Safety and Sensing

submodule of the Environment Observation Module

Bachelor Thesis

Sander van Leeuwen	4351436
Peggy Goris	4290569

June 19, 2017

DELFT UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE

ELECTRICAL ENGINEERING BACHELOR

THE ZEBRO PROJECT

Safety and Sensing

submodule of the Environment Observation Module

BACHELOR OF SCIENCE THESIS

For the degree of Bachelor of Science in Electrical Engineering at Delft University of Technology

Authors:

Sander van Leeuwen	4351436
Peggy Goris	4290569

Supervisor:
Chris Verhoeven

Daily Supervisor:
Daniël Booms

Advisors:
Edwin Hakkennes
Ronald Bos

DELFT UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE

ELECTRICAL ENGINEERING BACHELOR

THE ZEBRO PROJECT

Abstract

This bachelor thesis is about the design of the Safety and Sensing module, which is part of the Environment Observation Module for Deci Zebro robots. This Environment Observation Module is developed at the faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) of Delft University of Technology. Zebro robots were not able to walk around without falling off things or without collide against obstacles. Also Zebro was not able to collect data of its surrounding or of its own position and orientation.

With the Safety and Sensing Module Zebro will be able to collect data of its surroundings such as light intensity, humidity and temperature and is equipped with lift and fall detection and over-current and internal overheat protection. Also the safety of Zebro and its ability to walk around safely are improved with this module.

Because Deci Zebro is not able to walk at this time, the module is not tested under real circumstances. In simulated circumstances the module works as intended.

Contents

Chapter 1

Introduction

This thesis is written for the Electrical Engineering Bachelor Graduation project of group L at the EEMCS (Electrical Engineering, Mathematics and Computer Science) faculty of Delft University of Technology. For this project our project group created a module for the Zebro project. The Zebro project is a subgroup of a greater team of swarming robots that is part of the TU Delft Robotics Institute. They have the following vision:

> *Introduce innovative robotics technologies that will enable robots to work together with humans in human environments, contributing to all kinds of services and labour beyond 'confined' industrial environments.* [1]

Inspired by insects, the Zebro team made Zebro (see Figure 1.1) robots that are actually six-legged robots with minimum intelligence. These robots come in three different sizes (Nano, Deci and Kilo), but currently lack adequate sensors for communication and risk assessment. They also do not have an automated recharging system.

> *The main purpose of the Zebro is swarm-related behavior, where a swarm of relatively inexpensive, simple and straight-forward robots is able to perform tasks that are not suited for one bigger robot. An example would be the mapping of a disaster area.* [2]

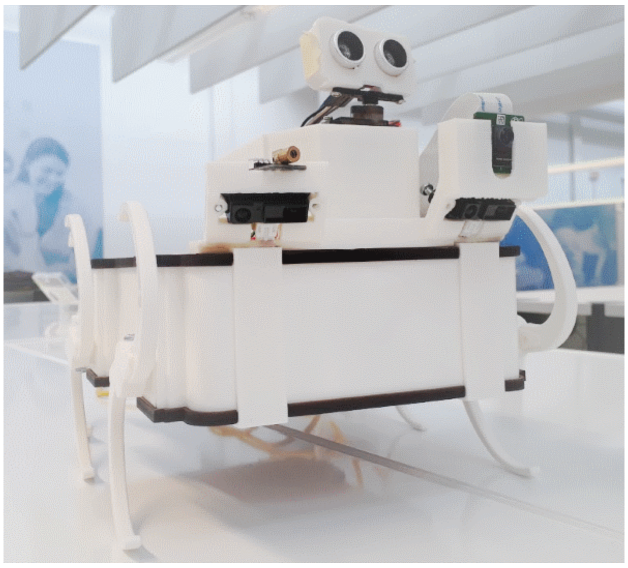


Figure 1.1: Photograph of Zebro

The assignment for our group was to create an Environment Observation Module for the Deci Zebro [2]. An overview of the modules created at the same time by other groups is shown in Figure 1.2.

