really meant for forced in the direction. An idea to solve this was to mount the linear actuator beneath the module and let it push forth the module over several extensions and retraction. The actuator would grip into a gear rack mounted under the module when it was extended, but loose the grip when retracted. The weight of the module itself should have been enough to ensure a reliable grip. The reason for rejecting this idea is simply because it appears as a one-way solution: The rack under the module would be designed to only be grabbed when being pushed in one direction, making it impossible to eject it by the next robot.

Ideas of using rubber wheels seemed as a really good idea (see middle picture of appendix A) as the modules could be inserted a bit off, but be pulled into the right position by the wheels. Unfortunately, it wouldn't be possible to know when the modules where fully inserted as the rubber wheels potentially might keep spinning after the module was in place. Other ideas included tipping it over, which would require a lot of space over the robot, while being sensitive to a good positioning of the robots before doing this. In the end, it was decided to use small geared motors with a pinion to get a grip of a gear rack under the module to move it back or forth. This solution was chosen, because it had no rotational limitations like, i.e. most servos, and it only required a small amount of power as a result of the gearing, which was 1:1000. The size of the motors were also very small, but because the motors had to be orthogonal to the direction of transferring of the modules, they had to stick out to the side of the robot. An alternative was to make them towards the center of the robot, which wasn't wide enough to

have the motors be next to each other. This would force one motor to be further away from the edge of the robot than the other, which could complicate the transferring of the modules. For this reason it was chosen that the motors would have to stick out of the robot's sides. As the design of the platform was made in SolidWorks it revealed that the motors could be put down the side of the robot - if a 90° gear box was added. Sadly, this was not to be found anywhere.

The chosen motors also came with the possibility to install small magnetic encoder in the end, which could be used to detect if the motors were still running during an insertion/ejection of a module. But please note, that when attaching the encoder circuits it shouldn't touch the motor pin and don't apply heat too long when soldering it to the motor as the plastic might melt. The wires used to the motors and encoders also had to be replaced several times, because the thin wires couldn't handle the working with the robots. At last some thicker wires were used, which appears to be holding very well.

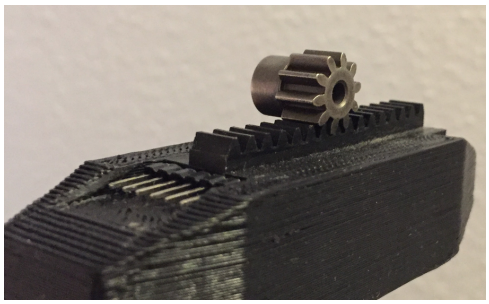


Figure 50: The chosen gear rack and pinion for the modules: Notice the rather large size of each tooth, which should help get a better grip in case these rack and pinion is not completely aligned.

The robot was designed to have the gear rack under the module slide into groove, which started with an outer width of 14 mm but decreased to just 7 mm (0.5 mm more than the width of the rack) after 44 mm of insertion. The groove was designed to be long enough to have the whole module inserted before it would hit in the end, forcing the motor to stall for a very short time. The robot would then detect this stalling based on the encoders and should then turn off the motor. During the initial phase a lot of different gear pinions and gear racks were tested. The gear rack that fitted best into the module and which gave a good grip was of the size 1M (see figure 50).

A little detail is that the wires for the motors were switched for just one of the motors. This was done deliberately as it would allow for less complexity in the code: Now a motor could be set to insert/eject the module without considering to which side the module was. Otherwise the left motors should have had a right rotation to eject a module, while the right motor should have a left rotation. The motors were controlled by Pololu's item 2961 motor driver[52]. This would make the electrical schematic less complicated as a H-bridge otherwise should be created. The driver allows for the motor to be controlled only with two pins: One for direction and one for speed (PWM).

The platform and all the modules are to be printed on a privately built 3D printer. The printer is up to the current standards for homemade printers, but has shown to sometimes to leave rough contours which potentially may effect the design, but could be corrected with a

[52]Pololu motor driver: `https://www.pololu.com/product/2961`

small file. Minor oozing from the printer was easily removed with a hot air gun. 3D printers from the IT University of Copenhagen were also used, but these appeared to be unable print the model of the platform[53]. The orientation of the platform was set to have the front facing up as this would prevent any filaments threads from hanging down from the ceiling in the slots, where the modules would go, which easily could interfere with the precisions of the design. Having the platform facing up during the printing, would also have the layers to be vertical when mounted to the Zumo, which should make any insertions of module more smooth as top surfaces of a 3D print often are a bit rough.

The control of the motors was done by the Teensy 3.2 which was mounted on the PCB at the top of the robot. Initially a Teensy LC was used, since it was less expensive and offered almost the same interface as the 3.2. But during the development, the space on the Teensy was all used, which is why it was replaced with the Teensy 3.2, which has 256 kb compared to the 62 kb on the LC[54].

**Modules**

The modules were, as mentioned, also printed on a PLA 3D printer. Preferably, these should be printed on the side as this would give completely smooth lines in the surface, which would be used to guide the module into its correct position. But as the hangings from the sides would occupy too much of the space within a module, and all of the space would most likely be needed, it was initially attempted to print the modules in their normal orientations.

The width and the depth of a module were decided by the amount of space available in an empty slot in the platform. The height was chosen by the height of a small geared motor as this was thought as the largest item, which were to be inserted into a module. In order to try and keep the modules small, so that they would be reasonable in size when compared to the platform, it was chosen not to implement a small micro controller in each module. A micro controller would have added great features to the system, but not some that would help proving the concept of a reconfigurable system. This meant that a communication protocol and physical way to make a connection between the robot and modules should be established.

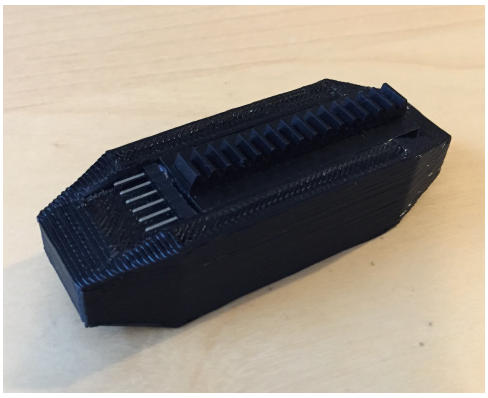


Figure 51: Initial design of a module (upside-down). The pins are in the bottom to protect the pins from dirt.

[53]The model was attempted to be printed multiple times on 3 different 3D printers from the IxD lab at the IT University of Copenhagen, but with poor results. Often the printer would gain an offset during the printing.

[54]Specifications for the Teensy LC and 3.2 can be found here: `http://www.pjrc.com/teensy/teensyLC.html`

To make the physical connection between a module and a robot, this initial design would use normal pin headers. These can, in later iterations, be replaced with SLCs, which are far more costly but adds reliability to the connection. To prevent the connectors to get dirty and increase the risk of a bad connection, the pins were positioned under the module (see figure 51).

In order to detect which type of module that would be inserted, a simple solution of using a voltage divider was used. Though such a voltage divider would drain the battery a bit, it wouldn't be much - especially because large resistors were used as these limits the amount of amps. Along with the type pin, other pins would have to be available between the module and the system. These would not only be for voltage and ground, but others to control and/or sense values from the modules. Therefore two pins were added, where one was a digital and the other was an analog, which should fit to most module configurations. A last pin for detecting the state of the module was also added, which adds up to a total of 6 pins. This pin should, depending on the type of module, be able to measure a high or low signal, that could be translated into a good or bad state. An example of this is the motor module, which would pull this pin high whenever power was added to the motor (see appendix D for the final electrical schematics).

Considering each type of module, some of these have designs that are worth adding a comment to: The battery module will be having an optocoupler controlling a relay which controls if power that should be given to the robot or not. The relay is activated by the battery modules own power, which completely separates the robot's circuit with the battery modules circuit, a bit like Luo, et al.[55]. The benefit of this is that a potentially faulty battery wouldn't be able to harm the robot at any time - not even if it is short circuited and burns some of its components. This is a big advantage, but has a minor negative effect of requiring power to keep the relay activated (closing the circuit) - power that otherwise could have been used by the robot. Unfortunately the battery module isn't big enough for one D (9v) or 3-4 AAA (4.5 - 6.0v) , which would be able to supply plenty of power. Instead tests were done to implement 2 A23 batteries, which are just 9 coin batteries that sums up to the 12 v - these can't supply much power though. It was first thought that the motor module should be controlled by a transistor or MOSFET, and thereby only have it run in one direction. But by adding a motor driver to the module, the ability to drive the motor on both directions was given almost for free. Whether a pin should be defined as input or output, is set upon insertion and ejection of modules, in the code.

The implementation of a service station will be done later, when the exchange of the modules is performed more or less reliably.

### 9.6.1 Evaluation

The tests of having a module transferred from one robot to another was initially *not* successful. It was discovered that the length of the gear under the module was far too short to have a robot eject it enough to have another be able to grab it. Therefore this evaluation included having the pin headers under the modules moved to the top instead, which allowed the gear rack to be extended far more compared to the first design. After having done this and cut the grove in the slot for the gear rack in full length of the platform (see figure 52), the modules could now be exchanged with even some centimeters between the robots, and approx. 5 mm of orthogonal misalignment!

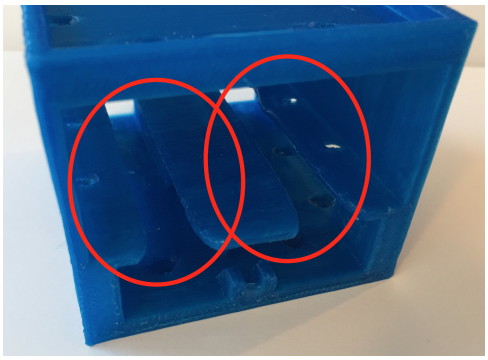


Figure 52: Rear of platform, where the grooves (red circles), for the gear racks under the modules to slide into, can be seen. The grooves were extended the full length of the platform during the evaluation.

Unfortunately, there were some communication issues: The robot to do the ejection of the module, would try to establish a connection to the robot which would do the insertion. Then it would start ejecting the module, but for currently unknown reasons the connection was occasionally broken right after it was established, which appear as a software related problem. The communication was achieved in chapter 6, even though this issue only was experienced when testing the reconfiguration.

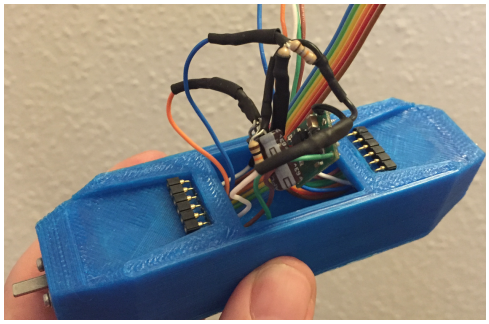


Figure 53: Installation of components for *one* end of the motor module. The small motor is already installed, leaving only very sparse room the components.

A lot of minor issues was also discovered, i.e. the grove on top of the edited module, where the pin headers were to be inserted was a bit too narrow. This caused the thin wires to break when all the parts where packed into the module (see figure 53). Another minor issue is that the screws holding the platform to the Zumo is right under the pinion gears for exchanging modules. This caused some trouble when the platform should be replaced for the next iteration. The screws for holding the motors also had a too large head and would force the pinion to not be completely centered. But the gears of size 1M did the work just fine: They were able to grab the rack under the module and the ”large” size of each tooth allowed for minor misalignments during the exchange. But the groove, which was manually cut during this evaluation to use a longer gear rack, would introduce some imprecision regarding creating the connection between the robot and the module.

It appeared as the module was well controlled until where the groove was extended, which imply that there may have been removed a bit too much.

A note about the 3D printing is that Windows 10 doesn't seem to work well with 3D printing host software (Repetier host[55] and Pronterface[56]) as Windows crashed several times[57] during prints. The reason for this could not be detected, but it often happened some hours into the print and with no warnings - neither was it possible to detect exactly what caused the crash. This *only* happen when attempting to print, but has never occurred when running Repetier host on a MacBook Pro, which therefore is highly suggested.

Regarding each type of module, all of the schematics and tests on breadboards were done. The laser and motor module was completed, but the battery module was only tested on the breadboard, where it worked fine. Because it hasn't been packed into an actual module, it will not be considered successful, which also applies for the LDR sensor module. A point to mention is that, because of the extreme compact design of, especially, the motor module, there aren't hardly any room for the electric parts and wires. The magnetic disc at the back of the module would very often experience friction from wires or parts that was pressed against the disc, which caused the motor to sometimes not being able to run.

The type pin on the robot has a pull-down resistor, but combined with a voltage divider on the module, it is simply too clumsy: There should only be the need of two resistors and not three.

Below is a list of the goals for this chapter that has been addressed:

**R-1** Must be able to activate a module's function
**R-2** Must be able to deactivate a module's function
**R-4** Must be able to handle request about reconfiguration
**R-5** Must be able to inform other robot about start of module ejection
**R-6** Must be able to detect when module is fully inserted
**R-7** Must be able to notify other robot module is fully inserted
**R-8** Must be able to autonomously detect module type
**R-9** Must have a tool module

**RX-1** Has several tool modules
**RX-5** Only the end in the front of the robot is activated, when module is used

[55]Repetier host website: `https://www.repetier.com/`
[56]Pronterface website: `http://www.pronterface.com/`
[57]The crashes were in terms of the "popular" blue screens with no further explanation of the why it crashed.

## 9.7 1st iteration - Service station and battery module

This iteration will attempt to have the service station start working, improve the geometrical design of the platform, and not only finish the battery and LDR modules, but also improve the modules in multiple areas.

**Service station**

The code for the service station was written in Java and used a bluetooth module, that has been attached to an FTDI driver. This means that the software of the service station will be able to run on multiple platforms, since Java is widely used and well integrated into most operating systems.

The service station is able to receive meta-data about the robots, which each robot reports in as a part of their setup process. The service station receives informations about which robot it is, its MAC address for the HM-10 module, and which modules it carry in every slot. The MAC address is automatically collected from the HM-10 module by the robot with AT commands, which means that a robot can have its HM-10 module replaced without updating any parts of the code.

**Platform design**

As the slots for the modules wasn't separated (see figure 54), a module wouldn't be guided into its correct position if it was too close to the center of the robot. Therefore a 2 mm wall has been added between the slots, which widened the platform with 1 mm to each side, protecting the motors slightly more by covering them better. These motors were also lowered a bit as the 3D printed model got some rounded corners, as a result of the 3D printer, where the edge of a motor's gearing should be which tilted them slightly upwards.

