TU Delft Delft University of Technology

ZebFlo

ENVIRONMENT OBSERVATION MODULE FOR THE DECI ZEBRO

Obstacle and Cliff Detection For Robotics Applications Using Miniaturized Sonar and IR Distance Triangulation

BACHELOR THESIS OF

Lode De Herdt 4390601
Jan Maarten Buis 4351266

June 19, 2017

DELFT UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE

ELECTRICAL ENGINEERING PROGRAMME

Abstract

This is a bachelor thesis about the design and development of an observation and sensing module for the Deci Zebro robot that is being developed at the EEMCS faculty of Delft University of Technology.

This part of the thesis is about the development of a cliff- and obstacle detection system, as well as the design and development of a fitting plastic enclosure for the module and the design of a PCB that integrates the module's electronics.

The report explores and compares different types of distance sensors and concludes that the best option is to use two infrared-based distance sensors on the left and the right of the robot for cliff detection, and a rotating ultrasonic transceiver on a servo motor to detect obstacles. For easy debugging and to allow communication with nearby humans, a LED ring was also fitted to the top of the module. The design process and the implementation of these components is described in detail.

The module's enclosure is designed using SolidWorks software and afterwards printed using a 3D-printer. The PCB is designed using the opensource KiCad software.

Contents

Chapter 1

Introduction

1.1 General

This is the final thesis report for the bachelor graduation project of group L. The project consists of designing the Observation and Sensing module for the Zebro research team at the EEMCS faculty. The thesis is split up into three parts, of which this report deals with the obstacle- and cliff detection system, as well as developing a full plastic enclosure for the module and a custom PCB.

For the final bachelor project, students are required to show off the academic knowledge and skills that they have adopted during the three years of the Electrical Engineering program. The goal is to come up with a creative and scientifically sound solution for a 'real world' problem. This problem can be formulated by the TU Delft or by a company.

1.2 About the Zebro Team

The Zebro project (Zebro is short for 'ZesBenige Robot', Dutch for six-legged robot) at Delft University of Technology has as aim to develop six-legged autonomous robots for numerous applications, of which one is studying swarm behavior in robotics. The team consists of Electrical Engineering students from the TU Delft as well as other institutions, both in the bachelor and master phase, led by researchers Dr. Chris Verhoeven en Dr. Edwin Hakkenes.

This year, the team has adopted three teams of bachelor students to develop three external modules for the robot as part of their final thesis. The modules are required to work with the existing Zebro interface that is developed by the Zebro team, called Zebro-bus. This interface is used to let the central system communicate with all of its modules [1]. The modules have information-gathering and communicating with the main system as its main goals, the modules are not supposed to make independent decisions based on the data from the sensors.

1.3 Project Organization

1.3.1 Organization within the project group

The total Environment Observation module is developed by a group of six bachelor students. This group is further divided into three subgroups of two people, each subgroup deals with a certain aspect of the module. The module was divided by the project group into three parts:

- Obstacle & Cliff Detection;

- Obstacle and Motion Detection using a Laser Rangefinder and Optical Flow [2];
- Safety and Sensor [3];

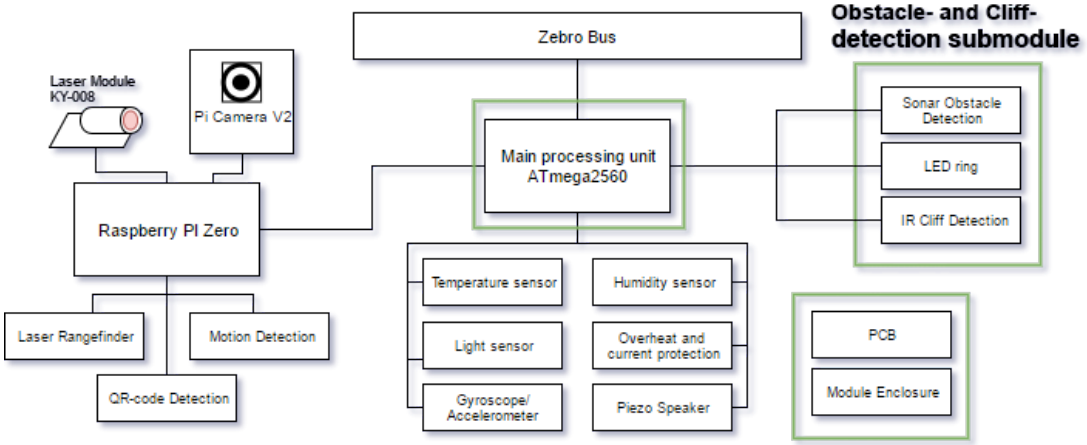


Figure 1.1: *Overview of the entire module*

Figure 1.1 shows an overview of the entire module. The submodule described in this document is highlighted in green. Note that the choice of microcontroller was also made by this group; this will be described in section 8.2.

The subgroups adhere to this division of labor with regard to solving the problems laid out in the requirements. When it comes to additional tasks, like implementing the communication with the Zebro main system, designing a PCB, designing the enclosure or handling the power distribution, subgroups or individual team members volunteered to work on these tasks.

Since all the parts of the module are closely tied together physically, power-wise and communication-wise, exchange of ideas and designs within the group is important before carrying it out.

1.3.2 Organization within the faculty

The module is developed for the Zebro robot that is developed by master and bachelor students of the TU Delft and other institutions, supervised by Chris Verhoeven and Edwin Hakkennes. This year, three groups of bachelor students develop a module for the robot as part of their final thesis. The bachelor graduation groups are also under the supervision of the Zebro team supervisors. All bachelor projects are also under the supervision of Ioan Lager.

1.3.3 Communication with supervisors

To keep the supervisors posted about the progress that is booked and the problems that are faced, every week a meeting is scheduled where at least one of the supervisors will be present to hear what is done and to give feedback where necessary. Since the Zebro team also acts as 'customer' for the project group, requirements and/or new ideas for the module can also be discussed.

Chapter 2

Program of Requirements

This chapter lays out the general criteria and the minimum requirements for developing and testing the module. Note that these are the requirements for the entire module, per sub-module more specific requirements have to be set. The description here is a summary of the Thesis Requirements as was formulated by the supervisors [4].

2.1 General criteria set by the Zebro team

- *The module should be self-contained*
  This means that it should be capable of gathering and processing its own data without the need of 'help' from the outside. Furthermore, it should be able to detect when faults occur in its electronics (e.g. short circuits or overvoltage) by implementation of an Autonomous Module Damage Protection System (AMDPS).
- *The module should have very well defined interfaces with the outside world*
  The interfaces that are implemented are a physical interface to attach the module to the Zebro body; a power interface to be able to provide power to the module; a control and status interface to be able to read out the sensor data and for debugging. The interfaces that the module is going to use should naturally adhere to the existing interfaces of the Zebro.
- *The module should be easily replaceable (preferably by another robot) so that repairs can be made without human interaction*
  In order for the module to be easily replaceable, it should be easy to detach the module from the Zebro, relating to the previous requirement, and it should be easy to build a replacement module.

2.2 Discrete requirements for the module set by the supervisors

The team has also set some minimal requirements for what the finished module should be able to do. They are numbered EM-1 til EM-6. They are:

- EM-1: Detect obstacles
- EM-2: Detect cliffs
- EM-3: Discern between scalable and dangerous obstacles and cliffs
- EM-4: Determine the system's temperature
- EM-5: Determine the system's voltages, currents and power flow.
- EM-6: Implement an Autonomous Module Damage Protection System

Relevant for this submodule is EM-1, EM-2, and EM-3.

2.3 Thesis Requirements

The thesis covers the development of the module. For the thesis, the minimum requirements are numbered BT-1 to BT-4. They are:

- BT-1: Develop a test bench to run and record different parameters of Environment Observation performance
- BT-2: Develop and test at least three types of sensor systems
- BT-3: Develop the necessary software, hardware, control and their architectures for requirement EM-3
- BT-4: Develop the necessary software, hardware, control and their architectures for requirement EM-6

2.4 Time Constraints

As the bachelor final project takes place in Q4, around ten weeks are reserved for working on the project. This thesis is handed in the 19th of June, the defense will he held on July 5th 2017. After the thesis has been finished, some time is left to continue working on the prototype.

2.5 Interface with the Zebro body

The module will be connected with the Zebro robot both physically and electronically. Therefore it is necessary to know the details about the interface.

2.5.1 Physical dimensions

As designing a plastic enclosure of the module is one of the tasks, it is important to know where the module will be located and how much space there is. The technical drawings of the Zebro body can be found in the appendix of this report.

2.5.2 Electronic interface

The electronic interface consists of a connection with a low-level communication bus shared by several parts of the robot that adheres to the I2C protocol, called the 'Zebro-bus'. All modules must use this protocol to communicate with the main controller [1].

Chapter 3

State-of-the-art

Before starting to solve the problem, it is important to see what has been done before and what are ideas that could be worked with. Since the main tasks of this sub-module are 'seeing', whether it's obstacles (matter) or cliffs (absence of matter), distance sensors would be a good bet. Distance sensors come in several flavors, of which the major ones are Time-of-Flight (ToF) based. In other words, these sensors send out some kind of pulse, in the form of light or as a sound wave, measure the time it takes for the pulse to come back and do some processing before returning the measurement result.

Generally, the task that has been given does not require any ground-breaking technology. Many different types of distance sensors that suit our purpose already exist and are widely used. It would therefore likely not be efficient to redesign products that are cheap to purchase and work well. The difficulty in this task is the appropriate choice of sensors and the efficient integration of many different devices into one streamlined system.

Other projects and studies with various distance sensors have been done by other individuals. It is therefore beneficial to look at their findings and build upon this. One such example is Kassim et al., who investigated the influence of different materials' properties on the performance of infrared (IR) and ultrasonic (US) distance measurements. One of the conclusions of this article is that the US sensor used generally performs better with different objects than the IR sensor used, and that the IR sensor's range is significantly smaller. [5]

The Ultrasonic Radar paper by Paulet et al. outlines an experiment using an ultrasonic distance sensor on a stepper motor to create a radar, and various experiments with this system. This is an interesting solution that could also be used on the Zebros. The paper also provides some calculations as to what the effect of temperature is on the results; this is an interesting avenue to explore for us too. [6]

One more aspect that will need to be considered in this project is that data received from sensors may not be reliable enough and some error-correction may be needed. Probabilistic Aspects in Mobile Robots Navigation by Stănescu et al. investigates this and provides some comparisons of different error-correction algorithms in real-time processing. [7] It will need to be investigated whether the algorithms described in this paper will be appropriate for our project.

Chapter 4

Obstacle Detection

This chapter describes the design process of the module's obstacle detection system.

4.1 Problem Description

The requirements that concern this subsystem, as was described in chapter 2, are EM-1 (detection of obstacles) and EM-3 (discerning between scalable and dangerous obstacles). This essentially means that the system must be able to detect the presence of an obstacle near the robot, and determine whether or not the robot will be able to climb over it. The system will pass this information onto the main Zebro controller (outside of our module) and this controller will make behavioural decision based on the received information. The main controller needs this information to ensure that the Zebro can navigate around obstacles, and does not get damaged by running into obstacles. It must be noted that many different things can be dangerous obstacles for the Zebro. As the robot will go out into the real world, this is not limited to stationary obstacles like walls, but also includes moving obstacles like people and vehicles. The detection also needs to be reasonably quick because the Zebro should react quickly if, for example, a person suddenly steps in its path.

Lastly, it is important that the chosen sensor is cheap. This is because the Zebro robots are designed with mass production in mind; in the near future, 100 robots will be produced. Ideally, the price of these robots will be limited to a few hundred Euros. As a consequence, this means that very expensive sensors should not be used.

4.2 Potential Sensors

After discussions with the group and a literature study, several possible solutions were available.

4.2.1 Camera

One possible solution would be to mount a small camera (or possibly a stereo camera) on the front of the Zebro. This camera would give a lot of information about its surroundings, but significant amounts of image processing would be necessary to get useful information about possible obstacles from the camera. Another benefit of a camera is that it can be used for a lot more things than just obstacle detection. For example, it could also be used for reading QR codes, recognizing objects/faces, recording video, etc. A camera system is therefore potentially very useful. However, because of the complexity involved in designing the image processing algorithms, it was decided that another subgroup would focus solely on the camera, and this camera would eventually be used to complement a second obstacle detection system.

4.2.2 LIDAR

LIDAR, short for LIght raDAR, uses a system of rotating lasers to determine distances to objects. However, the cheapest LIDAR systems available for purchase were about €100. This was considered to be far too expensive for our purposes, so LIDAR was not investigated further.

4.2.3 Feelers

In nature, animals also need to be able to detect obstacles in their path. Certain small animals, namely arthropods, use feelers (also known as antennas) to do this. A similar principle could also be applied on the Zebro robots; a pair of flex sensors could be attached to the front of the Zebro. If these sensors come in contact with an obstacle, they bend and their electrical resistance changes, which can easily be read by a simple circuit. This is a simple solution, but it does not give much information about the size and scalability of an obstacle. The detection range is also limited to the length of the flex sensors, so the obstacle would only be detected right before the Zebro walks into it. This is not ideal, so this sensor was abandoned.

4.2.4 Infrared

Another possible solution was to use an infrared sensor, consisting of a small IR transmitter, and an IR sensor. There are two main types of IR sensors. The first type is an IR transmitter with a simple receiver. The transmitter sends out IR radiation, which bounces off nearby objects. Next to the transmitter is a receiver; if this receiver receives a certain intensity of IR radiation, it knows that there is an object nearby. The major downside of this simple system is that it cannot measure distances, only whether or not an object is nearby.

A second type of IR sensor system is also available. Instead of a simple receiver, this uses an imaging sensor similar to what is found in digital cameras. This imaging sensor allows the system to determine the angle of the received IR radiation. As the distance between the transmitter and receiver is known, the angle of the received radiation can be used to calculate the distance to the object which reflected the IR radiation. This triangulation principle is shown in Figure 4.1.

This second system could work well for our purposes. Some research into available hardware showed that out of all the readily available sensors, the one with the largest range was limited to 150 cm. While this is acceptable for our purposes, it would be beneficial to have a higher range as this means obstacles can be detected earlier. One concern with IR sensors is that bright sunlight (which contains some IR radiation) may affect the readings, so this needs to be investigated.

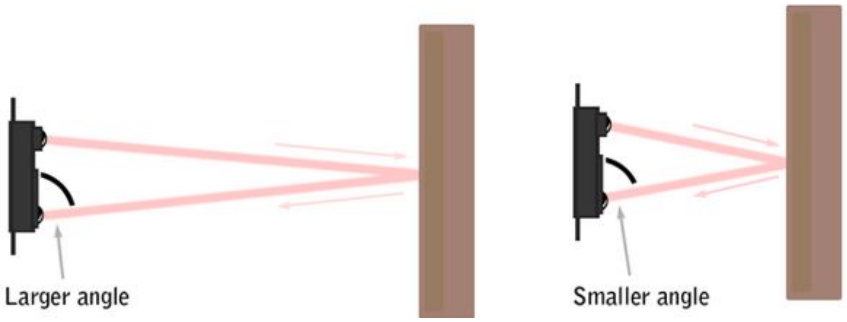


Figure 4.1: *Working principle of the Sharp IR distance sensor [8]*

4.2.5 Ultrasonic

Lastly, a possible useful sensor to detect obstacles is an ultrasonic sensor. This type of sensor relies on the 'time-of-flight' principle. The transmitter periodically sends ultrasonic pulses (usually 40 kHz). If there

is an object within range, then this object will reflect the sound waves and they will be received by the receiver. As the speed of sound is finite, there will be a measurable difference in time between the transmitting and the receiving of the pulses. Knowing the speed of sound and the difference in time, the distance to this object can then be calculated.