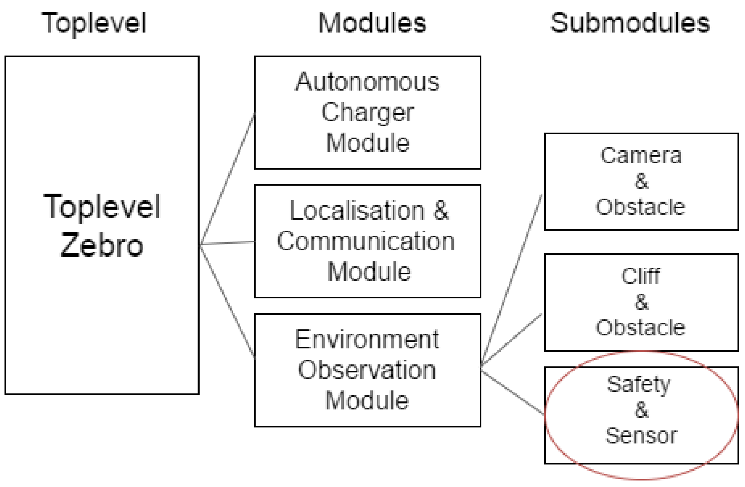

Toplevel
Modules
Submodules
Toplevel Zebro
Autonomous Charger Module
Localisation & Communication Module
Environment Observation Module
Camera & Obstacle
Cliff & Obstacle
Safety & Sensor

Figure 1.2: Overview of Zebro module division

The project group existed of six people divided in three subgroups, working each on their own submodule of the Observation Module (Figure 1.2). In this thesis the Safety & Sensing module, or $S\&S$ module for short, and the integration with the other submodules is described. The $S\&S$ module is able to protect the whole Observation Module against damage by switching off the power supply when an over-current is measured and by monitoring and controlling internal temperature. Some other things of Zebro its environment can be measured such as temperature, light intensity and humidity. The orientation and acceleration of Zebro are monitored, from which data can be derived whether Zebro is lifted or if it is walking on a too steep slope where it can fall off.

The design steps of the $S\&S$ module can be read in this thesis. The problem description is given in chapter 2, where also a global overview of the hardware is shown. In chapter 3 the specifications of the Observation Module and the $S\&S$ module are given. The orientation and acceleration monitoring of Zebro can be read in chapter 4. In chapter 5 the additional sensors are explained. The protection system of the Observation Module is explained in chapter 6, where in chapter 7 an overview of the top-level software is given, with a short description about the software used in the Observation Module. In chapter 8 the integration of the sensors and submodules are described and the end result of the Observation Module is discussed. The conclusion, in which the achieved specifications and improvements are shown can be read in chapter 9. Finally in chapter 10 the recommendations and future work is described.

1.1 State of the art

Back in 2003 C.-Y Chong and S.P. Kumar made the prediction that increasingly data would be available via sensor networks [3]. With the trend of IOT [4] this is happening as expected and it got so far as self deploying sensor networks using UAVs [5]. The Zebro project is also a self deploying sensor network, however it differs on that most of the data will not be send to a central terminal, but the data is used to swarm. Numerous simulations of swarms are available, but realistic swarm behavior needs to be studied using a realistically sized swarm. Therefore, a larger number of Zebros is needed [2]. With the help of simple sensors the Zebro robots become able to monitor its environment. Even though not all techniques are practical to implement, state of the art techniques will be discussed along in the sensor selection in this thesis.

Chapter 2

Problem description

Zebro is a swarm robot that is meant to walk around and collect data of its surrounding. These swarm robots can be used to do tasks for and in cooperation with humans. In the current situation Zebro is only designed to walk around without monitoring or seeing anything of its surrounding. In this way it is highly probable that it will fall off things, walk against objects or can come into a dangerous environment i.e. one with fire.