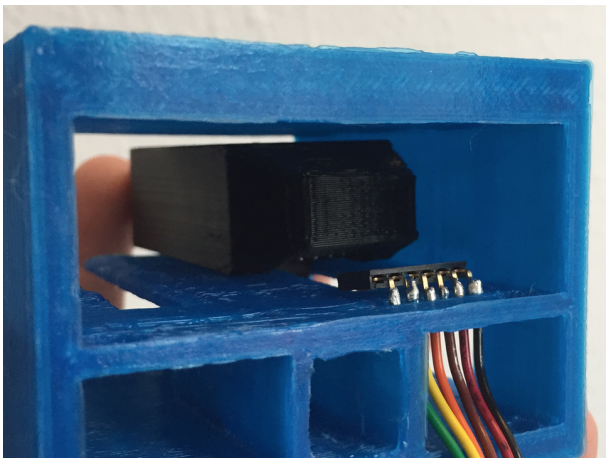


Figure 54: The initial design of the platform, didn't have a wall that separated the two slots. This caused the modules to not always benefit from geometrical structures that should guide the modules into position.

Contrary was the edges of the slots rounded with the *Fillet* feature in SolidWorks of 2 and 3 mm for respectively the top/bottom and the side edges . This should improve the potential offset from where the modules might be inserted, but still find their way into the slots of the robot. Even though this isn't a drastic improvement, it is still worth mentioning as the exchange of the modules is key feature in this project.

The platform was also increased such that the modules could have a squared design and now was 28 mm at the sides. It wasn't believed that the rectangular design of the modules would benefit in any way, and the space inside of the modules could certainly be used to fill in some of the electronics.

The top of the platform also got 5 mm high edges, which should protect the electrical circuits, but also have the PCB be parallel with rest of the robot so the magnetometer used in chapter 7 and 8 would be more precise, even though it was tilt-compensated.

**Modules**

The electrical parts for the battery module were now put together and inserted into the module. But a 9 v battery was too wide to fit into the module for which reasons, the sides of the battery module was reduced from 3 mm to only 1 mm each. The 9 v battery seemed like a good choice, as it had more voltage than what was required and a step-down regulator often can supply a current which are the same or greater than the input - and the 9 v held the ability to provide a decent amount of current.

Figure 55: Battery module with the 1 mm thing wall, so a 9 v battery could inter through the opening in the top. Unfortunately, the 9 v battery was too long to fully enter..

When the new design was printed, it was discovered that the battery sadly was a few millimeters too deep to be installed into the module through the opening in the top of the modules (see figure 55) - even with the new height of 28 mm. It was attempted to print the battery module in two parts, install the battery and the electrical circuit, but putting the two parts together never went well as any heating would bend the 1 mm thick walls. The heat was needed to install the gear rack to the bottom part of the module. Instead a battery case for two A23 batteries was used as it could just make it through the opening. An A23 battery is often used for digital camera, etc. an can supply 12 v, but because that it simply is 8 * 1.5 v coin cell batteries that are put into a single package, the current it can supply is very limited (55 mAh). Therefore the battery case was modified to put the batteries in parallel and not in series. This increased the amps of the battery module. But it wasn't only the battery module, which was completed: The LDR module, which will work as an example of a sensor module, was also finished.

A detail to the battery module is also that because of the relay, optocoupler, step-down regulator, etc. that had to be installed into the small module, the wires from the connectors in one end of the module was also connected to the other end of the module. Therefore the battery module would supply the 5 v power to both ends of the module at the same time, which

isn't ideal: Only the end being used show be provided with power as there now is a risk of short-circuiting if a conductive part connects the 5 v to the ground in the unused end - even though these to connectors are in each side of the pin row to prevent this from happening (see Appendix C and D). All of the other modules were created to only have the end that is pointing to the robot's front to be used.

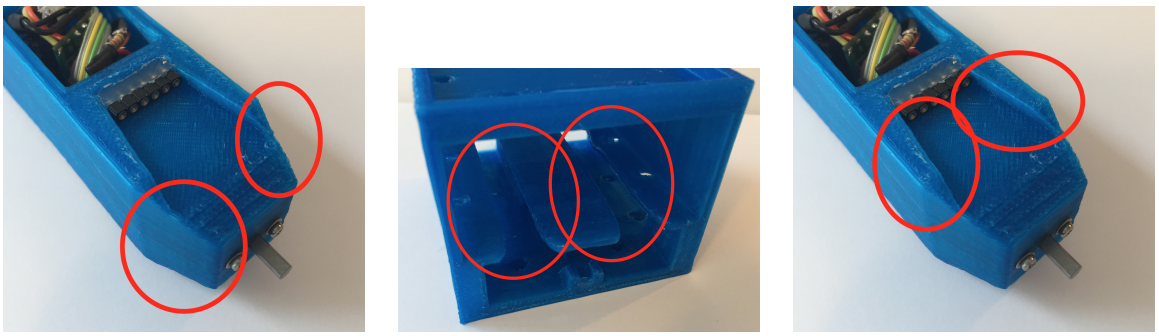


Figure 56: The 3 guidance steps for a module, when it is inserted: 1) The end of the module to hit the slot (red circles in the left image), 2) the groove in the platform for the gear rack under a module (red circles in the center image), and 3) the angles of the edges in the top of the modules, which will guide the SLCs to meet (red circles in the right image).

The 3 resistors used to define the value of the type pin were also replaced with just 2: On the robot side a $18\,\mathrm{k}\Omega$ resistor is pulling the pin down, while a resistor on the module side is connected to the 5 v supply for the module. The value of the last resistor clearly depends on which type of module it should represent, but the 4 types of modules returns a voltage that is evenly distributed across the 3.3 v, which the Teensy can interpret.

Another thing that was changed about the modules was that an angle was added to the guidance on top of the modules, which should lead the male SLCs in the ceiling of the robot's slots (see right image in figure 56) into the correct position. They now have an angle of 8°, which makes it a total of 3 steps, where the module is guided into the correct position (see figure 56): First the end of the modules will roughly make sure that the module is actually entering the slot. When the edge of the end has been fully inserted, the groove in the bottom of the slot for the modules will guide the rack, along with the attached module, into a more precise position, while also ensuring that the rack will be able to have a good grip with the pinion gear for inserting/ejecting the modules. As the module is about to be fully inserted, the groove in top of the module and the female SLCs in the ceiling of the slot, will guide the SLCs to be positioned right in front of

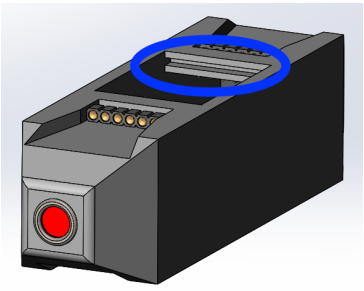


Figure 57: Image of module, where the groove (blue circle) in the side of the opening has been added. This may be used to add a layer of plastic or as for i.e. silicon to get a better grip and seal the module off.

each other, just before creating the physical connection.

In the end of this iteration there are 2 small notes: For the opening in the top of the modules, there are added a groove in the sides (see figure 57), which can be used to hold i.e. a layer of silicon or plastic in place to seal off the module from dust and dirt. The other note is that the groove for the pin headers in top of a module has got a small cut in the sides to prevent the use of support structure during the 3D printing as explained in chapter 8.8.

### 9.7.1 Evaluation

In general the improvements worked well in this iteration, even though some attempts turned out not to work. All of the modules were also now working and the robot was able to fully control each of the modules, independently even if it was a tool, sensor, or battery module.

#### Service station

Regarding the service station, it was now able to receive the meta-data about the robots, which they would transmit as a part of their start-up procedure. This would include their ID, MAC address to the bluetooth module, and what types of modules the robot's slots were holding.

#### Platform

The added walls could now guide a misaligned module into the correct position even if this was too close the center of the robot. Having added this wall and the fillets to the edges of the slots made the robot able to *most* of the time to have modules inserted with a misalignment of up to 8 mm orthogonally and a whole 70 mm collinear (see figure 58).

When transferring the modules, it was obvious which robot that had the most power left in the batteries as this would push/pull the module faster than the other. If the one with the most power left was pushing, the robots would actually have their rears lifted, because of the force (sometimes causing the module to get stuck). But if the it was pulling, the robots would get more aligned (if they were placed misaligned), which suggests that it might be worth looking into have the pulling robot run its motor faster than the other one is pushing.

| Collinear distance (mm) \ Orthogonal distance (mm) | 0 | 5 | 8 | 10 |
|---|---|---|---|---|
| 0 | 1 | NP | NP | NP |
| 10 | 1 | 1 | NP | NP |
| 20 | 1 | 1 | 1 | 0 |
| 30 | 1 | 1 | 1 | 0 |
| 40 | 1 | 1 | 1 | 0 |
| 50 | 1 | 1 | 1 | 0 |
| 60 | 1 | 1 | 0 | 0 |
| 70 | 1 | 0 | 1 | 0 |
| 80 | 0 | 0 | 0 | 0 |

Figure 58: Table over which misalignments the robot could transfer a module from: Green is successful, red is failed, and blue is not possible transfers.

Having lowered the motor a bit, increased the distance to the module a tad too much, which resulted in the pinion gear rarely would loose the grip with the gear rack.

**Modules**

As the modules now were 28 mm on every side, there was a lot more room for adding the electrical parts which filled up quite a lot of space, even though the design was as compact as possible. Especially the relay, battery case, motors, and the lasers occupied large parts of the limited space available. For this reason, it wasn't possible to have 2 battery circuits installed in 1 module, even if they shared the same battery: There wasn't enough room for 2 relays, which meant that the pin headers in each side of the module has to be connected, such that when the relay was activated in one side, the other unused side would also be powered. This isn't good, but if using robots that are just a bit larger, this wouldn't be a problem - robots that are to fit into the described scenario in the mining or industrial hall scenarios in chapter 9.1 will most likely be at least twice the size of this one.

Even though the A23 batteries aren't strong ones, they provide enough power to not only activate and hold the relay, but also give the robot some power. Initially, it was attempted to use a 9 v battery because of the higher volts and the large capacity, but it was impossible to install into the battery module: The opening in the top for installing the electrical parts wasn't long enough and printing the module in two halves didn't help as the thin sides of the module bend if the smallest amount of heat was added - which was needed for installing the gear rack to the bottom part of the module.

All of the modules were also changed slightly in terms of how the type of module was detected: Instead of using 3 resistors, 2 resistors were now used, as a $18\,\mathrm{k}\Omega$ resistor would pull the pin low (and prevent floating) while a resistor on the module side would be supplied with 5 volts, which created a voltage divider when a module was inserted. This worked extremely well, as the values now ranged between only 20/30 out of 1024 and not 80 out of 1024, which was the case in the previous iteration. The values that the voltage divider would give to the robot were divided evenly across the range of 1024. With a floating of only up till 30, this gave a very reliable determination of which modules that were inserted.

Having added the angle of the guidance on top of the modules for the SLCs, worked very well as none opposite force/obstruction was experienced after this was added.

Below is a list of designs which have been addressed or improved in this iteration:

**R-1** Must be able to activate a module's function
**R-2** Must be able to deactivate a module's function
**R-10** Must have a sensor module
**R-11** Must have a battery module
**R-12** Must be able to receive a module even if the other robot has an orthogonal offset of up to 8 mm and an axial offset of up to 15 mm

**RX-5** Only the end in the front of the robot is activated, when module is used

**Design goals - Service station**

**RS-1** Must be able to accept incoming meta-information about robots (robot ID, carried modules, etc)
**RS-2** Must hold meta-data about all available robots

## 9.8 2nd iteration - Minimizing backlash and increase handleable offset

The 2nd iteration will focus on lowering the backlash of the module and allow for the robot to have modules inserted with an even bigger offset than in the previous iteration.

**Platform**

To address the small backlash from the module, while they were being transferred, the width of the slots were reduced from 30 mm to 29 mm, which leaves 0.5 mm to each side for the module to move within. This may not seem as much, considering that the models are printed on a DIY 3D printer, that could use some adjusting, but compared to the amount of backlash, it seemed reasonable to reduce the space to just the half of it.

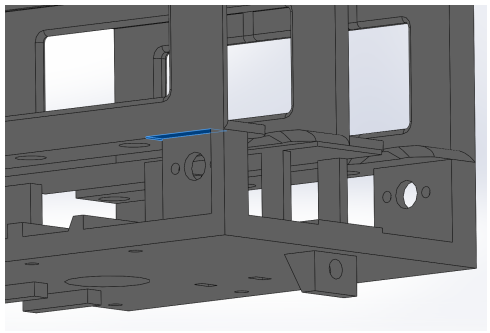


Figure 59: Rear left side of the platform: A cut (marked with blue) was made to prevent the motor getting too close the top edge as the 3D printer made the edges too round, in which case the motor could be tilted.

The motors was also raised by 1.5 mm to address the issue introduced in the previous iteration in chapter 9.7, where the gear rack and pinion gear occasionally lost the grip to each other. They did this because the motors had to be lowered as a result of using 3D printer, which had a hard time creating perfectly angled corners.

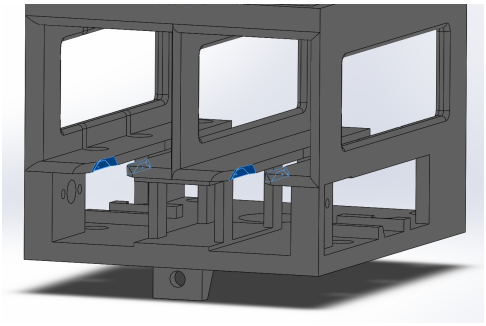


Figure 60: Rear of the platform: The blue areas are where the complex cuts were added, so the pinion gear wouldn't be blocked again.

To not risk to introduce this issue again, 1 mm of the nearby structure was removed (see figure 59) so that the motor could be well mounted without getting too close to the round corners. Raising of the motor also introduced a problem with the pinion gear, as this now was blocked by lower part of the slots for the modules. Therefore a few extra cuts were added to these parts (see figure 60), which allowed the pinion gear to spin, but without compromising the groove for guiding the gear rack.

When printing a new design of the platform it would take 8-9 hours to complete. To lower the print time, some large areas on the structure were removed (see figure 61). These areas were from the vertical sides for each slot, which wouldn't interfere with the use of the robot as the "guidance areas" for the modules wouldn't be affected and the electronics on top of the platform still was protected, since it was kept covered.

