

# Zebro onboard navigation system (ONS) Thesis, part 2

Version 1.0

June 22, 2015

To:  
Chris Verhoeven

## Abstract:

This document describes the Bachelor Graduation Project of the ONS-group of the Zebro Mars Rover Project. This team developed the telecommunication, a sensor set, the user interface and an autonomous navigation algorithm.

Bart Rijnders, 4103505  
Bart van den Bogert, 4215192

## 1 Foreword

For the Bachelor Graduation Project we are fortunate enough to work on the Zebro Mars Rover project. This challenged us to engineer, to become part of the team that will attend in the European Rover Challenge (ERC) this September. We would like to thank Chris Verhoeven and Maneesh Kumar Verma for the opportunity they offered us and assistance in designing our systems. We are ready to work on the Zebro, taking it to the next level and win the European Rover Challenge 2015.

Zebro!

*Bart Rijnders and Bart van den Bogert*



Figure 1: Design draft of Zebro

## Contents

<b>1</b>	<b>Foreword</b>	<b>2</b>
<b>2</b>	<b>Introduction</b>	<b>4</b>
<b>3</b>	<b>Specifications</b>	<b>5</b>
<b>4</b>	<b>Design choices</b>	<b>8</b>
4.1	Microcontroller . . . . .	8
4.1.1	Comparison . . . . .	8
4.1.2	Choice . . . . .	9
4.2	Global Positioning System (GPS) . . . . .	10
4.2.1	Distance measurement . . . . .	10
4.2.2	Calculating the position . . . . .	10
4.2.3	Comparison . . . . .	12
4.2.4	Choice . . . . .	13
4.3	Object detection . . . . .	14
4.3.1	LIDAR . . . . .	14
4.3.2	RADAR . . . . .	14
4.3.3	SONAR . . . . .	14
4.3.4	Working principles . . . . .	14
4.3.5	Comparison . . . . .	15
4.3.6	Choice . . . . .	16
4.4	Onboard Navigation Algorithm . . . . .	18
4.4.1	Implementation . . . . .	19
4.4.2	Comparison . . . . .	20
4.4.3	Choice . . . . .	20
<b>5</b>	<b>Implementation</b>	<b>21</b>
5.1	Microcontroller . . . . .	21
5.1.1	Implementation . . . . .	21
5.1.2	Results . . . . .	21
5.1.3	Discussion . . . . .	21
5.2	GPS . . . . .	22
5.2.1	Implementation . . . . .	22
5.2.2	Power Saving Mode . . . . .	22
5.2.3	Test Plan . . . . .	22
5.3	LIDAR . . . . .	23
5.3.1	Results . . . . .	24
5.3.2	Discussion . . . . .	24
5.4	Navigation algorithm . . . . .	25
5.4.1	Destination search . . . . .	25
5.4.2	Obstacle Detection . . . . .	25
5.4.3	State discription . . . . .	26
5.4.4	Signal discription . . . . .	26
5.4.5	results . . . . .	27
5.4.6	discussion . . . . .	28
<b>6</b>	<b>Conclusion</b>	<b>29</b>
<b>7</b>	<b>Ending</b>	<b>30</b>
<b>8</b>	<b>Bibliography</b>	<b>31</b>
<b>9</b>	<b>Appendix</b>	<b>32</b>
9.1	Appendix A . . . . .	35

## 2 Introduction

The Zebro Explorer (Zebro) is a Mars Rover from the Delft Robotics Institute of Delft University of Technology. The name Zebro comes from 'zesbenige robot', which is Dutch for 'six-legged robot', an idea of Chris Verhoeven. It is designed as a swarm robot that can assist in different applications. It can help in space exploration, health care for the elderly, rescue assistance and many other applications. The ultimate goal is to have a swarm of Zebro's, with the capability to repair and transport other Zebro's. Due to its unique leg design it has proven itself to be nearly unstoppable on any terrain.

The project started several years ago and the first big challenge is planned for this September: The European Rover Challenge (ERC). It is a rover challenge in Poland organized by the European Space Agency (ESA). To win this challenge a lot of new features are added to the Zebro design, such as long range communication, grabbing objects and autonomous navigation. For this challenge the Zebro organization was looking for new members and so we were offered a place in the Zebro team as part of our Bachelor Graduation Project. Since it is impossible for the whole project to be done by just us, we will only work on a selected set of modules and subsystems.

We chose to design the onboard navigation system (ONS). This system consists of multiple other subsystems such as the navigation algorithm, the sensors to image the environment, the communication system and the user interface. These subsystems are one of the most vital parts of the Zebro. The navigation uses the data received from the sensors to control the robot to its destination. The user interface provides us a way to control and monitor the Zebro from a distance. Since communication is very important, clear rules need to be discussed with the other teams on the communication between the subsystems.

At the end, Zebro needs to be assembled. This is the part where everything will come together. Because the project only lasts 8 weeks and not all the subsystems are ready by now, the assembling will be after we finished the graduation project. The assembling and testing of the whole robot will be done by us in the upcoming months. Experience teaches us that this will always cost more time than expected.

First the design choices will be specified. This will be done by explaining different problems. This will conclude with a comparison of the different possibilities where the choice will be made on what is the best solution. This choice will be based on multiple aspects like power consumption and efficiency, while also taking into account reliability and costs. Then the implementations of the solutions will be explained. An explanation on the results and a comparison of the results with the expectations will be given. Concluding in a discussion on whether good choices were made and how this will impact the future of this module and the rest of the Zebro. At the end, a general conclusion will be given.

The ONS-group has been divided in two subgroups. The division of the different subjects of the thesis can be found in table 1

To conclude: we believe we are going to win the European Rover Challenge this September. With the unstoppable power of Zebro and the drive of the team it will be a great success!

Table 1: Division of the subjects over the subgroups.

Part 1	Part 2
Communication system	Autonomous navigation system
PCB design	Environment imaging
User Interface	Sensor set

### 3 Specifications

Since the Zebro Explorer is competing in the ERC, it will need to fulfill certain tasks. The main task our group is focusing on is the so called 'blind traversal'. Since some essential solutions will be designed for the functionality of the whole Zebro, our responsibility also will be responsible for the functionality of our parts during other tasks. To summarize, the different problems in obstacle detection, positioning and autonomous navigation. Other problems that will be dealt with this period are picking up objects, movement by legs, the communication to the base station and combining this in a user interface, but these are done by other members of the Zebro group.

The main target in our design will be the reverse traversal challenge. In this challenge the robot needs to navigate from a starting point to the next 3 locations, these are given in GPS coordinates by the organization. Between location 1 and 2 there is a 2 meters wide obstacle blocking the shortest path. If the robot reaches the last location within 10 meters the challenge is completed. A visual representation of the challenge can be found in figure 2. A destination is reached when the robot is within 10 meters from the given coordinates.

This will result in 100 points for the competition. Bonus points are acquired by walking from the second location to the third autonomously. Only the position of the robot can be sent to the Zebro explorer, the rest of the computing has to be done on board. When the team succeeds, it will be granted 25 bonus points. When 2 navigation techniques are used to complement each other, again 25 bonus points will be rewarded to the team. The last 10 bonus points will be awarded when the Zebro Explorer can return back to the starting point after finishing the challenge.

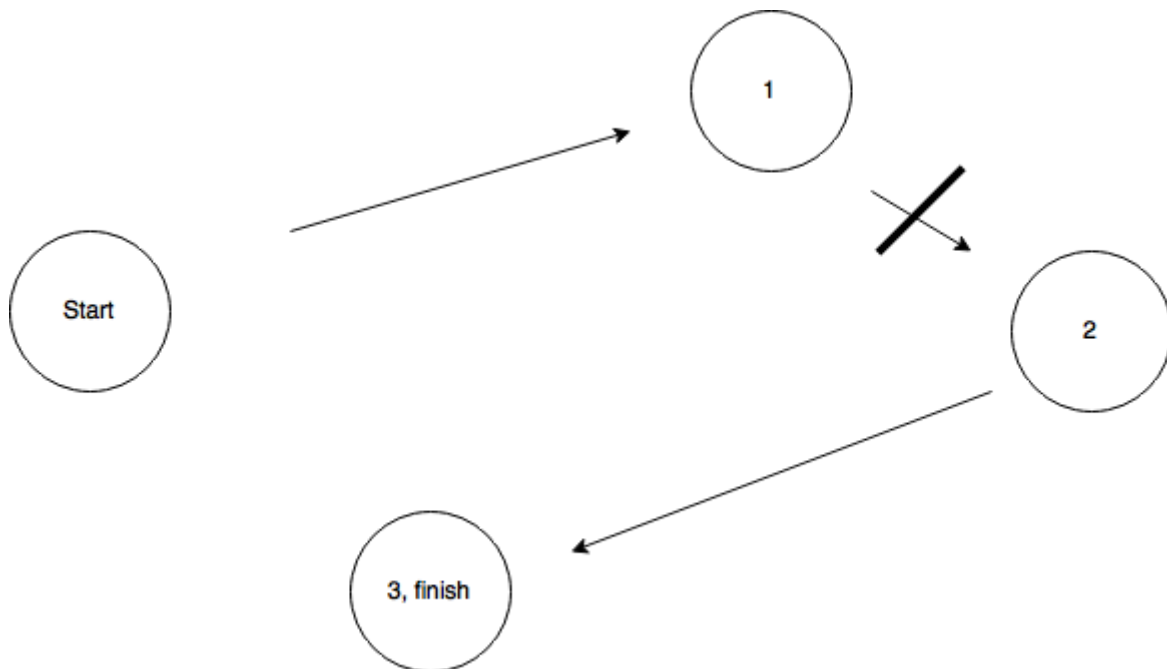


Figure 2: Reverse traversal challenge, starting from start going to respectively 1, 2, 3. A obstacle blocks the direct path from 1 to 2. This picture is not scaled.

The first thing that is absolutely necessary is being able to send and receive data to and from the Zebro explorer respectively. The communication system will be tested by the judges. The robot will be set 500 meters from the base station and then it should still be able to communicate with the base station. The second feature the robot should possess is being able to receive GPS coordinates about its position. For this a GPS module is needed of course, and more specifically one that has a better accuracy than 10 meters. The threshold during this challenge is 10 meters from the chosen target. Further, the robot should be able to be manually controlled by the user, it is preferable to have a video feed to see where the robot is heading and what is around the robot. The video feed should be of an okay quality, not full HD since the signal needs to be transmitted over a range of 500 meters. For this is decided to achieve a higher frames per second than a higher quality of the images. The video feed should be smooth thus the amount of frames per second needs to be as high as possible. When this subsystem will fail to work it will be very hard to control the Zebro manually without vision.