Ultrasonic distance sensors are widely available for purchase and they are relatively cheap (starting at a few Euros). Depending on the exact model, most of these types of sensors have a range of about 4 or 5 meters. This is significantly more than the infrared sensor discussed earlier. One possible concern with ultrasonic sensors is that, if many ultrasonic sensors are used near each other (as is likely to be the case in a swarm of Zebro robots), they may interfere when the ultrasonic pulses from one Zebro are reflected and then received by another Zebro. This will need to be investigated. Another limitation of ultrasonic sensors is that they generally cannot see obstacles which have their normal at an angle greater than 45 degrees to the transmitter. This is because the sound will deflect from this obstacle, and will not reflect. Therefore, the sensor will not receive the pulses it sent out. Certain absorbing materials also will not reflect the sound waves, so these will also be invisible to the ultrasonic sensor.

Due to the cheap price and good range of ultrasonic sensors, it was decided to use this type of sensor for the obstacle detection system. However, because of the limitations discussed above, it is useful to also have a camera system (which will be designed by a different subgroup) as discussed in subsection 4.2.1. Both systems inherently have unique limitations, so by combining the two systems the limitations will be minimized. If one system is unable to detect an obstacle, the other system will detect it.

4.3 Ultrasonic Sensor Testing & Comparison

Once the sensor technology was chosen, the exact model of sensor to be used also had to be chosen. After some research, it was determined there were two widely-used ultrasonic sensors: the Parallax Ping and the HC-SR04. Both were ordered so they could be tested and compared.

4.3.1 HC-SR04

The HC-SR04 has 4 pins: two for power, one 'Trigger' pin and one 'Echo' pin. To start a measurement, the Trigger pin must be asserted for at least 10μs. The sensor's transmitter then sends out eight pulses of 40kHz. This sound wave then travels through the air and, if an obstacle is in its path, reflects from the obstacle. After a certain amount of time, the reflection is received by the sensor's receiver. The sensor then asserts the echo pin with a width corresponding to the amount of time it took for the signal to travel to the obstacle and back. Thus, by measuring the width of the pulse on the echo pin and knowing the speed of sound, the distance to the obstacle can be calculated.

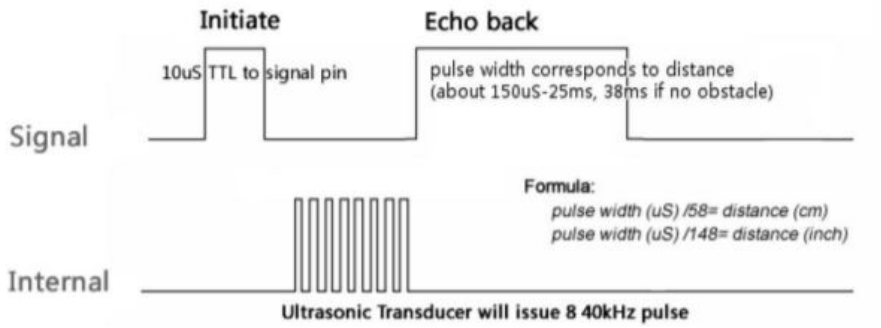


Figure 4.2: *HC-SR04 Timing [9]*

4.3.2 Parallax Ping

The Ping only has 3 pins: two for power and one signal pin. It operates on a similar principle as the HC-SR04, but the signal pin is used both as an input and an output. First the signal pin is asserted high for a minimum of 2μs to trigger a burst of 40kHz pulses. Next, the sensor sends a high signal on this pin with width proportional to the time the signal has travelled. Clearly, this is very similar to the HC-SR04, the only main difference being that the Ping uses the same pin both for the input and output. This may be beneficial as it means only one of the microprocessor's IO pins is needed to connect this sensor.

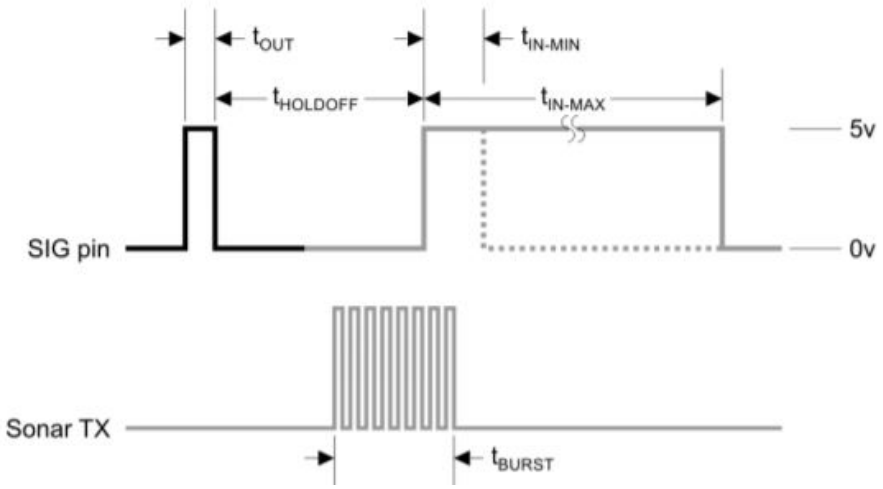


Figure 4.3: *Ping Timing [10]*

4.3.3 Comparison

In order to be able to decide which sensor should be used, both were tested. The first test was to determine the range and accuracy of both sensors. To do this, a flat solid box of approximately $1m^2$ was placed at various distances from the sensors, with the sensors on the box's normal, and the sensors' outputs recorded using an Arduino (which will be further discussed in chapter 6). As can be seen in Table 4.1, both sensors have similar accuracy until 300cm. The Ping is limited to 315cm, while the HC-SR04 is not, but the HC-SR04's accuracy decreases significantly after about 300cm.

Table 4.1: *Distances to a flat solid object measured by the Ping Parallax and HC-SR04 at normal incidence*

Real Distance [cm]	HC-SR04 Distance [cm]	Ping Distance [cm]
20	21	20
40	40	41
60	60	60
80	79	80
100	99	100
150	147	149
200	195	198
250	245	248
300	294	296
350	338	315*(1)
400	370	315*(1)

*(1)Note: the Parallax Ping is only rated up to 300cm according to its datasheet [10] . When an object is more than 315cm away, the sensor will simply output 315cm.

The ultrasonic sensors require a signal to be reflected from an object. The problem with this is that, depending on the angle of incidence of the sound wave, the wave may simple be deflected instead of reflected. To test the impact of the angle between the object and the sensor on the measurements, the same box as before was placed with its center at a distance of 1m and rotated between 90 degrees (perpendicular to the direction of propagation of the sensor's sound waves) and 0 degrees (parallel). The results can be found in Table 4.2. Clearly this is a major limitation of the ultrasonic sensors; they cannot see flat objects at small angles. This could be a problem if, for example, the Zebro is approaching a flat wall at a shallow angle. However, as mentioned before, another subgroup is also working on a camera system which will not have this limitation, so this system will be able to see the wall in this example.

Table 4.2: *Distance measured to an object at 1m for various angles*

Angle [degrees]	HC-SR04 Distance [cm]	Ping Distance [cm]
90	99	98
80	98	101
70	95	97
60	90*(2)	102
50	88*(2)	315
40	Timeout	315
30	Timeout	315
20	Timeout	315

*(2)At these angles, the HC-SR04 did not give consistent measurements. It alternated between the noted value and a timeout.

Lastly, an important comparison is the price. This is important as the Zebros will eventually be mass produced, so a small difference in price will become much more significant at higher quantities. The HC-SR04 is very cheap, and can be found for just under €1 from some sources. However, the Ping is about €30.

With the help of some experiments with both sensors, it was determined that the HC-SR04 and Parallax Ping have similar performance. However, it was decided to use the HC-SR04 as this sensor is significantly cheaper than its competitor.

4.3.4 Configuration

Another decision that had to be made was how the sensor would be mounted. There were a few options available:

- A single sensor mounted to the front of the Zebro
- An array of sensors at different angles, as in Figure 4.4
- A sensor rotating on a servo to form a sort of radar

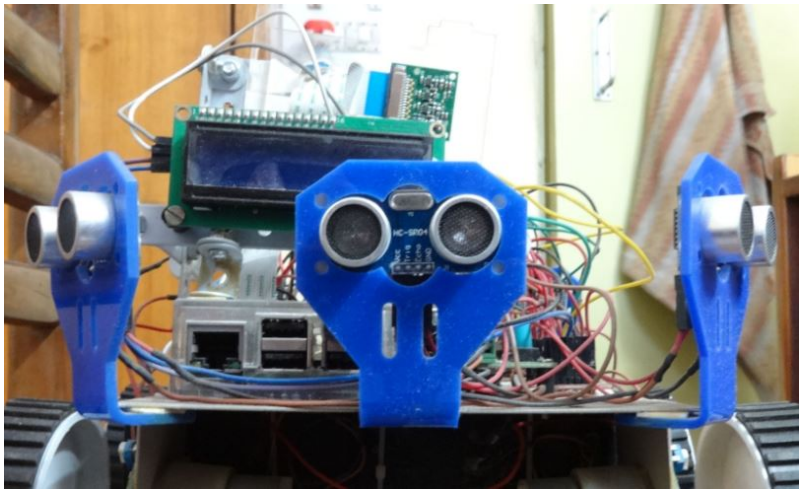


Figure 4.4: *Ultrasonic sensors connected in an array [11]*

The single sensor would only give information about the distance to the nearest object in front of the Zebro; it would give no indication about where the obstacle is. An array of sensors would work better, but this would require several sensors so a lot of connections would be needed.

The last option is putting a sensor on a rotating servo, and measuring the distances every few degrees. This allows the horizontal field of view to be customizable, and gives some useful information about the location of obstacles. Therefore, this option was chosen.

Chapter 5

Cliff Detection

This chapter describes the choices made during the design of the cliff detection system.

5.1 Problem Description

The requirements for the cliff detection subsystem (see EM-2 in the Requirements chapter) are more arbitrary than for the obstacle detection subsystem, since the robot only has to determine whether it comes near a cliff or not. A specification of the properties of the cliff is not necessary. Therefore a simple yet robust sensor should suffice.
For this problem, a similar distance sensor as for the obstacle detection system could be used, but since it only has to detect when the measured distance becomes larger than a certain standard distance (say, from the top of the robot to the ground), a much simpler setup can be used. For this subsystem, robustness is more important than precision.
Finally, while comparing sensors, it should be taken into account that in order to effectively sense a cliff in front of the robot, the sensor may be placed on the robot at a certain angle. This should not be a complication for the effectiveness of the sensor.

The sensor comparison from the last chapter still holds up. The ultrasonic and infrared sensors were deemed most suitable for this purpose. However, since the ultrasonic sensor works with sound waves, it is therefore less effective when used under an angle due to reflection losses. IR sensors have as a disadvantage that they can not measure as great distances as ultrasonic distance sensors, but for this subsystem, that was not a constraint. It should also be taken into account that their prices are very similar.
Therefore, for this subsystem, a setup that employed an IR sensor was chosen.

5.2 Infrared Sensor Testing

While looking into the sensors of this type that are available, it was found that only a few very similar sensors (most of them produced by Sharp) were suitable of our purpose, in a price range from about €10 to €25, with the more expensive ones having a longer range. However, since a range of about half a meter would be sufficient, (the exact depth of the cliff does not have to be measured), a cheaper sensor was chosen.

Since the product/brand variety is very limited for this type of sensor, only one type of sensor was ordered and tested, the Sharp GP2Y0A21YK0F.

5.2.1 Testing

One of the cheaper IR distance sensors is SHARP's GP2Y0A21YK0F sensor. It has a range that lies between 10 and 80 cm and costs about €8. Once ordered, the sensor was tested for different types of surfaces. Since the sensor is IR (light) based, it would be interesting to see how different materials, shades of colors and angles affect the distance measurement. Several measurements were done, one distance measurement for white paper, one for dark metal and one measurement for white paper under different angles. The results can be seen in table 5.1.

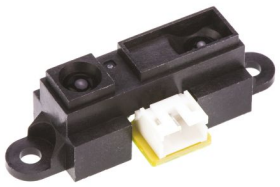


Figure 5.1: *The SHARP GP2Y0A21YK0F infrared distance sensor. [12]*

Real Distance [cm]	Measured Distance for white paper [cm]	Measured distance for black metal [cm]
10	11	12
20	22	21
30	33	33
40	43	44
50	54	60
60	62	70
70	76	81
80	100	random
90	126	random
100	random	random

Table 5.1: *The measurements done while testing the IR sensor. It can be seen that for a dark surface, the range that can be measured reliably is diminished.*

As can be seen, in most cases the sensor overestimates the distance by a few centimeters. For this purpose, this falls within the margin, since the robot does not need to know the exact distance between the sensor and the ground. Further, it can be noted that the measurement for light and dark materials for the same distance differs slightly, although it is only a few centimeters. Finally, for larger distances, especially those out of its range (as specified in the data sheet), the error margin becomes much larger, up to the point where the measured distance seems to be random.

Next, a the sensor was tested for different angles. The sensor was placed about 30 cm from a sheet of white paper. The results can be seen in Table 5.2.

Angle that was used [degrees]	Measured distance
90	32
80	33
70	33
60	32
50	33
40	32
30	33
20	33

Table 5.2: *Measurements testing the effectiveness of the sensor while placed at an angle. The distance used in the setup was 30 cm.*

In table 5.2, it can be seen that for different angles, the measured distance stays more or less the same. The results in this table mean that the sensor can still be used effectively when placed under an angle. This angle will be needed because the robot has to detect cliffs in front of him. If the sensor was to work only when placed straight downwards, it wouldn't be very effective as a cliff detection system since it would only be able to detect cliffs straight under the robot.

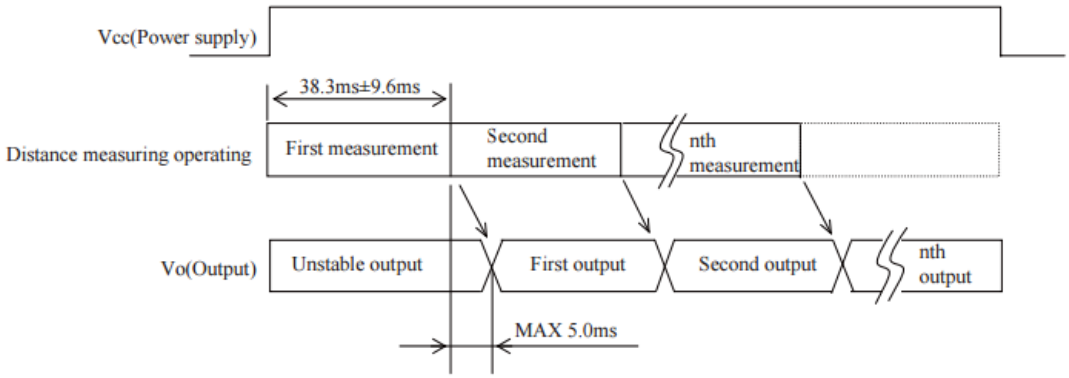


Figure 5.2: *Timing chart of the SHARP GP2Y0A21YK0F showing the output timings and intervals. [12]*

In figure 5.2, the timing of the output voltage of the sensor can be seen. The first output is available after about 43 ms, then, about 38 ms later, the second output is available, then after another 38 ms the third, and so on. Once set up, the sensor output is updated about 26 times per second.

