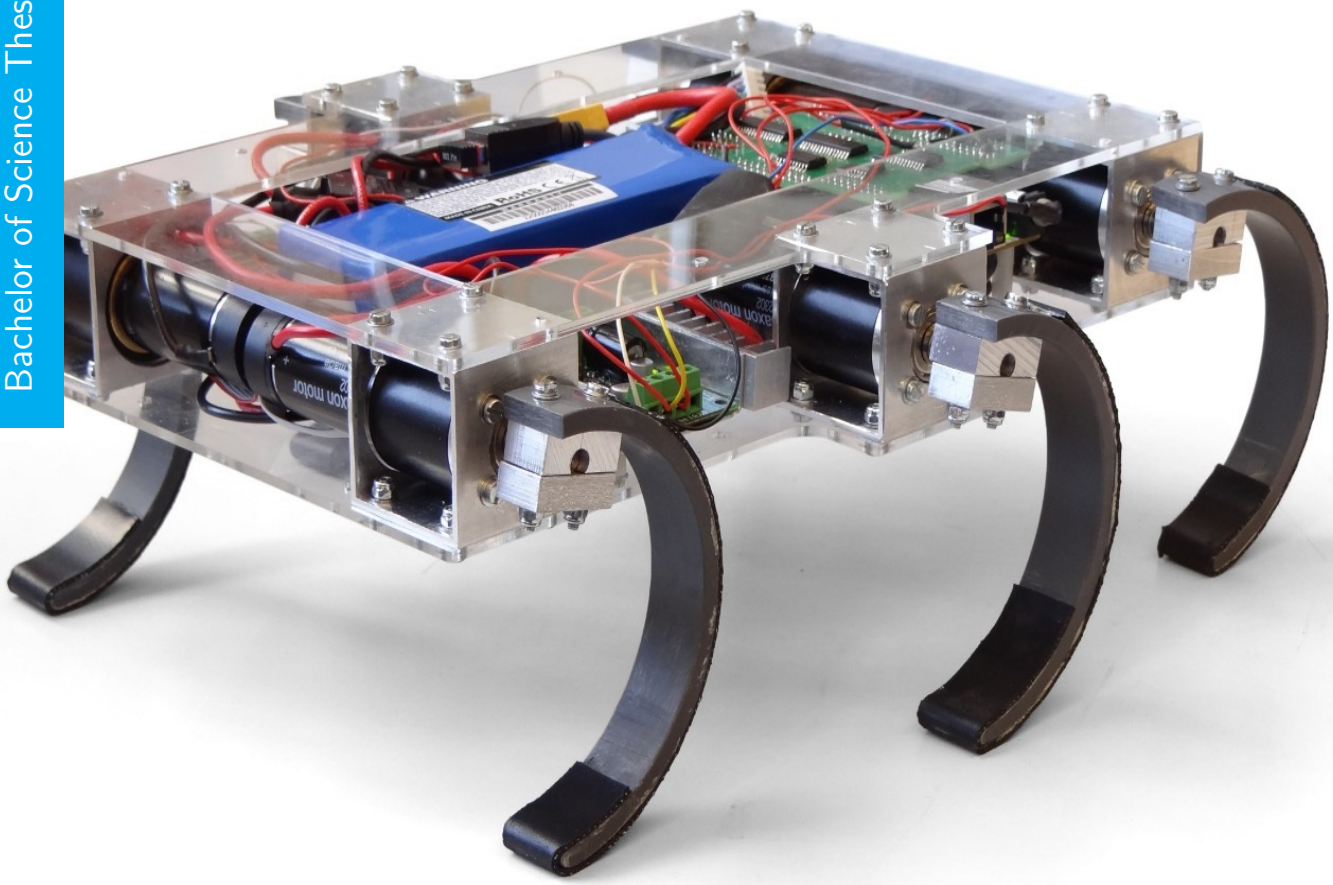
Fundamentals For An Autonomous Zebro Swarm

M. Ceelen 4110196
C. van der Geer 4156145
F. Rouwen 4076605
S. Seuren 4083121

Bachelor of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Fundamentals For An Autonomous Zebro Swarm

Bachelor of Science Thesis

by

Marcel Ceelen 4110196
marcel_ceelen@hotmail.com
+31 6 30 71 86 20

Cees van der Geer 4156145
ceesvdgeer@gmail.com
+31 6 52 26 46 13

Floris Rouwen 4076605
florisrouwen@outlook.com
+31 6 16 41 62 52

Stijn Seuren 4083121
seuren.stijn@gmail.com
+31 6 28 52 11 48

Thesis for the Bachelor Endproject (WBTP303),
in partial fulfillment of the requirements for the degree of

Bachelor of Science
Mechanical Engineering

Faculty:	Mechanical, Maritime and Materials Engineering
Project duration:	September 1, 2014 – January 13, 2015
Supervisors:	Dr. G.A. Delgado Lopes G.A.Delgadolopes@tudelft.nl +31 15 2785489
	Dr. Ir. C.J.M. Verhoeven C.J.M.Verhoeven@tudelft.nl +31 15 2786482

TU Delft Delft University of Technology

Acknowledgements

We would like to thank our supervisor Dr.ir. C.J.M. Verhoeven for his assistance during the project with fresh thoughts and guidance. Also many thanks to the other people in the Zebro group for helping us out when needed, in particular Maneesh Kumar Verma, Arash Noroozi and Kevin McElligott.

Stijn, Marcel, Cees, Floris
Delft University of Technology
January 13, 2015

”The Three Laws of Robotics:

1. *A robot may not injure a human being or, through inaction, allow a human being to come to harm;*
2. *A robot must obey the orders given it by human beings except where such orders would conflict with the First Law;*
3. *A robot must protect its own existence as long as such protection does not conflict with the First or Second Law”*

— Isaac Asimov

Summary

This thesis describes the research to the fundamentals of an autonomous Zebro Swarm. Swarm is defined in this thesis as:

> *"A swarm is a large number of homogenous, unsophisticated agents that interact locally among themselves and their environment, without any central control or management to yield a global behaviour to emerge."*
>
> [1, p. 25]

The bigger perspective is to have the units within a swarm operate for a potentially infinite amount of time, thus making algorithms and energy use as effective as possible. The research is split up in different sections: sensors, communication and behaviour. The behaviour is split again: individual behaviour, anti collision behaviour and swarming behaviour.

The research on the sensors and communication points out that the Bluetooth Low Energy (BLE) sensor is the most suitable for an autonomous Zebro Swarm. BLE can be used to transfer data as well as serve as a range sensor. As a communication device, BLE offers the ability to advertise data an has a range of 50 meters which is sufficient for application in a swarm. Furthermore, iBeacon makes use of the BLE protocol and therefore there will be a lot of information available. As a sensor, BLE offers fairly accurate measurements in distances below 1 meter. Due to the properties of radio signal strength and the always present radio frequency noise, the accuracy decreased with increasing distances.

The individual behaviour aims at exploring as much area as possible, with the amount of energy available. The energy use therefore has to be minimised and the explored area maximised. Since little is known on the energy use, this is assumed constant. Maximising area, however, has some key elements. The first is to keep moving; standing still results in energy use without exploring new terrain. The second is to make only gentle turns; the rotational centre must be outside of the detection distance. When this centre lies inside of the detection distance, the inner turn will cover area for a longer period making the turn ineffective.

The anti collision behaviour should prevent units from making physical contact. By using a contactless sensor an approaching collision can be detected. Changing the parameters of the movement of the units at the moment they enclose more than a certain value should prevent the approaching collision. Letting the agents turn an angle between $90 + arctan(\frac{W_u}{2 \cdot D})$ degrees and 180 degrees in a randomly chosen direction is proposed to be a simple and effective solution.

The anti separation behaviour should prevent units from separating from eachother. This can be done by making units turn 180 degrees once the relative distance reaches a certain limit. The turning on a certain limit is a simple algorithm and thus requires little processing power.

Apart from the anti collision behaviour and the anti separation behaviour, there is another approach to deal with these behaviours. Since the units are able to communicate with each other, they can also advertise their location to other units. There are several methods for a unit to acquire their location. These methods are divided in approximate and exact location estimation methods. The Trial and Error method is placed in the first category, the exact location estimation methods are: Time Based, Internal Reference, External Reference and Radar. The External Reference method is considered the most applicable of these methods since it is relatively accurate, simple and provides an direct location. The downside is that an external reference is needed, but that is acceptable.

Contents

1

Introduction

1.1. Introduction to the Zebro

The Zebro is a six legged robot [2], designed on the foundations of RHex. The name Zebro is derived from the Dutch translation to 'six legged robot': Zesbenige Robot.

Thanks to his six legs, the Zebro is capable of navigating through very rough terrain. The top and bottom of the Zebro are symmetric, and therefore the Zebro will be able to walk along if it tips over.

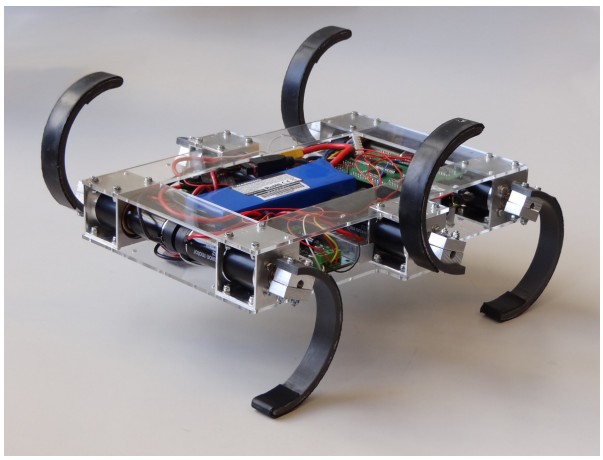


Figure 1.1: The six legged robot: Zebro.

At the moment the Zebro is controlled through a Wi-Fi chip. The Zebro creates a server which can be accessed through a tablet, computer or phone. The movement of the Zebro is controlled through the server and therefore isn't autonomous.

1.2. Research question

This project originated from a thesis proposal by the Technical University of Delft and was titled: *"Autonomous Zebro Swarm Navigation"*. The free translation of the project description is as follows:

"In this project a navigation system will be designed, with which Zebro Robots can swarm in and out of the CyberZoo. Every Zebro is equipped which a radiobeacon based on the Low Energy Bluetooth Protocol (as concluded from an earlier feasibility study). With these beacons the Zebro's need to stay together and form an effective swarm without colliding into each other. The swarm must be able to react to its surroundings, like following a beacon."

The description is later slightly changed to:

"Zebro is a bio-inspired robot, designed for operation in rough terrain. At the moment Zebro does not have any extereoceptive sensing capabilities, and as such it cannot move autonomously in the Cyber Zoo arena. The objective of this project is to enable the Zebro robots to perform the top level task of walking in the same direction with the same speed and synchronised leg movement. Before this top level task can be accomplished basic swarming behaviours need to be implemented. In particular, collision avoidance and exploration while maintaining connectivity is necessary. On top to that, when internal power runs low each individual Zebro must search and move towards a charging station."

The related research questions:

1. What type of module is suited to handle the communication traffic between Zebros?
2. What data needs to be transferred between the Zebros to achieve swarming behaviour?
3. Order of working towards the final task:

First	Avoid collisions
Second	Stay together as a group while exploring
Third	When low battery, individually search for charging station
Fourth	Engage in synchronous movement

1.3. Problem analysis

This section elaborates the main objective of this study. Besides the main objective, different smaller and more concise goals are required to eventually reach the main objective. By performing a problem analysis on these objectives it is possible to construct a List of Requirements (LoR). By complying to the list of requirements, the objectives will be met. To differentiate between solutions the bigger perspective and a set of criteria is used to indicate to what extend a solution fulfils the LoR.

1.3.1. Main objective

The main objective is as follows:

Design a physical autonomous swarm of robots on a 2D plane.

1.3.2. Bigger perspective

The design will have to fit within a bigger perspective. This bigger perspective will give an overall direction for the design choices that have to be made. The bigger perspective can be stated as:

Have a swarm operate for a(n) (potentially) infinite amount of time.

To reach this goal a robust system has to be designed. Since it is impossible to verify if this goal is reached, signs of the potency to operate for an infinite amount of time will be looked for/at. In this sentence, the operation of the swarm means that every unit within the swarm is either charging, moving or communicating to other units. The idea is that in later stadiums more and more restrictions are put on the system. To give an idea of the direction of the bigger perspective, a future goal could be:

Have as many units operate with the amount of energy available.

The swarm will consist of a number of units. These units will behave in an environment in which a certain amount of energy is available. The goal for the swarm is that as many as possible units are operative. The assumption is that this will be the direction in which the project will progress after completion. This gives direction to design choices.

1.4. List of requirements

To complete the partial objectives and finally the main objective, the requirements these objectives are subjected to have to be clarified. These requirements are divided in hard and soft requirements. Hard requirements have to be met, while soft requirements can help in ranking different design solutions and do not necessarily have to be met.
The following LoR is assumed to be complete and correct.

1.4.1. Swarm behaviour
The behaviour of the swarm is based upon the following hard requirements:

Hard requirements
1. Decision making must be decentralized
2. The path of the individual units may not be pre-defined
3. The swarm must be able to operate without any information from other sensors than the ones equipped on the units themselves
4. The swarm must be able to operate independent of its absolute location
5. The behaviour must be designed for operation within a 2D plane
6. The behaviour must be designed for operation in a finite space, which has unknown dimensions
7. The space in which the swarm operates is big enough for the units to move without colliding
8. The individual units may not make physical contact
9. The maximum distance of separation between the individual units is fixed
10. An individual unit has contact with at least one other unit
11. A unit in the swarm gains its energy from a charging station
12. The (absolute) location of a charging station is unknown until the moment of detection
13. The swarm becomes more effective when the amount of units within the swarm increases

Soft requirements
1. Units must be moving at any moment in time, except for when charging, when out of energy or when defect
2. The behaviour must be designed for operation in a finite space, which has unknown dimensions

1.4.2. Communication hardware
These are the requirements that have to be met for the communication hardware.

Hard requirements
1. The communication between units may not rely on anything else than the units themselves
2. The communication module must be able to communicate wirelessly with other communication modules
3. The communication module must be able to effectively communicate within a distance of 50 meters without any disturbing factors in between
4. It must be possible to use the communication module effectively in a 3D swarm
5. The communication module must be compatible with the Delfly communication module
6. The module is capable of an advertising mode

Soft requirements
1. Energy use must be as low as possible
2. The communication module may not be influenced by any weather conditions

1.4.3. Sensors
These are the requirements that have to be met for the sensors.

Hard requirements

1. The sensors have to fit on a Zebro

Soft requirements

1. Energy use must be as low as possible
2. The sensors may not be influenced by any weather conditions

1.4.4. List of assumptions

To reach a proper end result of this project within the limited time given, a few assumptions have to be made, scaling down on the difficulty of this project. Furthermore, since some features/requirements are yet unknown it is necessary to make assumptions as well. The list of assumptions (LoA) listed below.

1. Energy can not be transferred along units
2. A charging station only has place for one robot
3. A charging station can be detected whenever a unit is within a fixed perimeter around the station
4. The area covered by the perimeter around the station is much smaller the total area of the search field.
5. There is one active charging station at any point in time
6. The times at which the charging station becomes available is random
7. The amount of energy available within a period of time is variable
8. The geographical location of a charging station is fixed while available for units
9. The geographical location of a charging station is determined by a uniform random distribution
10. The speed of movement of the individual units is limited
11. The energy usage of the individual units is assumed to be constant for every speed of movement
12. The energy available for use by a unit is finite
13. There are no obstacles disrupting the communication between units

1.5. Plan of approach

The following list sums up the order in which the different tasks within the project are to be accomplished. To keep this project as general as possible, the term ”Zebro” is replaced by ”unit” since the ”Zebros” behave like an agent/unit within a swarm. Furthermore, the Charging Stations may be referred to as ‘Points of Interest’ (POI).

1.5.1. Plan 1.1

This was the initial plan, this was improved after receiving a brighter picture of the project.

Before building a physical swarm, the different algorithms need to be tested in a virtual simulation. This simulation model can be built in V-Rep, a simulation program specially designed for robot simulations. The algorithm that controls the simulation will be built in Matlab, which is virtually connected to V-Rep. In short the general plan consisted of the following steps:

1. Build a swarm model in V-Rep, in which the units are controlled by an algorithm running in Matlab
2. Test several swarm models in V-rep
3. Replace V-Rep model with physical units, controlled by an external computer running the algorithm in Matlab

4. Implement Matlab algorithm on a chip on the units themselves

Within this general plan are two parts that have a plan of their own: the algorithm and the communication.
The plan on the design of the algorithm is as follows:

1. (Randomly) walk around, staying within a certain perimeter
2. (Randomly) walk around in the perimeter without collisions between units
3. (Randomly) walk around in the perimeter in a group without collisions between units
4. Walk around in the perimeter in a group without collisions between units, while having for example synchronized movement or walking in a 'V' shape.