To score bonus points an extra technique for navigation is required. Furthermore, the robot needs to be able to dodge objects with a detection system. With this information the Zebro can determine its position and destination. Together with the GPS module the robot has two navigation techniques which complement each other.

The Zebro also needs to be able to communicate over a distance of 500 meters to the base station. This part will be designed by the other subgroup, but taking into account to not overload the communication and keep the exchange of information to a minimum while implementing these parts. When the communication fails the Zebro will be unable to complete its tasks.

In order to walk autonomously from the second location to the third a program needs to be written on the microcontroller of the Zebro Explorer itself. The last 10 points however are just a matter of time, being quick enough to get back to the starting point. To realize this the Zebro needs to make calculation onboard on a microcontroller. This means limited processing power is available and this will be taken into account for the design of the algorithm. When it is impossible to complete this we can do this part of the challenge with manual controls, thus missing the bonus points.

To let it navigate autonomously from one point to another it needs to know its destination, given by the organization, and the current position, which is not given. From this is determined what direction the Zebro is supposed to be heading. So a solution to locate the Zebro needs to be designed. When the position is unknown the autonomous navigation will be unable to know the direction. But manual control will still be possible through the camera feed provided from the Zebro.

Next is the problem that the Zebro needs to detect obstacles. During the whole competition the Zebro will be on a rocky surface. To avoid it getting stuck in one of those rocks a way to detect the rocks will be designed, so Zebro can determine how to move around them. If the Zebro is unable to detect the environment it has a high risk to not get to the destination and get stuck on the way to it. Then the challenges are not complete and the team loses points in the challenge.

During the whole competition an optical view of the environment is needed. This will be done by a camera with a frame rate of around 20 frames per second. To have a decent view it will have a resolution of 480p. The use of this imaging is necessary during the challenge to use it with the manual control and to monitor it during the autonomous navigation mode. When this subsystem fails to work it will be still able to complete the tasks of the challenge by the imaging of the environment sensing, but this makes it really hard.

The reverse traversal will be the only moment where the autonomous navigation will be used. During other challenges the manual controls will be used to complete the different tasks. So all the other sub-systems we will design need to be able to also be used during the manual control. The manual control itself will be designed by the other subgroup. When this part of the system is not working the whole Zebro cannot move and the system will fail to complete the challenges.

one of the most important specification is that the design needs to be as modular as possible. This is because when there is a better solution to a problem, it can be easily incorporated in the rest of the design. Plus when something breaks in Poland it can be easily swapped out with spare parts. With our future work in mind we need to guarantee this to make improvements to the Zebro as easy as possible.

For the design we also consider the costs and the power consumption of every part. For the whole Zebro a budget of 15.000 EUR is available. In the first predictions the total costs will be around 6.000 EUR, so the budget will not be a limiting factor, if no extraordinary systems are used. For the power consumption the specifications are not available yet, so therefore we will be choosing to minimize the power consumption of our modules. When these specifications are not met, problems completing the challenges could occur by running out of battery power.

To summarize the specifications:

1. A communication system that is able to send and receive over 1000 meter;
2. A way to let the Zebro navigate autonomously from one point to another with onboard processing;
3. A way to determine the location of the Zebro;
4. An obstacle detection system;
5. A camera with sufficient quality and enough fps;
6. Manual controls;
7. Autonomous controls;
8. The system should be as modular as possible;
9. The maximum budget is 15.000 EUR;
10. A maximum power consumption is required within the specifications;
11. The maximum weight of the Zebro is 50 kg.

These specification will be the guideline in the design of the systems. Some of the requirements are not realized by us, but by the other sub teams. This design can be found in their theses. With the specification, the different design choices can be made. This will be done by checking alternatives for the different solutions and then deciding what fits best for the Zebro, considering both time and quality of the system.

## 4 Design choices

Now the specification of the system are known, choosing the design will be done next. Here, decisions on the design are taken with respect to the specifications, so we will have a system that satisfies them. Here decisions on the microcontroller, GPS, obstacle detection and navigation algorithm are made.

### 4.1 Microcontroller

On our PCB a lot of different peripherals are needed, plus it needs to be able to communicate with the Zebro processing unit (ZPU). In order to do this the microcontroller needs to have an I2C and a UART interface. The UART is necessary for the XBee module. The other peripherals and the communication with the ZPU is done via I2C.

The next important aspect of the microcontroller is the power usage, which needs to be as low as possible. The Zebro explorer has a battery system, so it has a limited amount of energy for the operations. So power cannot be wasted and an efficient system is needed. Plus it needs to be able to operate in a wide range of temperature conditions.

The main task of the microcontroller is to gather data and commands and sending them to the right place, there is not much processing power needed to achieve this task. Only some processing power is needed for the autonomous navigation. In figure 3 you can see the data flow the ZPU has to deal with. And probably a very important thing to consider is that the whole Zebro team is using the same type of microcontroller, this makes communicating with other parts easier.

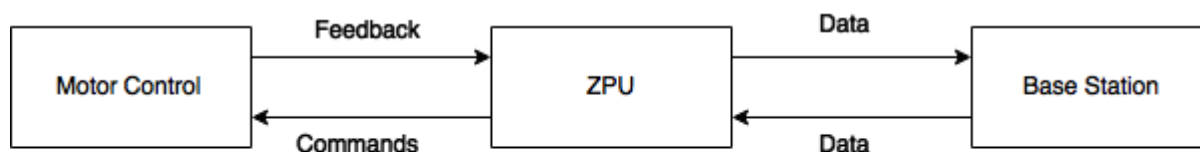


Figure 3: BlackBox model of the data flow for the Zebro Processing Unit (ZPU), format of the data lines is not specified.

#### 4.1.1 Comparison

The choice of the microcontroller can be based on various specifications. Since the industry offers a lot of different types of microcontrollers, with all different specification, you can almost always find a microcontroller that meets your desires. For this we can use a development board to see whether the controller works properly or not. This board can be seen in figure 4.



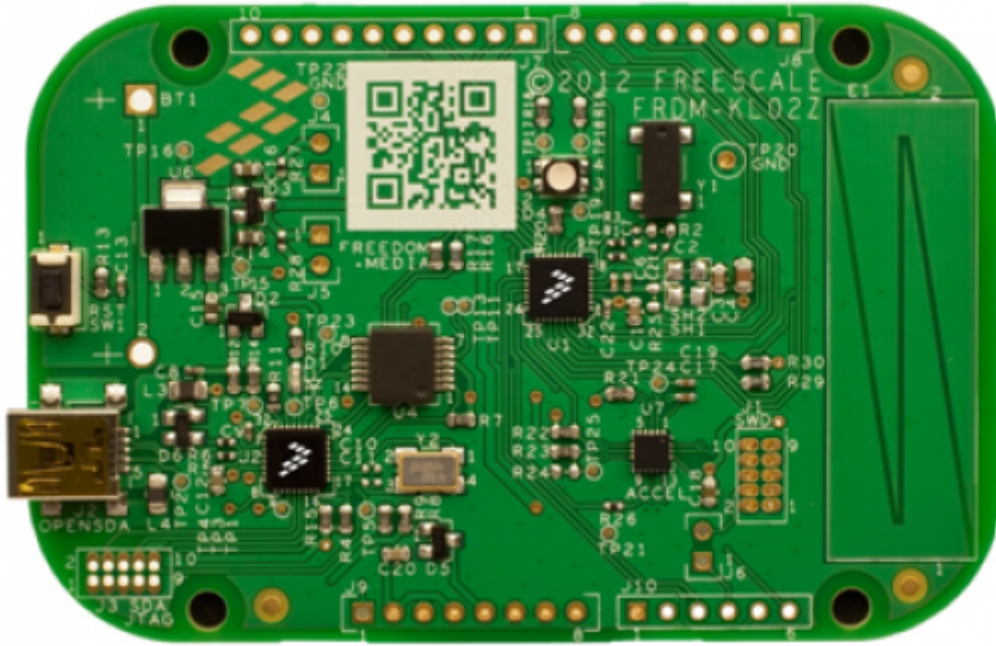


Figure 4: A development board, containing the Freescale Cortex M0+ processor, used for testing micro-processors.

#### 4.1.2 Choice

Because of the above mentioned reasons the complete Zebro team chose for a Freescale Cortex M0+. It has one I2C interface as well as one UART and SPI interface. The power it uses is 7.1 mW (as seen in equation1).

$$45 \frac{\mu A}{MHz} \cdot 48 MHz \cdot 3.3V = 7128 \mu W = 7.1 mW \quad (1)$$

The operating temperatures are from  $-55$  to  $150^{\circ}C$ . The Netherlands and Poland are both not known for their extreme temperatures so the temperatures fall in this range. It has a 4 KB SRAM and 32 KB program flash memory.

## 4.2 Global Positioning System (GPS)

[3]

For the ERC we will be using a GPS module, this because this works on earth and the organization of the ERC supports it. Of course when the robot would be operating on Mars this would not be an option since there is no GPS there. And when Galileo is in operation one of their modules could be implemented in order to get better accuracy compared to the GPS. Galileo is the European version of the American GPS But for now GPS is the best option since the challenge will also supply the location with GPS coordinates.

### 4.2.1 Distance measurement

The GPS receiver sends a signal to a satellite. When it reaches the satellite a signal is sent back to the GPS receiver which receives the exact same signal. The time difference is measured and from this time the distance can be computed following equation 2.

$$d = \frac{c * t}{2} \quad (2)$$

where  $d$  is the distance in meters,  $c$  is the speed of light ( $3 * 10^8 m/s$ ) and  $t$  is the total traveling time in seconds. The division by 2 comes from the fact that the time the receiver measures is the time it took to go to the satellite and back. This means the time measured is twice the time it took for the signal to travel to the satellite. A visual of this process can be seen in figure 5.

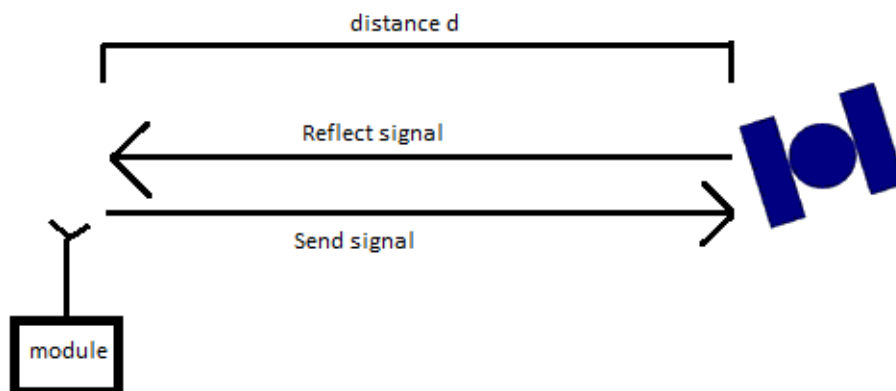


Figure 5: A visual on the propagation of the signals of a GPS module to a satellite, here we can see the signal travels twice the distance between the two points.