With the help of this self-contained Observation Module, Zebro should be able to walk around safely, inside and outside, without becoming a danger for itself or its surrounding. With the help of its distance sensors Zebro is able to detect objects or cliffs. It can monitor it surrounding with a temperature sensor, light sensor and humidity sensor. The orientation and acceleration will both be measured. The module using a self protection system to prevent other parts of Zebro and the module itself against damage. The data will be collected and processed where needed. This data will be sent to the top-level Zebro, the main part of Zebro were all the final decision will be made. An overview of the hardware of the created module is shown in Figure 2.1, where the emphasized and encircled boxes are the parts that will be discussed in this thesis. This solution will bring the Zebro project a big step further to come closer to their ideals.

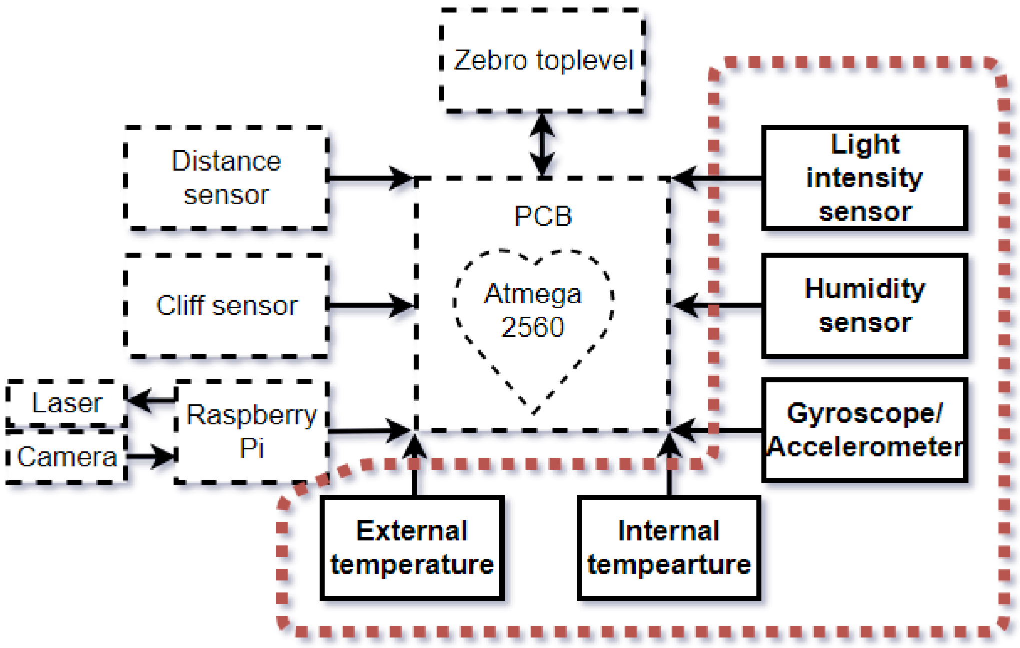


Figure 2.1: Overview hardware of top-level

Chapter 3

Program of requirements

This chapter shows the requirements for the Environment Observation Module. The general requirements, thesis requirements and criteria given by the Zebro team [2]. The requirements and criteria listed that are gray colored, are the ones not appropriate for this thesis. The requirements and criteria black colored are the ones used.

3.1 General requirements

In the assignment received from the Zebro team at the beginning of the project some minimal initial requirements for the Environment Observation Module were given, such as:

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 voltages, currents and power flow

EM-6 Implement an Autonomous Module Damage Prevention System (AMDPS)

3.2 Thesis requirements

The groups thesis covers the development of a Environment Observation Module. The minimum thesis requirements given 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

3.3 Criteria

The Observation module should also meet the following criteria:

C-1 It should be self-contained

C-2 It should have very well-defined interfaces with the outside world, that is, the Physical interface, the power interface and the control & status interface.

C-3 The module must be modular and easily replaceable by another robot

C-4 The system should report to central nervous system (called Zebro top-level in this thesis)

C-5 The system must be able to survive in an unknown environment, including rough terrain and public areas with people.