The plan on designing the communication structure is as follows:

1. Unit in simulation model receives relevant parameters from the model itself.
2. Physical unit is controlled by Matlab on a computer, communication through computer, parameters are received from camera system.
3. Physical unit is controlled by Matlab on a computer, communication through computer, parameters are received from sensors on physical unit.
4. Physical unit is controlled by Matlab on a computer, communication directly to other units, parameters are received from sensors on physical unit.
5. Algorithm implemented on unit and the unit behaves autonomous.

1.5.2. Plan 1.2

This was the second and final plan of approach. Neatly split up in different phases, the structure of the project is given here. What can be noticed is that the task of performing some kind of "synchronised movement" is left out. This is due to the decision of focusing on the main swarming behaviour.

Problem analyses

1. Set objectives
   (a) Main objective
   (b) Partial objective(s)
2. Set design goal
3. Set up LoR and LoA
   (a) LoR for the behaviour
   (b) LoR for the communication module

Synthesis

The following list is a sum of the activities that were to be performed in order to fulfil the main objective.

1. Sensors
   (a) Find out what variables have to be communicated from unit to unit, for proper functioning of the swarm
   (b) Decide upon which sensors to use
2. Communication
   (a) Decide which communication module should be used

3. Searching behaviour
   (a) Find out which type of walking/searching behaviour would best suit this specific situation
   (b) Create the walking/searching behaviour
4. Charging station
   (a) Find out what decisions have to be made when a unit finds a charging station

The underlined steps are the typical research steps. This project is focused mainly on walking through the design cycle once (Analyses, Synthesis, Simulation, Evaluation and Decision making). So the research steps are done with the goal of making a grounded decision, not in order to optimise to the full extend.

Simulation

1. Make a virtual simulation
   (a) Sensors
      i. Make a unit register the relevant variables
   (b) Communication
      i. Make a unit transfer the above variables to (an) other unit(s)
      ii. Make a unit receive/process the variables from (an) other unit(s)
   (c) Searching behaviour
      i. Program the walking/searching behaviour into the units
   (d) Charging station
      i. Make sure the units are able to detect a charging station
      ii. Make the units take the earlier determined decisions when a charging station is found
2. Test and troubleshoot
3. Implement the simulation on physical robots
   (a) Searching behaviour
      i. Program the walking/searching behaviour into the units
   (b) Sensors
      i. Make a unit register the relevant variables
         A. Unit receives relevant parameters from camera system via Wi-Fi through computer (and possibly personal sensors).
   (c) Communication
      i. Make a unit transfer the above variables to (an) other unit(s)
         A. Unit communicates to another unit over Wi-Fi (through computer) while receiving the relevant parameters over Wi-Fi from the camera system (and possible personal sensors).
         B. Unit communicates to another unit over Wi-Fi (through computer) and will determine the relevant parameters by itself (its sensors).
         C. Unit will determine the relevant parameters by itself (its sensors), and will communicate with other units by means of the communication module.
      ii. Make a unit process the variables from (an) other unit(s)
   (d) Charging station
      i. Make sure the units are able to detect a charging station
      ii. Make the units take the earlier determined decisions upon finding a charging station

Evaluation

Test the design to the design goal Find out what features contribute to the proper and improper functioning of the design.

Decision making

Advise on possible future steps to take.

2

Literature study

A numerous amount of papers are written on the subject of swarm intelligence (SI). From studying several references a clear view on this concept was constructed. The most applicable and comprehensive was considered to be the following:

> *"A swarm is a large number of homogenous, unsophisticated agents that interact locally among themselves and their environment, without any central control or management to yield a global behaviour to emerge."*

[1, p. 25]

An example of a field where use is made of swarm intelligence is optimization algorithms. These are algorithms in which local and/or global optima of a function are investigated [1]. A lot of SI based optimization algorithms are inspired by natural swarms/behaviours of animals. To name a few: ant colonies, bee colonies, fireflies, cuckoo, fish schools and bird flocks. From studying these models it soon became clear that a couple of ground rules make up the behaviour of these algorithms. These ground rules hold for each individual agent the algorithm generates to solve the problem of finding the optima. Another behavioural aspect which is easily noticed is the random factors involved in the behaviour of each individual agent, which are used to make sure that the solutions are not deterministic. Furthermore, in multiple algorithms (Hunting Search, Firefly) different behaviours are detected for different situations. In the firefly algorithm the type of behaviour of an agent is determined by weight factors, whereas with the Hunting Search algorithm the modes are switched between with the activity of one excluding the activity of the other. All of the above mentioned has mainly produced an extended inspiration for the design of a physical robot swarm.

3

Sensors

This chapter will discuss the use of sensors for a physical robot swarm: which sensors should be used?

3.1. Requirements

The requirements for the sensors as earlier described are:

- The sensors have to fit on a Zebro
- Energy use must be as low as possible
- The sensors must measure the distances between robots
- Range for object/Zebro detection should be up to 50 cm

The first requirement is because of the limited dimensions of the Zebro. The second has everything to do with the limited energy supply of the battery. To make a efficient exploration robot we would like to explore the maximum area with the least amount of energy used. If the sensor is efficient enough to be used in combination with a Delfly, that project could also benefit from this work.

The range requirement is because of the safety range around the Zebro. The safety range of 50 cm is enough to avoid any contact with other Zebro's or object's in the walking area. The last requirement is needed to keep the individual robots together. Knowing the relative distances, the robot can walk away form the group or if the distance becomes too large, the robot can decide to return to the group.

3.2. Avoid obstacles

The first thing to be sure of is that Zebro doesn't collide with an object in the walking area. A sensor which is suitable for this job is a ultrasonic sensor. The decision for this sensor is based on earlier research [3].

3.3. Measure relative distances

To measure relative distances there are multiple possibilities. In this research we divide the medium for communication into two. The first way of communication is sound, for example communication through ultrasonic acoustical sound. The second way is the use of light. The working principle of sound is that it essentially is moving air. Compared to light which is based on electromagnetic radiation. Light travels much faster than sound an because of this the delay of the signal becomes much lower.

Anohter important difference is the influence of the environment. For example wind speed has a great influence on sound waves. To keep the influences of the surroundings as low as possible light is the better option.

The spectrum of light gives us different zones like InfraRed an Radio Frequency (RF). Infrared uses more energy and is more sensitive for environmental radiation in comparison to RF. In this research we will continue with figuring out what the possibilities are for communication with RF between robots

in a swarm.

Modules which are suitable for this task are Zigbee, BLE113 and Wi-Fi. These are all based on Radio Frequency communication. Important factors for communication between robots in a swarm are the network size, range, number of channels and the advertising mode. Table 3.1 gives an overview of the characteristics of the different modules. In combination with the requirements for the communication module we will find a suitable communication module.

3.4. Requirements for a communication module

The process starts with a list of constrains and requirements. Out of the literature study the module characteristics became clear. The data sheets of the modules give the data to compare with the requirements.

The following design requirements and constrains are taken in account during the design process:
Hard requirements

1. The communication between units may not rely on anything else than the units themselves
2. The communication module must be able to communicate wirelessly with other communication modules
3. The communication module must be able to effectively communicate within a distance of 50 meters without any disturbing factors in between
4. It must be possible to use the communication module effectively in a 3D swarm
5. The communication module must be compatible with the Delfly communication module
6. The module is capable of an advertising mode

Soft requirements

1. Energy use must be as low as possible
2. The communication module may not be influenced by any weather conditions

3.5. Design Solutions

Zigbee(CC2520), BLE113 (CC2541) and Low Energy Wi-Fi (CC3000) have a 50 meter range data transfer, have a low energy use and are light weighted. First we compare the energy use on a data transmission scale between the modules.

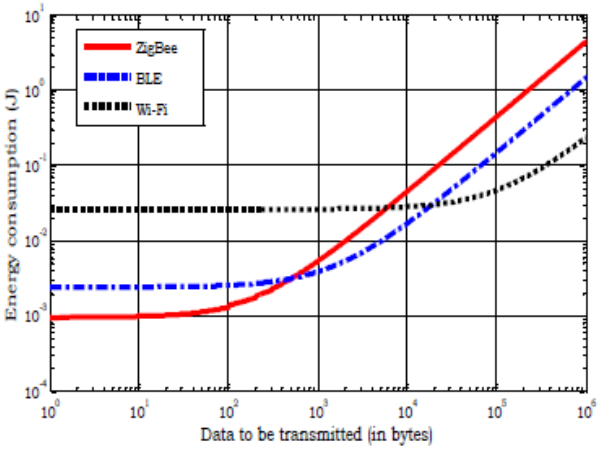


Figure 3.1: Energy consumption of the Zigbee, BLE113 and Wi-Fi for different data loads [4].

The energy use (figure 3.1) is different. We can see that it depends on the amount of data sent. The actual amount of data (in bytes) which will be send with the communication module is not yet defined.

Standard	ZigBee	BLE	Wi-Fi
Network size	65536	8	32
Range	<50m	<50m	<100m
Number of channels	1/10/16	40	11-14
Advertising mode	Yes	Yes	No

Table 3.1: A comparison of different parameters for ZigBee, BLE113 and Wi-Fi[5].

3.6. iBeacon

The BLE113 module is being used as a ranging tool in the form of an ”iBeacon”. This technology enables a device to perform actions in a close proximity to an iBeacon by advertising. This BLE’s advertising mode broadcasts packages of information in a set time frame. Another module can scan for these packages. With this advertisement mode we can range distances and transfer data between two modules and there’s no limiting network size. At this moment the iBeacon concept is in a development state but will be used in a lot of places. In the future costs will be low and the reliability will be high.

3.7. Conclusion

To build a physical swarm, the BLE113 module or the Zigbee module are the best options. Zigbee(CC2520), BLE113 (CC2540) have a 50 meter range data transfer, have a low energy use and are light weighted. Because the Wi-Fi doesn’t have the ability of advertisement it is less suitable.

So the characteristics of BLE and Zigbee are close. The choice for the BLE113 chip was because it is foreseen that in the near future Apple will significantly improve the functionality of BLE. This is beneficial for the project, because this save large amounts of research. Also, when this technique becomes widely used, it’s price will go down. This is especially important for a swarming robot which relies on large numbers to boost its performance.

4

Measuring RSSI with BLE

With the BLE protocol it is possible to measure Received Signal Strength Indication (RSSI) between devices with little effort. A ready to use, build in function displays, among others, advertisement data and RSSI. The goal is to use this RSSI to estimate the distance between devices. This measurement should be accurate with a maximum deviation of half a meter within an obstacle free space. To test whether the BLE device is suited for this purpose, RSSI measurements were done.

4.1. Goal and background information

The goal is to determine if the RSSI of a BLE module is suited to give an approximation of the distance between two devices in a space with no obstacles. There are no obstacles because the first Zebro swarm prototype will walk in an empty space. To give the swarm the intended behaviour it must have a sense of distance. This can be the precise distance in meters or just a sense of 'too close' and 'too far'. Too close is in this case one meter. This distance is just an estimate when looking at the dimensions of a Zebro and the speed with which it moves. With an accuracy of half a meter from the BLE module and thus half a meter distance between Zebros they should never collide. There should always be about ten centimetres of space left between the front legs sticking out.

The problem with RSSI is always the amount of noise of the signal. This noise comes from various factors which are for example hardware factors like antenna pattern or influences from the surroundings. Indicating and researching all factors is beyond the scope of this research.

An easy way to make BLE113 useful for Zebro is to try and define the RSSI values for 'too close' and 'too far'. These expressions are not really precise and allow a signal that's a little noisy. To see if it is possible to define 'too close' and 'too far', it was decided to do two sets of measurements. These sets differ in the distance between the two modules. The first set has a distance between modules ranging from zero to two meters with ten centimetre increments. The second set has distances ranging between two and twenty meters with two meter increments. The first set was measured indoors and serves to show what values of RSSI are detected from a nearing device. The second set was measured outdoors and is used to determine what the RSSI is at greater distances. Both measurements should also give a clear view of the amount of noise.

4.2. Setup

The BLE module is placed on top of the T-minus microcontroller board. One set was connected via USB to a laptop and the other set via USB to a power bank. This allowed for one set to move freely and for the other set to log the RSSI value to a log-file. To log the RSSI values coming from the T-minus board the program 'GoBetwino' [6] was used. Every time a `gap_scan_response` event is triggered because the device receives an advertisement packet, the RSSI value is logged on the PC with the program. This way around 100 values for every distance were collected. It is not possible to stop logging after precisely 100 values, so for every measurement a minute of data was collected. Sometimes this results in a little less than 100 values, sometimes a little more.

In the first measurement the modules were place on the floor as can be seen in figure 4.1 and in the second measurement on small tables. The laptop was always closed to eliminate potential reflections

Figure 4.1: The measurement setup

from the lid. Two extra single measurements were done out of curiosity and to create a little more insight. One measurement outdoors at 62 meters and one indoors with both BLE devices in different rooms with one closed door in between.

4.3. Results - post processing

All measured values were collected and assembled in Excel for post processing. To get insight in the collected values they were plotted for every increment. See figure 4.2 for the first set of measurements and figure 4.3 for the second set. There are multiple dots for every distance, which means multiple RSSI values are measured while the modules are this particular distance apart. The plot does not show how many times the different values were measured. Take for example the measurement at 1,4 meter in table 4.1.

dB	-63	-73	-74	-75	-80	-81	-82	-83	-84	-85	-86	-88	-93
# measured	36	3	27	7	3	7	7	2	5	2	2	1	1

Table 4.1: RSSI Values at 1.4m distance.

This measurement has 13 different values, but two of them are significantly more present than the others. It is not known why these two specific values are more present. However it is proof that the signal definitely has noise.

Before applying a filter it should be known what the real value for the RSSI should be. Otherwise it cannot be verified afterwards if the filter works as necessary. The relation between received signal strength and distance is described with the following equation. [7]

$$P_r(d) = P_{r,1m} - 10 \cdot n_p \cdot log_{10}(d) + X_\sigma \tag{4.1}$$

$P_r(d)$	Received RSSI at certain distance [dBm]
$P_{r,1m}$	Received signal strength at 1 meter distance [dBm]
d	Distance [m]
n_p	Constant value that describes accounts for all noise
X_σ	Zero mean Gaussian distributed random variable with standard deviation σ, accounts for the random effect of shadowing.

This equation is correct only when the right values for $P_{r,1m}$ and n are used and the devices are operated in an open space. [8]

There are multiple ways to determine the values for $P_{r,1m}$ and n. The most common approach for both is to try to measure them. For $P_{r,1m}$ this would mean building a set up with a receiver with a known antenna pattern (preferably uniform) to measure the exact received signal strength at one meter distance. The problem with this approach is that, even when assuming the custom receiver is lossless, the losses within the transmitting device are already taken into account.