In the last iteration is was mentioned how the robots tend to pull/push each other, when the

remaining power level of the batteries was significant different. To address this, the speed for ejection and insertion of a module was set to be different: A module would be ejected with a speed of 110 out of 400, while a module would be inserted with a speed of 140 out of 400. This should be ensure that, if anything, the robots would be pulled towards each instead of getting pushed away, which lead to both of them tilting and potentially prevent the module from being inserted.

Two small cuts was also added to the bottom of the platform, which was meant to keep any wires in place with a couple of strips. A lot of wires were starting to be near the motors, as the laser and TEPT4400 from chapter 8 now were moved to the rear in between the two motors, and the ultrasonic sensors from chapter 7 were added.

**Modules**

The end of the modules will now have more of the sides removed to shape the squared cone. Instead of just removing 5 mm, there will now be removed 7 mm at each side, which should increase the offset of a module may have while still being able have it inserted with 2 mm. By also added fillets to the end of the module, this should increase this distance further by 1 mm.

The angle for the shapes on top of the modules, which guides the SLCs to the female SLCs on the module was increased from 8° to 10°. The wasn't experienced any complications which suggested such an improvement, but the designed allowed for it, and since it only improved the design, it was done.

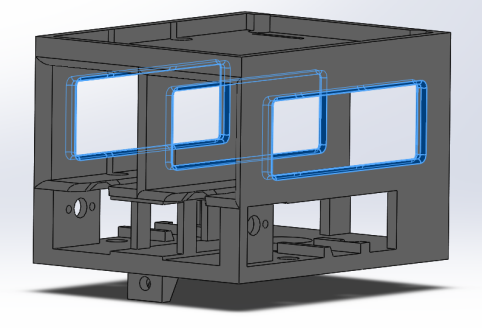


Figure 61: After removing unnecessary areas (marked blue) in the platform, the print time was reduced with 1 hour.

It was strongly considered to replace the A23 batteries with Li-Po batteries as these offers a larger amount of power compared to their size. It was noted that Li-Po batteries of 3-400 mAh would fit very nicely into a battery module and there was even enough room to install 3 of these, while leaving room for the rest of the electronics. This would provide a whole 1200 mAh compared to 2 * 55 mAh from the current A23 battery[58]. The only difference is that Li-Po batteries supply 3.7 v, contrary to the 12 v from the A23 battery. A proper step-up converter should be able to handle regulating the power to 5 v, while supplying enough current for the entire robot to run. But as the concept of having a working battery module already was shown, it was left for others to optimize the battery in the battery module.

[58]Capacity of battery types: `https://en.wikipedia.org/wiki/List_of_battery_sizes`

### 9.8.1 Evaluation

This iteration has improved in not only the printing time of the platform, but is now able to handle larger offsets. On the down-side the motors were raised too much, so the modules often got stuck when being inserted, for which reason they were lowered slightly to be able to better evaluate the rest of the iteration. The motors were lowered by melting the holes for the motors with a hot soldering iron. It was also noted that the bluetooth connection with the HM-10 modules was not always able to stay connected to the other robot, which might be a result of having the modules put in a horizontal position for when they need to wear the fiducial marker when using SwisTrack for documentation in the previous chapters 6, 7, and 8.

**Platform**

The printing time of the platform was reduced with 1 hour and a lot of filament was saved for this reason. Removing the areas didn't affect how well or reliably the robot worked.

Making the slots more narrow worked well, even when the filament was changed to a grey color, which unfortunately gave horrible results (see figure 62). The grey filament was very low-cost from China, and it might be wise to use something of at least a bit better quality. The narrowing of the slots gave a nice, controlled movement with almost no backlash when ejecting or inserting the modules. Even though the grey filament gave a very uneven surface, it is believed that making the slots even more narrow, will just increase the risk of preventing the modules from being transferred.

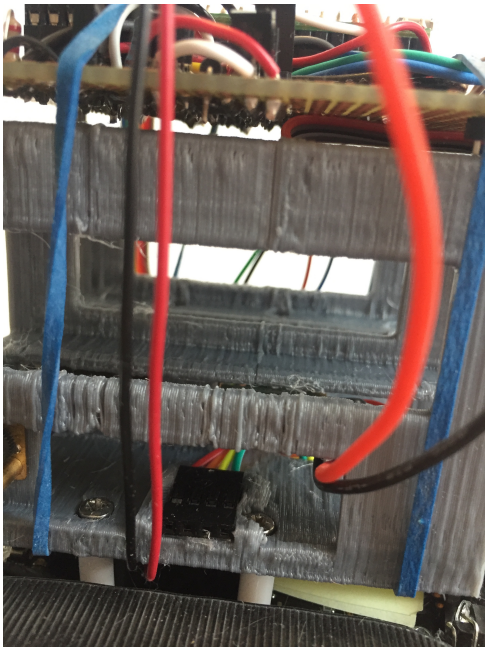


Figure 62: It was experienced that the grey filament produced very poor results, which clearly can be seen by the surfaces in this figure.

Having raised the motors introduced several issues. First the pinion gear would be touching the bottom of the slots, which is why the additional complex cuts from figure 60 were done. When the pinion gears could spin freely, it was tested how well a module would be transferred and very often the module would get stuck against the ceiling of the slot and the pinion gear. Therefore the motors were lowered by using a hot soldering iron against the filament, which were done in this iteration, so that further measurements could be done before starting a new iteration - without compromising the results!

The changed speed of the motors were successful as the only ”risk” when transferring the module was that the robot got pulled into a better angle for the transfer (if they weren't perfectly aligned). It can be discussed whether this is a reasonable way to address the issue of having the motors running at different speeds because of different levels of remaining power within the batteries. As mentioned in relation to the positioning of the robot in previous chapters, a better way to address such issues would be to use the encoder on the motors and drive the motor with a speed of ticks per second, as this would be completely independent of the battery level.

**Modules**

The end of the modules were made more narrow so that the robots could have a larger offset and still be able to perform a successful transfer of the modules. As shown in figure 63 a module can now be transferred even though the robots are as much a 10 mm orthogonal misaligned and up till 70 mm from each other, which is very impressive!

Following is a list of design goals from chapter 9.5, that have been addressed or improved in this iteration:

**R-12** Must be able to receive a module even if the other robot has an orthogonal offset of up to 8 mm and an axial offset of up to 15 mm

The robot is now well capable of addressing design goal **R-12** as it can transfer modules even if the orthogonal offset is 10 mm and/or it has an axial offset of 70 mm, which is more than 4 times the required distance from the design goal.

| Collinear distance (mm) \ Orthogonal distance (mm) | 0 | 5 | 8 | 10 | 12 |
|---|---|---|---|---|---|
| 0 | 1 | NP | NP | NP | NP |
| 10 | 1 | 1 | NP | NP | NP |
| 20 | 1 | 1 | 1 | 1 | 0 |
| 30 | 1 | 1 | 1 | 1 | 0 |
| 40 | 1 | 1 | 1 | 1 | 1 |
| 50 | 1 | 1 | 1 | 1 | 0 |
| 60 | 1 | 1 | 1 | 1 | 1 |
| 70 | 1 | 1 | 1 | 1 | 0 |
| 80 | 0 | 0 | 0 | 0 | 0 |

Figure 63: Table over which misalignments the robot could transfer a module from: Green is successful, red is failed, and blue is not possible transfers.

## 9.9 3rd iteration - Service station able to send reconfiguration commands

In the previous iteration it was described how the motors were raised too much and the modules would get stuck between the ceiling of the slot and the pinion gear. Therefore the motor has now been lowered with 1 mm, which gives a total raise from the initial iteration of 1.5 mm.

As the problems with establishing a connection via the HM-10 modules (which has turned out to be cheap replica as described in chapter 6) has started again, the delay after the connection has been established is now set to 1 second. This should give both of the robots and their HM-10 modules enough time to settle before starting to use the communication channel. The HM-10 has also been tilted into a vertical position as the horizontal position potentially could be limited since it was in the same height as the electrical circuit on the PCB on the top of the robot. Since this chapter isn't using SwisTrack for documenting the improvements, the robot doesn't need to have a fiducial marker on the top, which forced the antenna to be horizontal.

**Service station**

The service station has now been added some code, so that it can choose an available and fitting robot, which can help another robot with a reconfiguration. It has also been matured to create a connection to the chosen robot and send a command to it, such that the other robot knows which module that it should give to the robot that requested the reconfiguration along with the MAC address of that robot.

### 9.9.1 Evaluation

Even though this wasn't a big iteration, there were still some significant improvements, which means that the robot now can be reconfigured by another robot upon its request.

First of all, lowering the motor with 1 mm made the module able to be transferred again without getting stuck. The problem with establishing a connection from one robot to another in this scenario isn't completely solved, even though it had been improved. It is therefore strongly considered whether bluetooth is the best way to have a communication channel as the establishing also takes approx. 2 * 8 seconds every time: 8 seconds when the connection is established and 8 seconds, when the module is put back into slave mode.

Below is a list of the addressed or improved goals from chapter 9.5. Please note that the communication is a part of chapter 6, for which reason the small improvements won't be reflected here.

**Design goals - Robot**

**R-3** Must be able to request new module from service station, specifying module type and slot for insertion

**Design goals - Service station**

**RS-3** Must be able to choose robot, that can help other robot with reconfiguration

## 9.10 Discussion

This chapter has attempted to develop a way to transfer modules between two mobile robots and has come up with a platform and four modules (see figure 64). But can the system even be used in relation to underground mining robots as described in the scenario from chapter 9.1? And is the handleable offset of the robots even enough?

These questions, and a lot more, discussed in the rest of this chapter. To provide the reader a structured overview, it has been split into the following minor parts: Service station, Platform, and Modules.

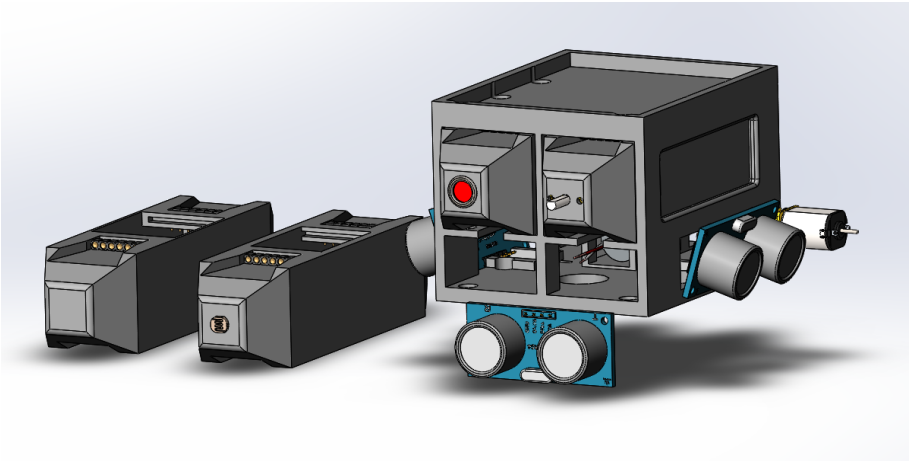


Figure 64: Screenshot from SolidWorks, which shows the final version of the platform with all four modules: (From left) are battery module, LDR module, laser module (in platform), and motor module (in platform).

**Service station**

The service station was written in Java, so that it could support the largest amount of platforms. For the same reason the FTDI driver was used as middle-man so that the service station was able to communicate to the HM-10 module as if it was a normal Serial port with on an USB cable. As the HM-10 module is connected to the service station through an USB cable, most computers will be able to use the exact same software and hardware as in this project. But it should noted that the HM-10 modules were a bit troublesome, which could have been caused by fake replica of the true HM-10 modules. This is mentioned in detail in chapter 6 where the general communication issues are handled - and other means of communication are suggest.

When the robot start up, it has been programmed to connect to the service station and inform this about all relevant meta-data in terms of the robot's ID, the MAC address of the HM-10 module, and what modules it carries. The MAC address from the HM-10 module is collected by the robot, which means that changing the HM-10 module while the robot is turned off, won't have any impact in terms of new MAC address or similar. This makes the robot more robust and flexible, as nothing is hard-coded into the robot's firmware.

The service station will store the incoming meta-data from the robots and save this in the RAM. It was considered to serialize these data and store them to the hard drive from where it could be read upon restart of the services station. In this way it would be tolerant to power failures, but imagining this scenario a bit more carefully it could really be prone to an unstable system, as the service station would automatically assume that all of the robots were still working great and available in case of reconfiguration request. Given that a robot had returned to, i.e. a dock or simply has been turned off, the service station would attempt to call a robot, that wouldn't be available. The HM-10 modules doesn't provide much feedback, and having to make this data persistence work reliably would have been unrealistic when using such modules. Instead it could be worth looking into the upcoming LoRa network[59] for IoT objects which has a range of a hundred meters in dense cities. This network comes encrypted and can run on a coin cell battery for a long time.

Even though the service is unable to handle reliable data persistence, it is absolutely able to choose an appropriate robot which should assist with a reconfiguration request, when such is being send to the service station. Which robot that is chosen depends on the type of module that is requested and in which slot it must be installed to: As the robots only have two slots, the requested type of module must also be in the same slot in the chosen robot, because they are standing back-to-back when transferring the modules. This may be a bit troublesome as a module then implicit is chosen to be in only one of the slots. An alternative is to have the robots dock with an offset of half the robots' width, which simply would require to extend the docking procedure from chapter 8 with an LDR in each of the robot: Tho robots could then use one of these LDRs when a module should change side.

The communication between the robots and the service station is done in clear text, which implies that if another communication protocol is wanted, this only requires to change to wanted the communication device, which must support the Serial communication protocol.

The service station does give a bit of feedback, even though this isn't marked as addressed as in design goal **RSX-2**. The reason for this is that it is merely simple console outputs, which are shown when a robot transmits its meta-data, requests reconfiguration, and when the service station has chosen a robot to assist another robot. Even though it is useful, it is not considered to be enough to fully qualify as addressing design goal **RSX-2**.

[59]LoRa website: `https://www.lora-alliance.org/`

**Platform**

The platform is more of less the entire upper part of the robot, and has been improved mainly in this chapter, while used in previous chapters for attaching sensors and other hardware.

The production of the platform was done on a DIY 3D printer, which could benefit from being calibrated, which implies that the quality of the printer could have been better if, i.e. a 1.400 USD purchased 3D printer was used instead. It was also experienced that not all colors of filaments printed in the same quality: For instance, a grey filament gave extremely poor results (see figure 62), nonetheless, the designs of the platform and the modules were able to work very well, which suggests a good and robust design.