During testing it was found that the sensor output was very robust, the measurement error always remained small. As the sensor requirements for this submodule are not very high, the SHARP sensor is considered good enough for this purpose.

5.2.2 Mounting

Once the sensor had been verified to meet the requirements for this submodule, it had to be decided in what setup the sensor would be used. As mentioned before, the sensor would be most effective when placed under an angle. Using this configuration, the sensor would be able to detect cliffs, but only if they are right in front of the robot. It would be better to check for cliffs on the right and left of the robot, since the robot might approach a cliff at an angle. Therefore, it was decided that two IR sensors should be placed on the left and right of the module, so that the system has a broader view. Later on, it was decided that both sensors should be placed at a horizontal angle of about 10 degrees. This way, the robot would be able to see cliffs in front of it and slightly next to it.

Chapter 6

Implementation on Arduino

In order to be able to read out all the sensors, perform necessary calculations, and communicate with the ZebroBus, a microprocessor is needed. This microprocessor will also need to be programmed. To facilitate easy prototyping, it was decided to use an Arduino board. These devices are relatively inexpensive, they can easily be programmed and powered over USB, and their default programming language is based on C/C++. Moreover, they generally have a sufficient amount of easily-accessible IO pins that can be used to connect external devices (like sensors in our case).

In our case, an Arduino was used to:

- Read the analog output of the two IR sensors
- Drive the servo motor to the appropriate position
- Perform the necessary timing to trigger the ultrasonic sensor and perform calculations to read results
- Display the presence of cliffs and obstacles on the LED ring

6.1 Main Function

In Arduino syntax, there are two standard main function: setup() and loop(). The setup() function runs once when the device starts or resets, and the loop() function then runs continuously in a loop. Both of these functions may contain calls to other functions when necessary.

This subgroup and another subgroup worked on Arduino code for their respective sensors. The other subgroup was responsible for integrating all Arduino code. The code was split into several classes so that each class would contain the necessary code for one sensor/device. To ensure uniformity, a few requirements were set:

- The class must contain an Initialize() member function. This will be called during setup.
- The class must contain an UpdateValues() member function which will be repeatedly called. When this function is called, the appropriate variables must be updated using the measurements from the sensors.
- The code must be as fast as possible and non-blocking, meaning it does not stop other sensors from being read

Figure 6.1 shows an overview of how functions are called. Note that 'Foo' is a placeholder for the name of the specific class in question. The classes relevant for this subgroup are Cliffclass, DistanceClass, and LEDringClass.

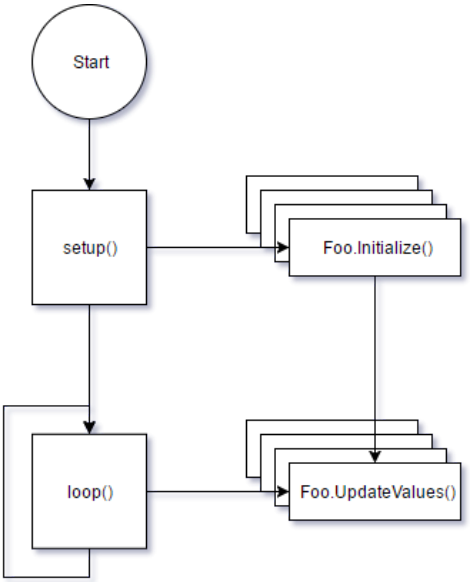


Figure 6.1: *An overview of how functions are called*

Furthermore, in the UpdateValues() a parameter can be sent. Some classes receive a pointer to 'Error_Byte', a variable of type uint8_t. In this byte, classes can set individual bits to indicate there is an error. Relevant for this subsystem is that bit 3 and bit 4 can be set high to indicate a cliff on the left or right respectively. This Error_Byte is displayed on the LED ring as will be discussed later. Because of this, if there is an 'error' such as a nearby cliff, this will be displayed on the LED ring. This data will also be sent over the ZebroBus, but the exact workings of this communication will be discussed by a different subgroup.

6.2 Cliff Detection

As discussed in chapter 5, an IR sensor which uses triangulation to measure distance is used for the cliff detection. This sensor can be read by reading the voltage on its signal pin. This voltage is inversely proportional to the distance, typically 0.4V for 80cm and 2.3V for 10cm [8].

The distance measured by the sensor is read as follows:

```
int cliffDistR = 4800 / (analogRead(irPinR) - 20);
```

The above formula converts the analog value of the output voltage of the sensor to a distance [13].

The variable maxCliffDist is the maximum distance that would be expected at the front of the Zebro. If the IR sensor measures a distance greater than this; the conclusion is that there is a cliff in front of the Zebro and the error byte must be adjusted:

```
if (cliffDistL > CONFIG.getmaxCliffDist() || cliffDistL < 0) {
  (*Error_ptr) |= (1 << 3); //Cliff detected on the left!
} else {
  (*Error_ptr) &= ~(1 << 3); //No cliff on the left
}
```

The same is done for the right sensor.

6.3 Obstacle Detection & Servo: The Distance Class

6.3.1 Reading Sensor Data

The obstacle detection and the driving of the servo, which is done in DistanceClass, is significantly more complex than that of the CliffClass. As discussed in subsection 4.3.1, the 'Trig' pin must first be asserted for at least 10 µs to start a measurement. After some time, the 'Echo' pin will then be high for the length of time there was between the transmitting of the US signal and the receiving of its reflection.

There are a few things that need to be considered. Firstly, the measurement cannot be triggered too often. If a new measurement is triggered before all reflections of the previous US pulses have died out, the sensor will give incorrect distances. Secondly, it is important that the code is non-blocking. When the length of the Echo signal is being measured, the rest of the code must keep running. If it is blocked, this means that other sensors like the gyroscope will not sense possible dangers.

The first solution that was designed used the pulseIn() function to measure the length of time that the Echo pulse was high:

```
duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound
    wave travel time in microseconds
```

This works, but the pulseIn() function stops execution of other code for as long as the echoPin is high, which can be up to 38ms [9]. This means that the updating of other sensors is paused for this amount of time, which means there is a large risk of missing crucial sensor data, for example a sudden acceleration or deceleration to indicate a fall.

A solution to this problem was to make use of interrupts in order to make the code non-blocking. This way, once the short pulse has been sent on the TrigPin, other code can run while waiting for the sound waves to return to the sensor. Once a change in the EchoPin is detected, other code can be stopped temporarily and the necessary calculations can be made. The basic idea of this is to attach an interrupt to the EchoPin so that the interrupt handler runs whenever there is a change detected on the EchoPin:

```
attachInterrupt(digitalPinToInterrupt(echoPin), distInt, CHANGE); //Attach an
    interrupt to echoPin, so that the interrupt handler is run when a change
    is detected
```

Where distInt() the interrupt handler which calls echo_interrupt(). When the interrupt handler runs, it first looks whether the EchoPin is now high or low; if it is high then it starts timing and if it is low then it stops timing and performs necessary calculations to determine the distance to an object.

6.3.2 Visual Radar

During online research, a project by Dejan Nedelkovski was found which also used an US sensor to create a radar [14]. This project also used an US sensor on a servo, both connected to an Arduino. The measurements of the US sensor are then communicated to a computer via the serial port, and then the open source processing software is used to visually display the results of the measurement on a computer screen. As this seemed very convenient for initial testing of the sensor and for a demonstration of the principle to the supervisor and colleagues, this code was adapted slightly and used for some demonstrations. A screenshot of a demonstration can be seen in Figure 6.2.

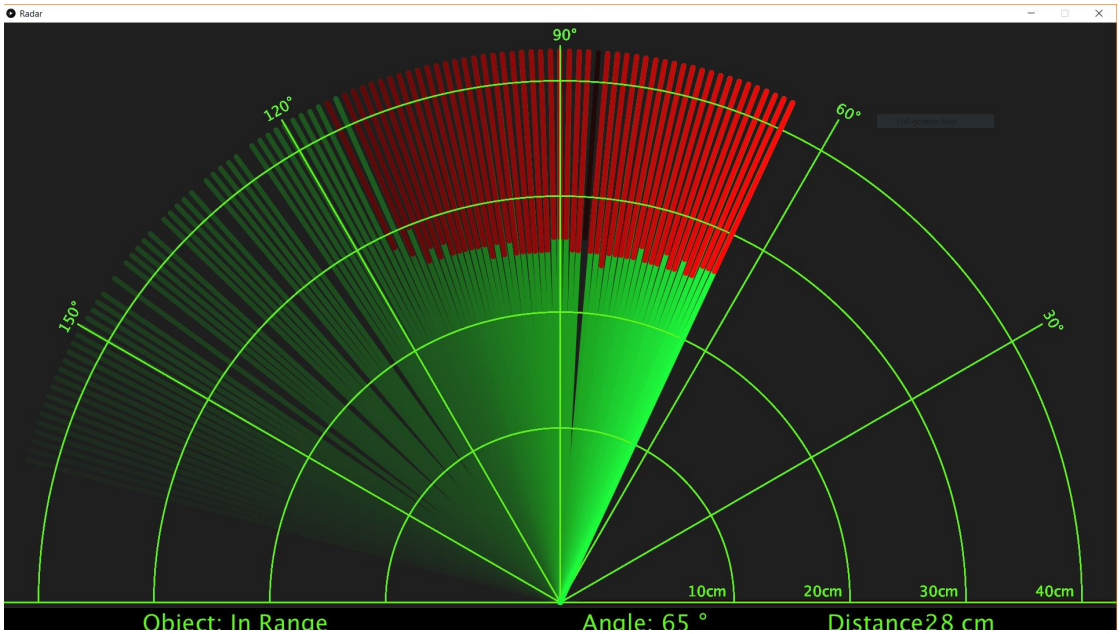


Figure 6.2: *A screenshot of the software used to display the radar measurements on a computer screen, adapted from Nedelkovski [14].*

During the test seen in Figure 6.2, a small object was placed 30 cm in front of the Zebro. The red bars on the screenshot symbolize a detected object. As can be seen, the object is detected in front of the Zebro at a distance of 28 cm.

6.3.3 Integrating code

The visual radar software descibed in subsection 6.3.2 proved effective for demonstrating the radar principle, but it was not very useful for the final product. Firstly, it used the pulseIn() function so the code was blocking, as explained in subsection 6.3.1. Secondly, a measurement was taken every degree between 15 and 165 degrees. This meant there were 150 measurements for each rotation, which was unnecessary for our purposes so slowed down the rotation speed needlessly.

For integration into the final product, a different software architecture was used in the distanceClass. A visual overview of the distanceClass can be seen in Figure 6.3.

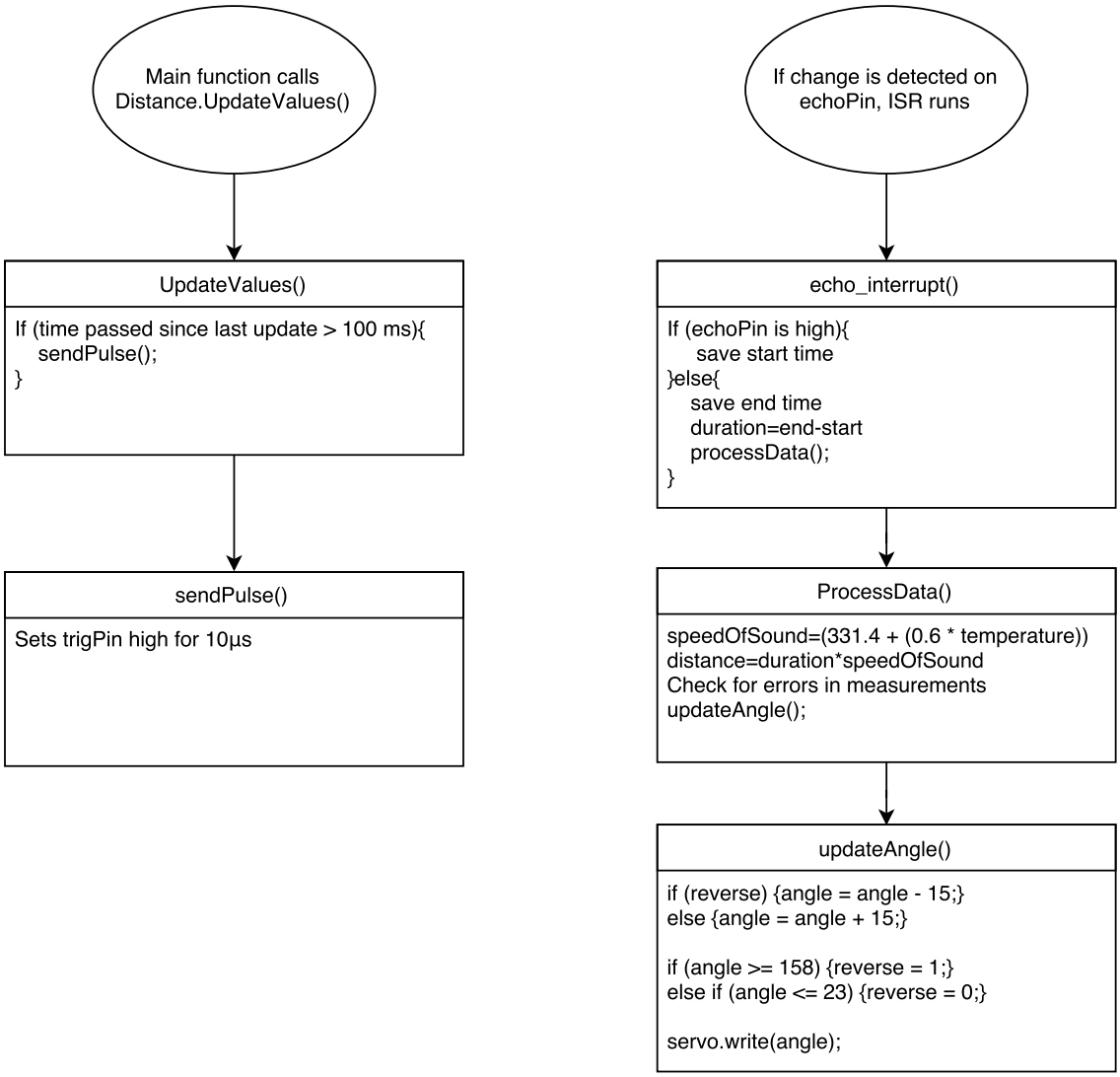


Figure 6.3: *Overview of distance class*

Note that this figure contains pseudocode and only the most important lines of code are shown. It only illustrates the main functionality of the class, not the exact implementation in the code.