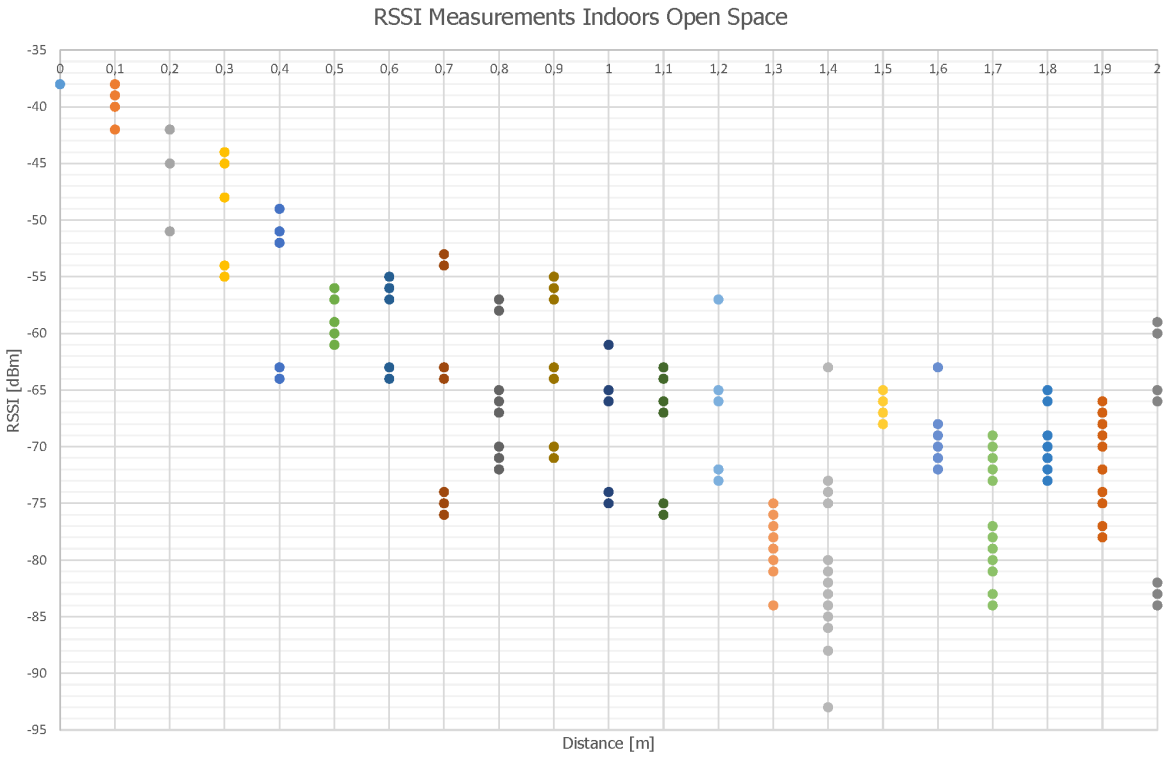


Figure 4.2: RSSI Measurements Indoors Open Space At Fixed Intervals

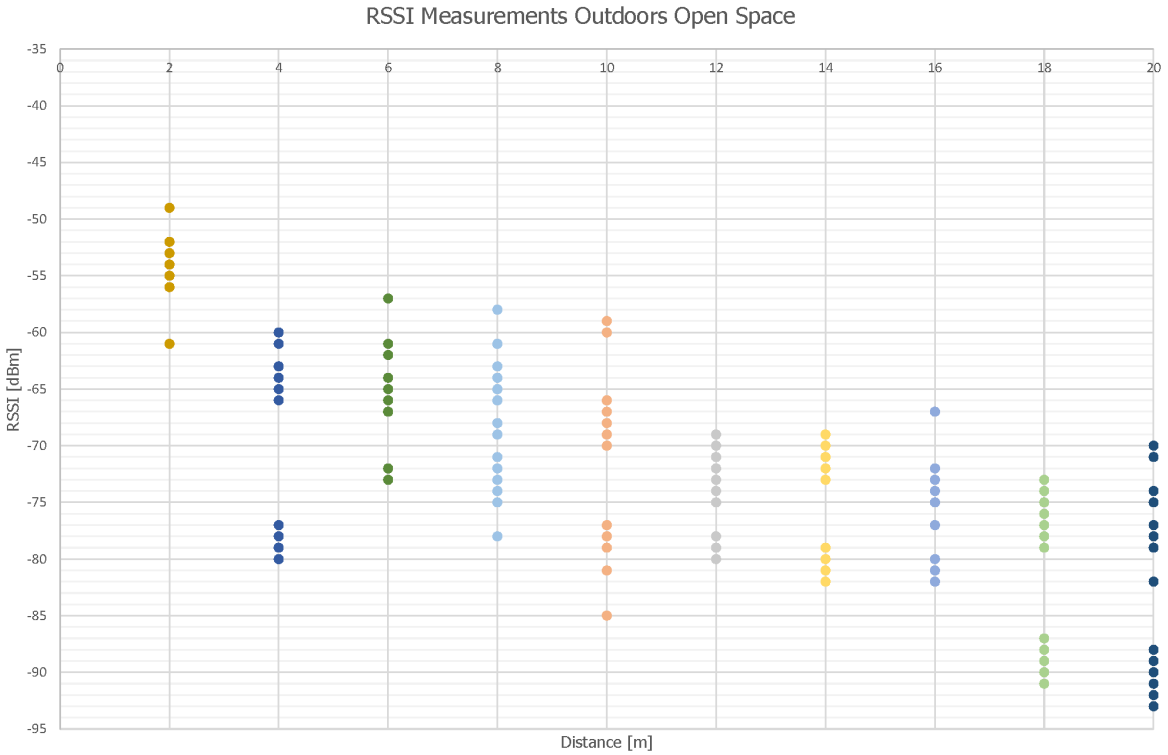


Figure 4.3: RSSI Measurements Outdoors Open Space At Fixed Intervals

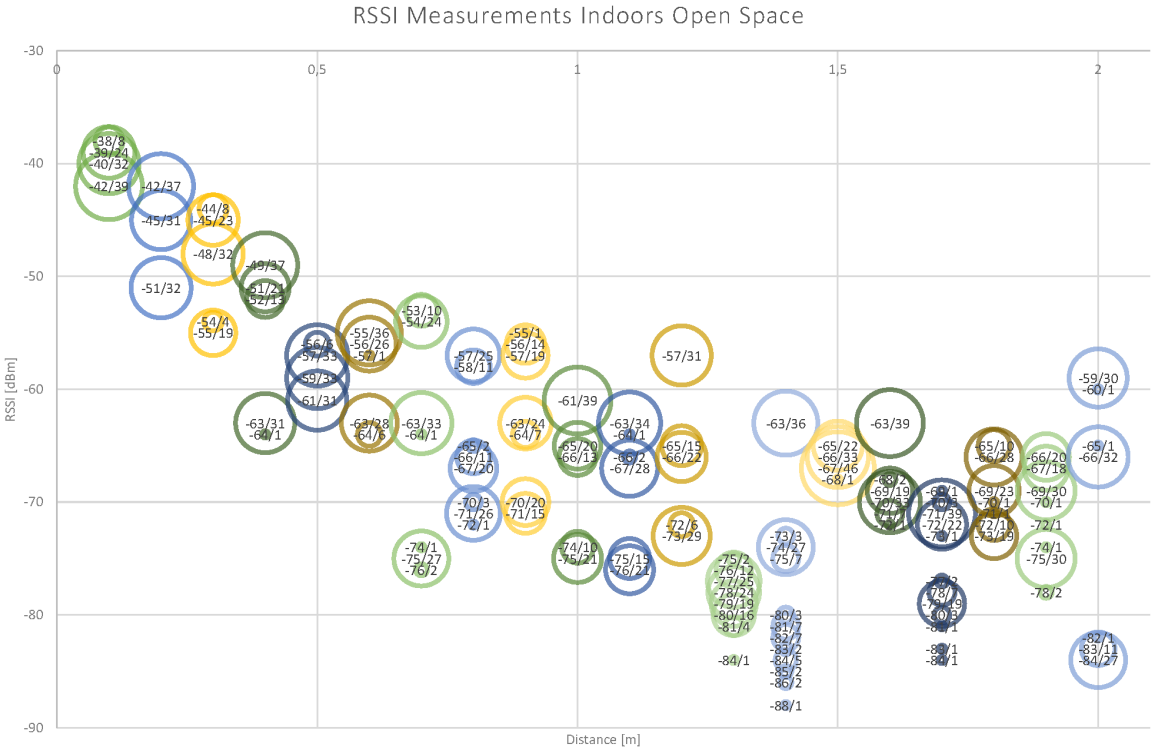


Figure 4.4: RSSI Measurements Indoors Open Space At Fixed Intervals With Measured Instances As Bubble Size

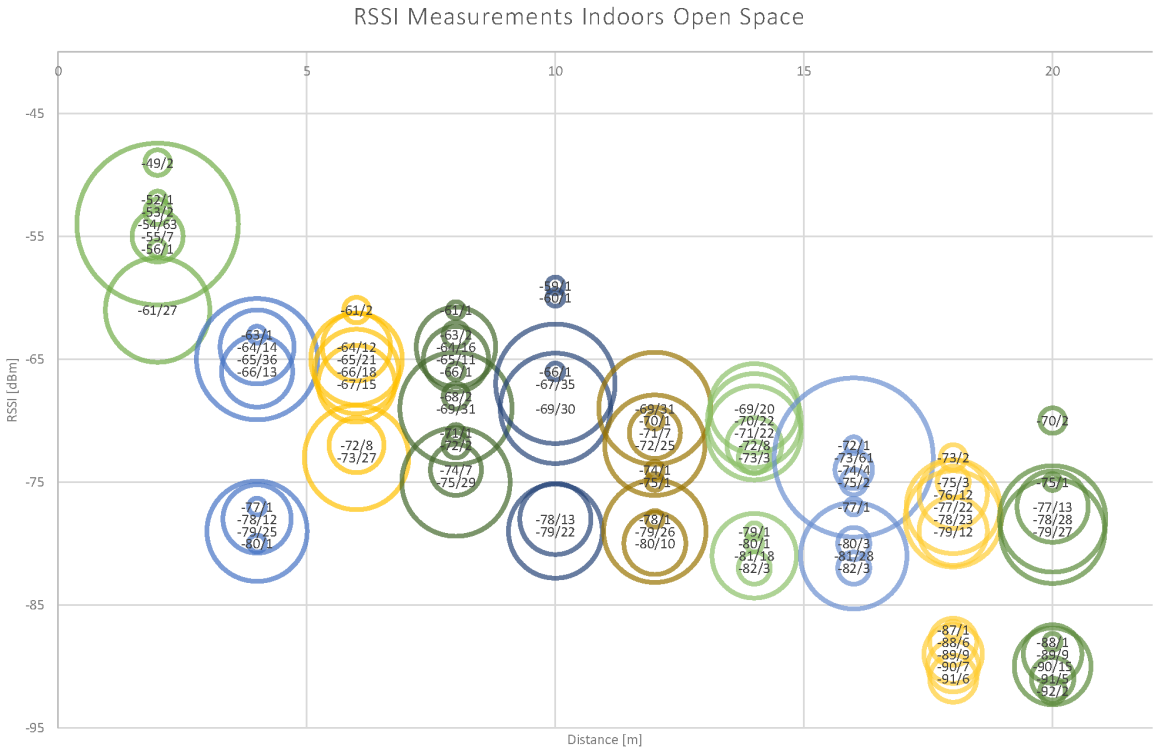


Figure 4.5: RSSI Measurements Outdoors Open Space At Fixed Intervals With Measured Instances As Bubble Size

Because the exact losses within the structure of the transmitter and the influence of the surroundings are always fluctuating, the value for $P_{r,1m}$ will also fluctuate when measured. This is not beneficial for the process keeping in mind that n also has this character. To prevent this from happening $P_{r,1m}$ is calculated with the following equation: [9]

$$P_r = P_t - 20 \cdot log\left(\frac{4 \cdot \pi \cdot f \cdot d}{c}\right) \tag{4.2}$$

P_t Transmitted RSS [dBm]
f Frequency for Bluetooth 4.0 [Hz]
c Speed of light [m/s]

For d one meter is used and according to [10] the frequency for the CC2541 radio chip is 2440 MHz. Also according to [10], the value for P_t is zero. Within the firmware of the BLE113 module this value can be changed between -23 dBm and zero dBm. To maximize range, the module is currently set to its maximum broadcasting signal strength of zero dBm. The equation now gives $P_{r,1m} = 40,196[dBm]$. With this value and the adapted equation 4.1 the result is the equation as follows:

$$d = 10^{\frac{P_r + 40,196 - X_\sigma}{10 \cdot n}} \tag{4.3}$$

In this equation the only unknown value is n. This value contains among other things losses in the RF path and antenna outside of the radio chip. Also the influence of the surroundings on the RSSI are within this parameter. To get the most accurate distance measurement with this equation, the value of n should cover all disturbance factors. To indicate and investigate all these factors is beyond the scope of this research. It is almost standard practice, when looking at other papers, to calculate a group mean and standard deviation for n. [8, 9] The way this is done differs. Of course it makes all the difference what kind of distribution is chosen for the mean and standard deviation. Noise on the RSSI signal is often modelled by a Gaussian white noise distribution [7], but other factors may have a different distribution. In the end every situation will lead to its own unique distribution.

Lastly, the result for the extra outdoor measurement at 62 meters showed little difference from the outdoor measurement at 20 meters. The other extra measurement (indoor with closed door) had values only slightly larger than the 20 and 62 meter measurements. What became clearer is that it took significantly longer to acquire about 100 RSSI packets.

4.4. Result and Discussion

Lots of research is being done on this subject. Also within The Zebro project a master's thesis is being centred around this problem. To check if distance measurement with BLE113s and RSSI can deliver an accuracy of 0,5 m has proven to be too difficult for this project. The measurement shows clearly how noisy the RSSI signal is and the fact that outdoor measurements contain far less noise when compared to indoor measurements. According to [9, 11] a fairly accurate indoor measurement can be obtained from RSSI when distances stay below one meter. This is a good thing for determining the 'too close' statement. If it's possible to accurately sense the 'too far' statement is unclear. Formula 4.1 shows that the difference in RSSI becomes smaller when distance increases which demands a more precise RSSI measurement for larger distances.

It is also unclear from literature or experiments how well RSSI performs when placed on moving robots.

The large amount of literature is very helpful in quickly discovering what has already been done on the subject. Especially [7] is a useful paper. For future work it is highly recommended to start with this paper and review its sources on roundtrip propagation time measurement and on equation 4.1.

5

Behaviour in general

This chapter will discuss the behaviour of a swarm in general. Aspects such as the energy use and the different parts of the behaviour will come up in this chapter. This will create a base for the following chapters on individual and interactive behaviour.

5.1. The concept of a swarm

In order to create a clear image on the swarm that is to be designed, it is of importance to define the concept of a swarm in first place. As seen in the literature study (ch. 2) the definition is:

> *"A swarm is a large number of homogenous, unsophisticated agents that interact locally among themselves and their environment, without any central control or management to yield a global behaviour to emerge."*
>
> [1, p. 25]

For this project, as can be concluded from the LoR, "large" is assumed to be more than 2 agents. In nature different swarms exist, to take some examples: ant colonies, bee colonies, fireflies, schools of birds and fishes. Each of those swarms have a different way of behaving. [1, ch. 2] In order to determine the behaviour of the swarm that is to be designed, we combine the features of the different swarm examples out of nature to design a swarming behaviour that fits with the LoR in the best possible way.

5.2. Ground rules

Since swarms exist in nature and have existed over a longer period of time, we can conclude that the design of these swarms is effective for their purpose. For this reason, inspiration for the design of the robot-swarm can be gotten from natural swarms. These swarms show to have several 'ground rules' which can be deduced from observing their behaviour. For the swarm that is to be designed, the set of ground rules have to be defined in advance. These ground rules have to lead to a successful functioning of the swarm. In other words, these have to make sure the design goal is reached (*Have a swarm operate for a(n) (potentially) infinite amount of time.*) Where possible making a good step in the direction of the future design goal: Have as many units operate with the amount of energy available. These ground rules are (restrictions to) rules concerning the behaviour of the individual units within the swarm. Every unit within the swarm follows this set of rules, resulting in the same behaviour for each individual unit.

Slightly corresponding to the structure of fish school behaviour [1, p. 30], the ground rules for the swarm that is to be designed are defined as follows:

Individual behaviour

- The individual units should perform an individual moving behaviour

Interactive behaviour

- The individual units should prevent physical contact
- The individual units should stay together

For each of these rules, design choices have to be made about what actions to take in order to comply with these rules. To do so, criteria and numerous possible solutions concerning each rule are come up with. What has to be kept in mind is that the effectiveness of the final algorithm is dependent on the combination of the different solutions.

5.3. Switching between different behavioural modes

As can be concluded from the previous chapters, three behavioural modes are distinguished:

1. The individual behaviour
2. The anti-collisional behaviour
3. The separation-prevention behaviour

Each individual agent has to decide upon the behavioural mode it will apply at any moment in time. The choice of this mode is based upon at least the relative distance between other units. This is because it is the least possible amount of information needed for an individual unit to interact with others.

With the relative distance between two units ranging from zero to the maximum range of the sensor, the activation of a certain mode should have a chance of occurring at every single one of these distances. Variables for describing these chances are taken to be the following:

$\gamma = f_i(D)$	The chance of execution of behavioural mode i, with f being a function dependent on D the relative distance between two units
ΔD	The $maximum\ D - minimum\ D$ for the same/constant γ

What has to be taken into account is that:

$$\sum_{i=1}^{i=3} f_i(D) = 1, \quad \text{for every } D \tag{5.1}$$

So that for D ranging from zero to the limit of the sensor, any of the three behaviours are activated. Possibly, a behaviour for the point at which no other agent is within range of the sensor could be implemented just as well. However, for this project this is out of scope, and the unit is considered to be separated from the 'swarm'. To get an idea of the possible distributions, the following plots were made. With the functions defined as follows:

$$f_1(D) = 1 - (f_2(D) + f_3(D)) \tag{5.2}$$

$$f_2(D) = \begin{cases} 1 - \left(\frac{D}{0.5 \cdot D_{max}}\right)^2, & \text{if } f_2(D) \geq 0 \\ 0, & \text{if } f_2(D) < 0 \end{cases} \tag{5.3}$$

$$f_3(D) = \begin{cases} 1 - \left(\frac{D - D_{max}}{0.5 \cdot D_{max}}\right)^2, & \text{if } f_3(D) \geq 0 \\ 0, & \text{if } f_3(D) < 0 \end{cases} \tag{5.4}$$

This gives a plot as can be seen in figure 5.1.