There is also one point given that is highly desirable:

D-1 The system must be simple, easy to reproduce, easy to repair, stable and low cost

3.4 Sensor and Safety requirements

In this thesis only the$S\&S$part will be described. From the requirements mentioned above, some specific requirements and criteria for the $S\&S$ Module are derived that are needed to accomplish the before mentioned requirements (these are number with S-1 to S-7, because these are the specific requirements for the $S\&S$ module):

S-1 The system should take care of its own right powering voltage. (criterion C-1)

S-2 The system should not be dependent on data from outside the module (derived from criterion C-1)

S-3 Should be able to monitor the incoming current and the internal temperature, and switch the power off when needed, to protect itself and other parts of Zebro from damage (derived from criterion C-1, requirement EM-4, EM-5 and EM-6)

S-4 Low-level commands and debugging information (derived from criterion C-2)

S-5 High-level commands and status needed for central nervous system (derived from criterion C-2)

S-6 Should be ready and easily to produce more Zebros (criterion D-1)

S-7 Should be able to communicate with the Zebro top-level (criterion C-4)

3.5 Requirements in chapters

In the chapters themselves are also chapter requirements. These are requirements we set ourselves, because of test results or results from calculations on the system. These chapter requirements are mentioned in the chapters themselves. In the chapters is also referred to the requirements or criteria used in that specific chapter by for example (S-4).

Chapter 4

Orientation and acceleration

This chapter is about the orientation and the acceleration of Zebro. An orientation sensor is needed to know if the robot is walking on a ground that is becoming too steep (see Figure 4.1). With this information the robot is able to prevent itself from role over, slipping or falling off the slope. With the acceleration data the Zebro can detect if it is lifted or has fallen down. Combining both the orientation and acceleration data makes it possible to correct the acceleration data according to the real world frame. In this chapter a list of potential sensors and the final design of this part of the system is shown. Also the applications implemented with the data are described.

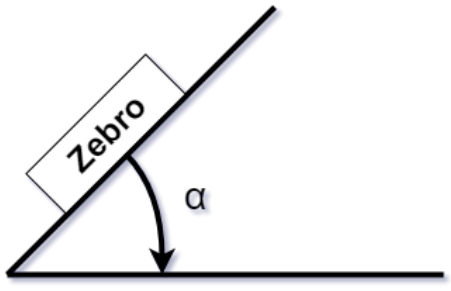


Figure 4.1: Zebro is walking on a slope

4.1 Requirements

In this chapter the following requirements have to be taken into account; S-2, and criteria C-1 and C-4 from chapter 3. On top of that we set ourselves some chapter requirements:

CR-4.1 Pitch and Roll should be known with a precision of 5 degrees* (reference system is as shown in Figure 4.1)

CR-4.2 Acceleration of Zebro should be measured in all directions (reference system is as shown in Figure 4.1)

CR-4.3 Zebro should be able to know if it is lifted or is falling off something (derived from criterion C-5)

CR-4.4 Zebro should be able to know if it is walking on a slope that is too steep (derived from criterion C-5)

*A precision of 5 degrees would be accurate enough to detect the steepness of a slope, as it would allow cheap(er) sensors (D-1).

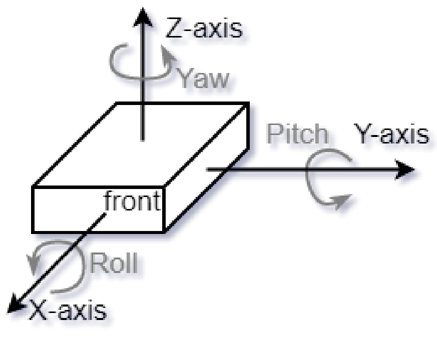


Figure 4.2: x,y,z axes and orientation vectors

4.2 Potential sensors

A selection of possible sensors was made. These sensors meet the requirements mentioned in section 4.1.