### 4.2.2 Calculating the position

To explain how the position is computed an example is used. When the distance from the satellite is known it results in a geometry problem using spheres. When the distance is known from one satellite the receiver knows it is somewhere located on a sphere with the satellite at its center and the radius equals the distance computed earlier. This is shown in figure 6

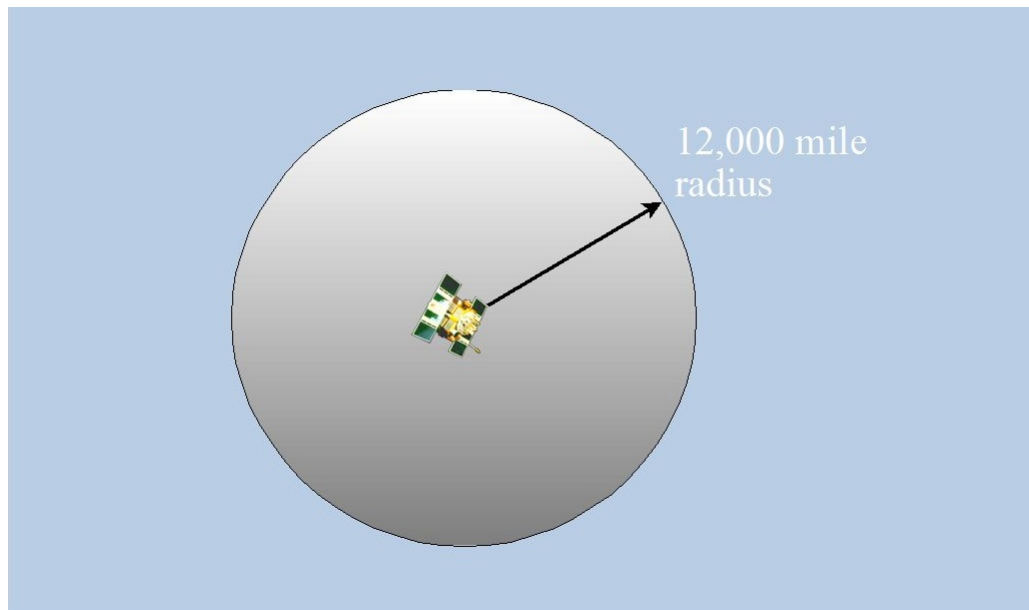


Figure 6: Possible position using 1 satellite. The possible positions of the receiver now are on the surface of the sphere.[2]

When a second satellite is added to this configuration the receiver knows that it is somewhere on the intersection points of those two spheres. These intersection points are located on a circle this is indicated with the black circle shown in figure 7.

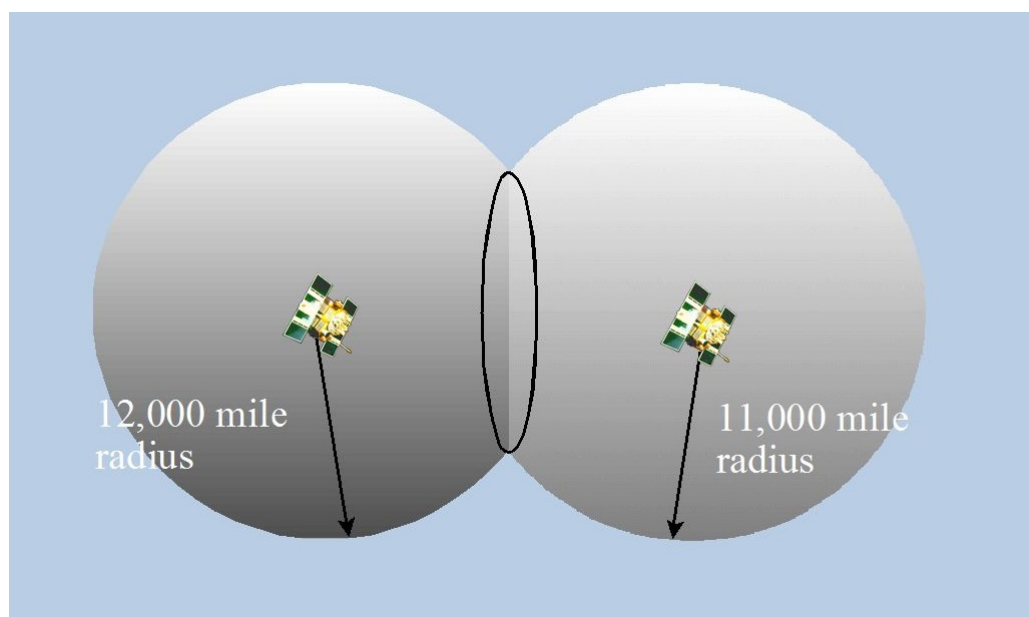


Figure 7: Possible position using 2 satellites. All possible positions are now in the circle made from the intersect of the two spheres.[2]

The next step is to add another satellite. This will result in the receiver knowing it is somewhere on the intersection points of the three spheres around satellite 1,2,3. As shown in figure 8 only two intersection points remain ( indicated with the red dots).

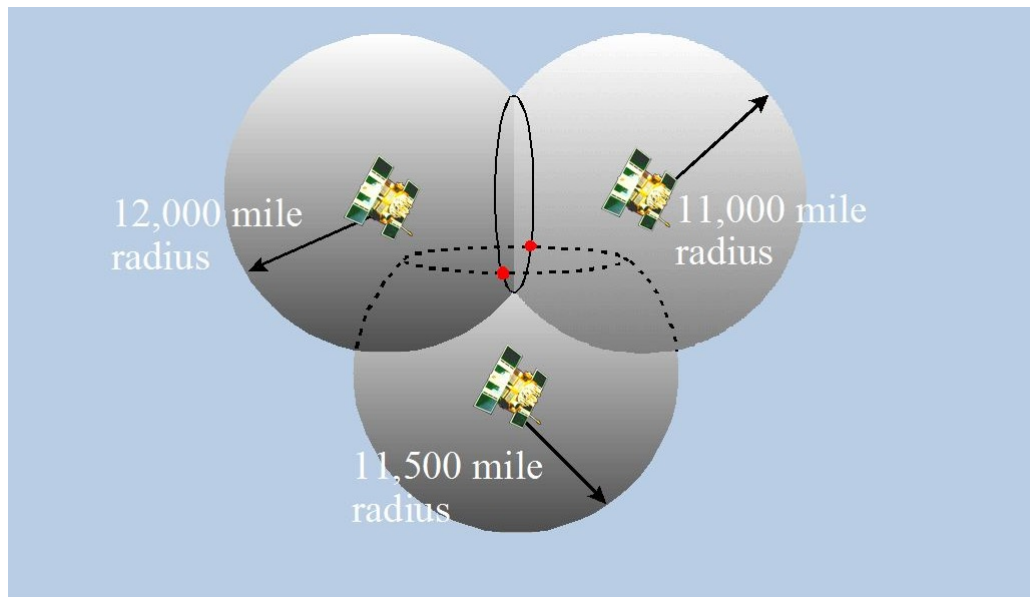


Figure 8: Possible position using 3 satellites. The possible locations are now indicated with the red dots, intersections of the three spheres.[2]

Most of the time from these two points the receiver can make a choice on its own which point it is located at since the other intersection point has a high chance of being nowhere near earth. If this is not the case the receiver will measure its distance from a fourth satellite and only one intersection point should remain, being the location of the receiver.

#### 4.2.3 Comparison

It would be a really straight forward choice to choose for a GPS system, since it is widely used for all kind of purposes, it has the preference above other systems. But nevertheless, we could also choose for an alternative. For example the Russian GLONASS system. This works by the same principles as the GPS, but on slightly different frequencies. Since GPS is the oldest, it has the most support from developers. But the biggest advantage of this kind of systems is that you are able to know your location accurate every where.

It is not allowed to add anything to the area, this means it will not be allowed to add something like a set of microphones or other detectors to track the Zebro. Therefore, the decision of using GPS over other systems is easy to make.

#### 4.2.4 Choice

Because the Zebro explorer needs to be within 10 meters of its target location the NEO-7N GPS module is chosen, seen in figure 9.



Figure 9: The chosen GPS model, NEO-7N GPS module.

This module has an accuracy of 3 meters, which satisfies our specification of 10 meters. It can update its position every 0.1 seconds which is fast enough since the Zebro explorer is not fast, the top speed which is aimed for is 1.1m/s. And quite important is the fact that it supports an external antenna since the Zebro explorer will have a case around it that could block the signal. Further it supports an SPI/UART/I2C interface which makes it easy it integrate into the rest of the system.

### 4.3 Object detection

Because the Zebro has to navigate autonomously for certain tasks, objects cannot always be avoided using the camera system and manual controls. For this task objects need to be detected using different methods. Suitable available technologies are RADAR, LIDAR and ultrasonic sensors. They can detect objects and pass it through to the navigation algorithm so that the Zebro can avoid collision with the detected objects.

#### 4.3.1 LIDAR

[8] LIDAR is an acronym for Light Detection and Ranging. It is a technique that uses a laser beam to detect or scan objects. It is an active form of remote sensing. So information is obtained by sending a signal from a transmitter which can be reflected by a target. A detector can pick up the reflected signal to determine different types of information about the object. These types are range, chemical properties and velocity. The only thing that is relevant now is the range of the object, since the goal is to avoid collisions with objects. The measurement of the range or distance to the object is based on precise measurement of time.

The advantage of LIDAR is that it uses light instead of sound, like in SONAR sensors. Most important is that it has a spread that is a lot smaller than for example a RADAR or SONAR. Due to the little spread and the use of light, this sensor is really favorable.[6]

#### 4.3.2 RADAR

[1] Radar stands for radio detection and ranging. It is widely used today in military, telecommunication, medical and many other applications. It utilizes electromagnetic waves to sense the environment. In this case it could be used to detect objects and thus avoid collisions.

#### 4.3.3 SONAR

It is also possible to use a SONAR sensor. This uses sound waves as the wave to determine information about the environment. Due to the limited speed of these waves, the speed of sound, it is more affected by the Doppler effect.

#### 4.3.4 Working principles

To detect objects, electromagnetic waves have to be emitted first. This is done by radiating electromagnetic energy from an antenna to propagate in space. When these electromagnetic waves collide with an object, the waves will be scattered in many directions. But some of these reflected waves will return to the radar antenna and can be detected. When these reflected waves are picked up by an antenna, the signal will be processed and the location plus possible other information about the object can be obtained. Because the emitted electromagnetic waves are usually sent in all directions, it is possible to detect objects in a certain radius of the radar. So it is not needed to aim the antenna.