To be able to determine the distance to an object based on the time travelled by the US pulses, the speed of sound is needed, which is dependent on temperature. As the module also contains temperature sensors (implemented by a different subgroup), the temperature measured by these sensors is used in the calculation of the speed of sound. This makes the estimation more accurate than if the outside temperature were not taken into account.

The updateAngle() function is responsible for driving the servo. The idea is that the servo rotates the US sensor back and forth from left to right. As can be seen in Figure 6.3, the servo rotates between 23 degrees and 158 degrees in steps of 15 degrees. Once a measurement has finished, the servo rotates 15 degrees further and stops there for the next measurement. As the servo turns 545 degrees per second [15], it takes about 28 ms for it to turn 15 degrees, so this is the minimum amount of time that must be waited before starting a new measurement.

There are 10 measurements per rotation from left to right (or vice-versa). It would have been possible to

have a greater resolution with more measurements per rotation, but this would mean it would take longer to complete a rotation. If this were the case and a sudden obstacle would appear in front of the Zebro, it may detect it too late. Therefore the trade-off between resolution and speed had to be made and it was decided that 10 measurements was sufficient. This is also convenient because each measurement can be mapped to a single LED on the LED ring, as will be discussed in section 6.4.

6.3.4 Error Correction

The measurements from the US sensor will not always be reliable. This may be due to occasional interference from, for example, the US pulses of other nearby Zebros. To reduce the amount of false positive or false negative detections of an obstacle and to increase the reliability, some form of error correction needed to be implemented. Many solutions were considered, including implementing a Kalman filter. However, it was decided that this would be more complex than necessary and would also introduce computational complexity, slowing down processing times on the microprocessor.

As a simpler solution, it was first decided to compare the current measurement with the previous measurement. If the difference between these two would be greater than a a set tolerance, the servo would remain in its current position and a new measurement would be done. This worked reasonably well, but it meant that the servo sometimes (whenever there was a large difference between the current and previous measurement) stopped for longer than other times. This caused the servo to have a sort of twitching effect, which looked strange and unnatural. Therefore, another method was implemented.

The new method was to take 3 measurements per step of 15 degrees. If all 3 measurements are similar, then they are all assumed to be reliable and the average of all 3 is taken. If 2 are similar and the other is very different, then only the average of those first 2 is taken. If all 3 are significantly different, it is assumed that none of them is reliable and '0 cm' is saved for this measurement to indicate the lack of a reliable measurement. This system proved to work better than the previous

During the testing of the US sensor, described in 4.3.3, it was evident that the results became significantly less reliable at long distances. For this reason, a maximum distance parameter was introduced. This is currently set at 3m but can easily be adjusted if necessary. If the distance measured is greater than the maximum distance, it will be changed to the maximum distance to indicate that this is outside of the range of the sensor.

6.4 LED Ring

One of the general tasks of Zebros is that they can autonomously navigate on pavements and roads where humans are also present. For this to go well, and also simply for debug purposes, it is useful to have some communication with nearby humans as to what the Zebro is seeing. This way, people can easily see whether or not the Zebro has detected an obstacle or a nearby danger. To do this, it was decided to use a LED ring mounted on the top of the module. A Neopixel ring with 24 LEDs was used for this purpose.

The front 10 LEDs were used to display the 10 distance measurements, where the color ranges from red for an object that's very close to green meaning no object within the given range. 8 of the LEDs at the rear of the ring were used to display the Error_Byte (discussed in chapter 5), of which mainly the left cliff and right cliff indications are relevant for this subgroup.

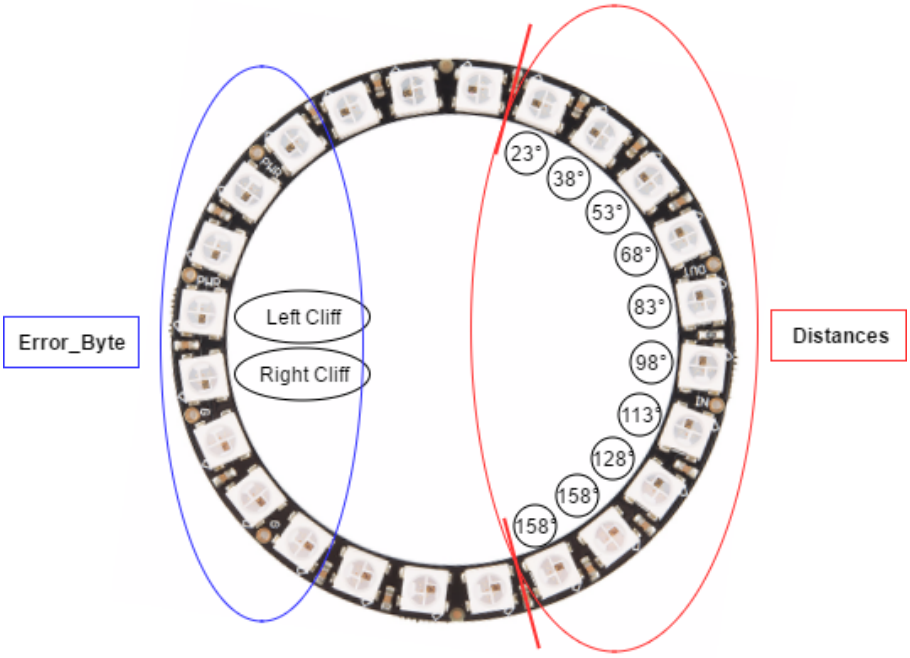


Figure 6.4: *LED ring showing the information displayed on each LED. Image source: SparkFun Electronics [16]*

For the Arduino implementation of the LED ring, the FastLED library [17] was used. This library is convenient for our purposes because the color can be controlled via CHSV instead of RGB like most other libraries. As can be seen in Figure 6.5, red has a 'hue' of 0 and green has a 'hue' of 96. This makes the calculation for the appropriate hue simple: $hue = 96 * distance/maxDistance$ where distance is the measured distance and maxDistance is a parameter which defines the minimum distance at which an object is considered to be outside the sensor's range. Because of this, the LEDs will be range from red at 0 cm to green at the maximum distance, as was required.

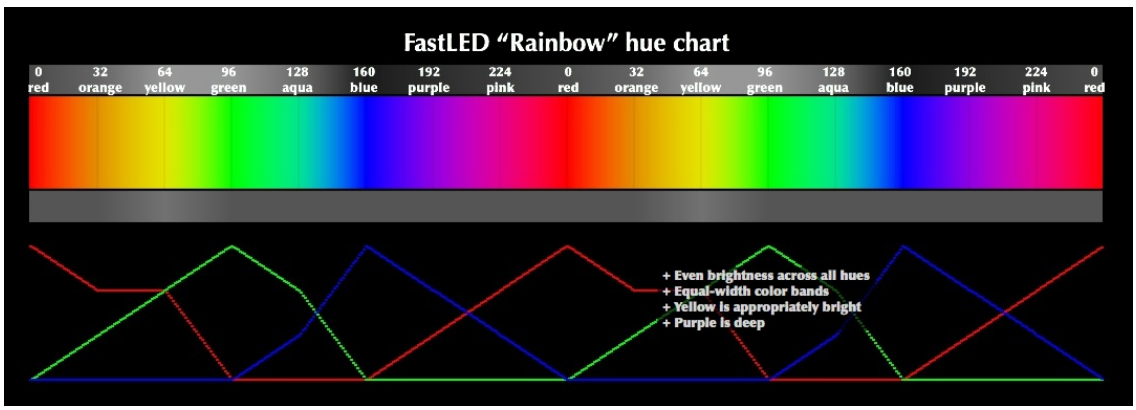


Figure 6.5: *CHSV hue chart for FastLED library. Image source: Garcia [17]*

6.5 Servo/LED Interrupt Issue

During small tests, it was found that when the LED-ring and the servo are used together, problems arise. The servo seemed to 'twitch' during rotating, making random turns at times. This was considered an undesired effect, however, it was not immediately clear what caused this. As it turned out, after some research

was done, this was a known issue. [18]

The problem is caused by the fact that the protocol that controls LED-ring uses a very tight timing. To prevent mistakes, the LED-ring controller temporarily disables all interrupts, including those used for the timer, which are in turn used in the 'Servo.h' library to control the servo motors. In short, the problem lies in two conflicting protocols that both use interrupts and of which one disables the interrupts used by the other.

These things are mostly controlled 'under the hood' of the Arduino libraries and are therefore not easy to change without shaking up other functionalities of the code. To cope with this problem, a library created by Adafruit was used. It changes the way the servo motors are handled, and uses the microprocessor's AVR peripherals to make sure that servos are not affected when the LED-ring disables all interrupts. When this code was implemented, the problem was indeed solved.

6.6 Testing & Practical Issues

To test the system, several setups were considered, including mostly outside situations like sidewalks, bikepaths, grass, hills and slopes.

However, the system was first tested numerous times on the floor in the practical hall. During these tests, some issues that were not anticipated during design were discovered.

Apparently, when not placed on a smooth surface, the ultrasonic distance sensor suffers from reflections from the ground. This gave the system the idea that an obstacle was close by, while the robot was actually placed in open space. It was found that tilting the sensor backwards reduced this effect. Therefore, as a solution, a new enclosure for the sensor was designed that was placed under an angle (see subsubsection 7.2.5.2).

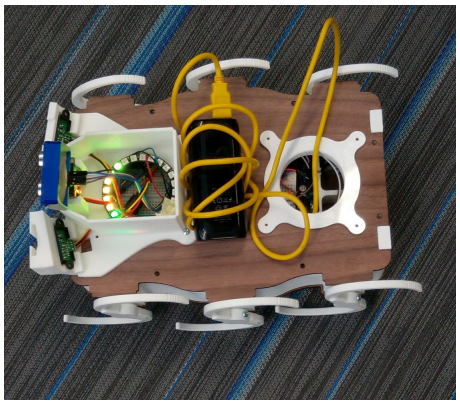
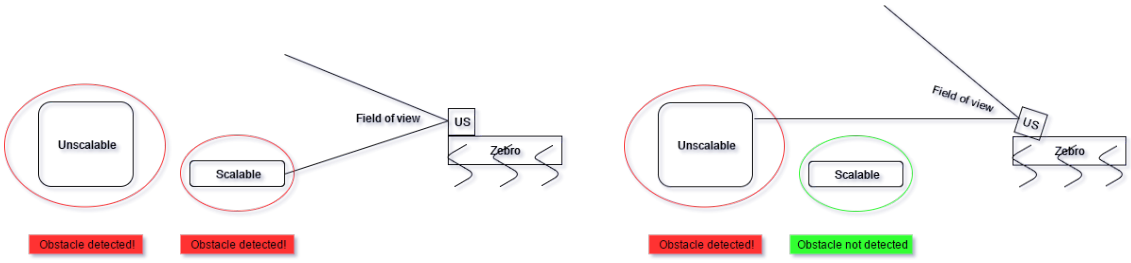


Figure 6.6: *The observation module attached to the front of the Zebro body during a test. The LED-ring (placed inside the enclosure for testing) can be seen displaying the relative distance. Here the robot runs off a USB battery pack.*

Placing the US sensor under an angle has another benefit: it helps in determining the scalability of objects. By rotating the US sensor backwards, objects smaller than itself will no longer be detected. This means that small scalable objects will not be detected, which is not necessary as the Zebro can just climb over these. This principle is shown in Figure 6.7. This adresses requirement 3; the ability to discern between scalable and unscalable objects, to some extent. However, it is not foolproof. If an object is taller than the distance Zebro can climb over (about half its height), but smaller than the height of the US sensor, this object will not be detected. However, this is one of the reasons why there is also a second obstacle detection system being developed by another subgroup [2].



(a) *Both scalable and unscalable objects are detected*

(b) *When the US sensor is placed at an angle, only scalable objects are detected*

Figure 6.7: *The angle at which the US sensor is placed helps determine scalability*

It was also found that, as a result of the way the code was written, the speed with which the servo motor rotated largely depended on the processing time of a measurement. This processing time included the time it took to send out a pulse and wait for its reflection. Consequentially, the servo motor rotated faster when an obstacle was near by and the distance was small. As this was considered a negative side-effect, it was decided that the transmit/receive time should not influence other parts of the code. Therefore, the structure of the code was changed to the non-blocking implementation using an interrupt, as described in subsection 6.3.1.

Another possible issue that was anticipated is the idea of interference when two Zebros both send out an ultrasonic pulse. As this could possibly hamper the performance of the robot's obstacle detection system in a swarm severely, it was tested with two HC-SR04 sensors aimed at each other. However, no severe negative influence that would be a result of interference was detected.

Chapter 7

Module Enclosure

In this chapter the design and development of a plastic enclosure for the whole module will be discussed. As the enclosure does not have any strict requirements other than to adhere to the existing Zebro physical interface, there is quite some freedom in what the final product will look like and how it is implemented. However, since the existing Zebro bodies and accessories were drawn in SolidWorks software and 3D-printed, it makes sense, as part of the Zebro team, to go along the same route.
This chapter lays the focus on the development of the enclosure rather than its exact specifications. However, more technical drawings that show the precise dimension of each model can be found in Appendix B.

7.1 Requirements

7.1.1 Considerations from within the project group

As mentioned before, the official module requirements (see chapter 2) do not contain any specifics about the enclosure other than that it should be (physically) compatible with the Zebros. However, within the project group, there has to be discussed in what way all the submodules will be incorporated in the enclosure. The enclosure has to fit on top of the Zebro body, so there is only so much space to put sensors or circuit boards. Nevertheless, it would be beneficial to all if the enclosure offers all subgroups the desired setup for the sensors and/or circuit boards they want to use.

The requirements for the enclosure are mostly communicated informally within the group. However, as they are constraints on the freedom the designer has while making a first design, they predetermine to a large extent what the final product will look like. In Table 7.1 the requirements for each sensor can be seen, including what kind of interface with the body they need.

7.1.2 General considerations

This section is about some general things that have to be kept in mind during the design of the enclosure.
First of all, the enclosure is to be 3D-printed, which means during the design steep (upwards) angles should be avoided if possible, as well as very small or very thin parts. These sections can be printed, but only by adding so-called 'supports'. These supports are light structures of printed plastic that can be removed easily. However, the higher the part that is sticking out, the more supports are needed and the more plastic is wasted.

Furthermore, even when parts have to fit exactly, a safety margin of a few millimeters has to be taken into account, as the final product may have edges that are printed slightly smaller or larger than was designed. Finally, also factors from outside the module should be kept in mind. The enclosure has to fit on the Zebro body but also not obstruct parts on the body that were designed for something else. For example, the Zebro body has two holes for two modules, so the enclosure of one module should not take space away or obstruct

the other module in any way. In the same way the enclosure has to be designed in such a way that other parts do not obstruct this module. For that, a schematic of the Zebro body is provided by the Zebro team, as well as already printed and assembled Zebro casts to fit the enclosure on (see figure B.1 in the Appendix).