Now, with a continuous function having a gradient inequal to 0, ΔD approaches 0. Increasing ΔD would give the functions a more stepped-like character. The simplest form of this would be when the functions are assumed to consist of stepfunctions which return 1 between two values:

$$\gamma = f_i(D) \begin{cases} f_1(D) = 1, & \text{for } 0 < D < L_1 \\ f_2(D) = 1, & \text{for } L_1 \leq D < L_2 \\ f_3(D) = 1, & \text{for } L_2 \leq D < D_{max} \end{cases} \tag{5.5}$$

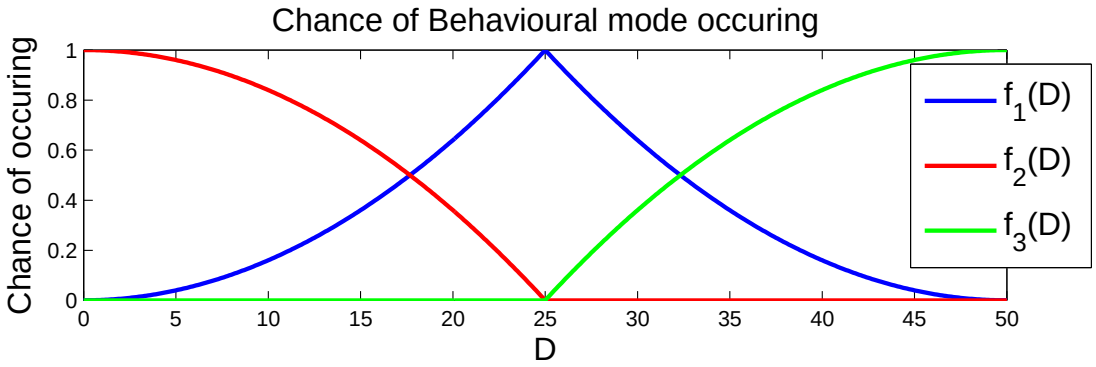


Figure 5.1: Behaviour distribution

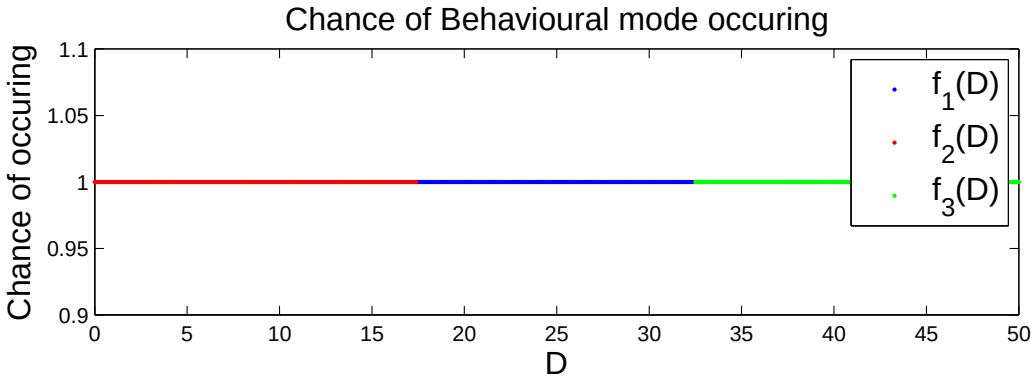


Figure 5.2: Binary behaviour distribution

With $L_{1,2}$ being the boundaries of where the step functions change value, and D_{max} being the maximum distance measured by the sensors.

This gives a plot like the one in figure 5.2.

In this manner, the distance D between two units is directly related to the type of behaviour these units perform. In addition, there is no chance of the agents executing a different behavioural mode while measuring the same distance between them.

5.3.1. Design choice

Setting hard boundaries at which a switch is made between the different modes (as in figure 5.2), results in a predictable behaviour of the system. This predictability is key to the functioning of the implemented choices which are made for the different modes later in this document (6,7,8). After this project, experimentation can be done on changing the functions that make up the chance of picking a behavioural mode. Taking the functions to be step-like functions as in figure 5.2, at any distance D, one mode will definitely be activated. The distances $L_{1,2}$ at which modes are changed can still be chosen. L_1 is the moment at which the unit stops the anti collision behaviour and starts it's individual behaviour. This limit depends on parameters such as the size of the unit and its speed. The best value for this limit has to be researched, but for units as Zebro's it is assumed to be around 0,5 meter. L_2 determines the moment at which the unit stops the individual behaviour and starts to move back to the swarm. Since the units need to keep contact, it is important to have L_2 smaller than R_{det}. Since the BLE module has a reach of around 50 meter, L_2 is assumed to be 45 meter. A higher L_2 value will mean that the unit will use the individual behaviour relatively longer and thus make the swarm more effective as a whole. The chance on losing contact also grows with a higher L_2, thus the best balance requires testing.

5.4. Movement parameters

When looking at the movement of a unit, it is possible to construct a list of base parameters. These parameters make up the movement of the agents. These are the following:

- Movement speed
  - Relative increase/decrease
  - Absolute increase/decrease
- Movement direction
  - None
  - Forward
  - Backwards
- Turning radius
  - Zero
  - Smaller than field of view
  - Bigger than or equal to field of view
- Turning angle
- Turn direction

The algorithm running on each agent should assign values to these parameters which on their turn have the task to generate an effective global behaviour. These values can be obtained in the following manners:

Movement speed	Movement direction	Turning radius	Angle	Turn direction
Fixed	Fixed	Fixed	Fixed	Fixed
Input based	Input based	Input based	Input based	Input based
History based	History based	History based	History based	History based
Random	Random	Random	Random	Random
None	None	None	None	None

Table 5.1: The movement parameters

The following chapters will discuss several parameters of importance to the different kinds of behaviour. The final combination of different parameters decides whether an algorithm is effective or not. The best combination of parameters differs for each situation, so it is impossible to determine the perfect way of assigning a value to a parameter. This will be discussed in the oncoming chapters.

5.5. Energy use

As explained earlier, searching for a charging station costs energy. The time in which a charging station must be found (at a specific point in time) is limited by:

$$T_{lim} = \frac{L}{dE_u/dt} \tag{5.6}$$

T_{lim} The maximum amount of time in which a charging station has to be found
L The energy level of the unit at a specific point in time
dE_u/dt The total energy use (per unit time)

As seen in equation 5.6, decreasing the energy use of the unit will extend the maximum searching time. This total energy use is made up of different subcomponents, these include:

- Energy used for movement

- Energy used for processing
- Energy used by sensors
- Energy used by communication devices

These possibly vary depending on the way of operation of the unit. Since nothing is known about these factors, assumed is that $^{dE_u}/_{dt}$ is constant.

6

Individual (independent) behaviour

This chapter focuses on the first behavioural mode, as described in chapter 5. The moving and searching pattern, along with its effectiveness, will be discussed in this chapter.

6.1. Individual moving behaviour

The goal of the swarm is to operate for a (potentially) infinite amount of time. Since a group of units make up the swarm, the individual units have to make sure that the swarm will reach this goal. To make sure this goal is strived for by the units, they are designed to have almost the same goal (operate for an as long as possible period of time). Assuming the LoR is met, the only variable that causes operation to cease, is the energy level. In order for a unit to operate for an as long as possible time, it has to maintain its energy level high enough to function. Now, by operating energy is used. To keep the energy level of the individual units high enough, the energy has to be replenished. The environment prescribes a non-moving charging station as a point where energy levels can be raised. So to keep up the energy level, the unit has to move towards the station in order to charge. In addition, the position of the charging station is unknown and is only known until the unit is within a certain range of the charging station. So, finding the charging station is key for the operation of the units. In order to make the chance of finding a charging station as big as possible, as much as possible area of the environment must be discovered, using as little as possible energy:

$$\frac{dA_c}{dE_u} \tag{6.1}$$

A_c Area covered by a unit from start of operations until a specific moment in time
E_u Total energy used per agent from start of operations until that same point in time

Equation 6.1 must be as big as possible. For evaluation of different options later on, the point in time at which the entire search field is covered and the point in time at which the charging station is found could be relevant. Furthermore, of importance here is the assumption that E_u/t is constant. The chance of finding a charging station increases with increasing area searched, because the amount of area covered by charging station relative to the undiscovered area increases with every bit of area searched

$$P_f = \frac{A_{cs}}{A_u} \tag{6.2}$$

P_f Chance of finding charging station
A_{cs} Area covered by charging station
A_u Undiscovered area

This means with every bit of area searched, the chance of the next bit of area being the area covered by a charging station increases. To decide upon the best individual moving behaviour, in first place different options were come up with. These options can be read in table 6.1. To rank the different options, criteria had to be set up. The criteria to rank the different algorithms are the following:

Method	Angle ($0 \leq \theta \leq \pi$)	Turn Direction	Distance
Brownian Motion	$\frac{1}{2}\pi$	Bernoulli, $p = 0,5$	
Lévy Flight	Uniform distribution	Bernoulli, $p = 0,5$	Lévy distribution
RARD		Bernoulli, $p = 0,5$	
FARD			
RAFD			

Fixed Stochastic

Table 6.1: Different methods, consisting of fixed and stochastic parameters

1. ${}^{dA_c}/_{dE_u}$
2. Required processing power (simplicity)
3. Time required to fill entire search space
4. Time required to find charging station

6.2. Construction of the individual behaviour

The movement algorithm is constrained to the physical characteristics of the (surrounding of the) units. These constraints include:

1. When moving from one point to another, the units must cross every point in between
2. Search space does not converge towards the solution (charging station)

The second constraint is inherent to the property of the detection of a charging station. It can only be detected when it is within a certain perimeter around the station. The rest of the search space does not give any directional information regarding the location of a charging station. For the construction of the algorithm, the movement of the units is split up in two:

1. Changing direction
2. Moving in a straight line

From which the following variables are extracted:

θ Angle of directional change
D_p Length of line moved

When taking in mind the requirement of not moving in a fixed pattern, these variables can not be considered constant (or predetermined) at the same time. Rather, at least one must be dependent on any variable which is independent of the variables giving information about the state of the unit, the relationship of the unit to its environment, or the environment itself. As in, a random factor should be added to the process of determining these variables. The chosen solution to this is making sure θ and D_p are dependent on stochastic variables. Some studies [1, p. 52] have hypothesized Lévy flights or Lévy walks as an optimal search strategy adopted by many foragers in nature. Where it is worth mentioning that foragers adapt their search strategy based on the density of prey , sometimes switching between Lévy flights/walks and Brownian motion [1, ch. 3]. These Lévy flights/walks (LF/LW) base D_p on a Lévy distribution, and θ on a uniform distribution. The options come up with are summed up in the following table. For the RARD, FARD and RAFD methods, the R stands for random, F for fixed, A for angle (θ) and D for distance (D_p).

Two remarks have to be made concerning table 6.1. The distribution of the ”random” variables of the RARD, FARD and RAFD methods could be of any kind. In first place, a uniform distribution was assumed. For optimization purposes these can be changed, likely altering their performance. Secondly, the option of basing a variable upon either an input argument or historical data was looked at. These will likely give a kind of behaviour that can be determined beforehand. The question is if the resulting path must be considered ”pre-defined” or not (LoR, 1.4). To be safe, in order for these variables to be used as a ”random” input, a random factor must be added to this variable.

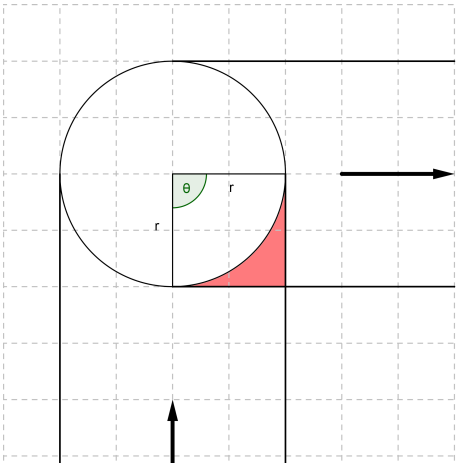


Figure 6.1: Turning

6.3. Design influencing factors

6.3.1. Turning

To start off, in order for a unit to be able to detect a charging station it has to have a sensorial device. Assumed is that the charging stations can be detected by the BLE module, which gives the unit a circular field of view. When performing a turn while having a circular field of view, the centre of rotation is of importance when looking at the $\frac{dA_c}{dE_u}$ ratio (see equation 6.1) while turning. This point could be either:

- Outside the field of view
- Within the field of view
- On the edge of the field of view
- On the midpoint of the field of view

For covering area while turning, it is essential to make sure the centre of rotation is either on the edge of the field of view or outside of the field of view. This is explained by the following.

Centre of rotation is on the midpoint of the field of view

In this case, with every turn energy is used without increasing the area of search. In addition, when moving in a straight line after having turned on a spot (after having moved), a piece of area is covered twice (coloured red in figure 6.1) . This total area covered twice is dependent on the angle turned, and the radius of the search field around the agent.

Centre of rotation is within the field of view

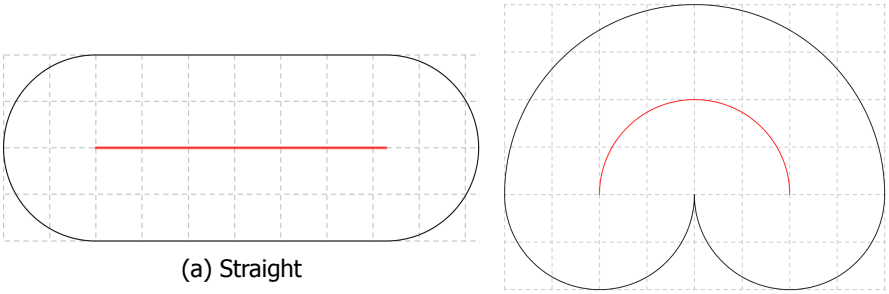
When the centre of rotation is within the field of view, there is an area covered twice as in figure 6.1. However, this area covered decreases with increasing distance between the midpoint of the field of view, and the centre of rotation.

Centre of rotation is on the edge of the field of view

When the centre of rotation is on the edge of the field of view, no area is covered twice whenever the turning angle (θ) is smaller than 180 degrees. With the centre of rotation being on the edge of the field of view, the smallest possible turn is made without covering area twice. This means as much area is covered as with walking in a straight line.

The area covered in 6.2a and 6.2b is equal. The red lines have a length π, the detectionradius is considered 1:

$$A_{straight} = \pi \cdot (2 \cdot r) + \pi \cdot r^2 = 3\pi \tag{6.3}$$



(a) Straight

(b) Turn

Figure 6.2: Area coverage in turns and going straight

$$A_{turn} = \frac{1}{2} \cdot \pi(2 \cdot r)^2 + \pi \cdot r^2 = 3\pi \tag{6.4}$$

Turning with the centre of rotation on the edge of the field of view therefore is as effective as walking in a straight line. However, turning with the centre in the middle of the field of view (on spot turning) covers area twice and therefore is less effective.

Centre of rotation is outside of the field of view

In this case no area is covered twice as long as the turning angle (θ) is smaller than 180 degrees. This means as much area is covered as with walking in a straight line.

6.3.2. Turning and the environment

As seen earlier (in 6.1), the requirements and assumptions regarding the environment influence the design choices made for the swarming behaviour. Next to choosing upon a behaviour which aims at searching area effectively, the requirement that operation is within a finite space has its influence on the way the individual movement is constructed as well. If a single unit would operate within an infinite space, performing the searching behaviour earlier described, it would not matter if turns would be made or not. As long as the centre of rotation is outside, or on the edge of the field of view of the unit. Furthermore, the sum of all turning angles (θ) can not exceed + or - 90 degrees. This of course regarding the $\frac{dA_c}{dE_u}$ ratio, with $E_u/unit\ time$ taken to be a constant (see 1.4.4). Now, restricting the unit to a finite space, the need for turning is introduced. In order for the unit to cover as much as possible area with its energy load, it can not stop its search at the boundary of the finite space. Nor can it cross the boundary to continue its search outside of the finite space. Instead, it should make a turn as effectively as possible, somewhere between the point of detection of the boundary and the point of crossing the boundary of this finite space. Where effectiveness is again measured in the amount of area covered, of which the area outside of the search space does not count as area. Zooming out, the combination of turns and movements in straight lines can add up to a point at which the unit intersects with its own historical path. This would mean area is covered twice, which makes its behaviour less effective.

6.3.3. Conclusion on turning

To sum up, effectiveness is lost where area is covered twice. This happens whenever a unit crosses its previous path (might it be immediately after turning, or later). Furthermore, effectiveness is lost where area is covered that certainly does not contain a charging station. This type of area is the area out of the boundaries of the finite space the unit operates within. Lastly, energy spent on turning while not moving forward is a loss in effectiveness as well.