To determine the range of an object, the time it takes for a radar signal to propagate to the object and back has to be measured. Because electromagnetic waves travel at the speed of light the distance to the object can be calculated. An advantage of radar is that it can measure this distance with great accuracy, even at long range. Because Zebro needs to detect objects at a relatively small range, high accuracy becomes even more important.

### 4.3.5 Comparison

All three types of distance sensors utilize the same method to calculate the range of detected objects. A signal is sent and reflected by the object, then the time is measured it took for the signal to propagate to the object and back. In the case of the ultrasonic sensor only the time is measured and the distance has to be calculated by the CPU.

Looking at the table below, it can be seen that the LIDAR Lite is best in every aspect except for the price and power consumption. In range and accuracy it is the absolute best, while the power consumption is not that much bigger compared to the ultrasonic sensor. The power consumption is also not a very important factor since sensor will only be active for very short amounts of time when the area needs to be scanned. The I2C connectivity is also very welcome, because the range is output directly compared to the PWM output which will need conversion. So the somewhat higher price of the LIDAR Lite is well worth it. The RADAR sensor has good range but very poor accuracy, it uses significantly more power than the other 2 sensors and it is very expensive. So LIDAR is the favorable option. A picture of the sonar and the radar sensor ccan be found in figure 10 and 11 respectively.

Sensor	Type	Power Consumption	Connectivity	Cost	Range	Accuracy
LIDAR Lite	LIDAR	5V DC and 100mA	I2C, PWM	94.90 EUR	40 m	1 cm
Ping Ultrasonic	Ultrasonic	5V DC 35mA	PWM	30 EUR	3 m	2 cm
QT50R-EU-AFH	RADAR	24V DC and 100mA	3 Bit Digital	660 EUR	24 m	1-2 m

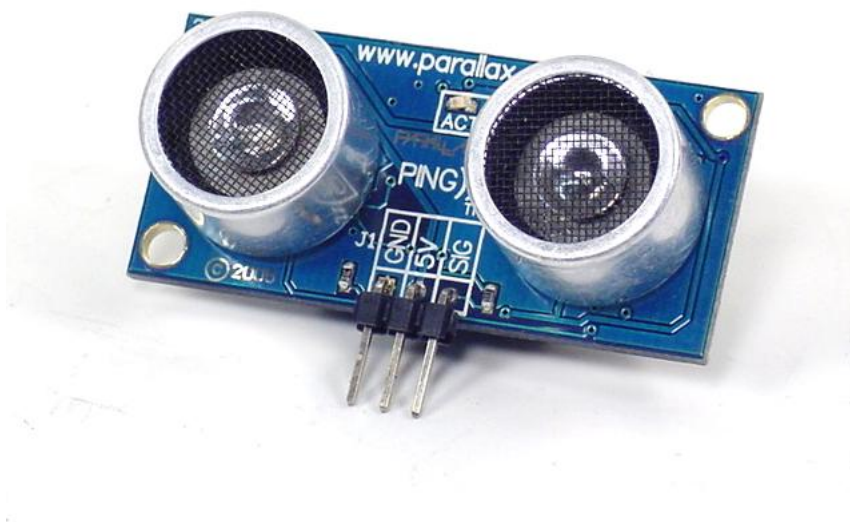


Figure 10: The Ping Ultrasonic sensor



Figure 11: The QT50R-EU-AFH

#### 4.3.6 Choice

The chosen sensor is a LIDAR system. This is because the LIDAR Lite has high accuracy, long range, low power usage and is very small. The performance parameters of the LIDAR Lite can be found in table 4.3.6. The RADAR systems available are much more expensive and have much lower range. The ultrasonic sensors have shown to be very unreliable in practice. Since these sensors only can measure the distance in one direction, imaging the environment is impossible without any adaptations. Now a way has to be designed to image the environment. This can be easily done by placing the LIDAR Lite on a rotating platform, for example a regular servo motor. By letting the servo turn and imaging the environment with the distance values of the LIDAR. One problem is the wire connected to the LIDAR. This will get stuck if the platform rotates, so a servo control is now needed to prevent the sensor from getting stuck.

To conclude the decision is to place the LIDAR lite on a servo. With this setup it is possible to make the imaging of the environment, without using the camera.



Table 2: These are the parameters for the LIDAR Lite sensor.

Parameter	Value
Range	0 m - 40 m
Accuracy	0.01 m
Update frequency	100 Hz
Power consumption	5 V DC / 100 mA = 0.5 W



Figure 12: The LIDAR Lite Sensor

#### 4.4 Onboard Navigation Algorithm

As a part of the European Rover Challenge, Zebro needs to autonomously navigate from an unknown point to a fixed destination. This is the 'blind traversal' challenge. In this challenge the robot has to navigate by a partly planned route, without using the input from the rover cameras. This means a system needs to be designed to navigate through the field and to avoid collisions with obstacles such as rocks, all autonomously. Since, the lack of knowledge about the area, sensors need to be included to provide information and determine what moves should be made.

From this is derived and formulates the specifications for our navigation system:

- The Zebro is not allowed to receive instructions from the base station;
- The Zebro needs to be able to detect obstacles;
- The Zebro need to be able to avoid obstacles;
- The Zebro needs to determine its position;
- The Zebro must be able to adapt its movement to avoid obstacles.

Furthermore, it is not allowed to navigate based on the input from the cameras on the Zebro. The only input that can be sent from the base station are the destination coordinates. Point subtraction will be given to rovers that go wide (10 meters) from the straight line from start to finish. Also the current position of the Zebro need to be send to the base station, so the team and the judges are able to monitor it on a virtual map. The autonomous navigating will award the team with bonus points during the challenge.

From these specifications is derived a black box model for the onboard navigation system, this can be found in figure 13. Here you can see that the algorithm is the central part of the autonomous movement. The algorithm get the destination from the base station and the data from the sensors to determine what motor commands are needed to reach its destination.

In the black box model, the four boxes of the abstraction are shown. Base Station, this will only receive output from the zebro, and in case of emergency, it is able to send an emergency stop command to the Zebro. The sensor set will receive requests from the ZPU, the ZPU will call certain sensors to give information, the output is send back to the ZPU. The motor control will receive the needed commands from the ZPU and will execute them for the motors. A feedback signal will be sent back to verify the command is executed successfully.

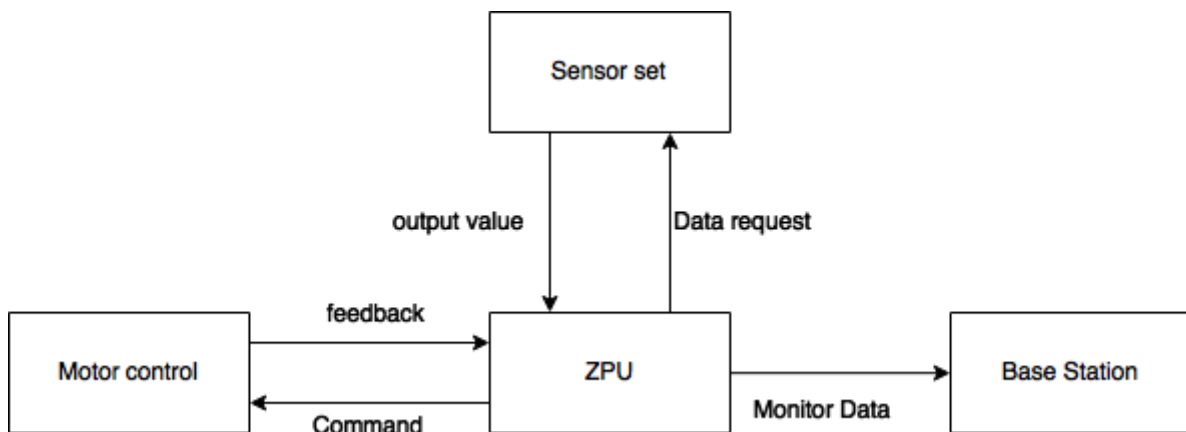


Figure 13: Black Box model for the onboard navigation algorithm, types of data are not specified, the emergency signals from base station to ZPU is not shown.

#### 4.4.1 Implementation

First of all, all the processing of the algorithm needs to be done on the Zebro, since the only communication that is allowed is setting the destination.

For obstacle detection a LIDAR sensor is used, by scanning the area it will be able to determine how far all obstacles are away and from this information it can be determined what move is needed to avoid the obstacle.

For locating a GPS module is used, this will return us the GPS coordinates of the Zebro. We do realize that we cannot use GPS on Mars, but we can use it in Poland, so for now it is good enough.

For our algorithm we choose to first design a finite state machine and then implement it in C. The FSM can be seen in figure 14.