Submodule	Sensor / Device	Requirements
Obstacle & Cliff Detection	IR Distance Sensor	Vertical angle of 60 degrees; Horizontal angle of 10 degrees; One on both sides of the robot; In the front of the robot; Needs a mount with screwholes;
	Ultrasonic Distance Sensor	Mounted on a servomotor; Being able to rotate; Somewhat in the front of the robot; Needs a mount with screwholes;
Range Finding & Image Processing	PI Camera	In the very front of the robot; Needs screwholes;
	Laser	Horizontal angle of 80 degrees; Needs screwholes;
Sensors & Interfacing	Temperature(1), light, humidity	Need to be exposed to outside air/light;
	Temperature(2), acceleration, gyroscope	No specific requirement;
	LED-ring	Needs to be visible on the outside;

Table 7.1: *Table containing the requirements for the enclosure for each device. The Temperature(1) indicates the temperature sensor used to measure the outdoor temperature, Temperature(2) for the internal temperature.*

7.2 Designing

This section is about the design process of the plastic enclosure. For creating the 3D-model of the enclosure, SolidWorks software was used. This software is provided by the TU Delft. SolidWorks can save the 3D-model as an .STL file, which can than be interpreted by 3D-printing software, such as Gembird Cura, the software that was used here.

7.2.1 Bottom plate

For the bottom of the enclosure, it was needed to study the Deci Zebros technical drawings to match the screwholes that were provided for the module. The distance between the screwholes and the edges of the Zebro body's top provides a general guideline for the final dimensions of the enclosure. The Deci Zebros body is designed with the idea that the modules are attached and can be removed without the need for screws. For this, a twist-lock interface was designed. However, for several reasons, this interface was discarded:
First, it was found that the Zebro body featured clips that hold it together that stick out on the top plate of the body. Since several of the sensors that are used need to be in the front of the body, these clips need to be covered by the enclosure, making it impractical to attach the module by turning it.
Second, since the enclosure will be printed from the bottom plate up, adding thin parts that stick out on the bottom will make the model hard to 3D print. Printing would be possible only by adding supports everywhere else, which would be a waste of plastic.
Third, the module can still be attached firmly to the Zebro casing with screws. Screwholes of 3 mm in diameter that match the ones already on the Zebro body were added to the bottom of the enclosure. This makes for a durable connection, leaving less space for the module to move around than the twist-lock would.
To cover up the clips that were mentioned earlier, a small space of about 3 mm was left open on the bottom

side of the enclosure. To be able to print this, small sections of supports were added in the 3D print software.

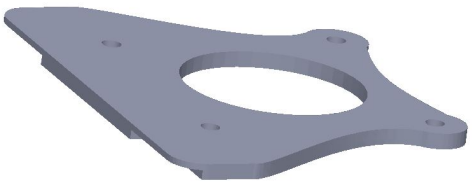


Figure 7.1: *Bottom plate seen from the side, with on the left the front of the plate.*

In figure 7.1, the bottom plate that was first designed can be seen. Under the bottom plate the spaces that were cut out for the clips on the front of the Zebro body are visible. Without these, the enclosure would not fit properly on the front of the Zebro.

7.2.2 Mounting the IR sensors

For the SHARP distance sensors that would be used for the cliff detection system, mounts were added under a vertical angle of 60 degrees and a horizontal angle of 10 degrees. This mount needed a space and screwholes that matched the sensor's dimensions plus some extra space to lead away the wires coming from the sensor. It was decided that the mount would enclose the sensor as a whole, as loose brackets with screwholes in it would be less structurally integer. Also, this needed to be designed twice, on each side of the enclosure.
First, the dimensions of the sensors were measured using a caliper. Then, two new structures with holes were added to the bottom plate, with one large cavity for the sensor and its wires and two screwholes to secure the sensor in its place. To keep a strong structure, the mounts and their connections were designed to be thick pieces.
In figure 7.2 the 3D model of one of the mounts is shown. The cavity at the bottom of the structure is for the connector and the wires of the sensor.

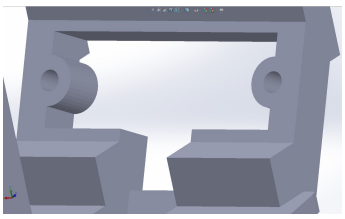
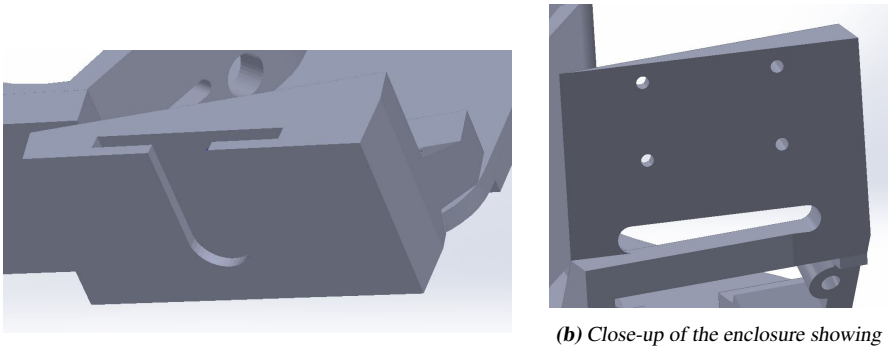


Figure 7.2: *Close-up of the enclosure showing the mount for the SHARP distance sensor.*

7.2.3 Mounting the PI camera & laser

Since the camera and its laser and the cliff sensors have somewhat the same requirements for the enclosure, the mounts are placed on top of each other. The infrared distance sensors are aimed downwards, so it makes sense to keep them located closest to the ground. However, this choice means the IR sensor mounts (see subsection 7.2.2) have to be adapted to support the camera and the laser.
For the PI camera, initially an enclosure was designed that would allow for the camera to be slided in.

However, as it proved difficult to design the right fit and working away the flat cable, it was decided that a simpler structure would be designed. A simple flat surface with screwholes and a cavity to lead away the cable was added.



(a) *Close-up of the enclosure showing the initial design for the camera mount. The cable sticks out on top while the PI camera is mounted upside down.*

(b) *Close-up of the enclosure showing the final mount for the PI camera. The flatcable is lead away through the whole below the mount*

Figure 7.3: *Two close-ups of different enclosure design iterations, showing two ways to implement the PI camera mount.*

In figure 7.3, two implementations of the PI camera mount can be seen. Although the first mount allows for screw-less attachment, the camera would have to be placed upside down in the holder, causing two issues: First, the camera image would be upside down. This can be solved in the image processing. Second, the camera flatcable would stick out significantly, in such a way that it would be visible to the ultrasonic distance sensor, causing interference. Therefore, a simpler mount was chosen, that did not have this problems. As the IR sensor mount is already placed under a horizontal angle of 10 degrees and the PI camera and laser are placed on top, the camera and laser mount need to have an angle themselves. The PI camera mount is placed under -10 degree angle to counter that of the IR sensor mount. The laser needs an angle of 80 degrees and is thus placed under an angle of 70 degrees relative to the IR sensor mount.

7.2.4 Mounting the PCB's

As the Raspberry PI module used for the image processing section and the custom PCB (see chapter 8) both measure 65 mm in length (the PCB is square), this should be the minimum open space in the enclosure. The wires of the sensors that are in the front of the body are lead back into the main space where all the electronics are stored. In the middle of this large space is a hole that matches the dimensions of the hole already present on the Zebro body. Through this hole all the wires from the PCB are connected with the main Zebro system. This section also features screw holes and mounts for the PCB and on top, the Raspberry PI.

7.2.5 Mounting the Ultrasonic sensor & the Servomotor

7.2.5.1 Servomotor Mount

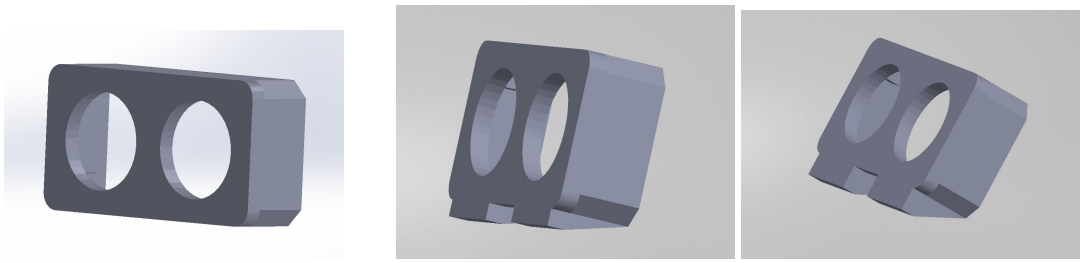
It was considered most practical to place the servomotor (which has the ultrasonic sensor attached to it) in the front of the enclosure. Since there was enough space left, it was placed inside the large space of the enclosure. It only needed two supports to set it at the right height. It was chosen that the upper part of the servomotor should stick out of the enclosure, as this height would prevent the ultrasonic sensor from detecting reflections from the frontal sensor mounts.

To achieve this, the servomotor was measured with a caliber and supports were placed inside the enclosure to hold the servomotor.

7.2.5.2 Ultrasonic Sensor Mount

Initially, during testing, the ultrasonic sensor was mounted on top of the servomotor using a cable tie. This proved quite effective, but ultimately, it was chosen to design a plastic casing for the sensor.
First, the large 3D model database *Thingiverse.com* was searched for an existing model of such a casing. A casing was found there and it was printed. As it turnt out, it was not the right fitting, the Parallax Ping sensor (which is not used) did fit in it, however. Therefore, it was chosen to design a custom casing anyway. This proved harder than anticipated because a millimeter offset on the HC-SR04 (the ultrasonic sensor) PCB was enough to obstruct the sensor from fitting in the casing. Moreover, several versions of the sensor existed, so one type ended up fitting while another was too large. Ultimately, however, this process ended up in a fitting casing.

For this mount, a small box was designed with holes for the transmitter and receiver of the sensor and small holes for screws on the inside and on the outside for mounting on the servomotor.
Later on, during testing with an assembled Zebro body cast, it was found that on a rough floor, the ultrasonic sensor detected many reflections, causing it to see obstacles that were not there. Therefore, a new version of the casing was made, this time the bottom of the sensor was placed under an angle, so that, once mounted, the sensor would be aimed slightly upwards. Two versions were made, one under an angle of 25 degrees and one for 15 degrees. The 15 degrees version proved sufficient.



(a) *Original casing of the ultrasonic sensor, no angle was used.*

(b) *Model of the casing tilted at 15 degrees.*

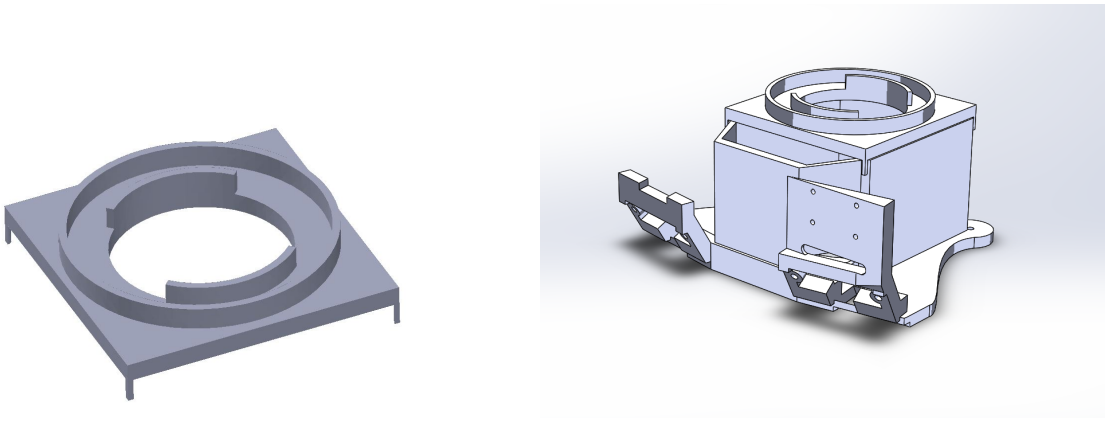
(c) *Model of the casing tilted at 25 degrees.*

Figure 7.4: *Figure showing the different iterations of the design of the mount for the HC-SR04 sensor. The version tilted at 25 degrees was chosen.*

In figure 7.4, the original casing of the HC-SR04 and the two altered versions can be seen.

7.2.6 Mounting the LED-ring

The LED-ring is used to show the Zebros current status, as well as measurement results for the distance and cliff detection system. To make it easy read out, it should be placed on top of the module. Therefore, a 'lid' was designed to partly close the open space of the enclosure in order to create space for the LED-ring. However, as closing off a large part of the enclosure creates issues with the freedom of rotation of the ultrasonic sensor (most notably, the cables connected to this sensor), a lid was designed with a hole in it, so that this part could still move freely. Further, it features an inner and an outer edge to hold the LED-ring in place. Lastly, it features small 'legs' that were designed to fit in the round corners of the main enclosure.
The LED-ring mount can be seen in figure 7.5a.



(a) 3D-model of the lid, showing the space where the LED-ring fits in.

(b) 3D-model of the entire enclosure with the lid for the LED-ring fitted on top.

Figure 7.5: *Figure showing the individual mount for the LED-ring and the whole enclosure with the lid put together.*

As can be seen in figure 7.5, the lid with the mount for the LED-ring, fits on top of the main enclosure. The hole in the lid is needed for cabling, in particular to guarantee movement for the ultrasonic sensor, but also to lead away the cables for the LED-ring itself.

7.2.7 Mounting the external temperature sensor, the light sensor & the humidity sensor

Since these sensors are very small, cutting a small hole in the casing and attaching them with a screw so that they are exposed to the outside air and light suffices for these sensors. No holes are being designed for this in the enclosure, since it is found that leaving very small screwholes are not always properly printed. Also, it is assumed that there will be sufficient space for these sensors in the enclosure once everything is put together.

7.2.8 Mounting the internal temperature sensor, the accelerometer and gyroscope

These sensors are attached to the inside of the casing, preferably close to the circuit boards, as that is where they are supposed to do measurements

Chapter 8

PCB

8.1 Problem Description

During most of the prototyping, an Arduino Mega2560 was used. This was very convenient as it is easy to program and communicate with over USB. However, these devices are not cheap and contain more pins and other connections like USB which will eventually not be needed when it is implemented in the Zebros. Because of this, they are also bigger than needed for our purposes. To solve this problem, it was decided to make a custom PCB with a chip loaded with an Arduino bootloader. Even though this was not explicitly part of the requirements, there was some time available to do this and it was decided that it would be beneficial. Loading it with an Arduino bootloader would mean that the code would not need significant adjustments. Designing the PCB ourselves would mean that its form factor could be adjusted to fit our purposes, and only the necessary components and pins would be on the circuit board.

The main requirements of the PCB are:

- Be small enough to fit in the module easily
- Have connections for all peripherals
- Connect peripherals to the power supply and to an appropriate microprocessor pin
- Have the necessary hardware and connections for the microprocessor to operate and to be programmed
- Be able to efficiently run the software described in chapter 6
- Be as cheap as possible to produce (preferably only 2 layers)

The PCB needs to connect the external devices to the power circuitry and to the microcontroller. The connections that the PCB will need to provide are:

- ISP
- Buzzer
- ZebroBus
- Temperature 1,2,3,4
- IR-R, IR-L
- LED Ring
- Ultrasonic Sensor
- Gyro
- Breakout
- Laser
- Servo
- Power
- Power 2
- Pi
- Extra IO1
- Extra IO2
- Extra Analog
- Extra UART

The ISP will be used to program the microcontroller. The ZebroBus will require a small transistor circuit as defined in the ZebroBus interface document [19]. The breakout board will contain the Lightsensor and humiditysensor, as these need to be in contact with outside air and thus cannot be on the PCB directly. The

laser and Pi signals need an amplifier and voltage divider respectively as the Pi works on 3.3V but most other components (including the laser, which is to be controlled by the Pi) work on 5V. There are also some extra pins in case it is later decided to, for example, add extra sensors to the module. If this is the case, the hardware will likely not need to be changed.