4.2.1 Orientation sensors

The orientation sensor should be able to measure the angle of the slope as is expressed by α in Figure 4.1, for both the pitch and roll (CR-4.1). To achieve this several techniques can be used, such as shown in Table 4.1.

Table 4.1: Techniques for measuring the orientation

Technique	Description	Advantages	Disadvantages
Gyroscope	Vibrating elements that when presented to rotation will feedback to a mechanical system, which can be read-out (MEMS [6]).	• Widely available • Small formfactor • Low computational power	• Drift can occur by accumulating errors
Vector Magnetometer	A mechanical motion sensor that is based on the Lorentz force acting on a current-carrying conductor (MEMS [7]).	• Widely available • Small formfactor • Low computational power	• Sensitive to non-earth magnetic fields • Problems in environments with metal surroundings
Camera	With the use of SLAM algorithms (i.e. see [8] [9]).	• Provides extra data like motion data • Small formfactor	• High computational power • Preferably needs extra sensors (i.e. laser range) • Lighting problems • Needs recognition points
Ball tilt switch	A switch which state depends on the orientation.	• Extremely low cost	• High hysteresis (up to 30 degrees) [10] • Exact orientation requires multiple sensors [11] • Mechanical wear

As can be seen from table Figure 4.1, the camera option needs high computational power that is undesirable and has lighting problems. Magnetometer has problems with other magnetic fields and metal surroundings, what is a probable environment Zebro would walk in. The ball tilt switch is too inaccurate. For this reasons the gyroscope is chosen, which may only have problems with drift. This problem can be reduced by software.

4.2.2 Acceleration sensors

The acceleration sensor should be able to give back the acceleration in the x, y and z direction (CR-4.2). Several techniques could be used to achieve this result. Possible techniques that can be used are shown in Table 4.2

Table 4.2: Techniques for measuring the acceleration

Technique	Description	Advantages	Disadvantages
Vector accelerometer	Structure that measures the deformation due to acceleration [12].	• Widely available • Small formfactor • Low computational power	• Accumulating error in distance measurements [13]
Barometer	Measures air pressure.	• Widely available • Small formfactor • Low computational power	• Can only measure heights (z-axis) • Low accuracy [14]
GNSS	Calculates position via satellites.	• Widely available • Small formfactor • Low computational power	• Problems indoors [14] and environments with metal surroundings • Low local accuracy [15]
Speedometer	Tracks distance via keeping track of the amount of revolutions.		• Mechanical wear • Errors due to diameter mismatches • Requires contact with the floor
Camera	With the use of SLAM algorithms (i.e. see [8]).	• Provides extra data like orientation data	• High computational power • Preferably needs extra sensors (i.e. laser range) • Lighting problems • Needs recognition points

The barometer is not a good option, because of its low accuracy. The GNSS has problems with metal surroundings and has a low local accuracy, which is needed in this case. The speedometer has to have contact with the floor, which can give problems to the Zebro walking and will give errors due to diameter mismatches. As is already said in subsection 4.2.1, the camera is also in this case not a good option for use, because of the high computational power and the lighting problems. For this reasons the vector accelerometer technique is used to measure the acceleration in all directions (CR-4.2), because this has no considerable disadvantages for its use. There would not be a distance measurement and thus this disadvantage is void.

4.2.3 Sensor chosen

The sensor that is finally chosen is a combination of both a gyroscope and an accelerometer (D-1 and S-6). This is the MPU-9150 [16], since this component was available at the Zebro research team. After research it seemed that the MPU-6050 [17] is software compatible with the MPU-9150. The only difference between the MPU-9150 and MPU-6050 is that the MPU-9150 has an integrated magnetometer. This extra feature is not needed to reach the specifications and will thus not be used, so the MPU-6050 is sufficient. The gyroscope drift is being calibrated for at run-time.