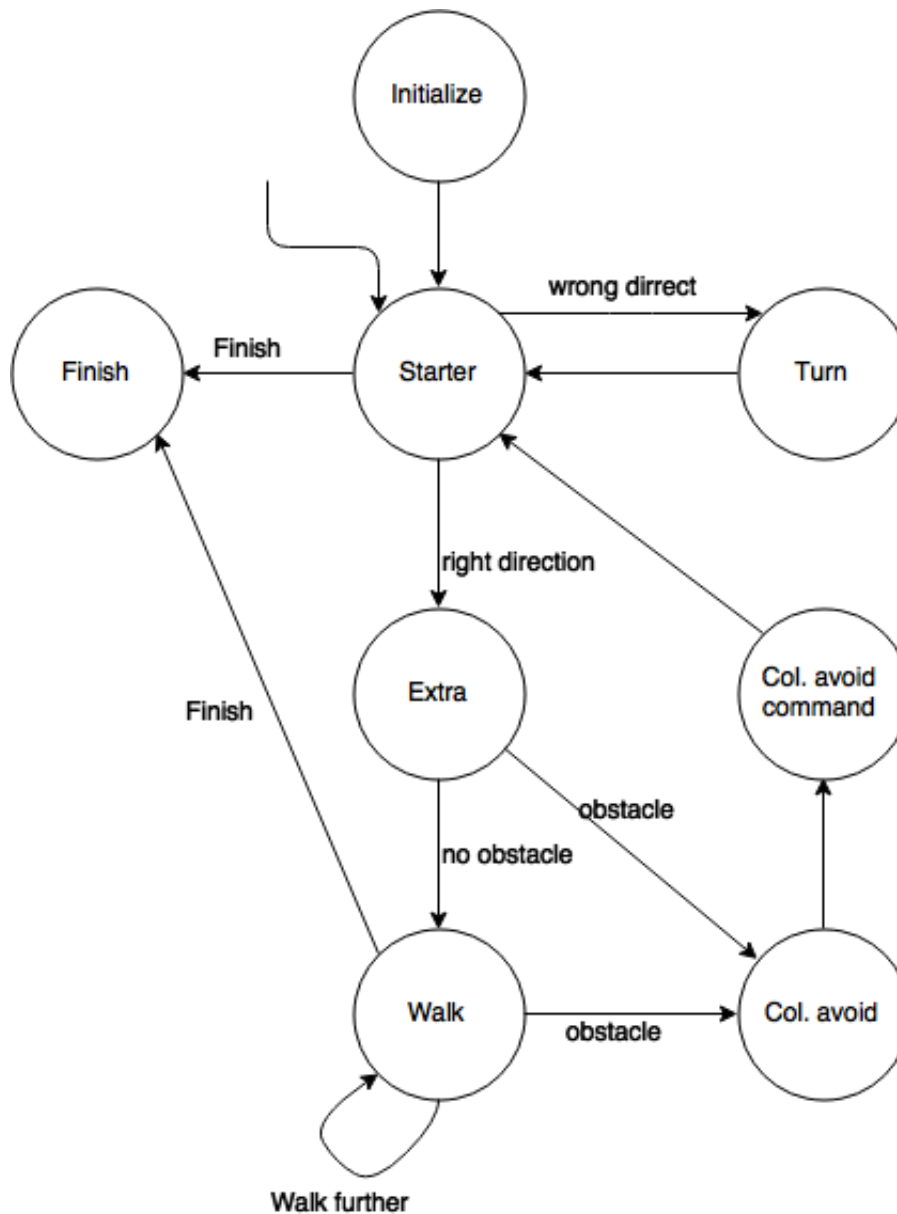


Figure 14: Finite State Machine for the Onboard Navigation System, the arrows between the states stand for change conditions. The start state is the 'initialize' state, reset state is the 'starter' state, as seen by the lightning arrow.

#### 4.4.2 Comparison

Here the use of a FSM-based algorithm is chosen to realize autonomous navigation. One of the most limiting factors in this part is the onboard processing of the algorithm. Since all the processing needs to be done onboard, an algorithm which does not use much processing power is used.

One of the weaknesses is that it has a limited range of detecting. This is since the Zebro is only interested in the front side of the Zebro. It only detects the area that is direct in front of the Zebro, so it will never detect an easier path if it is at the side or behind the Zebro. This can be a problem if it navigates around, but for the challenge it is good enough. It is possible to let it accurately make a image from rotating the LIDAR and visualize it, but this will cost too much processing power. Since the camera is used here, it is decided to use this algorithm.

An alternative way to determine the path of the Zebro can be used. For example combining GPS and accelerometer to create a new strategy. First it determines the position by GPS and then walks to the destination. By integrating the data from the accelerometer twice the distance the Zebro has traveled is calculated. Since double integration lets the value explode for long periods of time, a new position is needed after a certain time. When the new position is received from the GPS module we will compare it with the position where we want to be. Then recalculate is done for the new path and new commands are determined for the Zebro. This is one of the other solutions.

#### 4.4.3 Choice

The terrain looks mostly flat with a few rocks. This also mean that there will be no big cliffs or other big obstacles that could enclose the Zebro. Additionally, the GPS coordinates do not need to be asked continuously, updating the locating of the Zebro around each 10 seconds will also do the job. For this, concluded can be that an other algorithm that does this could also work, but is much too complex for this challenge in this terrain.

The main argument to not choose the accelerometer method is the processing power. When this the microprocessor of the Zebro is used to do a lot of calculations to determine its position. It has the advantage that it can wait longer with asking the GPS coordinates of the Zebro. Then the decision was made to use a stable low (processing) power for its positioning, so the use of the LIDAR/GPS tactic is chosen.

Furthermore the use of this system makes it more modular to replacing the LIDAR sensor with another sensor.

## 5 Implementation

The most important of the ONS is its ability to pass through communication with all the other subsystems next to being able to navigate, since it is the wireless communication hub. This communication must be well defined. An overview of the total Zebro system with the modules being used now can be seen in figure 15. The ONS is connected to the ZPU with an I2C bus. This way it is possible to use the system not only on the Zebro Explorer, but also on other Zebro robots, and even completely different systems.

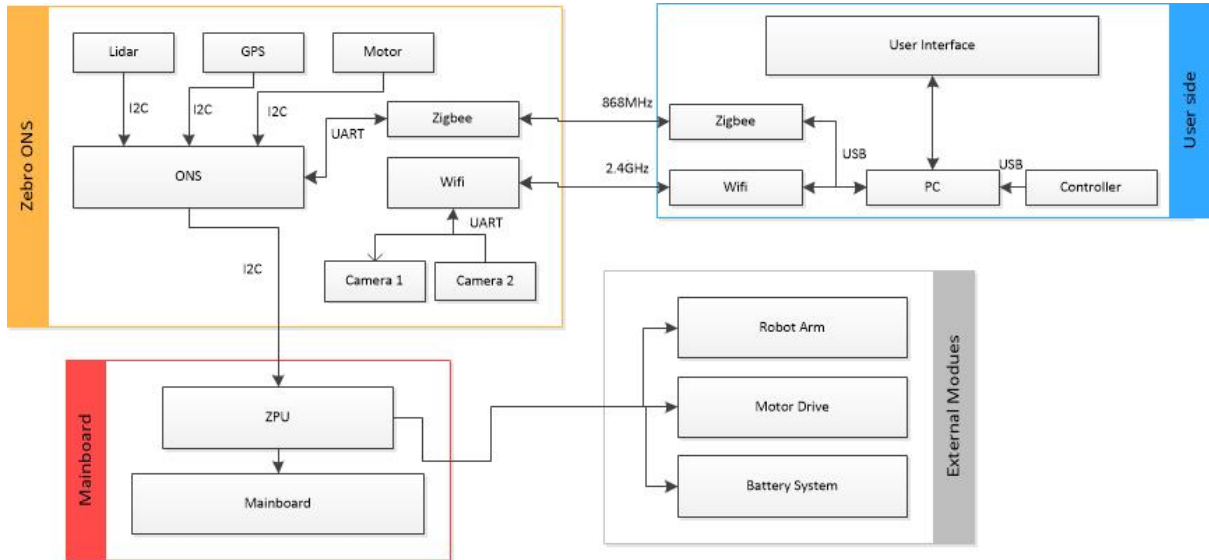


Figure 15: System overview

### 5.1 Microcontroller

During the operations of the Zebro they it will receive information and then send commands to different modules. All commands have bits that identify the destination of the command so the microcontroller knows where it has to deliver the command to.

#### 5.1.1 Implementation

To make a state machine a switch statement is used which switches state dependent on the command it received. The destination bits consist of bits 9 to 12 of a packet. So the destination is coded in binary and has to be converted to decimal first. This is because the input data is stored in ASCII characters and it is simpler to check an integer which identifies the command than to make if-statements for several characters in succession. The conversion to decimal is done with a simple function. When the destination bits are converted, the state machine switches to the right state and the command is output to the right subsystem via the right interface.

#### 5.1.2 Results

The microcontroller will be tested with a development board to see if everything is working as it should be. When it is completed, the microcontroller will be placed on the pcb and tested again. This way, if any errors occur when the microcontroller is on the pcb, it is known the pcb is the source of the error and not the microcontroller.

#### 5.1.3 Discussion

When the microcontroller is working correctly it will be able to transfer all the incoming data to the right subsystems and pass through outgoing sensor data to the user interface.

## 5.2 GPS

The chosen GPS module is the u-blox 7 GPS module. The specifications are shown below.

- GPS positioning system
- 3m accuracy
- 3V DC 3mA power consumption
- I2C protocol

To explain, the u-blox 7 GPS module is due to its low power consumption and high accuracy a good choice for the application in the Zebro. Also the I2C protocol is nice to work with.

### 5.2.1 Implementation

The module will be connected with an I2C bus, it can only operate in slave mode. The microcontroller will always act as master on the I2C bus. The message stream of the GPS module, where the location data is located, can be read from address 0xFF. Subsequent reads at this address will return the available data located in the transmit buffer, one byte at a time. This data will be parsed on the microcontroller and used in the navigation algorithm.

### 5.2.2 Power Saving Mode

Two power modes are available on the module, power saving and continuous mode. Since the Zebro only requests it's coordinates about every 10 seconds, power saving mode is the best choice to reduce power consumption. There are 2 power saving modes available, cyclic and on/off mode. The cyclic mode is best used if coordinates are requested somewhere between 1 and 10 seconds and on/off mode if the coordinates need to be requested with a frequency of every minute to hours. So the best choice is the cyclic power saving mode.

The power mode can be configured by editing the extended power management configuration register. In this register the position update interval and the satellite acquisition interval can be configured to lower frequencies to save power.

### 5.2.3 Test Plan

To test if the GPS module meets our desired specifications in practice, different aspects have to be tested. Including accuracy and signal strength. To test the accuracy of the module, it's output coordinates can be compared to the actual known coordinates from google maps or a smartphone.

When testing the signal strength, attenuation blocks will be used to lower the signal strength to the point where it is no longer able to properly function. These blocks can be placed between the antenna and the module. This gives the maximum signal loss the Zebro chassis can cause, to make sure it will always operate properly, considering both accuracy and reliability.

### 5.3 LIDAR

The LIDAR is mounted on a servo to scan it's environment and for communication with other systems the I2C protocol is used. The servo is a simple standard servo. It requires a PWM signal to determine the angle it has to rotate to. This PWM signal is generated by the microncontroller.

The LIDAR Lite has 2 connection options: an I2C bus and a PWM output. The PWM output gives a continuous output signal with the pulse width proportional to the distance. To read the distance over the I2C bus, a register can be read which contains 2 bytes: the upper and the lower 8 bits of the distance in centimeters. The LIDAR Lite can be configured via the I2C bus so it has to be connected at all times. To save pins and because a continuous stream of distance readings is not needed, only the I2C interface is used to read the distance measurements.

To start acquiring distances 0x04 is written to register 0x00, then 2 bytes can be read from register 0x8f which contain the upper and lower values of the distance in cm. This is done with a frequency of approximately 50Hz to complete the scan as fast as possible. The LIDAR Lite can reach a frequency up to 100Hz, but the servo is the bottleneck in this case since it cannot turn fast enough to use the full 100Hz measuring frequency.

A total scan consists of 90 measurements over 90 degrees which are stored in an array. When the scan is initiated the servo is turned to 0 degrees and a distance measurement is executed. When the measurement is complete the servo turns 1 degree further and the next measurement is executed. This continues until 90 degrees is reached. This vector is then forwarded to the navigation algorithm which uses it to maneuver around objects. A schematic visual of the Zebro with the LIDAR detecting an obstacle can be seen in figure 16.