On top of these connections, the PCB will also need debug LEDs, ESD protection, a reset button, decoupling capacitors, an oscillator circuit, and I2C pull-up resistors.

8.2 Possible Microcontrollers

The PCB will need to contain all the connections to the microcontroller. The devices that need to be connected to the microprocessor via the PCB are shown in Table 8.1.

Table 8.1: *Devices to be connected to the microprocessor*

Item	**Type of Connection**
Thermometers	4 Analog
IR Cliff Sensors	2 Analog
Ultrasonic Distance Sensor	1 Digital + 1 Interrupt
Gyro+Accelerometer	I2C (Soft) + 1 Interrupt
LED Ring	1 Digital
Raspberry Pi	UART
Light Sensor	1 Analog
Humidity Sensor	I2C (Soft)
Speaker	1 Digital
ZebroBus	I2C + 1 Interrupt
Servo	1 Digital PWM

With the requirements and the necessary connections mentioned in Table 8.1 in mind, an appropriate microprocessor had to be chosen. As the Arduino bootloader had to be loaded onto the microprocessor, the choice was limited to microprocessors used in Arduino devices. The microprocessors of some popular Arduino devices were compared, some of which can be seen in Table 8.2. The parameters which did not meet the required parameter are marked in red for each microcontroller.

Table 8.2: *Comparison of possible microcontrollers [20] [21] [22]*

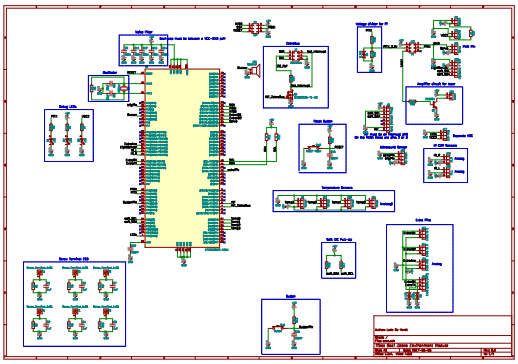
	Required	**ATmega328P**	**ATSAMD21G18**	**ATmega2560**
Arduino Used		Uno	Zero	Mega2560
Working Voltage (V)	5	1.8-5.5	1.62-3.63	4.5-5.5
Program Memory (KB)	35	32	256	256
CPU Speed (MIPS)		20		16
I/O Pins	20	23	38	86
I2C Busses	2	1	6 (UART or I2C)	1
UART Pin Sets	1	1	6 (UART or I2C)	4
ADC Channels	7	6	14	16

The ATmega 328P has too little memory, only 1 I2C bus, and only 6 ADC channels so this microprocessor is not an option. The ATSAMD21G18 has 6 serial communication interfaces which can be configured as either UART or I2C so this is very useful. However, its working voltage is too low as it had already been decided that the working voltage of all components would be 5V. The ATmega2560 is sufficient on all parameters except that it has only one I2C bus. This is problematic as the microcontroller should be slave for the ZebroBus and master for its own sensors, which requires two separate I2C busses. None of

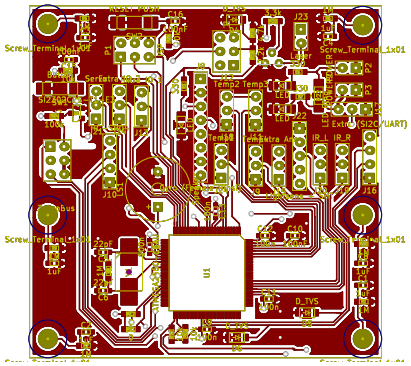
these microcontrollers are ideal, but it was decided to go with the ATmega2560 because it is possible to use regular I/O pins for a software implementation of I2C.

8.3 Designing the PCB

To design the PCB, it was recommended by the supervisor to use KiCad. First the schematic was made, then the appropriate footprints were added and the routing was done. After feedback from the supervisor, certain elements were adjusted. Note that at this point in time, the design of the PCB is not yet completely finished and still needs some minor adjustments.



(a) Schematic of the PCB



(b) Front copper layer of PCB

Figure 8.1: *Schematic and front copper of PCB.*
Note that this figure only serves to give an impression of the PCB design. A larger version of these figures can be found in the appendix.

The entire PCB is 65 mm X 65 mm. This means it will easily fit into the module. The Raspberry Pi will be mounted on the bottom half of Figure 8.1b to save space; because of this, all the connectors are at the top half so they are still accessible. In hindsight, it may have been useful to make the PCB bigger or place some connectors on the bottom because there is some lack of space in this version.

In a PCB like this, one concern is electrostatic discharge caused by someone touching the PCB, which could damage the microcontroller. To prevent this, some ESD protection was used. The ESD protection consists of a capacitor in parallel with a resistor connected between the screw holes and ground. There are also some transient voltage suppression diodes on certain pins which will be touched often, such as the pins used for debugging.

To filter noise on the ground and VCC inputs of the microcontroller, decoupling capacitors were placed between each pair.

The laser operates on 5V, but needs to be controlled by the Raspberry Pi. To achieve this, there is a laser signal as input to the PCB which goes to the base of a transistor via a resistor. This laser signal is then amplified by the transistor to 5V, which is put on the laser output pin.

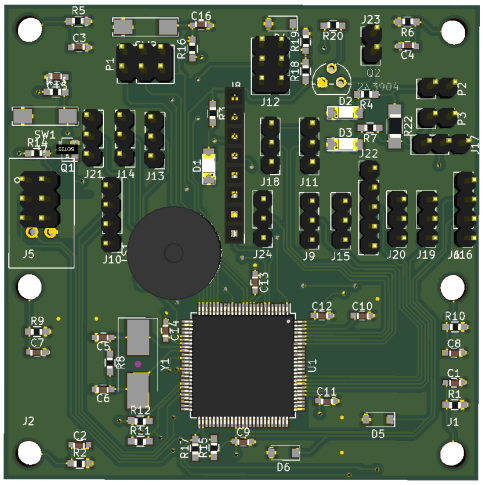
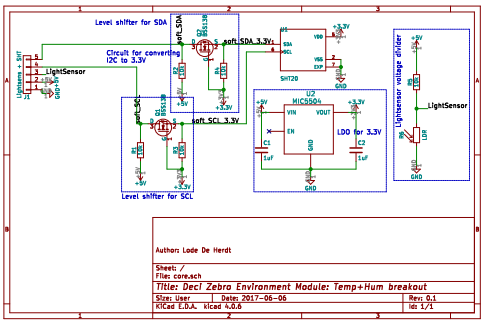


Figure 8.2: *A 3D render of the PCB*

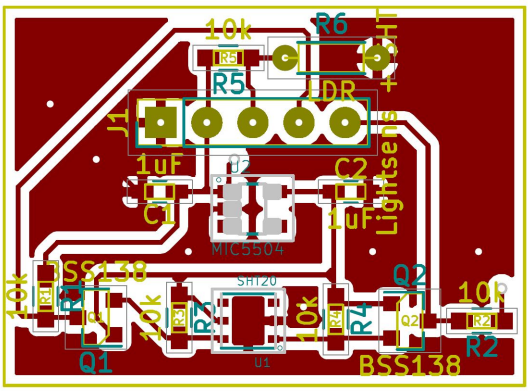
8.4 Breakout Board

As mentioned in section 8.1, the lightsensor and humiditysensor cannot be inside the enclosure as they need to be in contact with outside air and light. Therefore, it was decided that these a second, smaller PCB would be created to act as breakout board for these sensors.

These sensors also needed some other components. An LDO was required in order to convert the normal 5V to 3.3V for the SHT20 (humidity sensor). Similarly, a level shifter was needed to also convert the I2C lines (which are standard at 5V in our case) to 3.3V and vice-versa.



(a) *Schematic of the breakout board*



(b) *Front copper layer of breakout board*

Figure 8.3: *Schematic and front copper of breakout board.*
Note that this figure only serves to give an impression of the PCB design. A larger version of these figures can be found in the appendix.

8.5 Manufacturing of PCBs

Both the large PCB containing the microprocessor, and the smaller PCB with the light- and humiditysensor, need to be manufactured. In discussion with the supervisors, it was decided that it would be fastest and cheapest to make these PCBs on the milling machine present in the faculty for the prototype. Later on, when higher quantities of these PCBs are needed, it may be beneficial to use a PCB manufacturer. At this point in time, only a simple breakout board for the humiditysensor (different to the one described above) has been milled with success. In the near future, the two PCBs described above will be finalized, milled, the appropriate components will be soldered, the microprocessor programmed, and everything will be tested. Once the design is complete, a list of components will be generated to provide an overview of what should be soldered where on the PCBs.

Chapter 9

Conclusion

9.1 Conclusion

For this thesis, a submodule has been designed that does a selection of the tasks of the complete module. In combination with the other submodules (see [2] and [3]), the module fulfills the requirements laid out in chapter 2.

The submodule system developed in this thesis suffices the requirements EM-1 and EM-2 in section 2.2 as it is able to detect obstacles and get a rudimentary sense of its surroundings, it can detect cliffs by detecting an increase in measured distance to the ground. Also, hardware and software were designed and developed to make it work. Not a specific requirement, but according to the stated criteria, a PCB was designed to integrate the electronics and an enclosure to make it easily attachable and replaceable.
Requirement EM-3 proved more difficult to implement with the ultrasonic distance sensor; it is only partially working as is described in section 6.6. Fortunately the camera system developed by a different subgroup is better at detecting scalability [2].
Next to developing a module, several requirements with respect to the execution and documentation were set, also in chapter 2, namely BT-1, BT-2, BT-3 and BT-4. During this project, BT-2, BT-3 and BT-4 were achieved. BT-1, developing a test bench and doing a run has also been done to a certain extent (the test results are found in section 6.6), but it should be noted that at this time none of the Zebros can walk due to circumstances outside of our control. Thus from this report it can be concluded that the module works, but it was never tested on an operational Zebro.

Nevertheless, with the results of this project accompanied by documentation, a robust foundation has been laid out for anyone to improve upon.

9.2 Recommendations & Future Work

From the conclusion, it should be clear that testing the module on a walking Zebro would be a logical next step for anyone continuing with what has been developed thus far. As of now, it is not exactly clear how the modules' sensors and devices will react to the specific movements of the Zebro.

Another idea that could be worked out in the future is to let the ultrasonic sensor do a height and/or steepness estimation of an object in some way, as to see whether it's scalable or not. This could be done by, for example, rotating the US sensor vertically as well as horizontally. Also, some rudimentary motion checking could be done with this sensor if more memory is added to the module, by keeping previous measurements in the memory and calculating the speed at which obstacles are approaching the Zebro. One could also try to let the Sharp sensors discern between slopes downwards and upwards, instead of only looking for cliffs. Since the sensors are already looking at the distance between the module and the ground, a smaller

distance could mean a slope upwards and a larger distance a slope downwards or, in the extreme case, a cliff.

Lastly, the behavior of the Zebro is something that needs a lot of developing. The module discussed in this document focuses on gathering information about the Zebro's environment, but it does not do much in the sense of processing that information, except for some error-correction. What decision the Zebro then makes based on the information received by the sensors is a very important topic which still needs a lot of work.