In the initial iterations, it took 8-9 hours for the platform to be printed, but later this was reduced by an hours, as a result of removing unused space from the platform. This space was from the walls of the slots, but only areas that wouldn't help guiding the modules into place, or other important functions. For this reason, the ceiling of the slots wasn't removed as it was used to protect the sensitive electronics at the PCB on top of the platform. Having added the cuts, the platform still appeared strong and rigid.

The motors used to insert/eject the modules had a gearing of 1000:1, which at some point tested the strength of the module, when the motors had been raised a bit to high: At this point the modules were very likely to get stuck in between the ceiling of the slot and the pinion gear, which was attached to the motor. The high gearing made sure that the module was moved forward until it was sure that it couldn't go any further. Taking advantage of this strength, the platform would insert a module by running the motor until the encoder, at the back of the motor, revealed that the motor hadn't moved in 300 ms, at which point the module would be considered to be fully inserted, which also would be informed to the other robot. Because of the design of the module and how smooth it goes into one the slots, this is considered as a reasonable way to detect if an insertion has been successfully completed. Strictly speaking, something *could* block the module and the robot would interpret this as the module was inserted, even though it was still only half way into the slot. A way to check this, could be to detect if a module type of not *Empty* could detected in the slot - of this was to be the case, then the module would have been fully inserted.

One of the biggest flaws in this project is how the motors for the modules are pointing out from the platform. A lot of online and physical shops have been visited to find a solution to this, but none of them had a 90° gear box, which would have allowed the motors to stay inside of the platform and be very well protected. Having the electrical circuit for the encoder at the back of the motor, along with the magnet, to be one of the corners in a mobile robot is simply prone to errors: Two robots might crash into each other because of these outer corners.

But one of this systems biggest advantages is that the reconfiguration/repairing can happen on-site in contrast to other projects, like Hojmark, et al.[26], which required for the robot to

follow a line back to the service station before it could get repaired. Because each robot is capable of both inserting and ejecting modules, they can do this as long as they are able to find each other and dock, which is achieved in the chapters 6, 7, and 8. But this is only true as long as the robot has power to perform the actions which is required for the reconfiguration: Given that the robot's battery fails it wouldn't be able to do anything. But in chapter 10 a back-up battery is added, which is strong enough to power the entire robot. This will make it possible to perform a reconfiguration, which be may to get a battery module, as long as the main micro controller (the Teensy) and the motor for an empty module slot isn't damaged, and the robot hasn't backed up against a wall.

The situation where the wheel, or other mobility hardware, is damaged hasn't been addressed yet. But as the rack on a module is long enough to have both robots get a good grip, which was experiences when they dragged themselves closer because of different insertion/ejection speeds, they could use this to transport each other to the nearest service station. The only thing speaking against this is that the motors don't have a clutch, to prevent the motor from being turned as the robot was dragged.

One of the truly impressive achievements of the platform is how well it can handle another robot being significantly misaligned, while still being able to grab a module and have it well inserted. In figure 63, it was shown how much the other robot are allowed to be misaligned. This is partly because of the design of the modules, which will be discussed next.

**Modules**

The geometrical shape of the modules are one of the primary parts which has been iterated over in this chapter: Not only should it increase the possibility for another misaligned robot to be able to eject a module into the robot, but it should also guide the module into the intended position so that all the connectors would fit together. Beside this, it should also be possibly to install electrical components and parts into the module without too much hassle.

Because of the rather small size of the robot, and therefore also the modules, the modules ended up being 28 * 28 mm and 110 mm long[60], while the hole for installing the electrical parts only was 22 x 31 mm. Considering that real-life implementations of this system most likely will use larger robots, as they are intended for underground mining and industrial hall environments, the modules will also be bigger and more parts can be installed into a module, but also through a larger hole.

One of the limitations that was experiences through the geometrical design of the modules, was that a 9 v battery couldn't get through the hole in the top of the modules, which would have been preferable compared to the A23 batteries that was used for the battery module instead.

[60]Note that some of the modules are a bit longer because they have external parts mounted, like the LDR or the shaft of the motor.

The A23 batteries worked well enough to give an impression of the battery module, but given that these types of batteries only has a very limited capacity, it is encouraged to attempt to build the battery module with other types of batteries as i.e. LiPo batteries as they hold a very high power-to-size ratio and are rechargeable. It is unrealistic to use A23 in a real-life implementation, but it worked very well for showing the concept.

The battery module was also the only one which had to have so much installed into the module that there wasn't room enough for installing a circuit for each of the modules ends (using the same battery), for which reason this would be activated for both ends, when used. This wasn't the case for the other modules, which only would activate the end of the module that was in the front of the robot. Having so many components in the battery module (see Appendix D), was done deliberate as the module then would use it own power source to hold the relay, which forwarded the power to the voltage regulator and then to the power circuit of the robot. In this way, the circuit of the battery and robot were completely separated, when the relay wasn't activated, which protected the robot from being fried because of a faulty battery module. The disadvantage, because of the modules size, is that it supplies power to both ends, which means that it could potentially get short-circuited from the unused end, when activated - but again: This can easily be addressed, if the robot is scaled up.

Having the battery module constructed like this, also means that the robot has to give a 3.3 v signal before the module may supply power to the robot. Therefore one can't just expect that the robot will start up, if it's without power, when the battery module inserted.

But all for types of modules worked well, and showed that the robot could work with both tool, sensor, and battery modules. The types of modules was detected by a splitted voltage-divider across the SLCs: On the robot's side an $18\,\mathrm{k}\Omega$ resistor acted as pull-down, when no module was connected, but as a voltage divider upon connection as a resistor on the robot side would connect the 5 v power source to the type pin.

Based on the type of module, the pins on the Teensy going to the module, would dynamically be set to input or output. An example if that the pin controlling the direction of the motor in the motor module, would be used for analog input when the LDR module was inserted (see Appendix D). The laser module only had a single on/off pin, while the motor module had one for direction and one for speed (PWM controlled). By dynamically define the pins functions, the overall amount of connections between the Teensy and a module was reduced. If the amount of pins should be reduced even further, it should be done by adding a micro controller in each module, which also can provide meta-data about the modules. Preferably this can be done with the protocols that requires a low amount of wires, while being able to address several devices, or modules, like $I^2C$ and SPI as also described in chapter 9.4.

To ensure that a module will end in the correct position and with good physical contact between the SLCs, there has been made 3 stages to that gradually will guide the module: 1) The end of the module will make sure that the module actually hits the slot (see figure 63 for how much offset the module may have), 2) when the *end* of the module is fully into the slot, the rack under the module will hit the groove in the platform, which both guides the module, but also makes sure that the rack is in contact with the pinion gear, and 3) on top of the module are some angled cuts to guide the female and male SLCs into each other so all pins from the Teensy are well connected to the module. By having these 3 stages, the platform has shown that it can handle offsets with up to 10 mm orthogonally and 70 mm collinear! This a lot more that the offset that the robot was able to dock with in chapter 8.

## 9.11 Conclusion

The reconfiguration scenario has been addressed, based on the states goals, for a total of 4 iterations. Through these iterations, there has been significant improvements, but attempts to address/improve some of the goals have also revealed what not to to. In the end of the iterations, the design has addressed all of the required design goals for both the robot, but also the service station, while some of the extreme design goals have been addressed. The most impressive finding was how much offset a module could have, while still being able to be inserted into a slot: 10 mm orthogonally and 70 mm collinear.

This chapter has showed how 2 tool, 1 sensor, and 1 battery, modules can be developed, which all have shown to work very well during the tests after each iteration, where they have been created or improved. There was some trouble getting 9 v battery into the battery module, but two A23 batteries showed the concept, while it was suggested that 3 Li-Po batteries in parallel could be used, which would provide approx. 1,200 mAh.

The service station has also showed how it can receive, handle, and use meta-information from the robots. This includes choosing a fitting robot, when another robot has requested a reconfiguration. Even though the service station prints information to the console about any events, this was not considered enough to qualify to address the related extreme design goal.

All in all, this chapter is considered to be a success and it addresses the following design goals from chapter 9.5:

**Design goals - Robot**

**R-1** Robot is be able to activate a module's function
**R-2** Robot is able to deactivate a module's function
**R-3** Robot is able to request new module from service station, specifying module type and slot for insertion
**R-4** Robot is able to handle request about reconfiguration
**R-5** Robot is able to inform other robot about start of module ejection
**R-6** Robot is able to detect when module is fully inserted
**R-7** Robot is able to notify other robot module is fully inserted
**R-8** Robot is able to autonomously detect module type
**R-9** Robot has a tool module
**R-10** Robot has a sensor module
**R-11** Robot has a battery module
**R-12** Robot is able to receive a module even if the other robot has an orthogonal offset of up to 8 mm and an axial offset of up to 15 mm

**RX-1** Robot has several tool modules
**RX-5** Only the end in the front of the robot is activated, when module is used

**Design goals - Service station**

**RS-1** Service station is able to accept incoming meta-information about robots (robot ID, carried modules, etc)

**RS-2** Service station holds meta-data about all available robots

**RS-3** Service station is able to choose a robot, that can help other robot with reconfiguration

# 10 Error detection

In this chapter an error detection system will be evolved. Not only will it show how error detection in dynamic parts of a system can be implemented, and errors within these may be responded to, but it will also show how to address an error in a critical part of the system.

## 10.1 Scenario

Error detection is used to get feedback about if executed tasks are carried out well and if the system is working as intended. This can optimize the tasks in terms of adding a reaction to a detected error, which can either correct the task or prevent that part of the system to be used, if it would only cause further damage.

An example of such a situation is when a robot is working in a mine, where it uses a hammer to tear down some of the walls. If the robot isn't able to detect, i.e. the oil level of the hammer, it may continue to use it even though the hammer will get significantly damaged when operated without lubricants. In the situation where the robot can detect if the hammer it malfunctioning, it may stop operating it or execute a task to address this issue, which may be to call for a replacement tool.

Error detection can be used in several ways depending on the needs and what kinds of errors it is detecting: In case it is critical errors or timing is of great importance, it may be implemented based on interrupts so that it react to the error exactly when it happens. Some systems also benefit from being able to detect an error without the tool, or similar, is running - like when a car starts up: When the car is turned on it will do some basic tests of the engine even though it isn't running yet. Only a limited amount of situations can do this, while most would have the tool or equipment to actually run before it can determine if it is operating well.

## 10.2 Related work

There aren't really that much research in error detection as it is already a well-known technology, that have been applied to robotics for many years now. There are lots of examples of this, which often includes adding a validation number to a string of data[48], if it's within software, where the validation number relates to all of the data and will reveal if an error occurred during the communication. In terms of hardware, there are generally added a unit which will measure, i.e. the voltage level a specific point: In case the voltage drops or rises it would be a sign of an error.

An research project[8] focuses on a robot arm that constantly verifies its own position. In case the arm drops or a sudden offset is detected, it will calculate the best way to reenter its previous position. But similar systems have been used in many of the currently available industrial arms, i.e. from ABB or Universal Robots. Others have created a so-called artificial immune system[30], which has two types of failure detections mechanisms: One that will react to failures in a generic way by doing standard operations, while the other will try to learn and adapt to different types of failure. In this way the system will not only be able to correct itself from day one, but gradually get better as it learns how to tackle the failures in the best way.

## 10.3 Design considerations

There are multiple ways to implement an error detection system, which first of all comes down to whether it should be interrupt based or not. A possibility is also to have a second system running, which may be dedicated for error detection, that measures for errors (interrupt based or not) and forwards the status of the components it is measuring at. When forwarding the status, this could also be done by interrupts in terms of notifying the main system that *something* is wrong and it ought to ask for further details.

The first part, which concerns of actually detecting the error, comes down to measuring i.e. the voltage level from a certain point in the system or having an encoder to detect rotations. If the system is measuring for changes in the voltage, it may choose do this when it fits into the routine of the system, by having a section in the routine that is dedicated for this. If the system is setup to handle interrupts in *any* time of the routines, the system can have an interrupt routine which is triggered, i.e. when the voltage drops, changes, rises, etc. If the system really needs to have the error detection implemented with interrupts, but also have periods where it can't be interfered by an interrupt, most micro controllers have the ability to disable interrupts temporarily. But then the system must make sure to check if a potential error has occurred every time the interrupts have been disabled.

## 10.4 Implementation considerations

The error detection in this system is most likely not time critical and therefore doesn't need to be implemented with interrupts. But if there is nothing to hinder an implementation with interrupts, this should be preferred. In case that the system doesn't seem to support interrupts, there should be defined a routing which gets called rather frequent to minimize that a module, tool, etc. is operated in a faulty state and potentially cause damage.

Preferably the error detection should address most parts of the robot, like battery, radio devices, motors, and the modules. But as this would create almost identical sub-system, it will most likely be best to focus on just one part of the robot, like the modules, which later may be used as a starting point for others that wish to implement error detection to a robot system. But one of the big issues is also what the robot should do, when it detects an error. That might a bit out of the scope for this chapter, as this chapter simply addresses *how* an error can be detected, but it is definitely something that should be considered.

If the robot is going to have a dedicated micro controller to detect errors, this should be able to know what to look for: It shouldn't measure for high voltage on a motor that isn't running. So therefore the micro controller would need to be supplied with continuous information about which parts of the robot that is activated and in which way. Over time this might seem as a lot of overhead of information that need to be sent back and forth between the main controller and the error detection controller, but as the frequency of this is very low it shouldn't be an issue. If it is implemented like this, it could be beneficial to use an $I^2C$ protocol and give the error controller an unused address, so that it won't occupy any extra pins, as i.e. the magnetometer from chapter 6 also uses this protocol and the pins already are reserved for this.

Regarding being able to error detect on a module that isn't running, the robot could most likely benefit from this, as some modules might not be running all the time. If the robot first detects an error just as it starts to use the module, it would spend quite some time before being able to continue to work. A way to see if a module, or similar, is working, could be to give it just enough power so that it can detect if it is working, but not so much that the tool is fully powered.

## 10.5 Design goals

Based on the previous subsection, a list of design goals for error detection has been made. The items are named according to the name convention in 3.6 and will list what is needed for an error detection scenario to be considered successful. At the end are a few extreme goals, which are not vital for the scenario, but these will add some neat features to the system (note the X in names of the extreme goals).