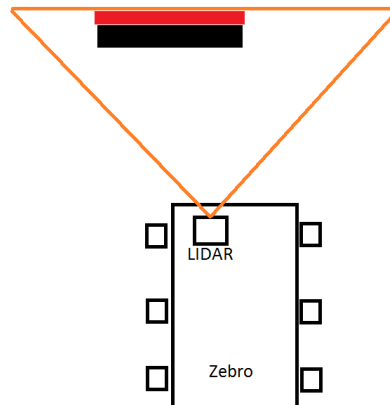


Figure 16: Zebro detecting a obstacle using the LIDAR Lite sensor, within the orange triangle is detectable for the system, the red area shows the area that is unknown, since no detecting can be done there.

### 5.3.1 Results

To test the LIDAR Lite an arduino is used to configure the registers and output the distance readings over a serial connection. This is a simple development board which configures the LIDAR Lite with I2C and outputs a continuous stream of distance measurements.

#### Measurements

With the arduino the accuracy of the LIDAR Lite was tested. By measuring a distance to the wall and then using the LIDAR to measure the distance and check the difference. Here it is found that the sensor is highly accurate, varying around 5 percent from the real distance. Also the range is high, it can even measure accurately above 20 meters. The system is only interested in distances of around 4 meters, since then the Zebro can easily curve around the obstacle.

Since this measurement value will be used in the navigation algorithm, a fail in this sensor will make it impossible for the anonymous navigation to detect the obstacles. Without the obstacles detected the Zebro will run into them and get stuck. So it is highly important the LIDAR can work perfectly.

#### Boxing

Since the challenge will be on a desert-like environment one of the possible problems is dust. When dust gets into the Zebro, especially the sensors, it could start to malfunction. To prevent the dust from getting in, encasing it is one of the main concerns in the Zebro design. This will be also applied to the LIDAR sensor. The LIDAR will be placed in a plastic box so the sensor is not affected directly by the dust. The only way it can now affect the sensor is when it gets on the box, but this will not be a big issue. Different casings were tested like plastic and glass and blurred them with various things. From the result of this, where all the measurement were the same as in the experiment without the casings. It can now be concluded putting the LIDAR in a casing on top of the Zebro will not cause problems. The actual box is not designed yet, but will be in the upcoming months.

### 5.3.2 Discussion

The LIDAR is now able to detect obstacles in a varying range and is unaffected by dust from the environment. This will make the LIDAR a complete successful module in the Zebro project, with modular capabilities.



## 5.4 Navigation algorithm

Now will be explained what decisions are made and what is considered during the implementation of the navigation algorithm. Here the algorithm is differentiated in two parts: a destination searching part and a obstacle detection part. The system is built as a finite state machine, as described in figure 14.

To distribute the commands over the subsystems a state machine is used. A state machine is a system which has a finite amount of possible states the system can be situated in. The current state depends on the input of the command. So the state system determines which state it has to be in by the destination bits of the command. When the required state is reached, the command can be routed to the correct subsystem in on the Zebro in the current state.

To make a state machine, a switch statement is used which switches state dependent on the command it received. The destination bits consist of bits 9 to 12 of a packet. So the destination is coded in binary and has to be converted to decimal first. This is because the input data is stored in an array of ASCII characters and it is simpler to check an integer which identifies the command than to make if-statements for several characters in succession. The conversion to decimal is done with a simple function. When the destination bits are converted, the state machine switches to the right state and the command is output to the right subsystem via the right interface.

The Zebro will be receiving commands via a wireless interface. Since the commands received can be targeted at any of the subsystems they have to be delivered to the right subsystem. All commands have bits that identify the destination of the command so the microcontroller know where it has to deliver the command to.

### 5.4.1 Destination search

For finding the right way to the destination the following method is used. Since the destination is known and the current position of the Zebro is known, it can be determined what direction it has to go. From the onboard compass it is known how it is positioned, then the difference can be calculated and the duration needed for the Zebro to face its destination. It sends the turning command and now the Zebro will be in the same direction as the destination. All that is need now is to move in a straight line to the destination. When the Zebro is positioned within a certain threshold distance from the destination it conclude it has reached its destination and sends a finish signal to the base station.

### 5.4.2 Obstacle Detection

One of the main parts of autonomous navigating is the part where it detects the obstacles. The input of this part is a distance given by the LIDAR Lite module. This is placed on a servo motor that is controlled to let it turn around to get an image from the area in front of the Zebro. The maximum distance of the LIDAR is around 40 meters, much further than is needed. For this is chosen to use an algorithm that distinguishes two different kinds of obstacles. The ones close and the ones out of reach. First thing it does is asking the LIDAR control to measure around in front of the Zebro, an angle of 90 degrees. This gives us 90 measurements. Then it converts all measurements below a threshold to 1 and above the threshold to 0. This makes a series of binary data. Then it filters possible errors by adapting '101' and '010' to respectively '111' and '000'. Then it will determine what side has more space (left or right) and what kind of command (curve or turn) is needed to get past the obstacle. Now it is possible to detect and move around obstacles, which solves one of the main problems for autonomous navigation.

### 5.4.3 State discription

Now each state is explained and what its purpose is:

- **Initialize:** This is the begin state, here the destination will be loaded and the system is prepared for using the algorithm. A rest command will be sent to the Zebro.
- **Starter:** This is the reset state, in this state the position of the Zebro will be determined with the GPS-module and its direction with the compass. In this state, a rest command will be sent to the motor control. If it finds that Zebro has reached its destination it will continue to the Finish state. If its direction is not corresponding with the destination direction, it will continue to the Turn state. When movement is required to reach its destination it will go to the Extra state.
- **Turn:** This state will calculate how many seconds of rotation are needed to let the Zebro be in the direction of the destination. Then it sends a turn command to the motor control. Then it returns to the Starter state.
- **Extra:** This state is to check whether or not a obstacle is nearby. If it is clear it will continue to the Walk state, otherwise it will go to the Col. avoid state. A rest command is sent to the motor controls now.
- **Walk:** Here, a walk command will be executed. Also the position is checked, when it reached its destination it goes to the Finish state. When a obstacle is detected it will go to the Col avoid state. Otherwise it will continue walking
- **Col. avoid:** A rest command will be sent to the motor controls, guaranteeing it will stand still now. Then it continues to Col. avoid command state.
- **Col. avoid command:** This state determines what command is needed to avoid the obstacle. It uses the information of the LIDAR to see how far it is from the obstacle. After that it sees whether it can curve around it or it has to turn around it first. Curve can be used when enough space is available, otherwise it has to turn first before it can curve further around it. When the command is done the system goes back to the Starter state.
- **Finish:** Finally when it reached its destination, the mission ends successfully. A rest command is sent.

### 5.4.4 Signal discription

In table 3 the in- and output signals in our system are described. This can also be found in the black box models given in figure 3, 13 and ??

Table 3: The signal description of the onboard autonomous navigation algorithm.

Signal	I/O	Description
Destination	Input	Destination for the Zebro
Direction	Output	Direction from the current position to the destination
Position	Output	Direction of the Zebro, received from a onboard compass
Collision	Output	Data from the LIDAR distance sensor, to detect obstacles
Command	Output	The command that goes to the motor control, ready to be executed

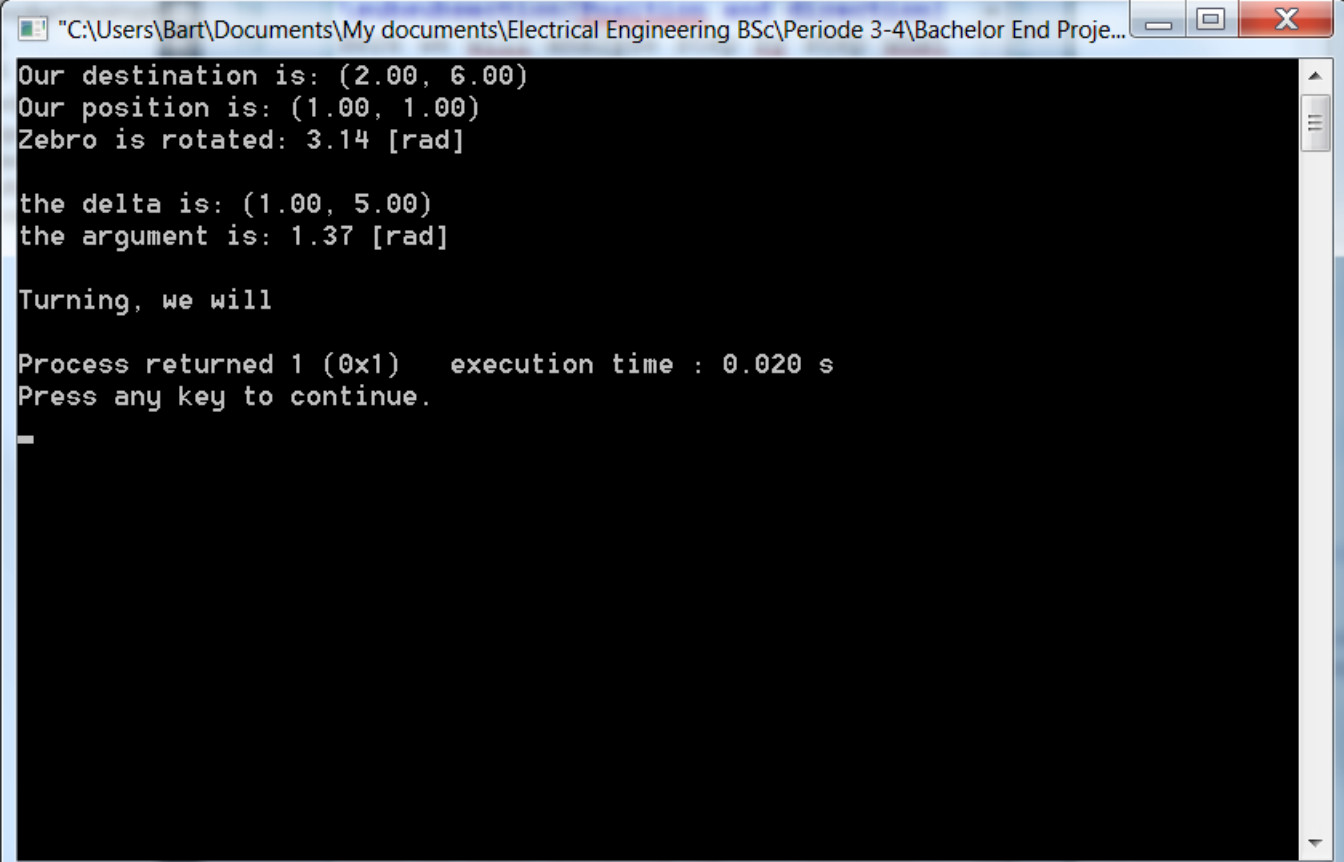
### 5.4.5 results

Testing this algorithm is a bit more complicated than it looks, because it needs input from a lot of the not yet available subsystems. Therefore, it is decided to test the different parts of the system separately and combine it as much as possible. This will be done by dividing the whole ONS algorithm in two main processing parts, the direction part and the collision avoid command part. This because the FSM that is made is just straight forward and the collision detection part is a sub part of the collision avoidance. Now, a few test cases are made to verify whether the parts work correct or not.