4.3 Orientation

The chosen sensor is a MEMS gyroscope. This means that the sensor measures deformations in a micro structure and converts this to orientation data. For using the chosen sensor a software library was found [18]. This library was used to run several tests, after which the code was slightly adapted to our top level code. The output is in degrees with respect to earth its horizontal.

4.3.1 Applications of the gyroscope data

Several functions can be fulfilled with the help of the data from the gyroscope. For example it can be helpful to select the relevance of the object detector data that is coming from both the object sensors of the ”cliff & object” [19] and the ”image recognition” [20] submodules of the Observation module. When Zebro is walking downwards a slope and is walking towards the ground, it is possible that it sees the ground as an object. This can make it difficult or even impossible for Zebro to get off a hill, if it detects objects everywhere. Another function for the data coming from the gyroscope is to detect whether a slope is getting too steep. If Zebro would walk further when the cliff gets too steep, there is a possibility that Zebro falls off, slides off or roles over and gets damaged.

Slope problem with object detectors

Two kind of object detectors were used to detect obstacles. With the help of the gyroscope, Zebro can know if it is walking downwards on a slope. This can help to find out which data of the object detectors is relevant or not. If Zebro is walking downwards as shown in Figure 4.3 it is possible that the detectors are seeing the ground as an object. One of the sensors is an ultrasound. It would be no problem for this sensor if the slope is less than 40 degrees [19]; then the sound will go the other way and not get back in the detector as shown by the dotted line.

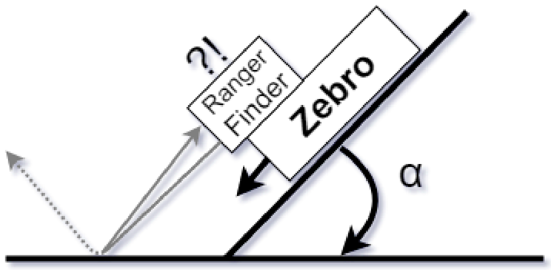


Figure 4.3: Zebro walking downwards on a hill with objectdetectors

With the assumption that the ultrasonic sensors won’t give any problems, only the other object detector can give a problem. This will give back that it saw an object in front of it when walking downwards the slope. This can be solved by ignoring the data of this sensor on a slope downwards. With this only objects that are on the slope itself will be detected by the ultrasound sensor. In Table 4.3 an overview of which data will be used in which cases is shown, where 0 means that no object is detected and 1 that an object is detected. The last column shows whether data is passed on to the Zebro top-level or not.

Table 4.3: Objectdetectors data relevance

Circumstances	Ultrasound detects	Image recognition detects	Send to top-level
flat ground or positive slope	0	0	0
flat ground or positive slope	0	1	1
flat ground or positive slope	1	0	1
flat ground or positive slope	1	1	1
negative slope	0	0	0
negative slope	0	1	0
negative slope	1	0	1
negative slope	1	1	1

Too steep

The second application is called the ”too steep” application (CR-4.4). This one is to protect Zebro from falling damage and has a high priority for the Zebro top-level (criterion C-5). If Zebro walks further and the slope becomes steeper it is possible that it will flip, slip or fall, which can cause damage to Zebro. Unfortunately, due to the fact that no Deci Zebros were walking during development of this Observation Module, no measurements can be done with Zebro walking on a slope. Because of the first application and the use of ultrasound as explained in subsubsection ”Slope problem with object detectors” in subsection 4.3.1, the slope should not be more than 40 degrees with its horizontal.

4.4 Acceleration

The chosen sensor is a MEMS accelerometer. This means that the sensor measures deformations due to inertia in a micro structure and converts this to acceleration data.

A software library [18] was found to make it able to use the sensor chosen. This library was used to run several tests, after which the code was slightly adapted to our top-level code. The output is in mm/s^2 in the x, y and z direction.

4.4.1 Applications of the accelerometer data

With the data from the accelerometer one main application can be derived. Zebro is meant as swarm robot and to collect data in its surroundings, which is difficult if people are constantly lifting Zebro (CR-4.3 and criterion C-5). Besides that can Zebro be stolen, because the Zebros will be walking in public places without supervision. On top of that it can be helpful for some sensors or other parts of Zebro, to know if Zebro is lifted.