**Design goals**

**E-1** Must start measuring for errors within one second after module insertion
**E-2** Must measure for errors at least one time per second
**E-3** Must not measure for errors on empty ports
**E-4** Must be able to detect errors on each type of module
**E-5** Must seek to find a solution, when a module has failed (module replacement)
**E-6** Must be able to detect errors on activated module

**EX-1** Is able to detect a failure in either of the proposed modules in real time (interrupt based)
**EX-2** Each module can be easily triggered to fail (for testing purpose)
**EX-3** Can detect errors on deactivated modules
**EX-4** The error detection can be conducted with only one pin per port
**EX-5** The robot has a battery backup, that can power the robot while waiting for a battery module, if the primary battery fails

As any of these goals get addressed or improved, they will be mentioned under *Evaluation* for each iteration.

## 10.6 Initial design - The error detection circuit

One of the big issues with the error detection is whether it should use interrupts or not. An interrupt based approach appears to be the preferred choice as the error would be detection exactly when it occurs. If a non-interrupt based approach where to be used, the error wouldn't be detected before the internal routine of the robot chooses to check for errors - which could be too late, as the error could have caused severe damages to the module.

During the implementation of the proposed reconfiguration system, it was experienced that the chosen communication modules (HM-10, mentioned in chapter 6) are very sensitive to the timing of the AT commands: Giving a valid command, that was sent with a minor time delay between two of the letters, would cause the HM-10 module not to interpret the command. Knowing that it is possible to temporarily disable interrupts within the Arduino environment, the continuing of implementing the error detection was done without interruptions, but a note about this was made. A second argument for not implementing the interrupts right away is that the Teensy 3.2 is running with a clock speed of $72\,\mathrm{MHz}$ (overclockable to $96\,\mathrm{MHz}$), which gives a good foundation for running the manual error detection routine at a high frequency. For which reasons, an internal routine would be called once every time in *the loop*[61].

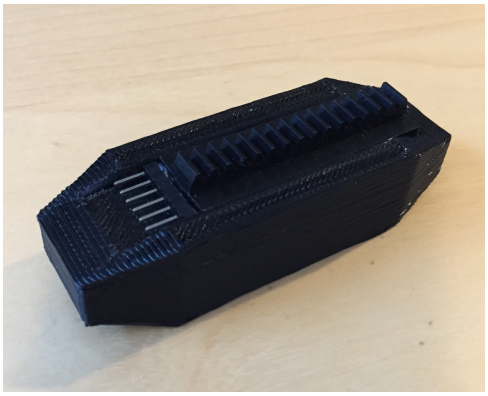


Figure 65: Bottom of a module - initial design. Note that there is exactly space for 6 pins.

As stated in chapter 9, there will be five pins needed for the modules in order to work correctly. Fortunately, the design and size of a module allows for exactly one more pin to be added, which adds up to a total of 6 pins for each module (see figure 65). This extra pin is used for detecting the state of the module, which can reveal if an error has occurred. Depending on the type of module an error state will be determined based on the following (see Appendix D for the final electrical schematics):

- **Laser module** Voltage level after the laser
- **Battery module** Voltage level (after voltage divider) available from the battery
- **Motor module** Voltage level before or after motor, depending on motor direction
- **LDR module** Voltage level after the LDR

During the design and testing of the modules, it was experiences that some of them needed a capacitor to balance the state signal between still being detectable when needed, but not so

[61] Sketches in Arduino has two main methods: The setup, which is called once and the loop which is called repeatably hereafter.

long that it couldn't be detected before +3 seconds had passed. An example of this is the motor module, where a $22\,\mu F$ capacitor and 10k $\Omega$ resistor is used. In general it took a lot of attempt to find the right values of the components for each of the four modules, but were eventual accomplished.

As an attempt to implement a mechanism that could trigger an error detection in the modules, a small reed switch was added to each of the modules. This would allow a module to trigger the error detection by having a strong magnet next to the robot and thereby avoid any physical contact. The pin for error detection had a pull-down resistor of $50\,k\Omega$, for which reason a normal-closed reed switch was used between the pull-down resistor and the module. It was also considered to use a hall effect sensor for this, but having a logic mechanism instead of an analog was more appealing.

### 10.6.1 Evaluation

The error detection worked in general well, as this would only measure the state of the module, when a known module was inserted. The worst-case runtime between every time the error detecting method is called is very difficult to say as this really depends on the robots current task. But when tested the robot notified, when a module was inserted and the state of this module, within a very short time - certainly less than a second. This would happen even if the module wasn't activated, by providing a bit of power to the correct pins of the module and measure the state. For three of the modules this was barely noticeable, but for the laser module, a vague, blinking line could be detected.

One of the things that didn't work in this iteration was the installed reed switch in the battery module as a battery apparently generates a fairly strong magnetic field around itself, when it is being used. Therefore the reed switch would constantly be one, because the voltage divider for detecting the voltage level and the voltage divider for testing the state of the battery, consumed a bit of power. Because the reed switch would always be on, it wasn't even possible to activate the module, as the robot shouldn't be able to active a failed module.

From the design goals specified in chapter 10.5, the following were addressed:

**E-1** The robot is able to start measuring for errors within one second after module insertion
**E-2** The robot is able to measure for errors at least one time per second
**E-3** The robot won't measure for errors on empty ports
**E-4** The robot is able to detect errors on each type of module
**E-6** The robot is able to detect errors on activated module

**EX-3** The robot is able to detect errors on deactivated modules
**EX-4** The error detection is conducted with only one pin per port

## 10.7 1st iteration - Back-up battery and jumpers for error triggering

This iteration will focus on the triggering of the error detection, as this will be used in chapter 11, but also how the robot should react to the detected error.

Instead of using the reed switches, a small jumper was added to the robot for each of the slots (see figure 66). This jumper would have the same position in the circuit, but instead of installing one for each module (which wouldn't be reachable when inserted), there was installed the two on the robot side. This allowed for the signal to be detected, but when a jumper was removed, the signal would be drawn to ground because of the previously installed pull-down resistor. By not using a magnetic signal, it should also remove the risk of accidentally trigger a potential other module in the robot.

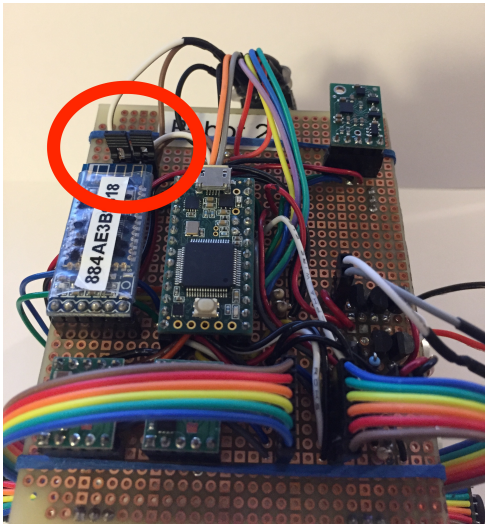



Figure 66: Two jumpers (redcircle) were added to the PCB, which were easy to reach and the risk of triggering two modules at once was eliminated.

In the code, it was added that the robot should react to an error by disabling the module and request a reconfiguration from the service station from chapter 9. As a part of chapter 9, the service station was coded to determine if an available robot could assist this robot by providing a working module.

An extra battery (see figure 67) was also added to the robot, which would kick in, if the 4 AA batteries in the Zumo failed. The system was based on Adafruit's PowerBoost C1000[62], which cost 19.95 USD, and a 3.7v Li-Po battery. There was spend a lot of time, attempting to add this power module in such a way that it could work just like an UPS[63], where it would be charged by the primary power source, but kick in if the power drops. Unfortunately, this wasn't possible with this type of module and the battery would therefore have to be charged manually, but worked as a proof of concept. A note is that the power module supplied 5.2 v for which reason a diode was added, that would make it only supply 4.5 v. This was not only to make sure that the module wouldn't get burned because of having voltage added to the supplying pin, when turned off, but also to have the effect of not adding any power before the 4 AA batteries got below 4.5 v. The battery was installed far away from the magnetometer as it was experiences how a battery makes a magnetic field, when being used. The electrical schematics can be found in Appendix C.

[62]Product page for Adafruit PowerBoost C1000: `https://www.adafruit.com/products/2465`
[63]UPS is short for uninterruptible power supply.

### 10.7.1 Evaluation

The added features of being able to trigger the error detection worked well and so did having the robot to disable a failed module and request for a reconfiguration.

When the robot should inform the service about the request for reconfiguration, it would first have to connect via the HM-10 module from chapter 6, which took approx. 8 seconds, before it could transmit the data string. The service station could find a candidate for assisting with the reconfiguration, based on the data that each robot had send to it as a part of their start-up routine (chapter 9).

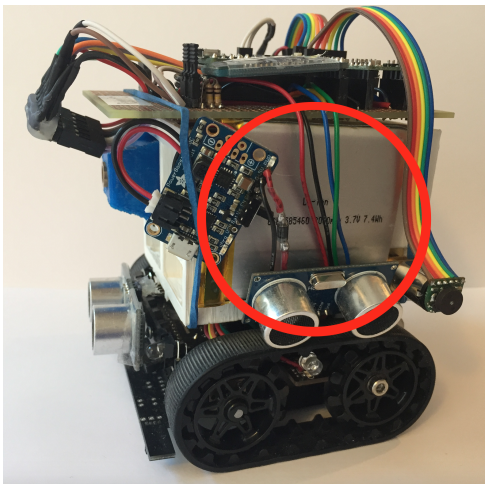


Figure 67: A back-up battery (red circle) was added to the robot and would kick in if the primary batteries failed. It was positioned far away from the magnetometer to limit the magnetic interference.

The added battery and power module worked well as a back-up: When having started the robot and set it to rotate, the primary batteries were removed, simulating a sudden drop of voltage. The back-up battery would kick in right away and the robot continued rotating as if nothing had happened. The Teensy was also providing correct print statements, which ensured that the Teensy hadn't been restarted by the sudden voltage drop. As the back-up battery only provides 4.5 v, and a substantial amount of hardware had been added to the robot, it could be beneficial to add a large capacitor as a temporarily power reserve. But it should be noted that the capacitor shouldn't be charged too fast and draw all the power from the Teensy and the Zumo at startup as this might interfere with their startup routines and prevent them from finishing these.

The installed battery had a capacity of 2000 mAh, but the runtime wasn't checked.

A list with addressed or improved design goals can be seen here:

**E-5** Must seek to find a solution, when a module has failed (module replacement)

**EX-2** Each module can be easily triggered to fail (for testing purpose)

**EX-5** The robot has a battery backup, that can power the robot while waiting for a battery module, if the primary battery fails

## 10.8 Discussion

In this chapter, it was shown how a simple error detection could be implemented into the modules of this system. To detect an error a certain method was called in the code every time the loop from the sketch was executed. This approach was chosen, because an interrupt based implementation could interfere with the communication device (HM-10 modules from chapter 6), which was controlled with AT commands where each of these couldn't be split by a small delay, as they then wouldn't be recognized by the device.

It could have been implemented with interrupts, but this would include to disable and enable the interrupts every time an AT command was send, but also to check for changes after each of these. If this was done, it would have made the code less structured. In this project, the timing wasn't critical as the error detection still may work very well - as shown in the iterations.

It was also shown how the robot could react to an error for each of the modules, which was demonstrated as the robot would turn the failed module off and request a new one from the service station. This approach should comply to most error detection system, as the system most likely would be interested in turning off a faulty module. Whether a system in general should replace a faulty module or try to repair it before requesting external help, completely depends on the overall system, but for this project, which has a focus on reconfiguration, it was a good way to trigger the need for a reconfiguration.

The error detection system is capable of determine if a module is faulty even though it isn't being used. As this is done by turning the module on at a very low power level, it is not truly able to detect error when it is turned off. But this was only done for the modules which had tool-like functionality: The motor or laser modules. The sensor module (LDR module) was never turned on, when measuring for errors, as the LDR inside of the module were constantly connected to 5v power source since it only used very little power. The same happens when a car is turned on (not started), and the electrical system checks if all the light bulbs are working and that the tires are inflated. Having to turn the tool-like modules slightly on to check for errors appears reasonable, as it is only enough let them run at the very lowest speeds.

In the last iteration, a back up battery was added to the robot, so that it could continue to work and request help, if the primary battery was broken. By added a diode with a 0.7 v drop from the 5.2 v power module, where the Li-Po battery was connected, the power module would supply the robot with 4.5 v and worked almost like an UPS. Unfortunately, it wasn't possible to implement the power module, so that it also could charge the battery, which means that it occasionally would have to be recharged from, i.e. the service station or a garage - but only after it had been in use. The reason for implementing the backup battery, was to show how to handle errors in critical parts of the robot: In this case by having a spare part. Other parts like a communication link, could require to have one to spare too, but a solution could also be to let the robot return to the service station, if such a part had failed.

## 10.9 Conclusion

This chapter has shown how a simple error detection system can be implemented and the high value of it, as it can prevent operating faulty equipment and have the system react when an error occurs. It was also shown how errors in a critical part of the system could be handled and allowed the system, in this case a robot, to carry on operating which gave it the opportunity to request help or move to a location, where it could be repaired.

The developed system, was founded in 6 design goals and 5 extreme goals, where the achieved ones are listed here:

**E-1** The robot is able to start measuring for errors within one second after module insertion
**E-2** The robot is able to measure for errors at least one time per second
**E-3** The robot won't measure for errors on empty ports
**E-4** The robot is able to detect errors on each type of module
**E-5** Must seek to find a solution, when a module has failed (module replacement)
**E-6** The robot is able to detect errors on activated module

**EX-2** Each module can be easily triggered to fail (for testing purpose)
**EX-3** The robot is able to detect errors on deactivated modules
**EX-4** The error detection is conducted with only one pin per port
**EX-5** The robot has a battery backup, that can power the robot while waiting for a battery module, if the primary battery fails

A robot, from this project, is now capable of detect if an error in of the modules occurs, in which case it can request a new one. This is the final part of the main topics of this project, and it should now be possible to use all of these to create a working reconfigurable system for mobile robots. In the following chapter, this will be put to the test!

# 11 Full reconfiguration experiment

All of the parts of implementing a robot system with modular reconfigurability have been developed and successfully tested in each of the previous five chapters. But to show that each part can perform well enough to lead up to the next part, this chapter will test them as *one* full system.

First the setup will be described to clarify which conditions the system is tested under, then a chapter will describe any interesting observations during the test and comment on this. The test is concluded with a discussion and a conclusion, which shows that the system is capable of reconfiguring a mobile robot with the help of another.

## 11.1 The setup

The setup is very similar to the experimental setup (chapter 5.4) and the exact same environment is used: As seen in figure 68, the primary area is the 118.5 x 105.0 cm flat arena covered with white paper. Because the robots are programmed to back a bit away from any work tasks, when they have requested or must assist with a reconfiguration, areas outside of the arena have been included as the robots otherwise would be right next to each other after they have backed up - which would make the test results misleading.