6.4. Conclusion

The conclusions that can be drawn from the preceding paragraphs are the following.
The first conclusion is that to cover as much as possible area with a given amount of energy, the behaviour should cover as little as possible area which definitely not contains a charging station. In addition, the energy use should be minimised. However, since the energy use for each manoeuvre is

not clear, a constant energy use per unit time is assumed.
Secondly, in order to fulfil the goal of operating for a potentially infinite amount of time, the chance of survival has to be 100% for each unit. For this, a unit has to be able to search the entire search space with its amount of energy available. This of course assumes that only one unit is present within the search space. Concluding from the LoR, this one unit does not make up a swarm. So in case of the unit not being able to accomplish this task nothing can yet be concluded about the potential of the swarm to fulfil its goal. The idea is that at least to make the chance of the swarm accomplishing the goal as high as possible, the most effective individual behaviour must be explored and chosen.
The last conclusion is that the different algorithms should be tested on the criteria 6.1. This is because nothing is known about the effectiveness of the different algorithms concerning this situation.

6.5. Design choice

To make sure the requirement of having a not-predefined path is met, the independent behaviour implements two random factors. A random angle is chosen, after which a random distance is chosen, etc. A uniform random distribution is chosen for the generation of the distances walked in a straight line. The influence of the type of distribution on the entire swarm is not known, and therefore taken to be uniform. Due to the assumption of a constant energy use, it does not matter if a turn is made or not, as long as the turning radius is bigger or equal than the field of view. For this reason the angle does not seem to matter either, the actual influence on the overall behaviour of the swarm is unknown and should be explored. This type of behaviour was earlier mentioned under the name RARD. Following table gives an insight in each variable and its value/way of generation.

Behavioural mode	**Anti collision**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	Fixed
	θ	Random (uniform 0-180)
	θ_{dir}	Random (ber(0.5))
Limits between modes	f_1	L_1-L_2

For movement on a 2D plane, sudden changes in speed are unlikely and seem unnecessary for an individual. For these reasons it is taken to be fixed. Due to the assumption of a constant energy use, it does not matter if a turn is made or not, as long as the turning radius is bigger or equal than the field of view. For this reason the angle does not seem to matter either, the actual influence on the overall behaviour of the swarm is unknown and should be explored. With R_{turn} approaching infinity, the turn approaches a straight line. This is not the goal, neither is the goal to have an as small as possible turning radius since this means unnecessary area is searched. So, to give the agents a versatile/dynamic behaviour while not wasting energy the centre of rotation is chosen to lay at the edge of the search field of the agent.

7

Anti-Separation behaviour

From the same,zoomed in perspective as in the chapter of individual behaviour, the behavioural mode of seperation-prevention is analyzed in this chapter. In order to prevent two units from crossing the maximum limit of the relative distance, a contactless sensor has to make sure an approaching separation is detected. Adjustments in the way of moving can be made in order to meet the requirement of staying within the fixed maximum range of separation. These adjustments are mentioned in the chapter dealing with the behaviour in general as well. To repeat:

- Movement speed
  - Relative increase/decrease
  - Absolute increase/decrease
- Movement direction
  - None
  - Forward
  - Backwards
- Turning radius
  - Zero
  - Smaller than field of view
  - Bigger than or equal to field of view
- Turning angle
- Turn direction

Deciding upon the generation of these parameters can be done in the following ways:
Where with random decision making the distribution can still be chosen. The criteria for the different methods are considered to be the following:

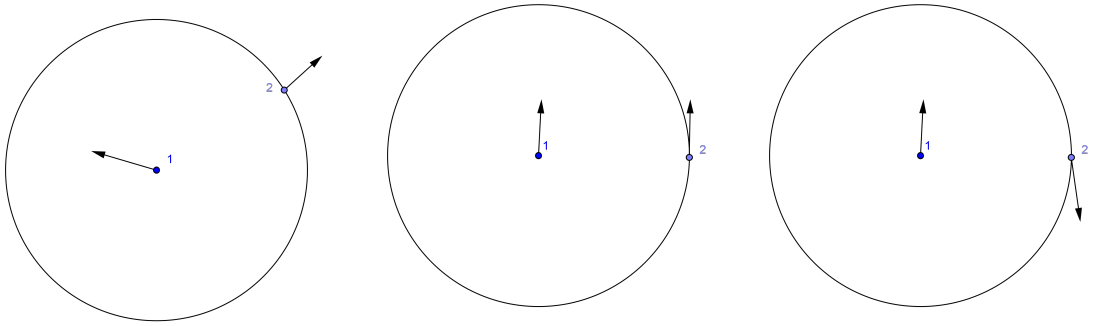
Movement speed	**Movement direction**	**Turning radius**	**Angle**	**Turn direction**
Fixed	Fixed	Fixed	Fixed	Fixed
Input based	Input based	Input based	Input based	Input based
History based	History based	History based	History based	History based
Random	Random	Random	Random	Random
None	None	None	None	None

Table 7.1: Parameters collision prevention

- $\frac{dA_c}{dE_u}$
- Effectiveness
  (how many times does a unit have to make the decision in order to prevent the collision from happening)

7.1. Design Choice

From analysing the way two units separate, a simple solution can be deduced which should theoretically prevent separation. This solution is only based on the distance between the two units, and is independent of the historical/future distances. It is, turning 180 degrees with an R_{turn} of 0. R_{turn} can be taken bigger, until half of the distance between the maximum distance of separation and the initiation point of this mode. Experiments must show in what way this influences the global behaviour. Whenever the units A and B are on the verge of separation, a turn of 180 degrees will prevent this. Even with small angles, as can be seen in figure 7.1b.



(a) agents leaving eachother

(b) slight increase in relative distance 1

(c) slight increase in relative distance 2

angles:

Figure 7.1: different movements

Basis to this solution is the fact that two units can only cross the maximum distance of separation whenever their movement is away from each other (the relative distance increases). Turning 180 degrees will change this into a behaviour which has a movement which decreases their relative distance. Variations might be chosen for this concept. But due to the simplicity, the above stated actions are assumed preferable. When displaying it in the table of movement this behavioural mode looks like:

Behavioural mode	**Anti separation**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	Fixed
	θ	Fixed
	θ_{dir}	Fixed/Random(ber(0.5))
Limits between modes	f_3	$> L_2$

8

Anti-collision behaviour

Units may not collide into each other and/or objects during operation. Therefore the unit should act accordingly on a potential collision: the anti collision behaviour. This chapter will focus on the small scale prevention: what should a unit do when it is driving towards an object or other unit.

8.1. Interactive behaviour: no collisions

In order to prevent physical contact, a contactless sensor has to make sure an approaching collision is detected. After that, a decision has to be made about a directional change in movement to prevent the collision from happening. The moment at which the decisions are executed is dealt with later.
Different variables can make up the behaviour to prevent physical contact between units. Each of these variables having their own way of generating. As with the Anti Seperation behaviour 7, the parameters could be based upon the following procedures:

Where with random decision making the distribution can still be chosen. The criteria for the different methods are considered to be the following:

- $\frac{dA_c}{dE_u}$
- Effectiveness
  (how many times does a unit have to make the decision in order to prevent the collision from happening)

8.2. Design Choice

From studying the possible ways of collisions between two agents to occur, one can deduce a simple solution which should theoretically prevent all of these collisions from occurring. This by only knowing the relative distance between two individual agents, without knowing the previous/future distances. The solution is to turn an angle between $90 + arctan(\frac{W_u}{2 \cdot D})$ degrees and 180 degrees in clockwise, or counter clockwise direction (trivial). Being a first concept, R_{turn} is taken 0, this radius likely has a maximum value dependent on the relative distance at which the mode is initiated and the dimensions of the agents though. See figure 8.1.

In this figure, a graphical representation is given of a single possible case in which the distance D between two agents reaches the point at which the anti collision behaviour is executed. The circles

Movement speed	**Movement direction**	**Turning radius**	**Angle**	**Turn direction**
Fixed	Fixed	Fixed	Fixed	Fixed
Input based	Input based	Input based	Input based	Input based
History based	History based	History based	History based	History based
Random	Random	Random	Random	Random
None	None	None	None	None

Table 8.1: Parameters collision prevention

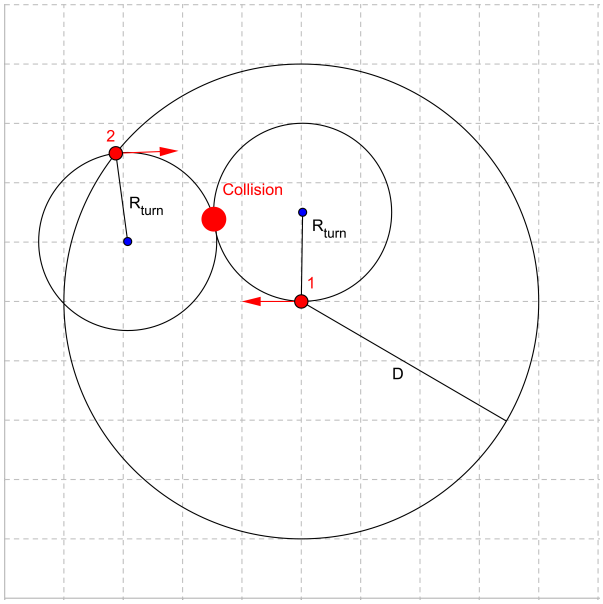


Figure 8.1: Collision prevention

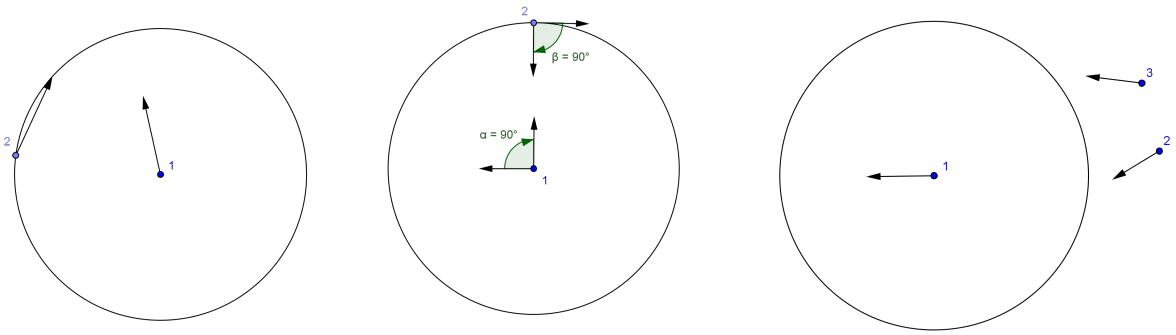
with the intersection point represented by the big red dot are the hypothetical paths of the agents given in small red dots. When choosing parameters leading to the above represented behaviour, a collision will occur. Here, the direction of the agents at the start of execution of the mode, the radius of turning and the dimensions of the agent seem to be of influence. An important key in the analyses of when collisions might occur is the assumption that units have the same speed.

Now, only with decreasing relative distance the units are able to cross the lower limit. This means the directional vectors of the movement of the agents, should be pointing towards each other see 8.2a for this situation. Figure 8.2c gives the situation of increasing relative distances.

If both units would have a relatively small size, a turning angle θ of a little more than 90 degrees would suffice, assuming a turning radius of 0. See figure 8.2b. Here, a boundary condition is shown. The point at which unit B only enters the range of relative distance in a slight amount. When turning for a little more than 90 degrees, the agents will not collide. Another case is given by figure 8.2c.

The above mentioned theory is assumed to be valid, could be proven later, but needs testing anyhow.

Looking at the area covered, turning on the centre of rotation makes this method not all to effective.



(a) Approaching agents (b) Turning agents (c) Trivial movement

Figure 8.2: Movements of agents

However, it is chosen for its simplicity and might be expanded later on if it is worth the addition in complexity.

With the above mentioned solution, the increase/decrease in angle could be randomly chosen between an (input based) boundary and a fixed boundary. Also, even simpler, it could be completely input based. Within the movement table earlier seen, the concepts would look like the following:

Basically, these could be seen as design concepts. As mentioned in the earlier chapter, the random distribution could be modified to alter the output behaviour. From these concepts, a (uniform) random distribution is chosen because it contributes to an undefined/ unpredictable overall behaviour of the different agents in the swarm. Which is desired. To give an overview:

Behavioural mode	**Anti collision**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	Fixed
	θ	Random (between boundaries)
	θ_{dir}	Random (ber(0.5))
Limits between modes	f_2	$< L_1$

9

Another Behavioural Approach

In the chapters 7 and 8 was discussed how individual agents, on a small scale, should react on other swarm members. By thinking of different situations it was made plausible that the interactive behaviour described in that chapter was the minimum to ensure agents stayed together and did not separate. Preventing collisions and staying together can also be regulated on a large scale. The same requirements have to be met, agents can not collide nor seperate. Preventing these occurrences on a large scale involves regulating the overall distribution of units. For instance, collisions are more likely to happen in very busy cities than in a remote desert. In this chapter a solution to the interactive behaviour is discussed from a swarm point of view.

9.1. Behaviour

As can be concluded from the swarm definition and the requirements in chapter 5, there can not be any form of central control. Every unit should decide for itself which way to go. The desired direction can be achieved by turning accordingly, thus the most important parameter from table 5.1 in this behaviour is therefore the Angle. The Angle parameter decides which way the unit goes. As can be seen in table 5.1, the different ways to assign a value to the Angle are as follows:

Angle	
Fixed	Assigning a fixed angle would make unit walk around in circles
Input based	The turning angle will be determined by the information the algorithm receives.
History based	The angle will be based on the earlier turns.
Random	The assigned angle will be random generated.
None	Not assigning a value to the angle would mean that the unit will keep driving straight

This chapter focuses on the input and history based angle. 'None' and 'Fixed' input will result in units walking in a straight line or circle, but since the swarm needs to stay together these ways of assigning a value will be neglected. The random input also doesn't result in a swarming behaviour, thus this method will also be neglected.

The coming sections will focus on input and history based value assigning. The values will be generated through the positions of the other units and the location of the swarm. Not using the positions of the other units will result in a behaviour as if the unit is not in a swarm. The way the positions are translated to an angle value will also be discussed.

9.1.1. Location estimation

To establish the connection between units and to make a searching pattern as effective as possible, it is important for the units to stay relatively close to each other. The maximum separation between the individual units is determined by the maximum connection range as well as the detection range. Besides the maximum distance, there is also a minimum separation distance. When different units come close to each other, the effectiveness of the searching algorithm will decrease and the chance of

colliding grows. The algorithm needs to cope with these limits on separation. In order to know if the distances are within the set limits, the distances between the units should be known. These distances can be calculated in several ways and depend on the available sensors. The units in this swarm will make use of a BLE chip. This chip can be used for transferring data as well as for measuring distance using RSSI.

Knowing distances between each other is not enough information for units within a swarm. The algorithm also needs to know how to walk towards or away from an other unit when the separation distance exceeded the limits. In other words, the algorithm needs to know how the unit is orientated relative to the swarm and what the distance is to the other units in order to calculate a desired heading. Taking in mind that the units will be equipped with a BLE chip, there are several ways to design the location estimation part of the algorithm:

1. Approximate direction
   (a) Trial and error
   This is a history based method. By measuring an increase or decrease in the RSSI a unit can determine whether it's walking from or towards another unit. The desired direction can be determined by logging the RSSI.
2. Exact location
   (a) Time based
   This history based method logs several RSSI measurements and then uses these to calculate the approximate location of the other units using trilateration.
   (b) Internal reference
   The Internal Refence method is input based. Every unit advertises the distances of itself relative to others. Therefore each unit knows all the relative distances within the swarm, which can be converted into the locations of the units in a 2D plane. This conversion doesn't have a unique solution, therefore each unit is also equipped with two additional range sensors so make trilateration applicable: internal reference.
   (c) External reference
   This input based method relies on a external reference in the term of three fixed sensors. Each unit calculates it's location relative to the reference point(s) and advertises it's location in polar coordinates to the other units.
   (d) Radar
   The radar method is input based and makes use of a reflector circling around the range sensor. This way the relative distances and angles of other units can be determined.

Trial and Error

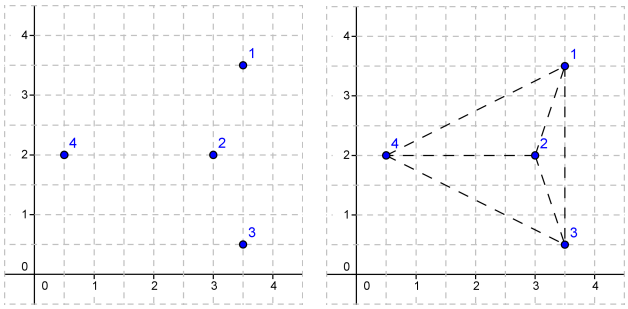
This method is based on a self-learning principle. The unit will log the RSSI while walking in a certain direction. This way it will know if a direction will result in a stronger or weaker signal of an other unit. The more RSSI measurements are logged by a unit, the more reliable the prediction on the signal strength will be. This method results in an approximate desired direction and only relies on RSSI measurements, no data transfer is required.
However, the swarm is continuously changing. All the units within the swarm are moving, so what once was a desired direction can be a wrong direction within a short time. This self-learning principle therefore has to adapt itself to the changing situations. Furthermore, the unit will have to walk in each direction to determine whether it is desired or not.