#### Position and direction

Here an analysis is made step by step what this sub part does and how it gives us the needed answer. First the coordinates of the zebro and the destination are put in. Then the desired direction and the argument are calculated, compensated for in which quarter it is. The whole coordinated system that is used here is a Cartesian system, so it can use earth's coordinates. Then it checks whether the difference is between the direction of the zebro and the destination is smaller than a threshold, so it will also walk when it is not perfectly aligned. From that it concludes with turning or not turning.

A little 'testbench' is created to check whether the code is working properly or not. Here the position for the Zebro is created, which is normally done by the GPS module and a direction the Zebro is standing is created, this will be done normally by the onboard compass. Some extra statements are added to create a proper output. Next is the output for an example, The output here will be whether it will turn or not.



```
"C:\Users\Bart\Documents\My documents\Electrical Engineering BSc\Periode 3-4\Bachelor End Proje...
Our destination is: (2.00, 6.00)
Our position is: (1.00, 1.00)
Zebro is rotated: 3.14 [rad]

the delta is: (1.00, 5.00)
the argument is: 1.37 [rad]

Turning, we will

Process returned 1 (0x1)   execution time : 0.020 s
Press any key to continue.
-
```

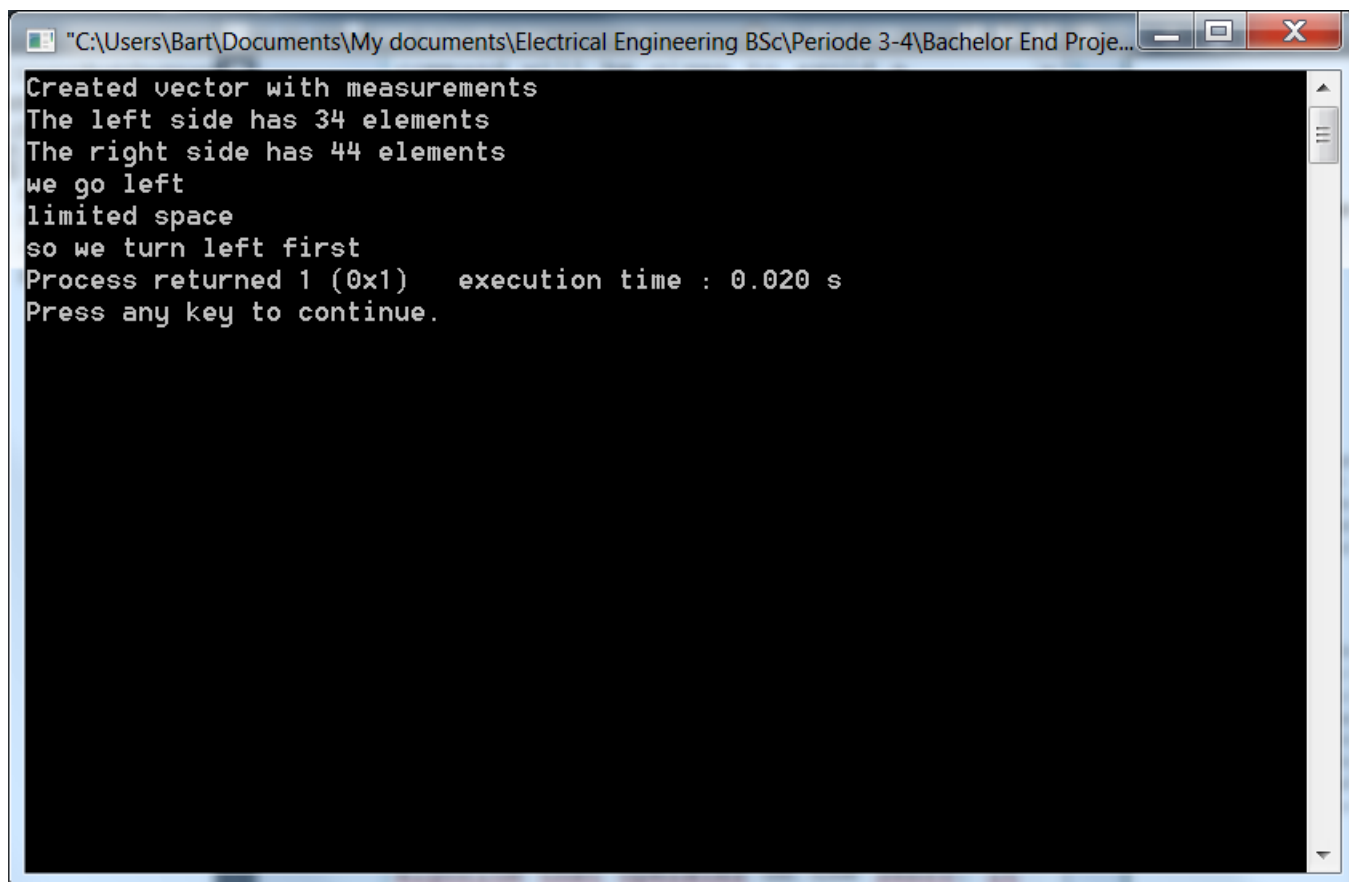
Figure 17: Test result from the test of the direction search part.

Here is seen that the output tells to turn. This is correct according to the given parameters.

#### Collision avoidance commands

Now is determined what command will be given to avoid a obstacle that is in the way of the Zebro. First a vector is generated that represents the values received from the LIDAR Lite, this done by a regular function and some overwriting. Then it determines if the distances are further than a certain threshold or not, this results in a 1 or 0 respectively. To correct faulty measurements the results are smoothed by

changing 010 and 101 to respectively 000 and 111. Then the side that has the most space is determined and if it has a lot of space or not. This is concluded with a command that curves to one side or turns first and then curves to one side.



```
"C:\Users\Bart\Documents\My documents\Electrical Engineering BSc\Periode 3-4\Bachelor End Proje...
Created vector with measurements
The left side has 34 elements
The right side has 44 elements
we go left
limited space
so we turn left first
Process returned 1 (0x1) execution time : 0.020 s
Press any key to continue.
```

Figure 18: The result of the obstacle detection part

Another 'testbench' is created to check if the code works. It correctly gives back the set of commands to the zebro. So now is known it does work as it should and the moves needed to be done can be determined. The result of the test can be seen in figure 18. Here a vector of distances from the LIDAR is given to test it.

#### 5.4.6 discussion

Now to conclude with reviewing our specification with the system that is realized. An autonomous algorithm that operates on the zebro is designed. It makes uses of a set of sensors to determine position, environment and direction to find its way to the destination. With this, the system will be able to successfully complete the challenge of the European Rover Challenge.

## 6 Conclusion

Now that the design choices are completed and their implementation of them. A conclusion can be drawn from the specifications. Now a check of the points from the beginning of the thesis with the found results is done.

1. *A communication system that is able to send and receive over 1000 meter:* This task has been done by the other subgroup. They have not tested the system yet, but they expect to be able to communicate on a range of around 5.000 meter.
2. *A way to let the Zebro navigate autonomously from one point to another with limited processing power:* For this the Onboard Navigation Algorithm is made, it works correctly in the simulations. When the microcontroller arrives, it will be tested to port it with the rest of the Zebro.
3. *A way to determine the location of the Zebro:* Here the decision is made for a GPS module, this one will arrive soon. Testing of the module is still needed to add it to the Zebro.
4. *A obstacle detection system:* For this, a LIDAR distance sensor on a rotating motor is used. With this it is possible image the environment to use it during movements. This module is tested and worked correctly, only the integration in the rest of the Zebro is needed.
5. *A camera with sufficient quality and enough fps:* For this the decision is made for a camera module. The implementation with the rest of the system to send the images is still needed in the upcoming months.
6. *Manual controls:* The user interface has been made by the other subgroup. They included an input of a XBox controller. Still, there had to be of this input.
7. *The system should be as modular as possible:* In the whole design modules are chosen that can be easily replaced by other ones.
8. *The maximum budget is 15.000 EUR:* Our modules did not exceed the available budget.
9. *A maximum power consumption is required within the specifications:* Since almost all module can be placed in a sleep mode when they are not used, power consumption is kept to a minimum.

From this is now looked to the future to determine what is still needed to complete the Zebro.

### Future plans

With the challenge upcoming in September, there is still a lot of work that has to be done. Mostly integrating the different parts that will be delivered in the next months. Integrating always is one of the most time consuming parts of engineering, so this will be tested with integrating them by testing on development boards. So after finishing up the exams work will go further on the project to complete the Zebro, so it is ready to win the European Rover Challenge!

## 7 Ending

To conclude this report, we want to thank the following people. First of all, we want to thank Maneesh Kumar Verma, for his time and energy to supervise us and let us be a part of the Zebro team. Second, we want to thank Chris Verhoeven, he invited us to take part of the team and helped us in engineering. At last, we want to thank all other the people that helped in designing the Zebro. A lot of time and effort was put in the Zebro when we started. We will continue evolving the Zebro to the next level.