Figure 68: The setup of the full reconfiguration experiment: The environment from chapter 5.4 is used, and the two robots (blue circles) are set to ”work” by blinking on the wall in front of them with a red line laser. Then a module is simulated to fail and the reconfiguration process begins.

The reason for programming the robots to move away from the work position is that real-life implementations could easily have tools in the modules, that would be in physical contact with, i.e. the walls in the mine, and could affect the transferring of the modules. The robot, which has requested the reconfiguration, is also more available to the other robot, when not being right next to the wall.

All events by the robots and the service station will be recording and provided as documentation.

**Scenario**

The simulated scenario will be that two robots are working in the same location. Their task is to work with the modules, that they are equipped with, which will be a laser module for each of them. They will therefore turn the laser module on and off in a random pattern, while moving forward until they are 5 cm from the walls. To make the scenario more realistic, they will work on each section of the test area.

After a while, the error detection on one of the robots will be triggered, so the robot thinks the module is broken. It should then make a connection to the service station, where it requests another similar module so it can continue working.

## 11.2 Observations

The video footage of the scenario is available here[64], which any readers are *highly* encouraged to watch! In the video, it is shown that the entire reconfiguration goes rather well, as one of the robots can detect an error in the module and gets a working one after a reconfiguration request has been sent to the service station. All of this happens, as intended, and without any of the robots colliding into each other in a non-controlled way.

There are few things, which isn't ideal, though: The calibration (36 seconds) and communication (8 seconds for starting, opening, or ending) takes a long time, even though the calibration only is done once. Considering that the robots would spend 16 seconds every time they need to communicate, and that this might happen rather often so the service station has an updated status on all the robots, the implemented HM-10 modules aren't the best choice.

Because all of the topics in this project are put together sequentially, some of the actions by the robot is done twice after each other. An example of this is when the robot has completed the first part of the positioning (Rendezvous), where it stops because the other robot is 10 cm in front of it. The first step in the second part of the positioning is where the robot estimates the center of the other robot and increases/decreases the distance to 10 cm.

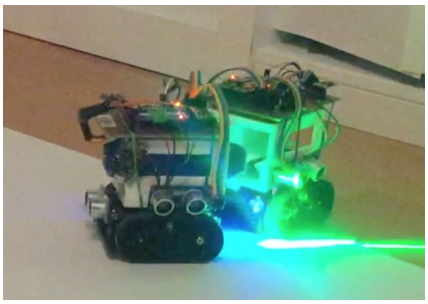


Figure 69: Final steps of the docking: Robot is almost undocking, when attempting to hit the receiver with its laser. (Robot will fully dock 2 seconds later, when moving fast left and right.)

Another thing is when the robot is starting the docking procedure and turns almost 360° to the left, instead of a few degrees to the right. This is a result of the inconsistent magnetometers, but appears foolish when the robot is exposing everything in the location to the laser - including humans. During the test the author wore safety goggles because of the strong laser. Locations where humans or animals are allowed access, doesn't appear as a good environment for the robots, as the laser could give irreversible eye damages.

The last thing to notice happened during the docking, when the robot actually had got so close to the other robot that it was fully docked, but almost undocked in the attempt to trigger the laser receiver on the other robot, (see figure 69), before realizing that it was docked. 2 seconds later, the robot had realized that it was very close to the other robot and would "wiggle" itself towards the other robot, which resulted in a perfect docking. During the wiggle procedure, the robot moved the other robot a few centimeters, which isn't ideal, as a larger scale of these robot could risk damaging each other under this maneuver.

[64]URL of video footage: `https://www.yndal.dk/owncloud/index.php/s/7jDDa36lYnWroA7`

## 11.3 Discussion

The full reconfiguration experiments clearly shows that not only is every subpart of the system working, but they can also work sequentially as *one* reconfigurable system for mobile robots: A robot was able to detect a simulated error in a module, request another module, and have it delivered by another robot.

A few optimization opportunities were also revealed under the tests, but none of them had a critical affect on the system: The most critical situation was where during the docking procedure, where the robot almost undocked, in the attempt to trigger the receiver on the other robot (see figure 69). The robot had actually docked well into the other robot, but because of its programmed behavior, it thought that the other robot had missed the laser and the robot now was turning too far to one side. Therefore it started to scan left and right to trigger the laser receiver (TEPT4400), which lead to the almost critical situation, where it was close to undock.

As the five topics of this project were put together and tested in the full experiment, some redundancy were also experiences. This is what to expect when combing that many procedures into a single one, as each of them has to take its precaution and make sure that the environment is as each procedure think it is: The very first step of the alignment procedure, is to determine the direction to the other robot, which also is done in the previous procedure, the rendezvous. Even though these are not serious errors/flaws, they are a waste of time, battery, and hardware. If the system is to be used further as a single procedure, it is therefore advisable to have a quick look in the code before fully implementing it.

When the system was put together, it was really obvious how long time, the communication actually takes. Spending 8 seconds every time a communication link needs to be established or closed, apparently summed up substantially, and another mean of communication might be worth considering if the robots shouldn't waste too much time, which also is mentioned in chapter 6.

## 11.4 Conclusion

Even though the system has showed to have minor optimization potentials, this full experiment clearly showed that all of the subparts of this project is capable of working sequentially together, turning them into *one* large system. This project can therefore contribute to more than just addressing the five primary topics of the project, as it also has shown how these topics can be combined and make the foundation of a reconfigurable system for robots.

The optimization potentials are not in any way critical as they all worked generally well, but they could be matured to increase the potentials of the system.

# 12 Discussion

Throughout this project, a total of 5 chapters have focused on each part of the hypothesis about if it is possible to create a mobile reconfigurable robot system. Each of these chapters have developed, tested, and matured, their topic in an iterative way, which in the end all were considered successful, even though minor issues could be matured. All of these were then combined into one full experiment, which also was considered a success. But can these robots even be used in underground mines and industrial halls? They are so small, only have power for approx. 45 minutes when the laser is on, and none other parts, except for the modules, can be replaced!

The immediate answer is no, as the terrain in an mine is far from suitable for these small robots, and the amount of force required from the machinery is by far too much compared to what this little robot can deliver. But if the robot was scaled up to, i.e. the size of a car or bigger, it would absolutely have potential to address these requirements for mining machinery: Clearly it could carry a larger and better power source, but also the tools for the modules could be scaled up to contain i.e. a hydraulic hammer or drill, which may break down the walls in the mine. The amount of sensor may also be increased, as the proposed system has been developed for the lowest amount of simply, low-cost sensors.

The system isn't capable to navigating on roads or in tunnels, but by combing this project with, i.e. the Google Self-Driving Car project[22], one can easily imagine how relief supplies can be sent into cities like i.e. Aleppo, Syria, which are in great need of these: If modules were filled with relief supplies and these were sent to a stationary storage instead of another robot, the robot could deliver these and bring back empty ones. The system could work with the stationary storage as it is only the robot delivering the modules, which is required to be mobile.

But as this project has tried to address the gap between fully modular robots and most robots/every day items, which are non-modular, one of its issues is that it can only exchange the modules. Considering a scenario, where the robot's caterpillar tracks broke, the robot would be unable to get assistance by another robot, as these aren't modular replaceable, and a trained technician would have to repair this, which is one of the situations that this project attempts to prevent. It could be argued that since the robot is capable of finding the other robot, positioning itself correctly, and exchanging *a* module, one could implement for the caterpillar tracks to be a module and exchange this too. Another possibility is to have a module which could lock two robots together and the robot could drag the other robot to the nearest service station.

# 13 Conclusion

After having developed each part of the proposed system and conducted a full reconfiguration experiment, the system can be concluded as successful: A total of 43 out of 44 required design goals were achieved, while 8 out of 17 extreme design goals were achieved. The one design goal that wasn't achieved was that the robot should be able to detect when it is fully docked, but never showed to be an issue at any point for the entire project.

The proposed system consists of a service station, a robot, and four types of modules. It is capable of having a robot detect if a module fails and request a reconfiguration. The system can also choose another robot, which can locate, align, and dock with the robot, before exchanging the requested module.

Even though the robots are too small to be working with heavy equipment, the design of the robot can be scaled up and the current behavior of the robot be used. The hypothesis of creating *"...a system with robots, where the robots are able to locate and repair/reconfigure each other."* has therefore been proven.

All source files are available from GitHub[84].

# References


[1] Paulo Aguiar, Luís Mendonça, and Vasco Galhardo. Opencontrol: a free opensource software for video tracking and automated control of behavioral mazes. *Journal of neuroscience methods*, 166(1):66–72, 2007.

[2] DJ Arbuckle and Aristides AG Requicha. Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations. *Autonomous Robots*, 28(2):197–211, 2010.

[3] Jungyun Bae, Sooyong Lee, and Jae-Bok Song. Use of coded infrared light for mobile robot localization. *Journal of mechanical science and technology*, 22(7):1279–1286, 2008.

[4] Joseph Nsasi Bakambu and Vladimir Polotski. Autonomous system for navigation and surveying in underground mines. *Journal of Field Robotics*, 24(10):829–847, 2007.

[5] Curt Bererton and Pradeep K Khosla. Towards a team of robots with repair capabilities: a visual docking system. In *Experimental Robotics VII*, pages 333–342. Springer, 2001.

[6] Joydeep Biswas and Manuela M Veloso. Wifi localization and navigation for autonomous indoor mobile robots. 2010.

[7] Wojtek Borowicz. How precise are estimote beacons?, 2015. Accessed 10:56, Oct. 30th, 2016 at `https://community.estimote.com/hc/en-us/articles/201302836-How-precise-are-Estimote-Beacons-`.

[8] Daniele Brambilla, Luca Massimiliano Capisani, Antonella Ferrara, and Pierluigi Pisu. Fault detection for robot manipulators via second-order sliding modes. *IEEE Transactions on Industrial Electronics*, 55(11):3954–3963, 2008.

[9] David Brandt, David Johan Christensen, and Henrik Hautop Lund. Atron robots: versatility from self-reconfigurable modules. In *2007 International Conference on Mechatronics and Automation*, pages 26–32. IEEE, 2007.

[10] Casey Carlson, Andrew Drenner, Ian Burt, and Nikolaos Papanikolopoulos. Modular mobile docking station design. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4722–4727. IEEE, 2006.

[11] Chih Liang Chen, Song Huei Huang, and Jia Hong Zhou. Mobile robot localization by tracking built-in encoders. In *Computer, Consumer and Control (IS3C), 2014 International Symposium on*, pages 840–843. IEEE, 2014.

[12] Long Cheng, Cheng-Dong Wu, and Yun-Zhou Zhang. Indoor robot localization based on wireless sensor networks. *IEEE Transactions on Consumer Electronics*, 57(3):1099–1104, 2011.

[13] David Johan Christensen. Evolution of shape-changing and self-repairing control for the atron self-reconfigurable robot. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2539–2545. IEEE, 2006.

[14] David Johan Christensen. Experiments on fault-tolerant self-reconfiguration and emergent self-repair. In *2007 IEEE Symposium on Artificial Life*, pages 355–361. IEEE, 2007.

[15] David Johan Christensen, David Brandt, and Kasper Støy. Towards artificial atron animals: scalable anatomy for self-reconfigurable robots. In *The RSS Workshop on Self-Reconfigurable Modular Robots*, 2006.

[16] David Johan Christensen, Ulrik Pagh Schultz, and Kasper Støy. A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots. *Robotics and Autonomous Systems*, 61(9):1021–1035, 2013.

[17] Hung-Yuan Chung, Chun-Cheng Hou, and Yu-Shan Chen. Indoor intelligent mobile robot localization using fuzzy compensation and kalman filter to fuse the data of gyroscope and magnetometer. *IEEE Transactions on Industrial Electronics*, 62(10):6436–6447, 2015.

[18] DaWei Dai, GuoLai Jiang, Junbo Xin, Xiang Gao, LiangLiang Cui, YongSheng Ou, and GuoQiang Fu. Detecting, locating and crossing a door for a wide indoor surveillance robot. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pages 1740–1746. IEEE, 2013.

[19] Carrick Detweiler, Marsette Vona, Yeoreum Yoon, Seung-kook Yun, and Daniela Rus. Self-assembling mobile linkages. *IEEE Robotics and Automation Magazine*, 14(4):45, 2007.

[20] Frédéric Evennou and François Marx. Advanced integration of wifi and inertial navigation systems for indoor mobile positioning. *Eurasip journal on applied signal processing*, 2006:164–164, 2006.

[21] Zahid Farid, Rosdiadee Nordin, and Mahamod Ismail. Recent advances in wireless indoor localization techniques and system. *Journal of Computer Networks and Communications*, 2013, 2013.

[22] Google. Google self-driving car project, 2016. Accessed 12:52, Nov. 23rd, 2016 at https://www.google.com/selfdrivingcar/.

[23] Ernesto Martín Gorostiza, José Luis Lázaro Galilea, Franciso Javier Meca Meca, David Salido Monzú, Felipe Espinosa Zapata, and Luis Pallarés Puerto. Infrared sensor system for mobile-robot positioning in intelligent spaces. *Sensors*, 11(5):5416–5438, 2011.

[24] Roderich Groß and Marco Dorigo. Self-assembly at the macroscopic scale. *Proceedings of the IEEE*, 96(9):1490–1508, 2008.

[25] Hongling Han and Fenglei Yang. Path planning of an indoor mobile robot navigated by infrared. In *E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on*, pages 1–4. IEEE, 2010.

[26] Anders Hojmark, Phillip Phoelich, and Morten Frederik Therkildsen. Autonomous robot repair system (arrs), May 2014. Bachelor thesis, supervisor: Kasper Støy.

[27] Wei Hong, Shaoping Wang, and Daoyan Shui. Reconfigurable robot system based on electromagnetic design. In *Fluid Power and Mechatronics (FPM), 2011 International Conference on*, pages 570–575. IEEE, 2011.

[28] Sungsik Huh, Unghui Lee, Hyunchul Shim, Jong-Beom Park, and Jong-Ho Noh. Development of an unmanned coal mining robot and a tele-operation system. In *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, pages 31–35. IEEE, 2011.

[29] iRobot. irobot's danish website, 2016. Accessed 14:13, Nov. 16th, 2016 at http://www.irobot.dk/.

[30] Bojan Jakimovski and Erik Maehle. Artificial immune system based robot anomaly detection engine for fault tolerant robots. In *International Conference on Autonomic and Trusted Computing*, pages 177–190. Springer, 2008.