Time based

The Time Based algorithm logs the RSSI measurements and applies trilateration [12] on these measurements to determine the location of the other units. Trilateration relies on at least three different measurements, taken from at least three different locations that are not in line. The exact position of these locations relative to each other are required to calculate a reliable solution, which is hard to achieve since the turns and distances walked are hard to measure on rough terrain. Furthermore, the measured object, the other units, should stay on a fixed place while the three measurements are made. Since the other units are continuously moving, it is impossible to calculate their exact position using trilateration.

Internal reference

Since all the units are equipped with a range sensor and they advertise their own collected data, each unit can construct a matrix consisting of all the relative distances. The size of this matrix is $n \times n$ with n the amount of units, thus the data transferred grows significantly with an increasing amount of units. The information gathered in this $n \times n$ matrix can be converted into the locations of all the units in a 2D plane using Multi Dimensional Scaling [13]. There are, however, some downsides on using this method.



(a) Dots (b) Relative distances

	1	2	3	4
1	0	1.581	3	3.354
2	1.581	0	1.581	2.5
3	3	1.581	0	3.354
4	3.354	2.5	3.354	0

(c) Relative distance matrix

Figure 9.1: From physical orientation to a relative distance matrix

When the 2D distribution is expressed in relative distances, all the information concerning rotation and mirroring is lost [13, p. 22]. The constructed 2D plane can therefore be rotated (relative to a fixed north) and mirrored in all directions, while the constructed 2D plane still copes with the relative distance matrix.

As can be seen in figure 9.2, a 2D plane without a fixed orientation can't be used by the algorithm. When a unit for instance needs to move away from a crowded place, the algorithm can't tell the desired heading since it doesn't know its own direction relative to the location of the crowded place. This can be accomplished by installing 2 extra range sensors on the unit. Using these 3 sensors, placed as far from each other as possible, the location of other units can be determined using trilateration.

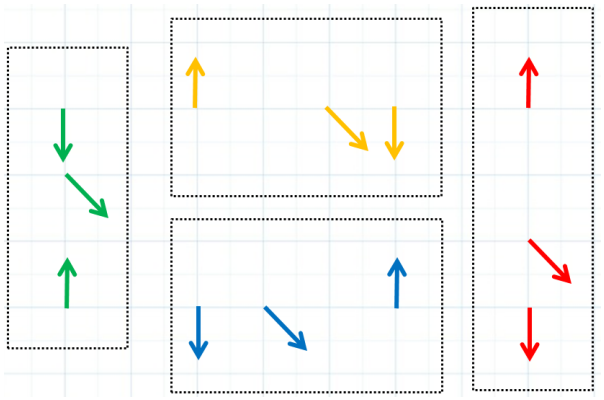


Figure 9.2: Orientation issues without reference

Radar

The radar method is similar to the 'Internal Reference' method since it is also based on the relative distances between the different units. However, this method uses a radar instead of extra sensors and trilateration for reference. This radar method is derived from a lighthouse; around the range sensor circles a material that blocks all radio frequencies. This way the range and the direction of the other units can be determined, thus a unique 2D orientation of the whole swarm can be constructed.
Since the range sensor, a BLE chip, is also used for data transfer, the blocking of the RF signals also interrupts the data transfer. Furthermore this method makes use of an extra moving part: the circling

RF blocking material. Moving parts break down relatively easy and since the functioning of the algorithm depends on the rotating material, this method isn't appropriate for this situation.

External reference

Where other methods make use of the relative distances between the units in a swarm, the 'External reference' method uses the relative distance to a specific reference point. The reference point consists of 3 separate detectable points, so the units can determine their distance as well as their angle relative to the reference point using trilateration. The acquired distance and angle can then be advertised, so all the units within the swarm know each others' exact location.
An other downside of the method is the detectable range of the reference. When a unit is too far away from the reference point, it can't calculate its own exact position. This issue can be resolved by calculating its own position relative to three other units that know their exact position. Three units that know their own exact position can act as a reference point themselves.

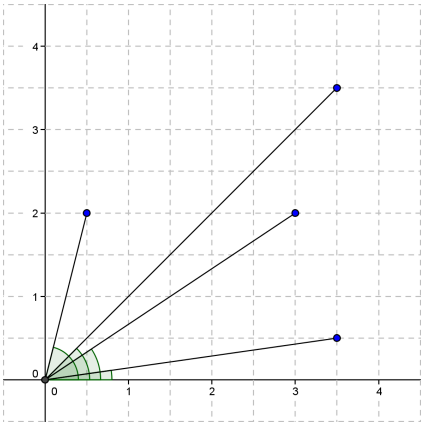
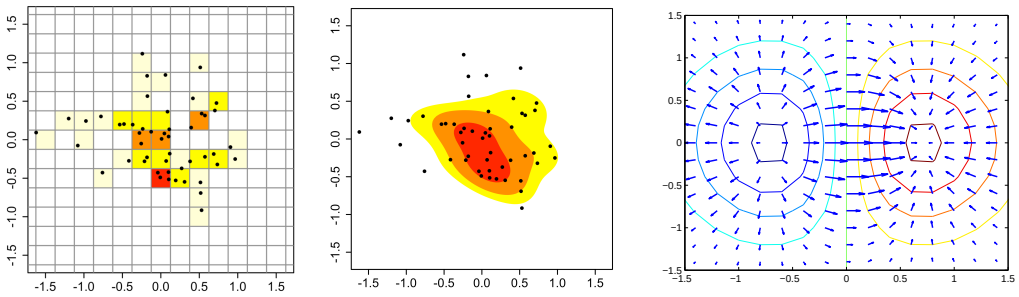


Figure 9.3: Polar reference

9.1.2. Determining heading

Now that each unit knows the position of all the other units relative to itself, the algorithm needs to determine the desired heading. When this desired heading is know, the algorithm can assign a value to the Angle parameter accordingly. Since the units may not get too close or too far from each other we can conclude that the algorithm needs to aim for a certain spread of the units, which can be achieved by sending units towards open areas and sending units away from areas that are too busy. This spread can be expressed by counting the amount of units per area; the density.
The local density in the constructed 2D plane can be calculated using the Multi Variate Kernel Density Estimation [14]. The obtained density field can be considered as an inverted potential field: dense areas have a low potential and open areas have high potential. After a unit localised itself on the



(a) Density calculation (b) Density plot (c) Vectors from density

Figure 9.4: From 2D orientation to a vectorised density plot

potential field, it can calculate the local slopes of the potential field (figure 9.4). The steepest slope connects the highest local density to the lowest local density and is therefore the fastest direction towards or away from a dense area. Thus, if the unit is in a too dense area and needs to move to an open area, the direction can be determined by calculating the steepest slope of its local density field. This method can be used as separation prevention and collision prevention. However, for this method to be usable on collision prevention more research is needed. The exact location of the other units is known, but how these locations translate into an evading manoeuvre is still unknown.

9.2. Conclusion

By comparing all the methods on location estimation, we can construct table 9.1. Some of the methods have downsides that make them inapplicable for this purpose. The Trial and Error method can't provide an accurate desired heading, and can therefore not be used in a swarm. Also, in order to be able to estimate an approximate desired direction, the unit itself has to gather several measurements by walking into different directions. This isn't always possible. The Time Based method is unusable due to the lack of accuracy. It may be possible to achieve the desired accuracy, but then again the algorithm will become a lot more complex. Since the algorithm should be as simple as possible, the Time Based method is not desired for this application. The Internal Reference and Radar method are very familiar, the only difference is the way the methods implement a reference. The Radar method implements a reference with a circling RF blockade and the Internal Reference uses 2 extra sensors and trilateration. These methods are both less suitable for this project due to the limited accuracy of the Internal Reference and the interruption of the data transfer in the Radar method. The method most suitable for an autonomous swarm is the External Reference method. Although it requires an external reference point, it's the best solution for implementation in a swarm: it's a relatively simple and accurate method for location calculation. The required reference point can be provided by the instance that places the units on their location. For example, if the units are on Mars the shuttle they came with can act as a reference or when the units are to discover a cave, the engineers can take care of the reference point by placing at least 3 Bluetooth modules.

	Pros	**Cons**
Trial and Error	No data transfer required Independent of surroundings	Information input dependent on movement No exact direction or location calculation Needs to adapt to changing orientation
Time Based	No data transfer required Independent of surroundings	Inaccurate due to moving of measured units Inaccurate due to own movement Information input dependent on movement
Internal Reference	Direct location calculation Independent of surroundings	At least 2 extra range sensors needed Sensor spread and thus accuracy limited Data transfer grows exponentially with swarm
Radar	Independent of surroundings	Moving parts Indirect location calculation Interruption data transfer Data transfer grows exponentially with swarm
External Reference	Direct location calculation Data transfer constant	External reference required

Table 9.1: Comparing the different methods on location estimation

The use of Kernel Density Estimation provides the possibility to give units a direction in such way that they don't wander too far from the swarm. The units will be directed from dense areas to open areas and vice versa when certain density limits are exceeded. Using the external reference, the movement parameters will be as follows:

Behavioural mode	**Anti collision**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	0
	θ	input based
	θ_{dir}	input based
Limits between modes	f_2	$< L_1$

Behavioural mode	**Anti separation**	
	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	Fixed
	θ	Input based
	θ_{dir}	Input based
Limits between modes	f_3	$> L_2$

10

Conclusion

After a literature study the conclusion about the definition of a swarm is:

> *"A swarm is a large number of homogenous, unsophisticated agents that interact locally among themselves and their environment, without any central control or management to yield a global behaviour to emerge."*

[1, p. 25]

Out of this definition we made up the requirements for the swarm. The most important requirements are:

- The individual units may not make physical contact
- The maximum distance of separation between the individual units is fixed
- The path of the individual units may not be pre-defined

The sensor choice is made by the requirements and the idea to build a physical swarm. A BLE 113 is a module based on RF communication. This module is chosen for ranging the distance between different units. A disadvantage is that it is not possible to see obstacles. To avoid obstacles the advise would be to use ultra-soon sensors. The BLE 113 is besides of measuring the distances, also suitable for translating data between the modules. For now we can't say in what way it will be used but an expectation is that it will be influence the effectivity of the swarming behaviour. By using virtual simulations different algorithms are tested to find out which is suitable for the physical swarm. The goal for this algorithm is discover as much area as possible. Looking at the individual way the best options are the RARD, FARD and RAFD methods. These methods have potentially the highest rate of area discovering for an individual unit. A swarm of these individuals need a algorithm to avoid collisions and keep the group together. This algorithm is based on the characteristics of our sensors. The efficiency of the swarm algorithm depends on the area which is covered twice. The less area covered twice the better the efficiency. The location estimation is a way to achieve a better efficiency. Some ways to design the location estimation algorithm are approximation direction and exact location. The exact location is in favourite because of the algorithm is less complicated and the distribution of units in the area is better.

10.1. concept 1

The design choices made for the creation of the global behaviour from a bottom up perspective are contained in the following table:

Behavioural mode	**Anti collision**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	0
	θ	Random (uniform) between $90 + arctan(\frac{W_u}{2 \cdot D})$ and 180 degrees
	θ_{dir}	Random (ber(0.5))
Limits between modes	f_2	$< L_1$

Behavioural mode	**Independent**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	R_u
	θ	Random (uniform, 0-180 degrees)
	θ_{dir}	Random (ber(0.5))
Limits between modes	f_1	L_1-L_2

Behavioural mode	**Anti separation**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	0
	θ	180 degrees
	θ_{dir}	Fixed/Random (ber(0.5))
Limits between modes	f_3	$> L_2$

10.1.1. Discussion

Implementation of the above mentioned design choices in a (model of a) physical swarm will have to proof if this chosen path is the way to approach the problem. The risk is in the focus on the individual (pairs of) agents, where the interactions with more than one agent are left out. The global behaviour that will emerge is not certain and likely needs multiple iterations in order to fine-tune. It could even fail as a whole. However, it made it possible to make design choices leading to an unsophisticated behaviour.

10.2. Concept 2

From a top down approach, regarding the behaviour of the entire swarm, it seems preferable to average the density of the group of agents. This sthrives to equal the spacing between the agents. In this way, the agents have a 'natural' tendency t to prevent collisions. In addition, since for the functioning of this concept a single agent has to know the place of other agents with reference to itself. Knowing this, the agents can make the choice of the turning angle input based, turning away from a collision or towards an other agent as they leave eachother. The design choices look as follows in this manner:

Behavioural mode	**Anti collision**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	0
	θ	input based
	θ_{dir}	input based
Limits between modes	f_2	$< L_1$

Behavioural mode	**Independent**	
Variables	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	R_u
	θ	Random (uniform, 0-180 degrees)
	θ_{dir}	Random (ber(0.5))
Limits between modes	f_1	L_1,L_2

Behavioural mode	**Anti separation**	
	V_u	Fixed
	M_{dir}	Forward
	R_{turn}	Fixed
	θ	Input based
	θ_{dir}	Input based
Limits between modes	f_3	$> L_2$

10.2.1. Discussion

As with concept 1, the concept has to be implemented in a (model of a) physical swarm in order to verify the expected result. The major setback in this approach is the large amount of processing power needed. The swarm will likely have a more predictable behaviour though, than concept one.

A

Using the BLE Module

A.1. Connecting

The setup uses UART to make the MCU (T-Minus PCB) communicate with the BLE module. To do this we need to connect signal and power wires. The MCU can deliver either 5V or 3.3V. To make it deliver 3.3V the 0 Ohm resistance needs to be removed from the PCB so it becomes an infinite resistance. Then using the data-sheet found in [15, p. 32-34], a 3.3V pin and a ground are connected to the supply pins of the BLE module. See [16] for the pins. Both AVDD and DVDD of BLE modules need to be connected but this specific BLE module has these two internally connected. This means two wires to power the device on one of the supply pairs is sufficient. Next up is the UART connection. In the original programs of J. Rowberg he makes use of the SoftwareSerial library which allows users to assign any pin on the MCU to be RX or TX. This works in theory, but not in practice at the moment. When the signals coming from the MCU using SoftwareSerial are analyzed, they appear to have lost their start bit. For this reason we use a second hardware serial on the MCU: serial 3. [15, p. 33]

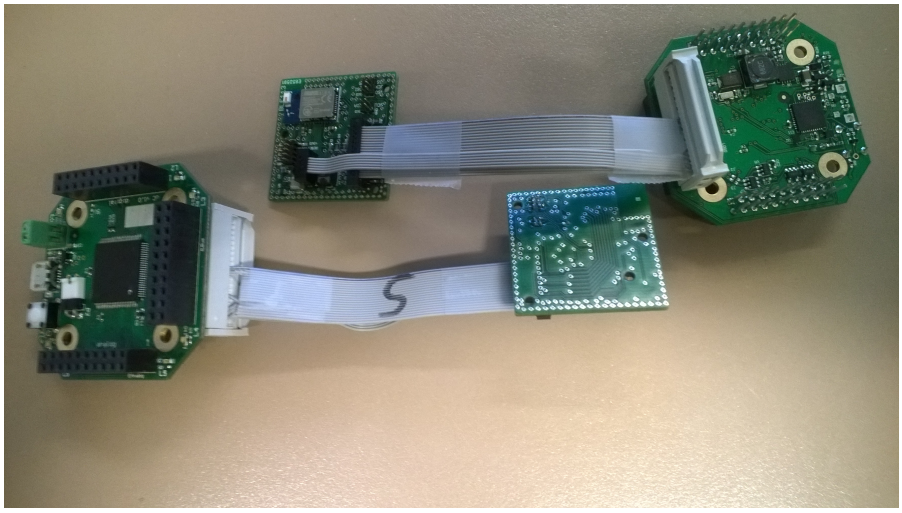


Figure A.1: T-Minus boards connected to BLE modules

To connect the RX/TX of the BLE module the pins need to be defined within the firmware. Pin P0.4 is used as TX and P0.5 as RX. [17] Flow Control pins do not have to be assigned yet. Now there are four cables going from MCU to the module as can be seen in the photo. Here extra cables are seemingly connected for extra physical stability, but they have no function.