Zebro is lifted

If Zebro is lifted, the acceleration in one or several directions will change. Detection of the lift can be implemented with a limit. This limit is set by testing the setup and compare the data of normal walking with the data of Zebro that is lifted. See section 4.6 for more details about the test. If the acceleration in one direction gets above the limit, the error bit (explained in subsection 7.1.1) is set high. This error bit causes a buzzer to make noise. This sound will continue till the error bit is set low again. For testing purposes this can be done by pressing a button that can be found on the enclosure. In chapter 10 another way, that is not implemented, to get the error bit low again is shown.

4.5 Applications of data from MPU

Since both the accelerometer and the gyroscope are implemented in the MPU-6050 it is also possible to fuse this data. The Digital Motion Processor (DMP) that is also integrated on the chip can be used to fuse this data, however the manufacturer has a discouragement policy for the use of the DMP [21]. With software it is possible to correct the chips [x y z] frame to the real worlds [x y z] frame. The library for the MPU-6050 has reverse-engineered code to control the DMP and it can correct the data to the real world [x y z] frame. All previous mentioned applications that use the accelerometer are implemented using this sensor fusion, so that orientation does not affect the data.

4.6 Tests

To test the accuracy of the sensor and the technique of the applications explained, some tests were done. First a test with the accelerometer is done to check if the technique for Zebro being lifted works properly. After that the gyroscope accuracy is measured.

4.6.1 Zebro lifted test

To test the Zebro lift detection, first some measurements were done on the walking Zebro and on Zebro that stands still. Because Zebro is not walking in real life at the time of this test, the walking of Zebro is simulated by moving the sensor by hand. The measurement of the accelerometer by moving it up and down in a constant motion, were done three times. After this the measurements were done again three times, but this time with Zebro being picked up in a normal manner along the z-axis. The results of this tests are shown in Figure 4.4.