[31] Carlos Fernando Crispim Junior, Cesar Nonato Pederiva, Ricardo Chessini Bose, Vitor Augusto Garcia, Cilene Lino-de Oliveira, and José Marino-Neto. Ethowatcher: validation of a tool for behavioral and video-tracking analysis in laboratory animals. *Computers in biology and medicine*, 42(2):257–264, 2012.

[32] Akiya Kamimura, Eiichi Yoshida, Satoshi Murata, Haruhisa Kurokawa, Kohji Tomita, and Shigeru Kokaji. A self-reconfigurable modular robot (mtran)—hardware and motion planning software—. In *Distributed Autonomous Robotic Systems 5*, pages 17–26. Springer, 2002.

[33] Serge Kernbach, Benjamin Girault, and Olga Kernbach. On self-optimized self-assembling of heterogeneous multi-robot organisms. In *Bio-inspired self-organizing robotic systems*, pages 123–141. Springer, 2011.

[34] Serge Kernbach, Florian Schlachter, Raja Humza, Jens Liedke, Sergej Popesku, Sheila Russo, Tommaso Ranzani, Luigi Manfredi, Cesare Stefanini, Rene Matthias, et al. Heterogeneity for increasing performance and reliability of self-reconfigurable multi-robot organisms. *arXiv preprint arXiv:1109.2288*, 2011.

[35] Serge Kernbach, Oliver Scholz, Kanako Harada, Sergej Popesku, Jens Liedke, Humza Raja, Wenguo Liu, Fabio Caparrelli, Jaouhar Jemai, Jiri Havlik, et al. Multi-robot organisms: State of the art. *arXiv preprint arXiv:1108.5543*, 2011.

[36] Myungsik Kim and Nak Young Chong. Rfid-based mobile robot guidance to a stationary target. *Mechatronics*, 17(4):217–229, 2007.

[37] Myungsik Kim, Nak Young Chong, Hyo-Sung Ahn, and Wonpil Yu. Rfid-enabled target tracking and following with a mobile robot using direction finding antennas. In *2007 IEEE International Conference on Automation Science and Engineering*, pages 1014–1019. IEEE, 2007.

[38] MyungSik Kim, Hyung Wook Kim, and Nak Young Chong. Automated robot docking using direction sensing rfid. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4588–4593. IEEE, 2007.

[39] Myungsik Kim, Kwangsoo Kim, and Nakyoung Chong. Rfid based collision-free robot docking in cluttered environment. *Progress In Electromagnetics Research*, 110:199–218, 2010.

[40] Andrejs Kostromins, Vitalijs Osadcuks, et al. Infrared active beacon application evaluation for mobile robot localization in agriculture. *Engineering for Rural Development (Latvia)*, 2014.

[41] Hakan Koyuncu and Shuang Hua Yang. A survey of indoor positioning and object locating systems. *IJCSNS International Journal of Computer Science and Network Security*, 10(5):121–128, 2010.

[42] Michael DM Kutzer, Matthew S Moses, Christopher Y Brown, Mehran Armand, David H Scheidt, and Gregory S Chirikjian. Design of a new independently-mobile reconfigurable modular robot. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2758–2764. IEEE, 2010.

[43] Frederic Lambert. Tesla's battery swapping magic revealed in new patent application drawings, 2016. Accessed 10:39, Nov. 2nd, 2016 at https://electrek.co/2016/10/25/teslas-battery-swapping-magic-revealed-patent-drawings/.

[44] HYK Lau, AWY Ko, and TL Lau. The design of a representation and analysis method for modular self-reconfigurable robots. *Robotics and Computer-Integrated Manufacturing*, 24(2):258–269, 2008.

[45] Sooyong Lee and Jae-Bok Song. Mobile robot localization using infrared light reflecting landmarks. In *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*, pages 674–677. IEEE, 2007.

[46] Sooyong Lee and Jae-Bok Song. Use of coded infrared light as artificial landmarks for mobile robot localization. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1731–1736. IEEE, 2007.

[47] Olivier Lefebvre and Florent Lamiraux. Docking task for nonholonomic mobile robots. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 3736–3741. IEEE, 2006.

[48] Aiguo Li and Bingrong Hong. On-line control flow error detection using relationship signatures among basic blocks. *Computers & electrical engineering*, 36(1):132–141, 2010.

[49] Dazhai Li, Hualei Fu, and Wei Wang. Ultrasonic based autonomous docking on plane for mobile robot. In *2008 IEEE International Conference on Automation and Logistics*, pages 1396–1401. IEEE, 2008.

[50] Georgios Lidoris, Florian Rohrmuller, Dirk Wollherr, and Martin Buss. The autonomous city explorer (ace) project—mobile robot navigation in highly populated urban environments. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1416–1422. IEEE, 2009.

[51] Wenguo Liu and Alan FT Winfield. Autonomous morphogenesis in self-assembling robots using ir-based sensing and local communications. In *International Conference on Swarm Intelligence*, pages 107–118. Springer, 2010.

[52] Thomas Lochmatter, Pierre Roduit, Chris Cianci, Nikolaus Correll, Jacques Jacot, and Alcherio Martinoli. Swistrack-a flexible open source tracking software for multi-agent systems. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4004–4010. IEEE, 2008.

[53] Ren C Luo, Ogst Chen, and Pei Hsien Lin. Indoor robot/human localization using dynamic triangulation and wireless pyroelectric infrared sensory fusion approaches. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1359–1364. IEEE, 2012.

[54] Ren C Luo, Chien H Huang, and Chun Y Huang. Search and track power charge docking station based on sound source for autonomous mobile robot applications. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1347–1352. IEEE, 2010.

[55] Ren C Luo, Chung T Liao, and Shih C Lin. Multi-sensor fusion for reduced uncertainty in autonomous mobile robot docking and recharging. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2203–2208. IEEE, 2009.

[56] Ren C Luo and Kuo L Su. Multilevel multisensor-based intelligent recharging system for mobile robot. *IEEE Transactions on Industrial Electronics*, 55(1):270–279, 2008.

[57] Martin Magnusson and Tom Duckett. A comparison of 3d registration algorithms for autonomous underground mining vehicles. In *Proceedings of the European Conference on Mobile Robotics (ECMR 2005)*, pages 86–91, 2005.

[58] Ling Mao, Jiapin Chen, Zhenbo Li, and Dawei Zhang. Relative localization method of multiple micro robots based on simple sensors. *International Journal of Advanced Robotic Systems*, 10, 2013.

[59] David Leal Martínez and Aarne Halme. Marsim, a simulation of the marsubots fleet using netlogo. In *Distributed Autonomous Robotic Systems*, pages 79–87. Springer, 2016.

[60] Marek Matusiak, Janne Paanajärvi, Pekka Appelqvist, Mikko Elomaa, Mika Vainio, Tomi Ylikorpi, and Aarne Halme. A novel marsupial robot society: towards long-term autonomy. In *Distributed Autonomous Robotic Systems 8*, pages 523–532. Springer, 2009.

[61] Yan Meng, Yuyang Zhang, and Yaochu Jin. Autonomous self-reconfiguration of modular robots by evolving a hierarchical mechanochemical model. *IEEE Computational Intelligence Magazine*, 6(1):43–54, 2011.

[62] Wang Ming-Xiao, Zhang Tao, Xie Miao-Rong, Zhang Bin, and Jia Ming-Qiu. Analysis of national coal-mining accident data in china, 2001–2008. *Public Health Reports*, 126(2):270, 2011.

[63] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002.

[64] Paul Moubarak and Pinhas Ben-Tzvi. Modular and reconfigurable mobile robotics. *Robotics and Autonomous Systems*, 60(12):1648–1663, 2012.

[65] Satoshi Murata and Haruhsa Kurokawa. Self-reconfigurable robots. *IEEE Robotics & Automation Magazine*, 14(1):71–78, 2007.

[66] Nicolás Navarro-Guerrero, Cornelius Weber, Pascal Schroeter, and Stefan Wermter. Real-world reinforcement learning for autonomous humanoid robot docking. *Robotics and Autonomous Systems*, 60(11):1400–1407, 2012.

[67] NIOSH. National institute for occupational safety and health (niosh) - statistics: All mining statistics, 2016. Accessed 13:42, Dec. 7th, 2016 at `https://www.cdc.gov/niosh/mining/statistics/allmining.html`.

[68] Noldus. Noldus - video tracking with ethovision xt, 2016. Accessed 11:22, Sep. 8th, 2016 at `http://www.noldus.com/EthoVision-XT/more-about-ethovision-xt`.

[69] Jung H Oh, Doojin Kim, and Beom H Lee. An indoor localization system for mobile robots using an active infrared positioning sensor. *Journal of Industrial and Intelligent Information Vol*, 2(1), 2014.

[70] Panlab. Panlan - smart video tracking software, 2016. Accessed 10:47, Sep. 8th, 2016 at `http://www.panlab.com/en/products/smart-video-tracking-software-panlab`.

[71] Mike Peasgood, Christopher Michael Clark, and John McPhee. A complete and scalable strategy for coordinating multiple robots within roadmaps. *IEEE Transactions on Robotics*, 24(2):283–292, 2008.

[72] Alfonso Pérez-Escudero, Julián Vicente-Page, Robert C Hinz, Sara Arganda, and Gonzalo G de Polavieja. idtracker: tracking individuals in a group by automatic identification of unmarked animals. *Nature methods*, 11(7):743–748, 2014.

[73] Miguel Pinto, António Paulo Moreira, and Anibal Matos. Localization of mobile robots using an extended kalman filter in a lego nxt. *IEEE transactions on education*, 55(1):135–144, 2012.

[74] Pololu. Pololu minimu-9 + arduino ahrs (attitude and heading reference system), 2016. Accessed 13:57, Oct. 14th, 2016 at https://github.com/pololu/minimu-9-ahrs-arduino.

[75] Pololu. Zumo 32u4 robot kit (no motors), 2016. Accessed 12:00, Aug. 17th, 2016 at https://www.pololu.com/product/3124.

[76] Se-gon Roh, Jae Hoon Park, Young Hoon Lee, Young Kouk Song, Kwang Woong Yang, Moosung Choi, Hong-Seok Kim, Hogil Lee, and Hyouk Ryeol Choi. Flexible docking mechanism with error-compensation capability for auto recharging system of mobile robot. *Int. J. of Control, Automation, and Systems*, 6(5):731–739, 2008.

[77] Michael Rubenstein and Wei-Min Shen. Scalable self-assembly and self-repair in a collective of robots. In *2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 1484–1489. IEEE, 2009.

[78] Behnam Salemi, Mark Moll, and Wei-Min Shen. Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3636–3641. IEEE, 2006.

[79] Steven Scheding, Gamini Dissanayake, Eduardo Mario Nebot, and Hugh Durrant-Whyte. An experiment in autonomous navigation of an underground mining vehicle. *IEEE Transactions on robotics and Automation*, 15(1):85–95, 1999.

[80] Milo C Silverman, Dan Nies, Boyoon Jung, and Gaurav S Sukhatme. Staying alive: A docking station for autonomous robot recharging. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 1050–1055. IEEE, 2002.

[81] Guangming Song, Hui Wang, Jun Zhang, and Tianhua Meng. Automatic docking system for recharging home surveillance robots. *IEEE Transactions on Consumer Electronics*, 57(2):428–435, 2011.

[82] Kasper Støy. Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems*, 54(2):135–141, 2006.

[83] Kasper Støy, W-M Shen, and Peter M Will. A simple approach to the control of locomotion in self-reconfigurable robots. *Robotics and Autonomous Systems*, 44(3):191–199, 2003.

[84] Lars Yndal Sørensen. Open repository with all source files, 2016. Available from 14:00, Jan. 2nd, 2017 at https://github.com/Yndal/Thesis.

[85] Shuai Tao, Mineichi Kudo, Bing-Nan Pei, Hidetoshi Nonaka, and Jun Toyama. Multi-person locating and their soft tracking in a binary infrared sensor network. *IEEE Transactions on Human-Machine Systems*, 45(5):550–561, 2015.

[86] Hamid Teimoori and Andrey V Savkin. Equiangular navigation and guidance of a wheeled mobile robot based on range-only measurements. *Robotics and Autonomous Systems*, 58(2):203–215, 2010.

[87] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*, chapter 9, 10, 11, and 12. MIT Press, 1 edition, 2005.

[88] ugo basile. ugo basile - 60000 - any-maze video-tracking software, 2016. Accessed 13:57, Sep. 8th, 2016 at `http://www.ugobasile.com/catalogue/application/memory__learning__alzheimer/product/60000_any_maze_video_tracking_software.html`.

[89] Alex Couture-Beil Richard T Vaughan. Adaptive mobile charging stations for multi-robot systems. 2009.

[90] Jue Wang, Fadel Adib, Ross Knepper, Dina Katabi, and Daniela Rus. Rf-compass: robot object manipulation using rfids. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 3–14. ACM, 2013.

[91] Wei Wang, Zongliang Li, Wenpeng Yu, and Jianwei Zhang. An autonomous docking method based on ultrasonic sensors for self-reconfigurable mobile robot. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 1744–1749. IEEE, 2009.

[92] Wei Wang, Wenpeng Yu, and Houxiang Zhang. Jl-2: A mobile multi-robot system with docking and manipulating capabilities. *International Journal of Advanced Robotic Systems*, 7(1):9–18, 2010.

[93] Hongxing Wei, Yingpeng Cai, Haiyuan Li, Dezhong Li, and Tianmiao Wang. Sambot: A self-assembly modular robot for swarm robot. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 66–71. IEEE, 2010.

[94] Hongxing Wei, Youdong Chen, Jindong Tan, and Tianmiao Wang. Sambot: A self-assembly modular robot system. *IEEE/ASME Transactions on Mechatronics*, 16(4):745–757, 2011.

[95] Brian H Wilcox, Todd Litwin, Jeff Biesiadecki, Jaret Matthews, Matt Heverly, Jack Morrison, Julie Townsend, Norman Ahmad, Allen Sirota, and Brian Cooper. Athlete:

A cargo handling and manipulation robot for the moon. *Journal of Field Robotics*, 24(5):421–434, 2007.

[96] Yi-Cheng Wu, Ming-Chang Teng, and Yi-Jeng Tsai. Robot docking station for automatic battery exchanging and charging. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1043–1046. IEEE, 2009.

[97] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

[98] Mark Yim, Babak Shirmohammadi, Jimmy Sastra, Michael Park, Michael Dugan, and Camillo J Taylor. Towards robotic self-reassembly after explosion. 2007.