A.2. Firmware

Firmware consists of a combination of files. In this case:

- gatt.xml
- hardware.xml
- project.bgproj
- script.bgs (optional BGScript application source code)
- cdc.xml (optional usb descriptor file, not used with BLE113)
- config.xml (optional application configuration file)

The first three are essential for the functioning of the module. The other three are not. The script file (bluegiga-script; bgs) is used to run a program on the BLE module itself. The language is relatively easy and well documented in [18] and [19]. The module can do a lot on its own and for example decide what to advertise or send. Or it can run a script which sends every message received through UART to another specific or non-specific module. In this setup the BLE module requires no script. All commands are executed on the MCU and the results are being send to the module. Therefore the firmware on the module only consists of the first three files given by the list above.

A.2.1. Gatt.xml

In this file there are two things defined: services and characteristics. Services define a group of characteristics used for something. Take the first service:

```
<service uuid="1800">
<description>Generic Access Profile</description>
<characteristic uuid="2a00">
<properties read="true" const="true" />
<value>Zebro Control 115200</value>
</characteristic>
</service>
```

This service is used to group all characteristics that define the device. This service is a standard one and therefor has a UUID of only 4 characters (16 bit). A non-standard UUID will have 32 characters (128 bit) and needs to be randomly generated by the user. On the site [20] there's a complete list of standard UUIDs.

This service is not good for much, but is mandatory! It enables advertising a device name which needs to be done when GAP (generic access profile) is used. In practice this is almost always the case and that's why it's mandatory. Other services and characteristics can be defined by the user. For this consult [21].

One last thing: this file differs for master and slave on only one point. The difference lays within the Cable Replacement Service. After the UUID of the master there's 'advertise=true'. If this is absolutely necessary is yet unclear.

A.2.2. Hardware.xml

This file is very important and its contents are described very well in [22]. Only the USART command will be discussed because [22] describes all other commands very well. The USART command has two attributes that need special attention: 'channel' and 'alternate'.

Channel can be either 0 or 1 and Alternate 1 or 2. This results in four combinations. And it's no coincidence that there are four possible ways of connecting the BLE module via UART shown in table A.1.

When channel 1, alternate 1 is chosen this must be declared within the USART command. RTS and CTS are pins used for flow control and not in use at the moment. This means flow control needs to be disabled with the 'flow=false' statement. This has one consequence: only packet-mode can be used. Because packet mode is used, a length byte at the beginning of a string of bits needs to be added.

Channel	Alternate	RX	TX	RTS	CTS	
0	1	P0_2	P0_3	P0_5	P0_4	
0	2	P1_4	P1_5	P1_3	P1_2	
1	1	P0_5	P0_4	P0_3	P0_2	(Selected)
1	2	P1_7	P1_6	P1_5	P1_4	

Table A.1: UART pin assignment table

A.2.3. Config.xml

This file is optional. [22] Describes what is possible with config.xml. This project uses the following command to ensure maximum throughput.

```
<config>
<manual_confirm />
<throughput optimize="performance" />
</config>
```

A.2.4. Project.bgproj

This file is also well documented in [22]. This file starts a compiler which outputs a hex-file. This file is then flashed onto the BLE module. To flash firmware onto the module the development kit with Bluetooth Smart Software and SDK v1.3.1 are needed. The documentation [20] will help where needed.

The firmware is downloaded onto the module with the help of the development kit from Bluegiga. Connect the development kit to the pc and the module to the kit as shown in the picture.

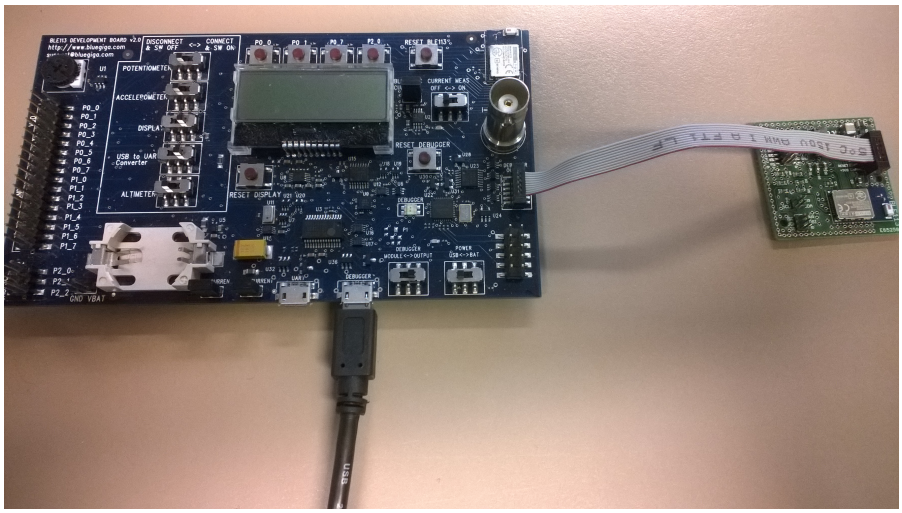


Figure A.2: Bluegiga Development Board connected to BLE module

Press the 'reset debugger' button to check the connection. If the LED turns red put the debugger switch from 'module' to 'output' and press the button again.

A.3. Software

When programming with BGScript all commands are already defined. This is not the case when code is run on an external MCU. Because of this the library from Bluegiga adapted by Jeff Rowberg is used. This library ensures that every command is translated into correct packets which the BLE module can read. For example:

```
ble_cmd_system_hello => [ 04 00 00 00 01 ]
```
(The first number is the length byte)

This library makes programming on a separate MCU as easy as programming in BGScript. To 'install' this library, put the three files in the same folder as the software file. (In this case an Arduino file.) The files are: BGLib.cpp, BGLib.h and BGLibConfig.h. They were made as any other Arduino library, so no changes are necessary.

A.4. Swarming Zebro Software v1.7

This project enables the master to send numbers 1-9 to the slave and make it turn on the corresponding LED. The code speaks for itself, but there are a few fundamental things that are important. First the master needs to scan for other modules and if present, discover these modules. When a module is discovered the `scan_response` event is triggered. Within this event a `connect_direct` statement is placed to connect to the discovered module. When the connection is made the `connection_status` event is triggered. This is conformation that connecting was successful. For the slave we only need to put the module in discoverable mode via `gap_set_mode`. Now a value must be changed on the slave within the GATT file. This can be done with the `attclient_attribute_write` statement from the master. When the write is successful the slave creates an `attributes_value` event. This event shows a value was changed. Within this event a `write_response` needs to be send confirmation back to the master to let it know the write was successful. All of the above ensure steady communication.

A.5. Used materials

- Ble module designed by Elpasys
- Modified Arduino-like pcb by T-Minus (ATMega 2560)
- Bluegiga Development Kit
- Farnell-componenten: order number
  - 20 core ribbon cable: 1207757
  - RECEPTACLE, IDC, S/RELIEF, 2.54MM, 20WAY: 1103920
  - RECEPTACLE, IDC, 1.27MM, 20WAY: 2289785
  - CONNECTOR, RECEPTACLE, IDC, 1.27MM, 6WAY: 1865334

B

V-Rep & Matlab

B.1. Introduction

As can be read in 1.5.2, the project started of with creating a model in V-Rep [23]. V-Rep is a robot simulation environment in which it is possible to simulate all kinds of robots. In this project is used to simulate the behaviour of the units. In this project a model is derived from the tutorial robot 'BubbleRob' which has very basic controls:

- Distance sensor
- Motor on right wheel
- Motor on left wheel

The big advantage of V-rep, is that it can be linked to third party programs as Matlab. This enabled us to design a algorithm for the units in Matlab and check the results in V-Rep.

B.2. Linking V-Rep with Matlab

It is quite hard to link V-rep with Matlab. There are several `.dll` files needed and there are several settings that need to be set.

The files required for linking Matlab with V-rep:

- `remoteApi32bit.dll`
- `remoteApi64bit.dll`
- `remoteApiProto.m`
- `remApi.m`

These original files can be found in the V-rep directory on your Hard Disk. In order to make the simulation run on all computers (32bit, 64bit, Mac), the `remApi.m` is adjusted together with the names of the `.dll` files. It is possible to receive the adjusted files by contacting one of the authors.

Furthermore, it is important to get the settings right. For this project the following settings were used, as described in the V-Rep help file:

IP adress	127.0.0.1
Port	19997

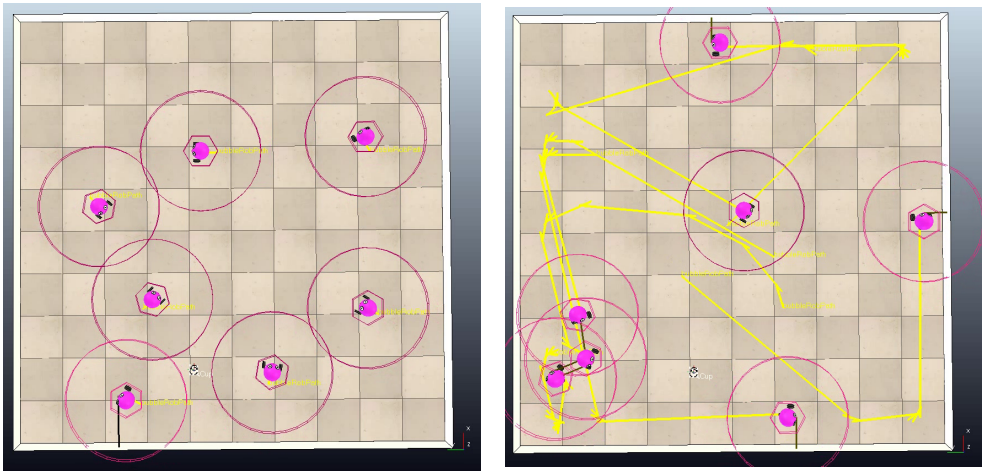
B.3. Running algorithms in Matlab

The simulation is controlled by two different Matlab files:

- `Main.m`
- A controller (i.e. `RARD.m`)

The `Main.m` file sets up the link with V-rep. It also clears everything, sets some parameters and prepares Matlab and V-Rep for the oncoming simulation. In the end `Main.m` calls for the controller. This controller can be interchanged. Development of a controller is an ongoing process. In the example in figure B.1 you can see the behaviour of a very basic swarm:

Units drive straight until they detect something. When they detect something, they drive backwards while turning a fixed angle.



(a) Starting position (b) Position after 41 seconds

Figure B.1: V-Rep model at starting position and after 41 seconds

Later in the project this controller is used to implement Multi Dimensional Scaling. This controller can be found in C.2.

C

Matlab files

C.1. Main.m

This file starts the connection between V-rep and MATLAB. After the connection is established the simulation will be started with the chosen controller.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Main.m                 %
%              v2.0 10-11-2014            %
%           Bachelor Eind Project         %
%                                         %
%       Stijn Seuren / Floris Rouwen      %
%     Marcel Ceelen / Cees van der Geer   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
USE OF MATRIXES
Handles Matrix
 Used to store all the robot handles in 1 matrix. Rows from left to right:
                            [S LM RM]
Error Matrix
 Used to store all the error values, returned by V-rep. These values are
 checked regularly throughout the file to make sure model runs fine and
 can be used for debugging. Rows from left to right:
   [SHandle LMHandle RMHandle SValue LMValue RMValue Heading Reverse]
%}


%% Clean Matlab
clc
clear all
close all
disp('Program started');
disp('Matlab cleared');

%% Settings for the connection and simulation
% Connection settings
ip   = '127.0.0.1';      % Localhost
port = 19997;            % Choose port, below 19997 are preferred for server
    services starting at V-REP start-up

% Simulation settings
t                 = 100;        % Simulation time [s]
v                 = 5;      % Initial speed

% Matlab settings
```

```
39 pause on
40 hold on
41
42 % Controller settings
43 detobjrange      = 0.25;     %Detection object range
44 detgoalrange     = 0.5;      %Detection goal range
45 dbtime           = 3;        %Drive back time [s]
46 longdrive        = 10e7;
47
48 % Initial settings
49 stop             = 0;
50
51
52 %% Connect to V-rep using API key
53 vrep=remApi;
54 vrep.simxFinish(-1); % Close all opened connections
55 clientID=vrep.simxStart(ip,port,true,true,5000,5); % Initiate connection with V-
       rep
56 if (clientID>-1)
57   disp('Connected to remote API server');
58     vrep.simxAddStatusbarMessage(clientID,'Matlab connected to V-rep',vrep.
           simx_opmode_oneshot);
59 else
60   disp('Failed connecting to remote API server');
61 end
62
63 %% Simulation Code
64
65
66 if (clientID>-1)
67     [res,objs]=vrep.simxGetObjects(clientID,vrep.sim_object_proximitysensor_type,
           vrep.simx_opmode_oneshot_wait);
68     if (res==vrep.simx_return_ok)
69     fprintf('Number of robots in the scene: %d\n',length(objs));
70         units = length(objs);
71   else
72     fprintf('Remote API function call returned with error code: %d\n',res);
73     end
74
75     % Form matrices
76     speed       = ones(units,2)*v*3.1415;
77     % speed(1,:)= 0;
78     error       = zeros(units,9);
79     handle      = zeros(units,3);
80     distance    = zeros(units,units);
81     Robotinfo   = zeros(units,(10+units));
82
83
84     % Receive handles
85     disp('Receiving handles..')
86     for i=1:units;
87     [error(i,1),handle(i,1)] = vrep.simxGetObjectHandle(clientID,['
           remoteApiControlledBubbleRobSensingNose#',num2str(i)], vrep.
           simx_opmode_oneshot_wait);
88     [error(i,2),handle(i,2)] = vrep.simxGetObjectHandle(clientID,['
           remoteApiControlledBubbleRobLeftMotor#',num2str(i)], vrep.
           simx_opmode_oneshot_wait);
89     [error(i,3),handle(i,3)] = vrep.simxGetObjectHandle(clientID,['
           remoteApiControlledBubbleRobRightMotor#',num2str(i)], vrep.
           simx_opmode_oneshot_wait);
```