*Bart van den Bogert and Bart Rijnders*

## 8 Bibliography

1. Skolnik, M. 2008, Radar Handbook, McGraw Hill
2. O'Donnell, Robert M. 2002, Slides Introduction to Radar Systems, Lincoln Laboratory Massachusetts Institute of Technology
3. Global Positioning System: Theory and Practice ,authors B. Hofmann-Wellenhof,H. Lichtenegger,J. Collins.
4. Digital and Analog Communication Systems - Leon W. Couch II
5. Structured Electronic Design, by Chris Verhoeven et al.
6. O'Donnell, Robert M. 2002, Slides Introduction to Radar Systems, Lincoln Laboratory Massachusetts Institute of Technology
7. Skolnik, M. 2008, Radar Handbook, McGraw Hill
8. Marcoe, K. 2007, Slides LIDAR: an Introduction and Overview, Portland State University.
9. Shan, J. & Toth, C.K. 2008, Topographic Laser Ranging and Scanning: Principles and Processing, CRC Press Taylor & Francis Group

## 9 Appendix

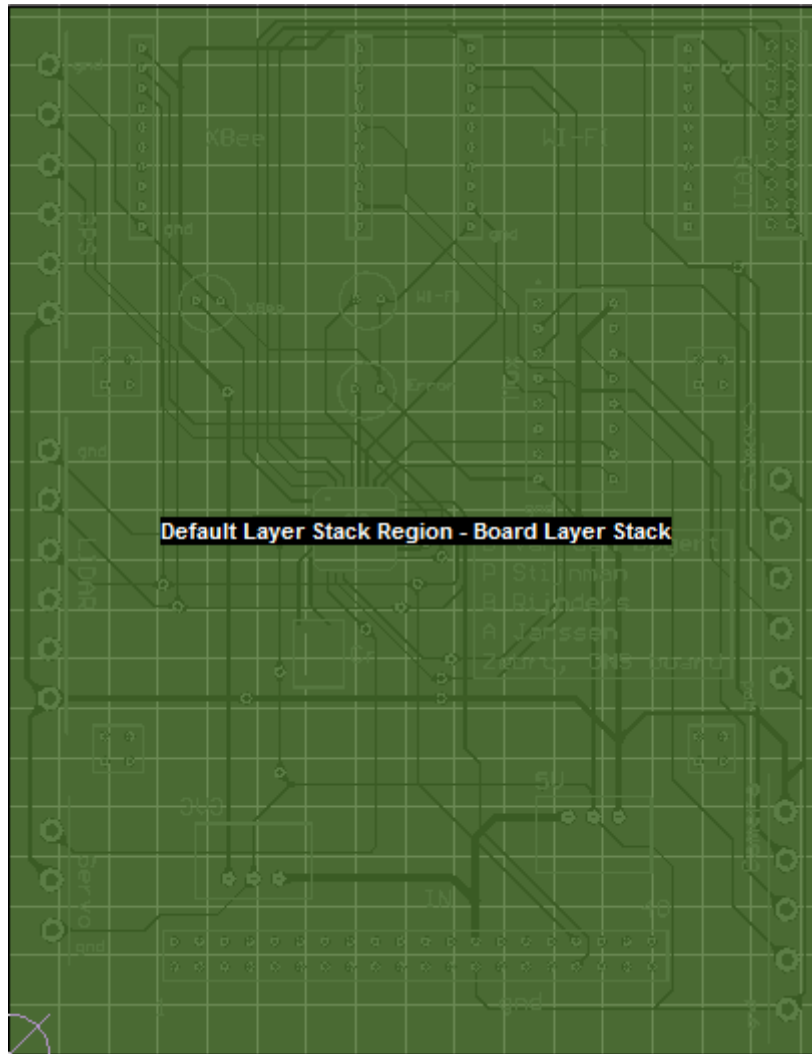


Figure 19: The board size.



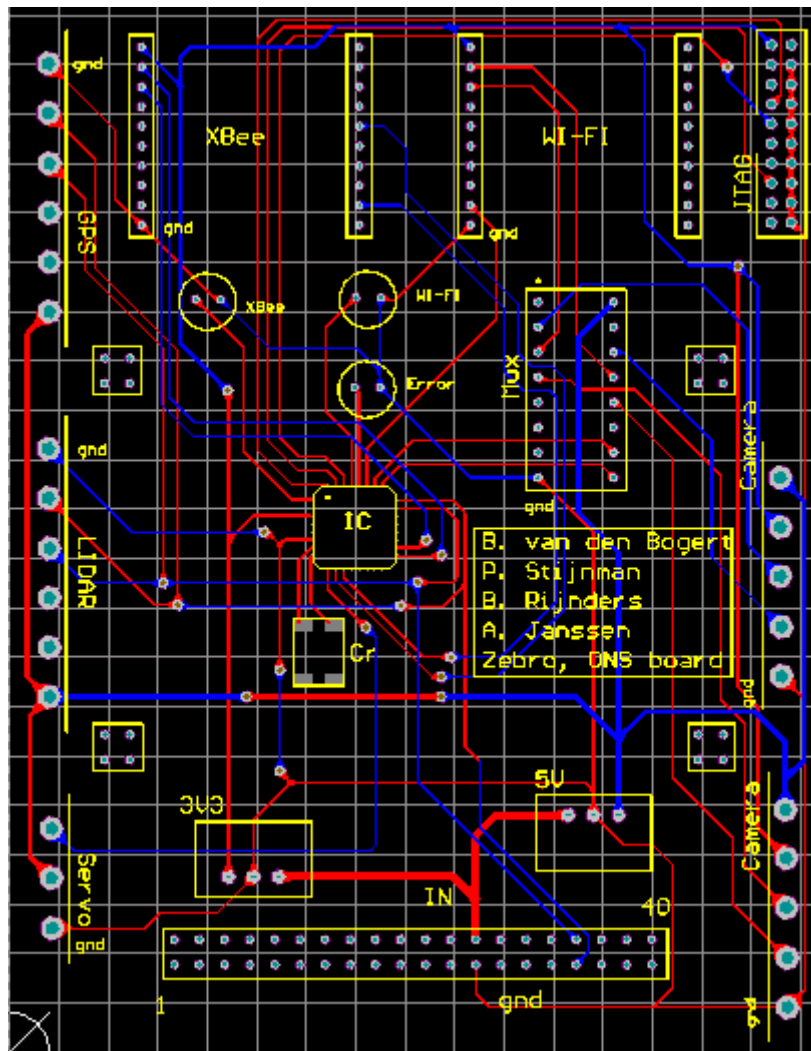


Figure 20: The PCB layout and routing.

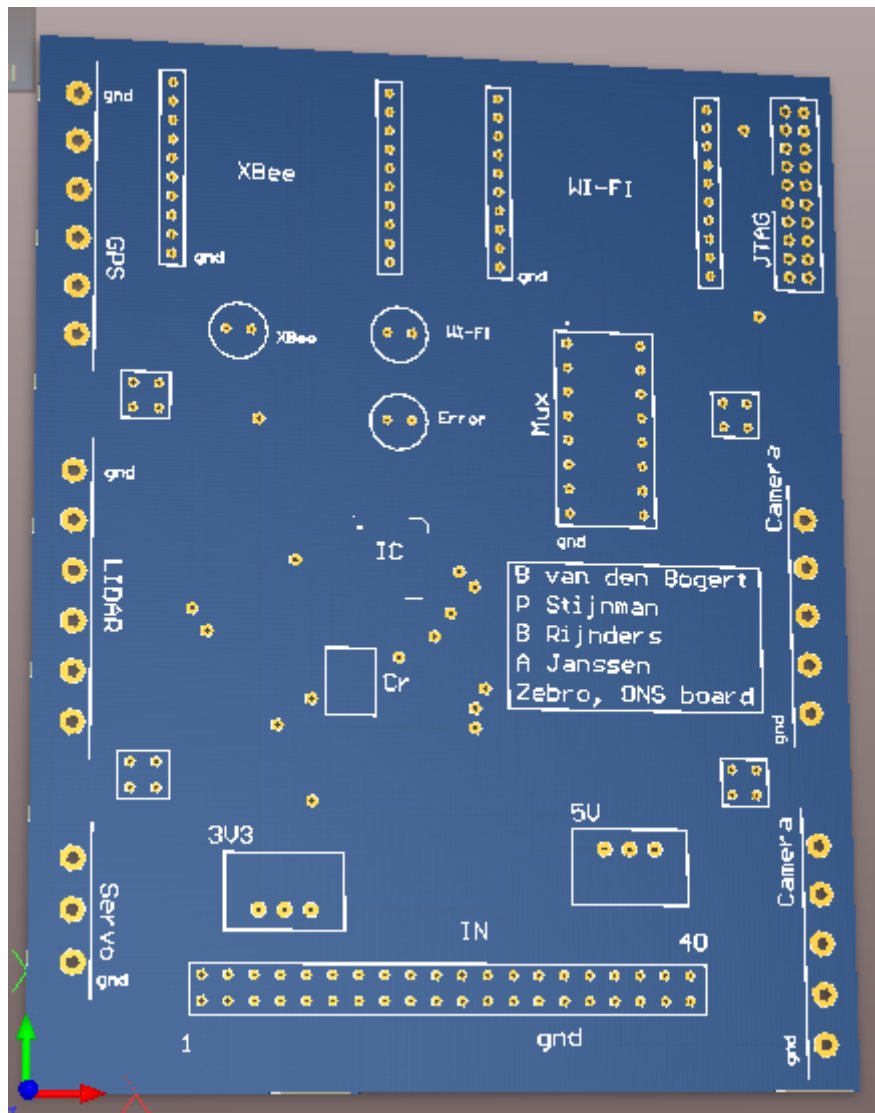


Figure 21: 3D look of the PCB.

## 9.1 Appendix A

Table 4: Microcontroller pin configuration

Pin number	Signal	Comment
Pin 1	Output	XBee LED
Pin 2	Not connected	
Pin 3	3v3 supply	
Pin 4	Not connected	
Pin 5	Not connected	
Pin 6	Ground	
Pin 7	External oscillator	32.768KHz
Pin 8	External oscillator	32.768KHz
Pin 9	PWM output	PWM signal for the servo
Pin 10	Output	Request to send for XBee
Pin 11	Input	Clear to send from XBee
Pin 12	Not connected	
Pin 13	Not connected	
Pin 14	Not connected	
Pin 15	Not connected	
Pin 16	Not connected	
Pin 17	UART-RX	connected to UART-TX of the XBee
Pin 18	UART-TX	connected to UART-RX of the XBee
Pin 19	Not connected	
Pin 20	Not connected	
Pin 21	Not connected	
Pin 22	Not connected	
Pin 23	I2C SCL	Connected to LIDAR,GPS and the ZPU
Pin 24	I2C SDA	Connected to LIDAR,GPS and the ZPU
Pin 25	Output	MUX B selector input
Pin 26	Output	MUX A selector input
Pin 27	Output	Sleepcontrol WiFi
Pin 28	Output	Error LED
Pin 29	Output	WiFi LED
Pin 30	SWD_CLK	Programming pin
Pin 31	RESET_B	Programming pin
Pin 32	SWD_DIO	Programming pin

Table 5: Multiplexer pin configuration

Pin number	Signal	Comment
Pin 1	Not connected	
Pin 2	UART-TX of camera	camera 1
Pin 3	UART-RX of WiFi	connected to UART-TX of camera 1 & 2
Pin 4	UART-TX of camera	camera 2
Pin 5	Not connected	
Pin 6	Ground	Inhib needs to be 0
Pin 7	Not connected	
Pin 8	Ground	
Pin 9	Input B	Connected to pin 25 of the microcontroller
Pin 10	Input A	Connected to pin 26 of the microcontroller
Pin 11	Not connected	
Pin 12	UART-RX of camera	camera 2
Pin 13	UART-TX of WiFi	connected to UART-RX of camera 1 & 2
Pin 14	UART-RX of camera	camera 1
Pin 15	Not connected	
Pin 16	5V power supply	