[99] Mark Yim, Paul White, Michael Park, and Jimmy Sastra. Modular self-reconfigurable robots. In *Encyclopedia of complexity and systems science*, pages 5618–5631. Springer, 2009.

[100] Da Zhang, Feng Xia, Zhuo Yang, Lin Yao, and Wenhong Zhao. Localization technologies for indoor human tracking. In *2010 5th International Conference on Future Information Technology*, pages 1–6. IEEE, 2010.

[101] Houxiang Zhang, Wei Wang, Zhicheng Deng, Guanghua Zong, and Jianwei Zhang. A novel reconfigurable robot for urban search and rescue. *International Journal of Advanced Robotic Systems*, 3(4):359–366, 2006.

[102] Victor Zykov, Efstathios Mytilinaios, Mark Desnoyer, and Hod Lipson. Evolved and designed self-reproducing modular robotics. *IEEE Transactions on robotics*, 23(2):308–319, 2007.

# A Mechanical ideas for the module transferring

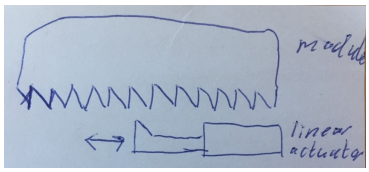
The figures show the initial ideas for how a module could be transferred from one robot to another. The upper left image shows how the module has a jagged bottom, which an linear actuator could grip into. By moving the linear actuator back and forth, the module would slowly be pushed away - onto the other robot.

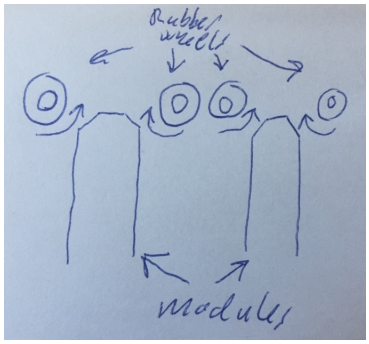
The center image is showing four rubber wheels from above. The four wheels should apply enough friction to grip the modules and insert/eject these by simply rotating. As the modules would be a bit narrow in the front, a precise insertion could be avoided, as the rubber wheels would guide the module into position.

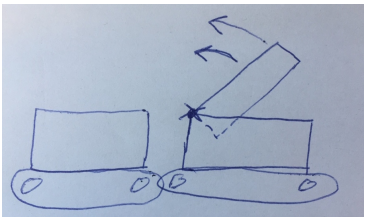
The upper right image shows how the module should be attached to a bar at the end of a robot. The module should then be rotated around this bar and after rotating 180° + the height of the module, it would safely land on the other robot. Some of the issues about this approach was how to rotate it safely around the bar, while being able to detach it when done.

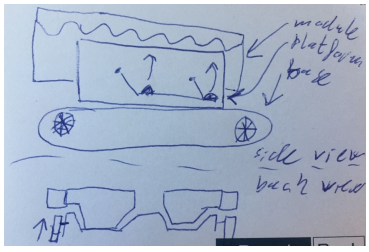
The image at the bottom to the left shows how a bar rotating around two joints could move a module: The bar should be attached to the joint with movable arms. When the arm were rotated, the bar would follow and lift up the module while also moving it away from and onto the other robot.

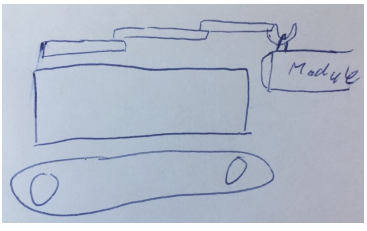
In the lower, right image is shown how an extendable arm could move the module: The lower part of the arm in the back is mounted to the robot case, while the end closest to the module should be able to be raised a bit, before extending the arm and moving the module. This would avoid friction, but appeared rather complex and error prone to develop.



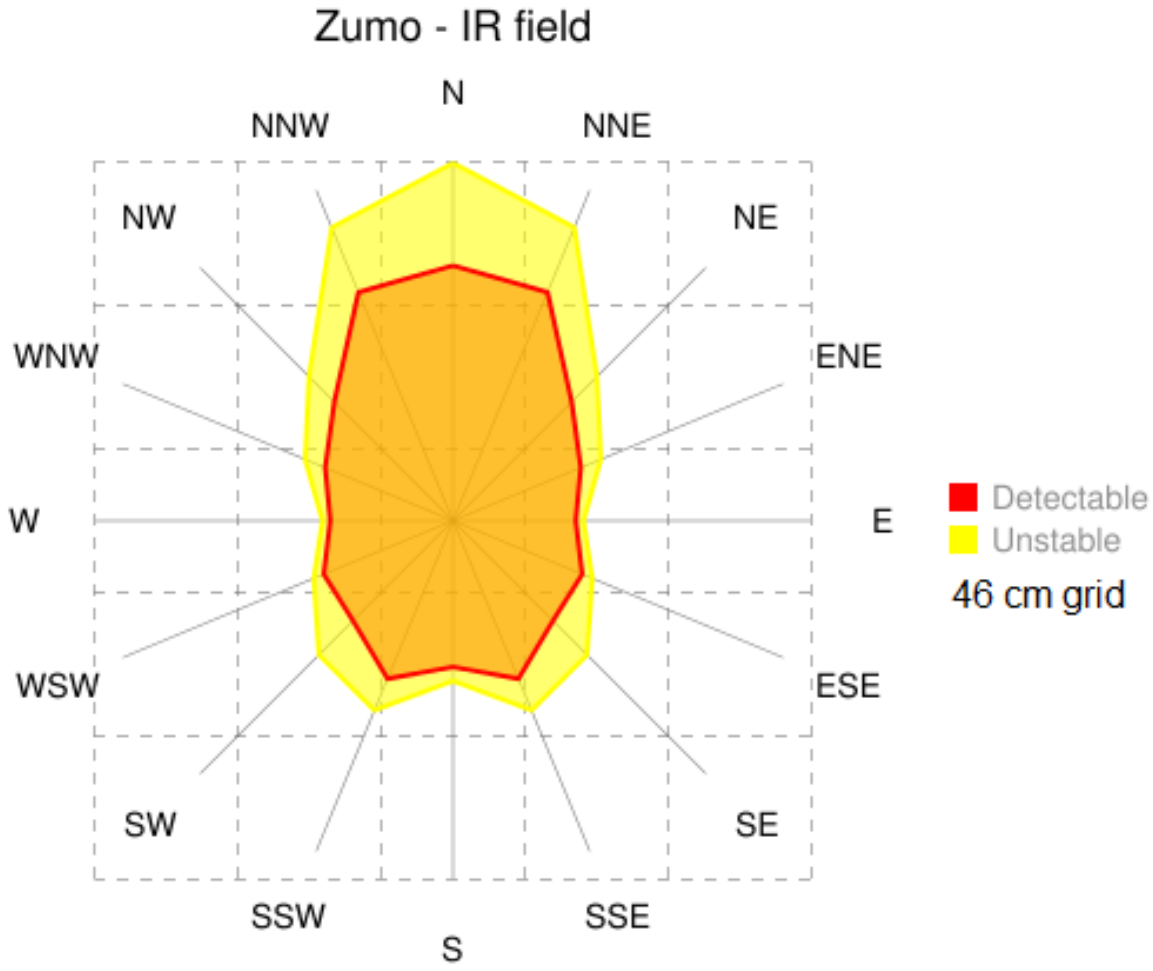





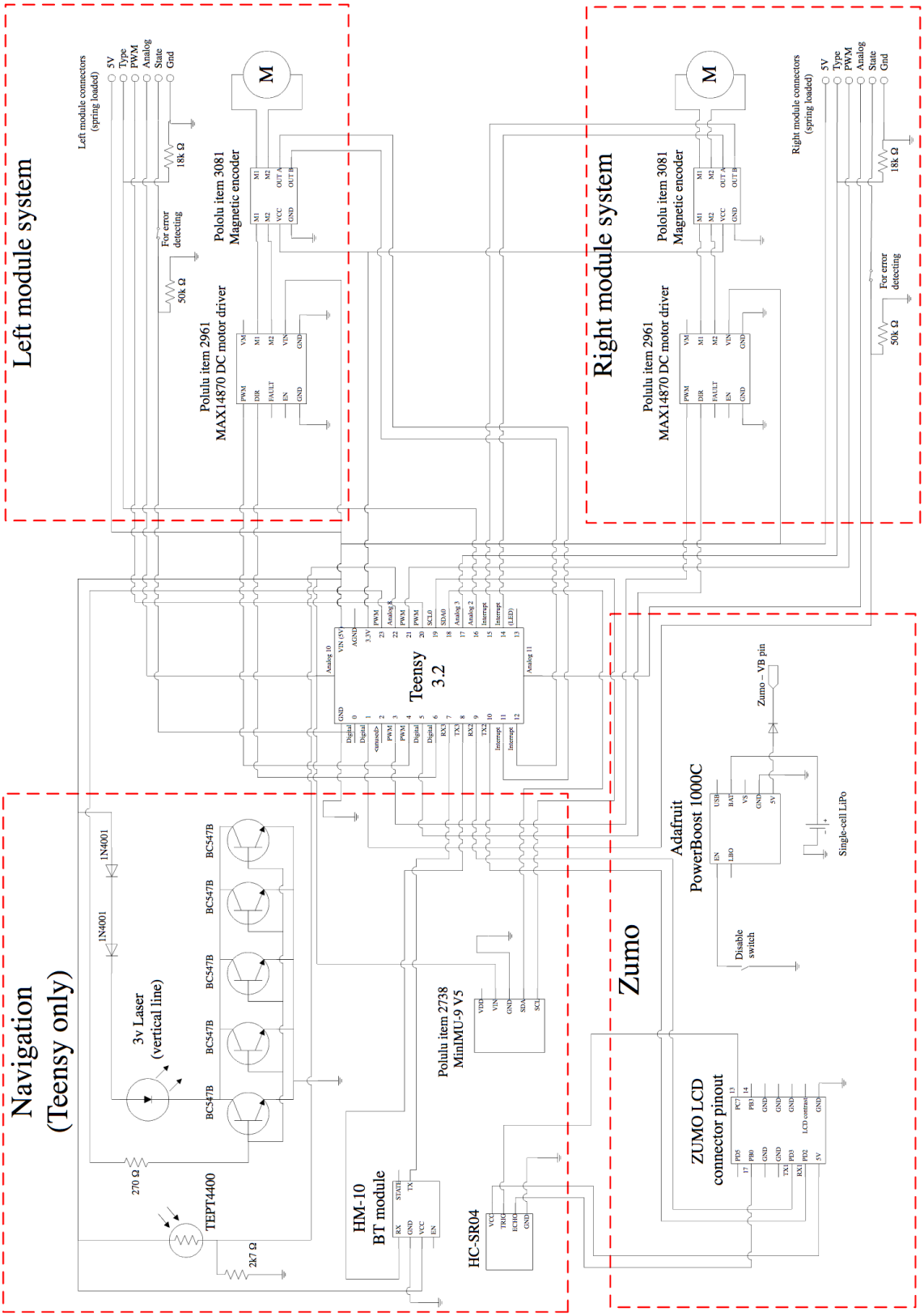



# B Zumo IR field

Map of the detectable IR field from a Zumo, with additional IR LEDs in the back, which is standing in the center pointing North with the IR on. Another Zumo is pointing towards the Zumo in the center: The red area is where the signal was well received, while the yellow area is where it was unstable. Outside of the yellow field, no IR signals were detected.

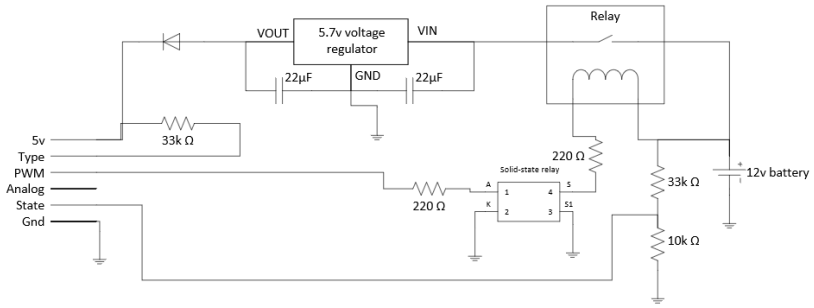


# C Electrical schematics - Robot

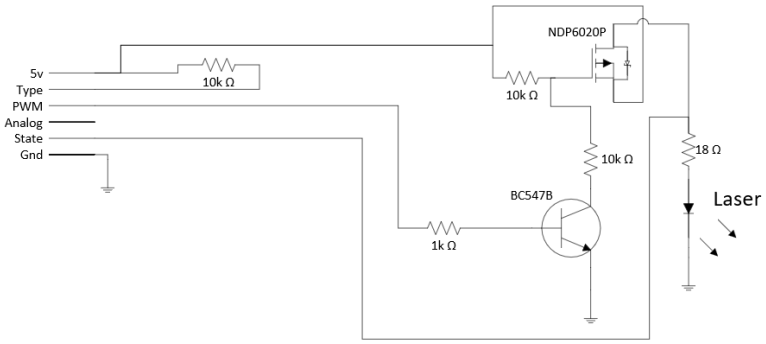


# D Electrical schematics - Modules

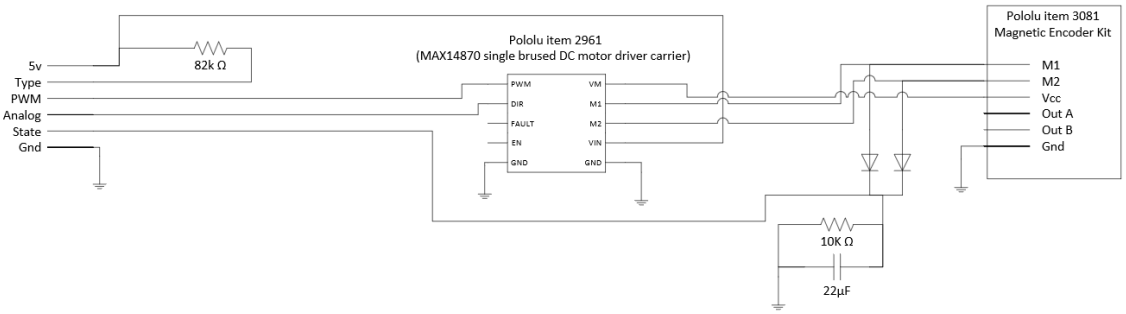
## Battery module




## Laser module




## Motor module




## LDR module