```
90      [error(i,5)]= vrep.simxSetJointTargetVelocity(clientID,handle(i,2), speed(i,1)
            , vrep.simx_opmode_oneshot);
91      [error(i,6)]= vrep.simxSetJointTargetVelocity(clientID,handle(i,3), speed(i,2)
            , vrep.simx_opmode_oneshot);
92      end
93      if any(error(:))<= (vrep.simx_return_ok+1) % Error debug
94          disp('All handles received')
95
96      else
97          disp('Failed to load all handles');
98          disp(['Sensor handle error: ',mat2str(error(:,1))])
99          disp(['Left motor handle error: ',mat2str(error(:,2))])
100         disp(['Right motor handle error: ',mat2str(error(:,3))])
101         stop = 1;
102     end
103
104     % Set Goal to find
105     [Goalerror,Goal] = vrep.simxGetObjectHandle(clientID,'Goal', vrep.
            simx_opmode_oneshot_wait);
106
107     vrep.simxStartSimulation(clientID,vrep.simx_opmode_oneshot);    % Start
            simulation
108     disp('Simulation started');
109     testtime = datenum(clock + [0, 0, 0, 0, 0, t]);
110
111     %FARD %Fixed Angle, Random Distance
112     %RAFD %Random Angle, Fixed Distance
113     RARD %Random Angle, Random Distance
114     %Zebrocontrollerv2 % Start the Zebrocontroller
115
116     axis([-4 4 -4 4])
117     text(location(:,1)+0.1,location(:,2),strsplit(num2str(1:units)))  %Add numbers
            to graph
118
119 %     [bandwidth,density,X,Y]=kde2d(location,128);
120 %     contour3(X,Y,density,50), hold on
121 %     plot(location(:,1),location(:,2),'r.','MarkerSize',8)
122
123
124     vrep.simxPauseSimulation(clientID,vrep.simx_opmode_oneshot); %Stop simulation
125     disp('Simulation ended');
126 end
127
128
129 %% Closing down
130 vrep.simxAddStatusbarMessage(clientID,'Matlab closes connection',vrep.
        simx_opmode_oneshot);
131 vrep.simxFinish(clientID); %Close connection with V-rep
132 vrep.delete(); % Unload the library
133 pause off
134 disp('Program ended');
```

C.2. RARD.m

This is the RARD controller.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Zebro.m                     %
%               v1.0 23-09-2014                 %
%            Bachelor Eind Project              %
%                                               %
%       Stijn Seuren / Floris Rouwen            %
%     Marcel Ceelen / Cees van der Geer         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This m-file contains de controller for the individual Zebro's. It can be
% run multiple times by Matlab to control multiple zebro's in one V-rep
% scene.
%%

% Create reference
[reterr, ~, ~, retFloats, ~]= vrep.simxGetObjectGroupData(clientID,vrep.
    sim_object_proximitysensor_type,9,vrep.simx_opmode_oneshot_wait);
error(1:units,7) = reterr;
movement         = vec2mat(retFloats,6);
position         = movement(1:units,1:3);
orientation      = movement(1:units,4:6);

for i=1:units
    for j=1:units
        distance(i,j) = pdist([position(i,1:3);position(j,1:3)],'euclidean');
    end
end
MDS          = squareform(distance);
location_ref = cmdscale(MDS);
%location_ref = location_ref(:,1:2);
tic

%%
while (vrep.simxGetConnectionId(clientID)>-1) && (stop == 0) && (now < testtime)
    A = rand*45; %Random degrees
    D = 2;  %Random distance driven [s]

    [reterr, ~, retInts, retFloats, ~]= vrep.simxGetObjectGroupData(clientID,vrep.
        sim_object_proximitysensor_type,13,vrep.simx_opmode_oneshot_wait);
    error(1:units,4) = reterr;
    detection        = double(vec2mat(retInts,2));
    detectedpoint    = vec2mat(retFloats,6);
    detdistance      = sqrt(sum(detectedpoint(1:units,1:3).^2,2)); %Calculate
        distance to detected object

    [reterr, ~, ~, retFloats, ~]= vrep.simxGetObjectGroupData(clientID,vrep.
        sim_object_proximitysensor_type,9,vrep.simx_opmode_oneshot_wait);
    error(1:units,7) = reterr;
    movement         = vec2mat(retFloats,6);
    position         = movement(1:units,1:3);
    orientation      = movement(1:units,4:6);

    for i=1:units
        for j=1:units
            distance(i,j) = pdist([position(i,1:3);position(j,1:3)],'euclidean');
        end
    end
```

```
54      orientation       = 360 - ((radtodeg(orientation(:,2)).*sign(orientation(:,1)))
            +90);
55      robotnumber       = (1:units)';
56      spacer            = zeros(units,3);
57
58      %Robotinfo= [     1           2         3-4         5       6-7  8-9-10 10:
            units];
59      Robotinfo = [robotnumber,orientation,detection,detdistance,speed,spacer,
            distance];
60  %     MDS = squareform(distance);
61  %     location = cmdscale(MDS,2);
62  %     [d,location] = procrustes(location_ref,location,'Reflection',true,'Scaling',
        false);
63  %     plot(location(:,1),location(:,2),'.')
64  %     location_ref = location;
65
66      for i=1:units;
67          if (error(i,4)<= vrep.simx_return_novalue_flag) % Proximity sensor read
                successful
68
69
70  %             % CASE: Driven long in a straight direction
71  %             if  (now < longdrive)
72  %                 speed(i,1) = 0.5*3.1415;
73  %                 speed(i,2) = -0.5*3.1415;
74  %                 [error(i,5)]= vrep.simxSetJointTargetVelocity(clientID,handle(i
        ,2), speed(i,1), vrep.simx_opmode_oneshot);
75  %                 [error(i,6)]= vrep.simxSetJointTargetVelocity(clientID,handle(i
        ,3), speed(i,2), vrep.simx_opmode_oneshot);
76  %                 if [error(i,5) error(i,6)] > vrep.simx_return_novalue_flag  %
        Error debug
77  %                     disp('Failed in setting velocities')
78  %                     disp(['Left motor handle speed error: ',num2str(error(i,5))
        ])
79  %                     disp(['Right motor handle speed error: ',num2str(error(i,6))
        ])
80  %                     stop = 1;
81  %                 end
82  %                 longdrive = 10e7;
83  %                 error(i,9)=1;
84  %                 disp('long')
85  %
86              % CASE: Robot detects an object, other than the Goal
87              if (Robotinfo(i,5) < detobjrange) && (Robotinfo(i,5) > 1e-30) &&
                    Robotinfo(i,3) && error(i,8)==0
88                  speed(i,1) = 0.5*3.1415;
89                  speed(i,2) = -0.5*3.1415;
90                  [error(i,5)]= vrep.simxSetJointTargetVelocity(clientID,handle(i,2)
                        , speed(i,1), vrep.simx_opmode_oneshot);
91                  [error(i,6)]= vrep.simxSetJointTargetVelocity(clientID,handle(i,3)
                        , speed(i,2), vrep.simx_opmode_oneshot);
92                  if [error(i,5) error(i,6)] > vrep.simx_return_novalue_flag  %
                        Error debug
93                      disp('Failed in setting velocities')
94                      disp(['Left motor handle speed error: ',num2str(error(i,5))])
95                      disp(['Right motor handle speed error: ',num2str(error(i,6))])
96                      stop = 1;
97                  end
98                  error(i,8) = Robotinfo(i,2)+ A; %Starting angle
99                  if error(i,8)>= 360
100                     error(i,8)=error(i,8)-360;
```

```
101                 end
102                 error(i,9) = 1;
103 
104             % CASE: After evading object
105             elseif (Robotinfo(i,5) > detobjrange) && (error(i,9)==1) && (Robotinfo
                    (i,2) >= error(i,8))
106                 speed(i,1) = 3.1415;
107                 speed(i,2) = 3.1415;
108                 [error(i,5)]= vrep.simxSetJointTargetVelocity(clientID,handle(i,2)
                        , speed(i,1), vrep.simx_opmode_oneshot);
109                 [error(i,6)]= vrep.simxSetJointTargetVelocity(clientID,handle(i,3)
                        , speed(i,2), vrep.simx_opmode_oneshot);
110                 if [error(i,5) error(i,6)] > vrep.simx_return_novalue_flag  %
                        Error debug
111                     disp('Failed in setting velocities')
112                     disp(['Left motor handle speed error: ',num2str(error(i,5))])
113                     disp(['Right motor handle speed error: ',num2str(error(i,6))])
114                     stop = 1;
115                 end
116                 error(i,8) = 0;
117                 error(i,9) = 0;
118                 longdrive = datenum(clock + [0, 0, 0, 0, 0, D]);
119 
120             % CASE: Goal found
121             elseif (Robotinfo(i,5) < detgoalrange) && (Robotinfo(i,5) > 1e-30) &&
                    (Robotinfo(i,4) == Goal) && Robotinfo(i,3)
122                 vrep.simxAddStatusbarMessage(clientID,'Goal found!',vrep.
                        simx_opmode_oneshot);
123                 disp('Goal found!')
124                 speed(i,1) = 0;
125                 speed(i,2) = 0;
126                 toc
127                 vrep.simxPauseSimulation(clientID,vrep.simx_opmode_oneshot)
128                 stop = 1;
129             end
130 
131         else
132             disp('Proximity read failed!')
133             disp(['Proximity error: ',num2str(error(i,4))])
134             stop = 1;
135         end
136     end
137 end
```

D

Group planning

Week	Groupwork	Activity
36	Structuring Project	
37	Orientation in subject matter	
38	Defining design goal	
39	Setting up LoR	
40	Finishing LoR and start coming up with solutions	Plenary session
41	Design concepts	
42	Design/work out concepts	
43	Work out concepts	
44	Reserved for delays/exams	Methodology test
45	Reserved for delays/exams	
46	Reviewing/testing concepts	
47	Reviewing/testing concepts	
48	Reporting tests/implementation in physical devices	Plenary session
49	Implementation in physical devices/mini paper	
50	Implementation in physical devices/mini paper	Plenary session
51	Testing physical devices	Hand in concept-mini paper
52	Report/minipaper	Christmas holiday
1	Report/minipaper	Christmas holiday
2	Report/minipaper	
3	Report/minipaper/presentation	Hand in mini paper
4	Report/minipaper/presentation	Presentation

E

Meetings

In this appendix, the minutes of the different meetings can be found. The minutes are ordered chronologically and in Dutch.

E.1. Meeting 02-09-2014

Aanwezig:
Groep + Chris

Gepraat over plan van aanpak en doelstellingen. Deze doelstellingen zijn:

1. Randomly walk around the room
2. Randomly walk around the room without collisions between Zebros
3. Randomly walk around the room in a swarming group without collisions between Zebros
4. (Randomly) Walk around the room in a swarming group without collisions between Zebros while having for example synchronized movement or walking in a figure V.

To Do's:

Stijn

- V-rep uitzoeken.
- Cees helpen met Matlab voor uitlezen coördinaten en communicatie met Zebro en wellicht V-rep.
- Mailtje over bonding-activiteit

Marcel

- Regelen afspraak met Sjoerd Tijmons voor vrijdag 05-09-2014.
- Regelen afspraak met Marnix voor maandag 08-09-2014 13:00.
- Regelen met Lopes dat het project ingezonden wordt + opgeven studienummers bij de course via deze link:
  http://threeme.collector-survey.tudelft.nl/nq.cfm?q=77CB2358-6B75-442A-91BA-E858B3250A30
- Met Floris verdiepen in Linux op Zebro met hulp van Marnix. Op 08-09-2014.

Floris

- Verdiepen in swarming behavior.
- Met Marcel verdiepen in Linux op Zebro met hulp van Marnix. Op 08-09-2014.

Cees

- Planning maken.
- Opzet van het verslag maken.
- Samen met Stijn verdiepen in Matlab.

E.2. Meeting 08-09-2014

Aanwezig:
Groep + Marnix van der Heide

Marnix kwam langs om uit te leggen hoe de communicatie tussen de Zebro en een ander device werkt. Deze communicatie verloopt via Wifi. In de Zebro zit een Beaglebone (Linux Computer) die via een dongel via een router voor het Zebro project verbinding maakt. Er kan dan met een andere pc, tablet of telefoon op de localhost van de Beaglebone ingelogd worden om de bewegingen van de Zebro aan te sturen.

Het advies dat Marnix gaf is om zo min mogelijk met de Zebro zelf te doen. Het is een buggy en instabiel systeem en kan ons project benadelen. Hij gaf het advies om met kleine bordjes als T-minus aan de slag te gaan en proberen onze doelen daarmee te bereiken voordat we het een en ander proberen te implementeren in Zebro. Wanneer alles op een klein bordje werkt kunnen we ons zorgen gaan maken over de aansturing van de Zebro. De focus ligt namelijk op het ontwikkelen van autonoom gedrag en niet op het opnieuw ontwerpen van de Zebro.

E.2.1. Losse aantekeningen

In browser ip intypen om in de Zebro in te loggen: IP= $192.168.1.7 : 81$ of $192.168.1.8 : 81$. Wachtwoord: $zebrolight$

In een soort commandcenter van een linuxpc kun je de mappenstructuur van de Beaglebone uitlezen.

Typ dan:
$sshroot$@192.168.1.8 (of 192.168.1.7) verschilt dus per IP adres. (ssh=secure shell)
Wachtwoord: zebrolight

Als je dan de opdracht $./ZebroLight$ ingeeft kom je in de structuur van de Zebro-map terecht en kun je zien in hoeverre de opdrachten die gegeven worden doorkomen.

De Zebro wordt aangestuurd door zogenaamde 'websockets'. Dit zijn poorten die uitgelezen worden door de beaglebone. De website die op de beaglebone draait geeft de aansturingen door via deze websockets (Marnix zou deze webcode doorsturen, ter referentie voor ons). Matlab kan ook de aansturing via deze websockets verzorgen. Frankain, de man die aan de grote zebro op 3ME werkt, weet hier heel veel over. Frankain heeft ook code geschreven om een matlab server op te zetten waarmee de websockets aangestuurd kunnen worden, dat stuurt Marnix ook door. Uiteindelijk kan de aansturing ook via UARD plaatsvinden (een verbinding die op veel controleboards zit, zowel arduino als beaglebone) wanneer de bluetooth geïnstalleerd wordt. De websockets zijn dan niet meer nodig.

Glossary

BLE Bluetooth Low Energy.

CCW Counterclockwise.

CW Clockwise.

FARD Fixed angle, random distance.

KDE Kernel Density Estimation.

LoA List of Assumptions.

LoR List of Requirements.

MDS Multi Dimensional Scaling.

POI Point of Interest, i.e. a charging station.

RAFD Random angle, fixed distance.

RARD Random angle, random distance.

RSSI Received Signal Strength Indication.

SI Swarm Intelligence.

Unit General term for an agent. Could be a Zebro.

V-Rep Virtual Robot Experimentation Platform.

Nomenclature

λ	Wavelength
θ	The turning angle for a single turn
θ_{dir}	The turning direction: clockwise or counterclockwise
A_c	Area covered by a unit from start of operations until a specific moment in time
A_{cs}	Area covered by charging station
A_{tot}	The total area of the search space
c	Speed of light [$^m/_s$]
D	Distance between sender and receiver
d	Distance [m]
D_{max}	Maximum distance of separation
D_p	The distance an agent travels (in a straight line)
D_{tot}	The total distance travelled by an agent
E_u	Total energy used per agent from start of operations until a specific moment in time
E_{tot}	The total amount of energy used by an agent
f	Frequency for Bluetooth 4.0 [Hz]
L	The energy level of a unit
M_{dir}	The moving direction: still, forward of backwards
n	The number of agents
n_p	Constant value that describes accounts for all noise
$P_{r,1m}$	Received signal strength at 1 meter distance [dBm]
$P_r\,(d)$	Received RSSI at certain distance [dBm]
P_r	Received power at the receiver
P_t	Transmitted RSS [dBm]
R_{cs}	The radius of the perimeter around a charging station in which it can be detected
R_{det}	The 'detectionradius' or 'the field of view' around an agent
R_{turn}	The turning radius
T_{lim}	The maximum amount of time in which a charging station has to be found
V_u	Speed of a unit [m/s]
X_σ	Zero mean Gaussian distributed random variable with standard deviation σ, accounts for the random effect of shadowing.

Bibliography

[1] X.-S. Yang, Z. Cui, R. Xiao, A. Hossein Gandomi, and M. Karamanoglu, *Swarm Intelligence and Bio-Inspired Computation* (Elsevier, 2013).

[2] G. Lopes, *Zebro - six legged robot,* http://www.robotics.tudelft.nl/?q=research/zebro-six-legged-robot (2010), [Online; last accessed 07-01-2014].

[3] M. van der Heide, B. Koopman, K. Salz, and Y. Schouten, *Autonomous Hexapod Robot Zebro Light*, Tech. Rep. (Delft University of Technology, 2014).

[4] K. Shahzad and B. Oelmann, *Comparative Study of In-sensor Processing vs.Raw Data Transmission using ZigBee, BLE and Wifi for Data Intensive Monitoring Applications*, Tech. Rep. (Mid Sweden University, 2014).

[5] P. Smith, *Comparing low-power wireless technologies,* DigiKey Online Magazine (2011).

[6] MikMo, *Gobetwino,* http://playground.arduino.cc/Interfacing/GoBetwino (2011), [Online; last accessed 07-01-2015].

[7] G. Mao, B. Fidan, and B. D. Anderson, *Wireless sensor network localization techniques,* Elsevier: Computer Networks **51**, 2529 (2006).

[8] J. Xu, W. Liu, F. Lang, Y. Zhang, and C. Wang, *Distance measurement model based on rssi in wsn,* Wireless Sensor Network **2**.

[9] O. Katircioglu, H. Isel, O. Ceylan, F. Taraktas, and H. B. Yagci, *Comparing ray tracing, free space path loss and logarithmic distance path loss models in success of indoor localization with rssi,* Wireless Sensor Network **2**.

[10] *Data Sheet CC2541*, Texas Instruments (2013).

[11] Y.-H. Lee, K.-W. Ho, H.-W. Tseng, C.-Y. Lo, T.-C. Huang, J.-Y. Shih, and T.-H. Kang, *Accurate bluetooth positioning using large number of devices measurements,* IMECS **2** (2014).

[12] O. Oguejiofor, A. Aniedu, H. Ejiofor, and A. Okolibe, *Trilateration based localization algorithm for wireless sensor network,* International Journal of Science and Modern Engineering **1**, 21 (2013).

[13] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications* (Springer, 2005).

[14] T. Duong, *Kernel density estimation and kernel discriminant analysis for multivariate data in r,* Journal of Statistical Software **21**, 1 (2007).

[15] *AE4S06P Part 1 V18* (2014).

[16] *BLE113 Data sheet*, Bluegiga, 1st ed. (2014).

[17] *Data Sheet BLE Module*, Elpasys (2014).

[18] *BGscript Scripting Language*, Bluegiga, 4th ed. (2014).

[19] *Bluegiga Bluetooth Smart Software*, Bluegiga, 3rd ed. (2014).

[20] Bluegiga, *Ble113 documentation,* https://www.bluegiga.com/en-US/products/bluetooth-4.0-modules/ble113-bluetooth--smart-module/documentation/ (2014), [Online; last accessed 07-01-2014].

[21] *Bluetooth Smart Profile Toolkit*, Bluegiga, 3rd ed. (2014).

[22] *Bluetooth Smart Module*, Bluegiga, 3rd ed. (2014).

[23] C. Robotics, *V-rep,* http://www.coppeliarobotics.com/contact.html (2014), version: 3.1.3 [Online; last accessed 07-01-2014].