Appendix A

PCB Files

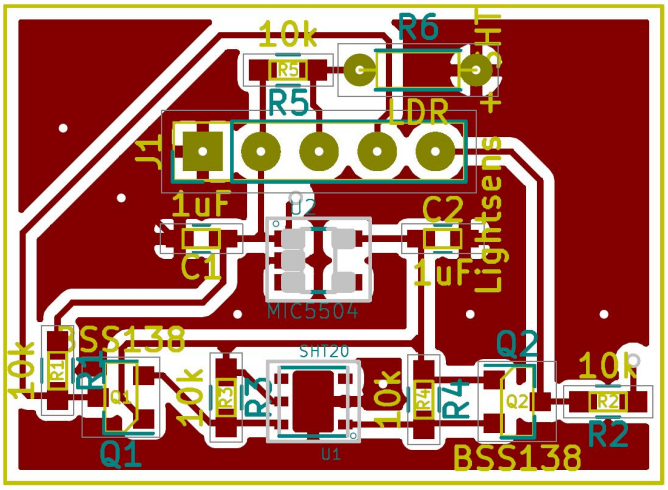


Figure A.1: *Front copper of breakout PCB*

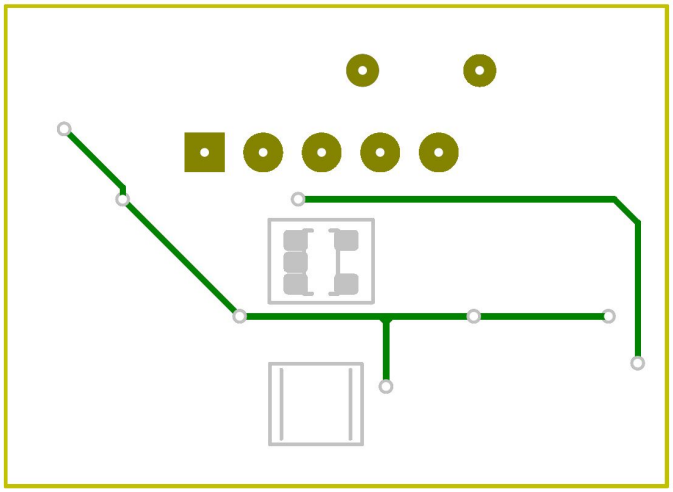


Figure A.2: *Back copper of breakout PCB*

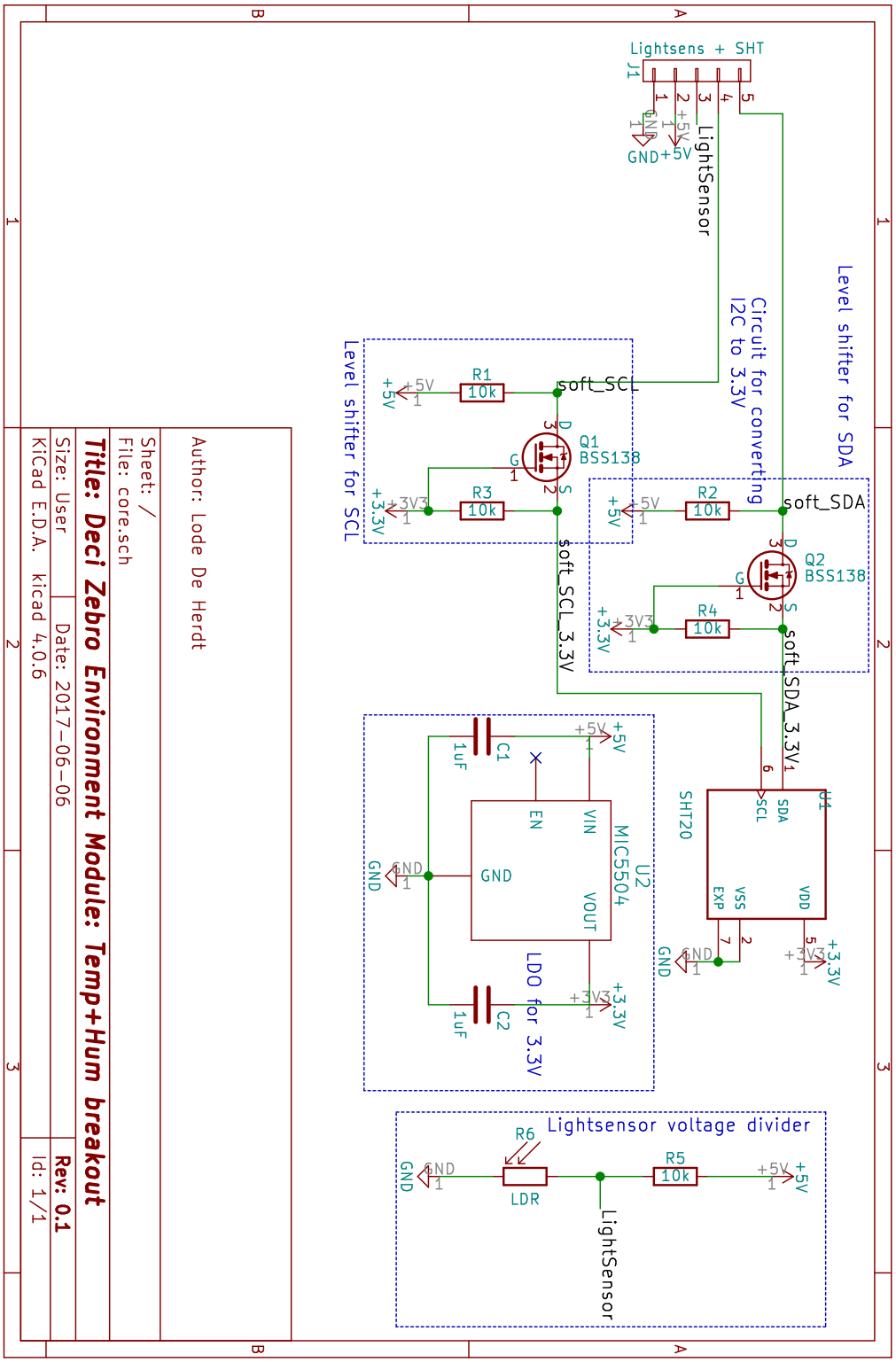


Figure A.3: Schematic of breakout PCB

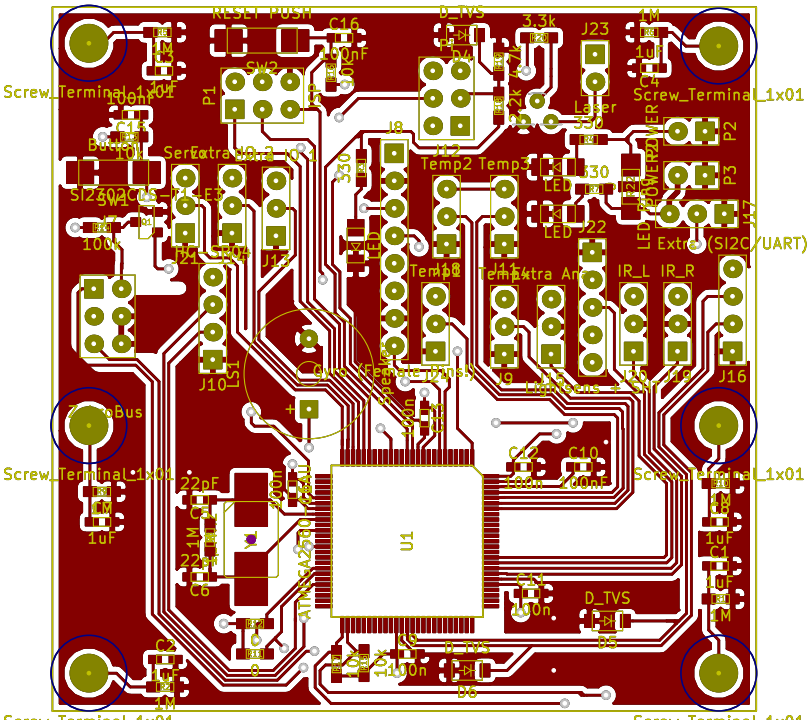


Figure A.4: Front copper of PCB

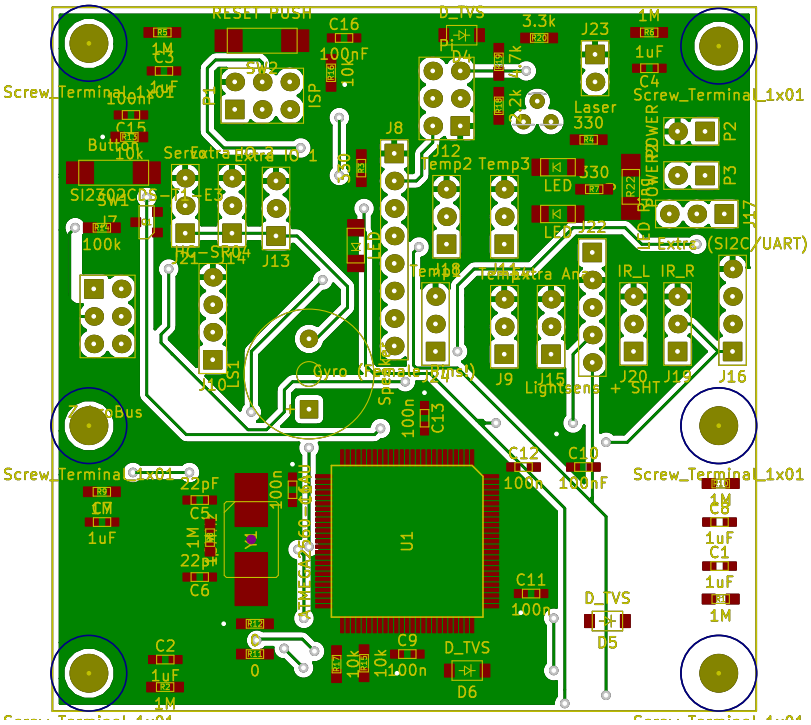


Figure A.5: Back copper of PCB

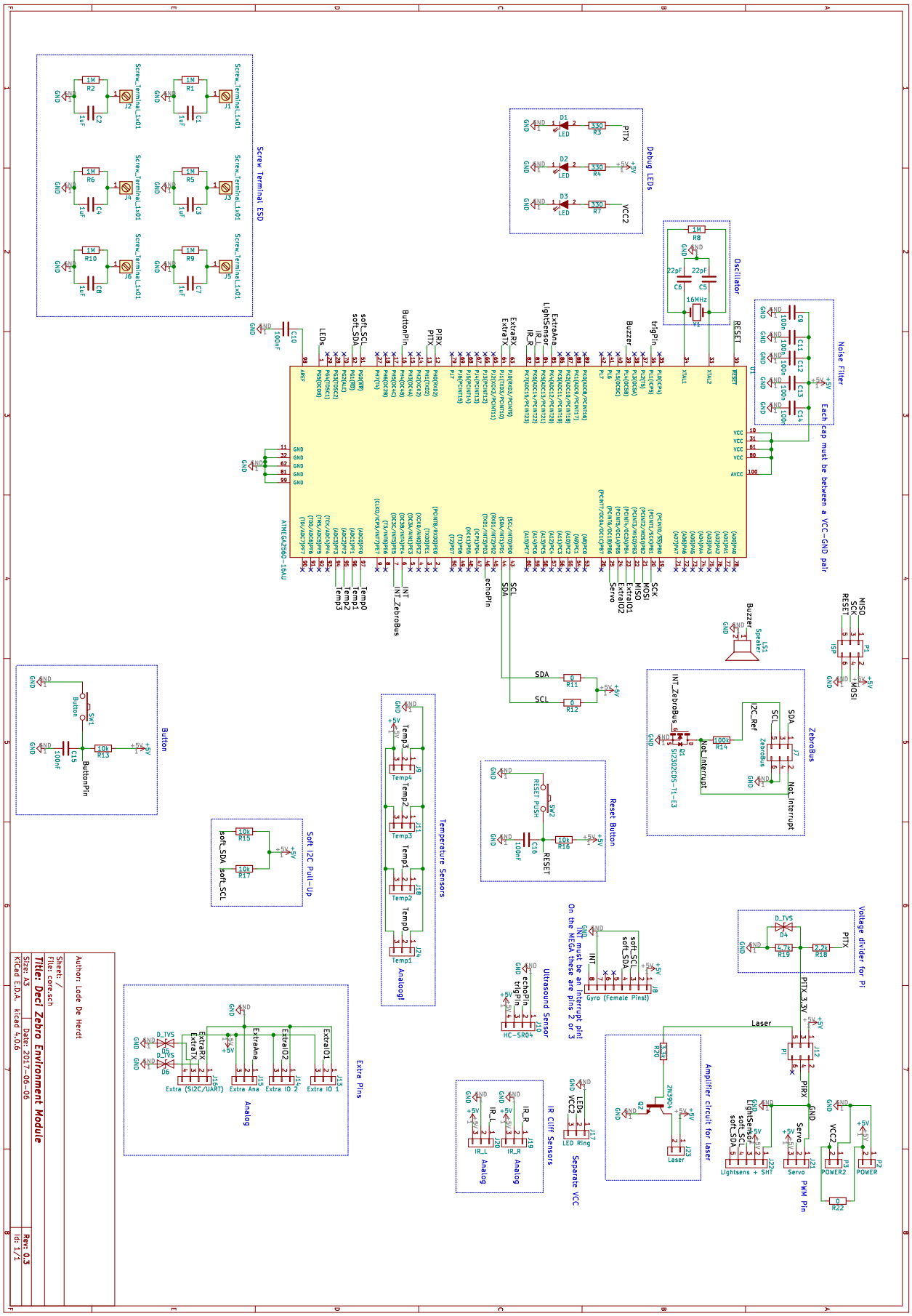


Figure A.6: Schematic of PCB

Appendix B

Technical Drawings

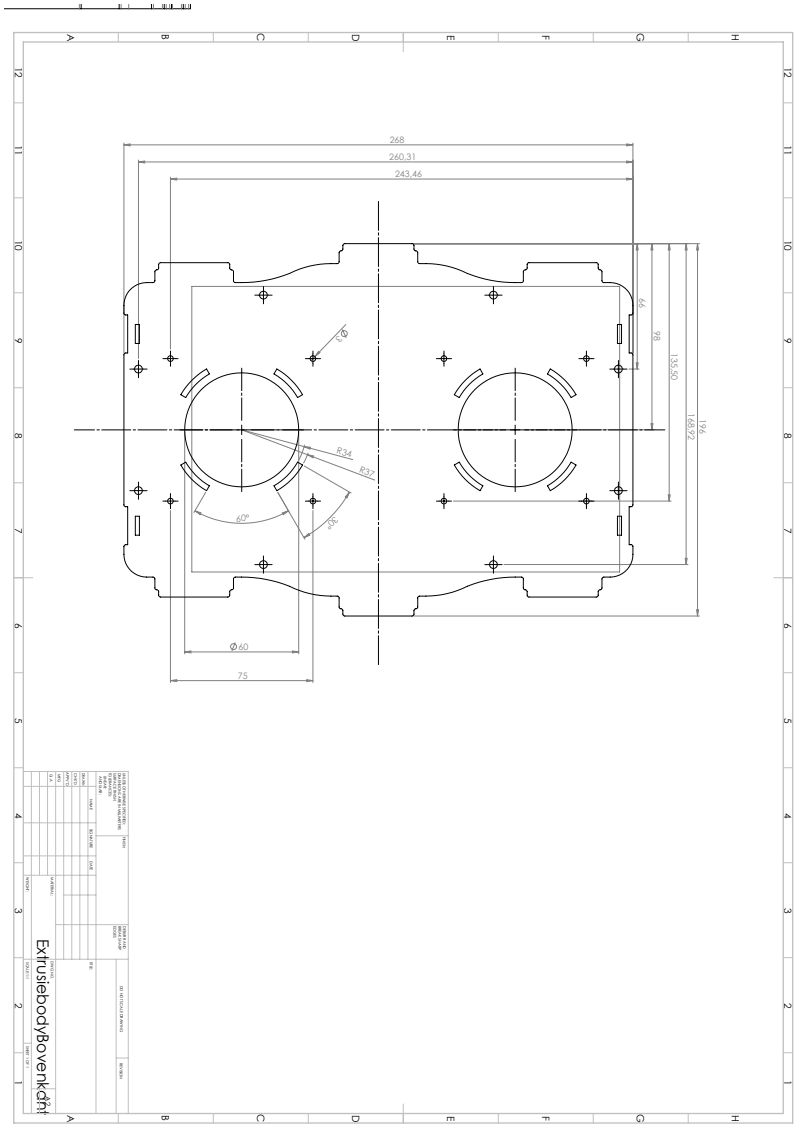


Figure B.1: *Technical Drawing of the Zebro body (top), showing the dimensions.*

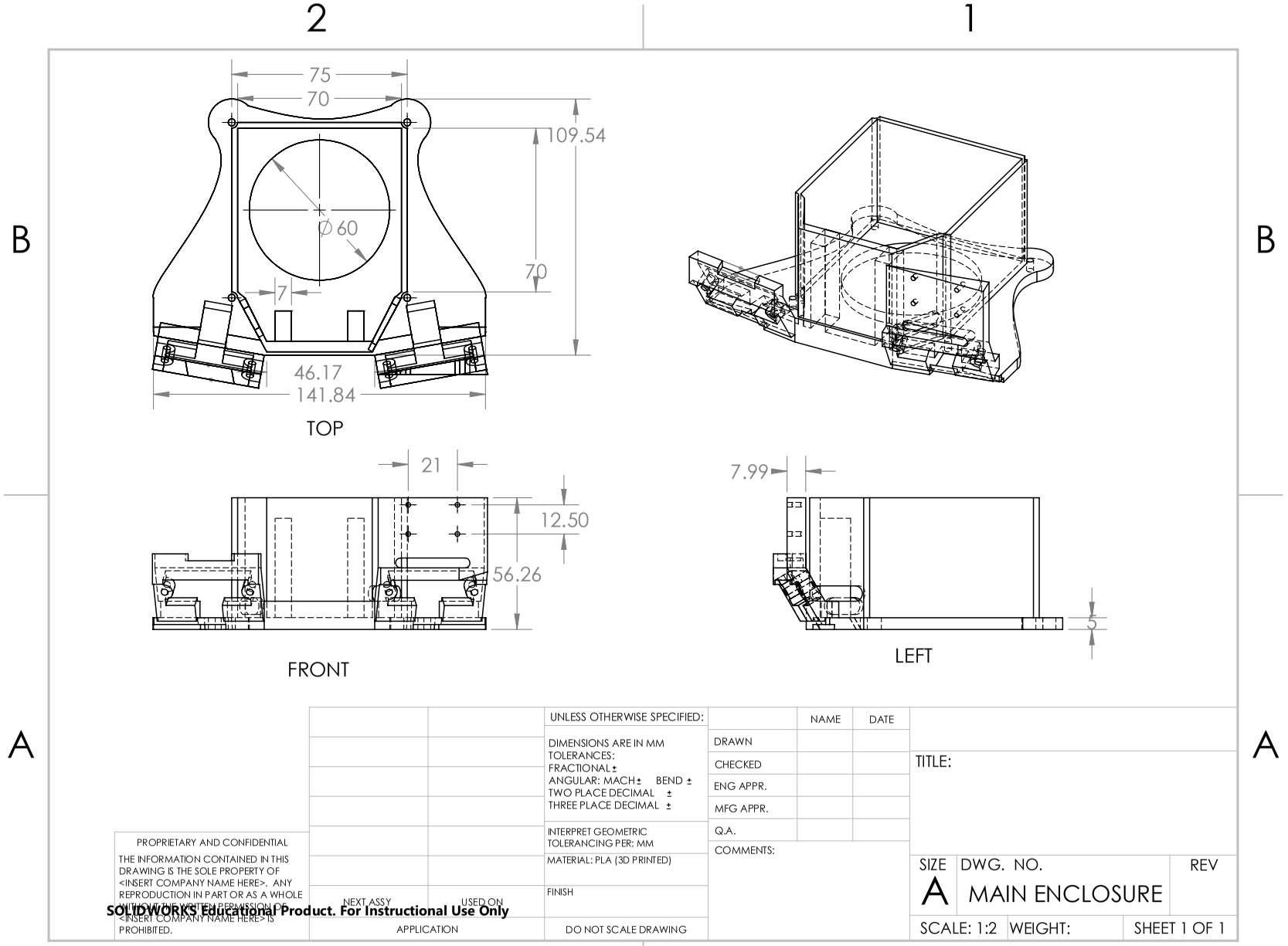


Figure B.2: Technical drawing of the main enclosure, showing the dimensions.

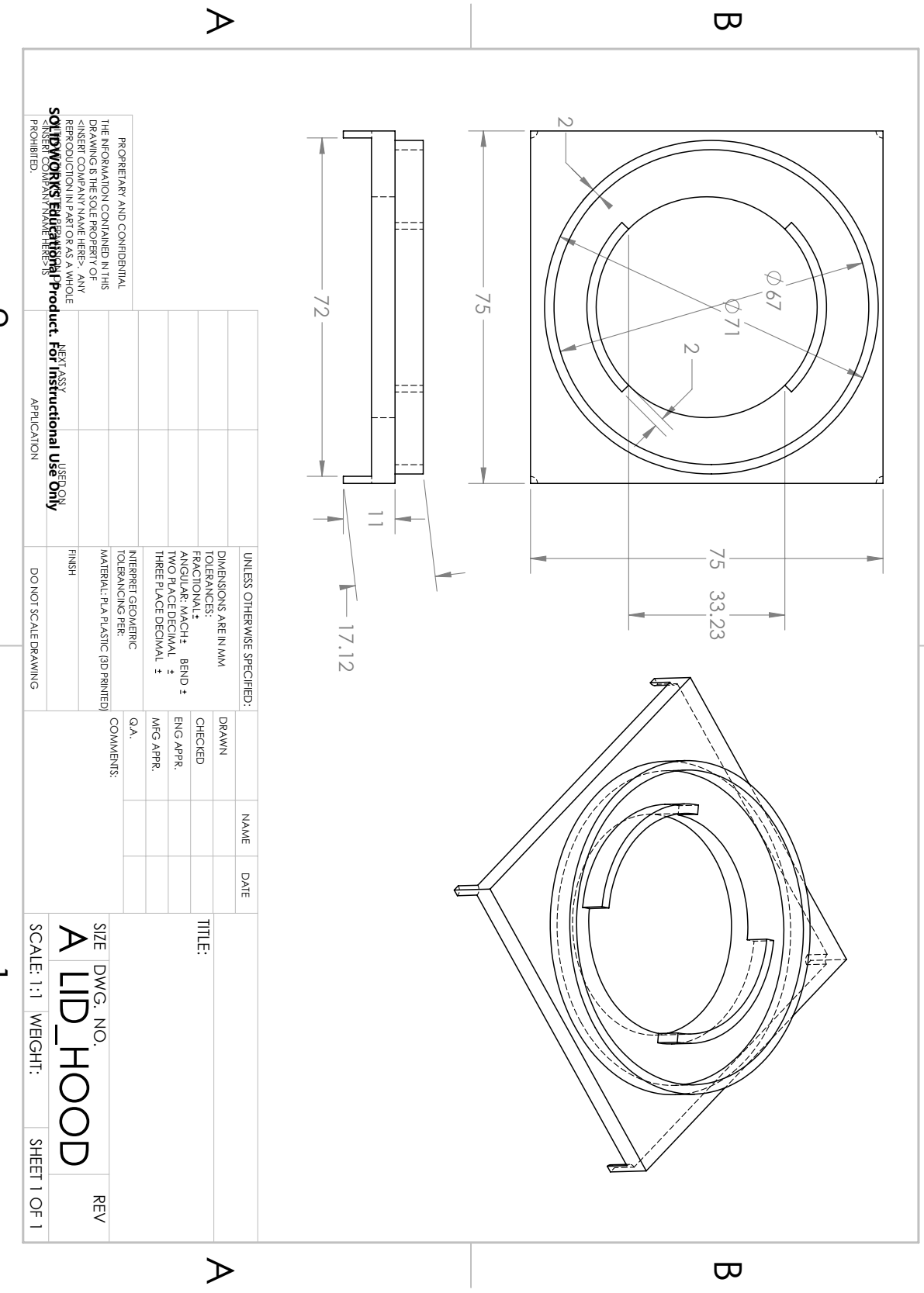


Figure B.3: *Technical drawing of the lid of the enclosure, showing the dimensions.*

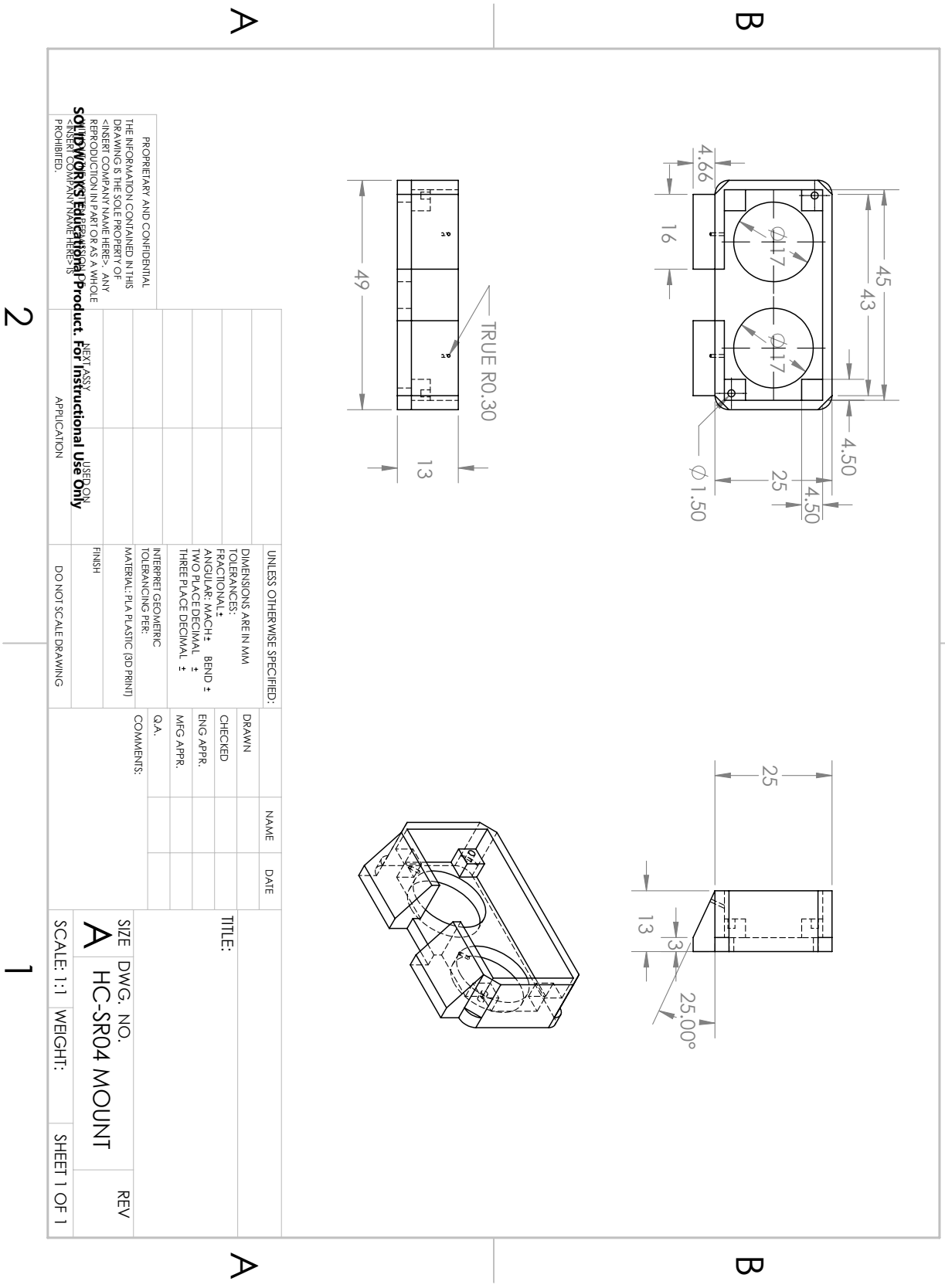


Figure B.4: *Technical drawing of the ultrasonic sensor mount, showing the dimensions.*

Appendix C

Price List

Table C.1: *Price List for all components in submodule*

Component	**Part number**	**#**	**€/ 10**	**€/ 100**	**Reseller NL**	**€/ 100**	**Reseller World**
Buck Converter	173010578	1	8.32	6.70	Farnell	0.50	AliExpress
Temperature sensor	MCP9701-E/TO	4	0.94	0.712	Farnell	0.712	Farnell
Gyro/accelero	MPU-6050	1	7.24	6.97	Farnell	2.30	DX.com
Light intensity sensor	LDR	1	0.871	0.743	Farnell	0.045	AliExpress
Humidity sensor	SHT21	1	4.10	3.74	Farnell	1.86	AlieExpress
Speaker	ABT-414-RC	1	1.81	0.896	Farnell	0.09	AliExpress
Distance sensor	HC-SR04	1	3.75	3.55	Antratek	0.79	AliExpress
Cliff sensor	GP2Y0A21YK0F	2	8.66	7.18	Farnell	3.45	AliExpress
Laser	KY-008	1	1,95	1,95	hobby-electronica	0.67	AliExpress
Pi Camera		1	17,97	17,97	Farnell	17,78	AliExpress
Raspberry Pi Zero W		1	11	11	Kiwi Electronic	11	Kiwi Electronic
SD Card for RPi (8 GB)		1	7,40	7,40	Kiwi Electronic	5	AliExpress
Pi Camera (long) flex cable		1	4,43	3,93	Kiwi Electronic	1,70	AliExpress
Microcontroller	ATmega2560	1	12.67	10.51	Farnell	3.80	AliExpress
loose components PCB	Various	1	5.82	4.55	Farnell	4.55	Farnell
PCB Manufacturing		1	8.82	2.89	Euro-Circuits	0.66	ALLPCB
Plastic Enclosure		1	13.14	11.68	3Dhubs	11.10	3Dhubs (China)
			130.37	111.68		71.59	

Bibliography

[1] P. De Vaere and D. Booms, "Zebrobus," Internal documentation, Delft University of Technology The Zebro Project, May 2017.

[2] O. Oosterlee and S. Peterse, "Obstacle detection using laser triangulation and opticalflow," Jun 2017.

[3] P. Goris and S. van Leeuwen, "Safety and sensor submodule of the environment observation module," Jun 2017.

[4] C. Verhoeven, E. Hakkenes, and D. Booms, *Research and Development of Environment Observation Module for Nano Zebro*, Zebro Team, EEMCS Faculty Delft University of Technology.

[5] A. M. Kassim, H. I. Jaafar, M. A. Azam, N. Abas, and T. Yasuno, "Performances study of distance measurement sensor with different object materials and properties," *2013 IEEE 3rd International Conference on System Engineering and Technology*, p. 281, Aug 2013.

[6] M. V. Paulet, A. Salceanu, and O. M. Neacsu, "Ultrasonic radar," *Proceedings of the 2016 International Conference and Exposition on Electrical and Power Engineering*, p. 551, Oct 2016.

[7] T. Stănescu, A. Mondoc, and V. Dolga, "Probabilistic aspects in mobile robots navigation," *Romanian Review Precision Mechanics, Optics and Mechatronics*, no. 45, pp. 125–130, Jan 2014.

[8] "What is different about the sharp sensor?" http://education.rec.ri.cmu.edu/content/electronics/boe/ir_sensor/4.html, Carnegie Mellon Robotics Academy, accessed: 6 2017.

[9] *Ultrasonic Ranging Module HC - SR04*, Elec Freaks.

[10] *PING))) Ultrasonic Distance Sensor*, Parallax Inc., Feb 2013, rev. 2.0.

[11] "Add 6 ultrasonic distance sensors to existing raspberry pi robot," http://www.instructables.com/id/Add-6-Ultrasonic-Distance-Sensors-to-Existing-Rasp/, Instructables.com, accessed: 6 2017.

[12] *Distance Measuring Sensor Unit GP2Y0A21YK0F*, SHARP Corporation, Dec 2006.

[13] "3521_0 - sharp distance sensor (10-80cm)," http://www.phidgets.com/products.php?product_id=3521, Phidgets Inc., accessed: 6 2017.

[14] D. Nedelkovski, "Arduino radar project," http://howtomechatronics.com/projects/arduino-radar-project/, July 2015, accessed: 6 2017.

[15] *Towerpro MG90 Micro Servo*, Phidgets Inc.

[16] "Neopixel ring - 24 x ws2812 5050 rgb led," https://www.sparkfun.com/products/12665, SparkFun Electronics, accessed: 6 2017.

[17] "Pixel reference," https://github.com/FastLED/FastLED/wiki/Pixel-reference, FastLED/FastLED Wiki, GitHub, accessed: 6 2017.

[18] P. Burgess, "Using neopixels and servos together: An introduction to avr peripherals," https://learn.adafruit.com/neopixels-and-servos/overview, Adafruit Learning System, Adafruit Industries, accessed: 6 2017.

[19] *Deci Zebro module interface specifications*, TU Delft Robotics Institute, Jan 2017.

[20] *8-bit AVR Microcontrollers ATmega328/P*, Atmel Corporation, Nov 2016.

[21] *32-bit ARM-Based Microcontrollers SAM D21E / SAM D21G / SAM D21J*, Microchip Technology Inc, Jan 2017.

[22] *8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash*, Atmel Corporation, Feb 2014.