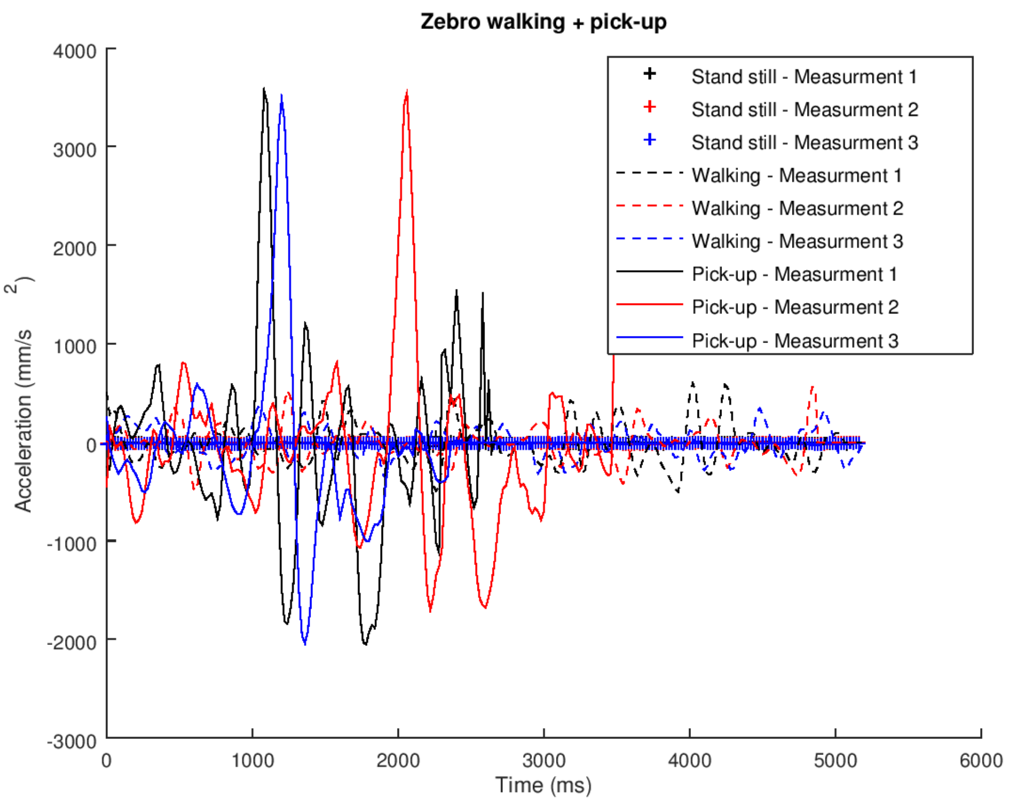


Figure 4.4: Zebro walking and lifted up while walking

Zebro may not be completely lifted in the z-direction so the same kind of measurement is done in the forward direction. First three measurements were done by moving Zebro with an approximately constant velocity in the forward direction. After that three measurements were done by picking it up in the same direction as the movement. The results of this tests are shown in Figure 4.5.

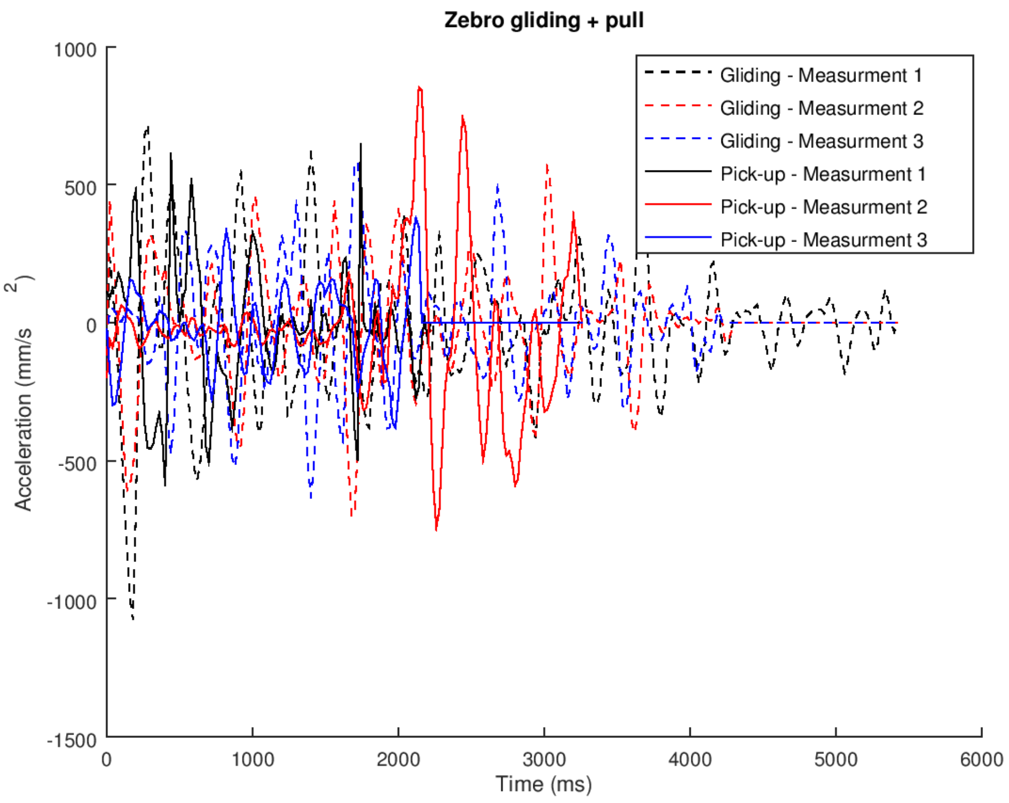


Figure 4.5: Zebro walking and lifted in same direction

Finally a combination of two directions were measured. This means that Zebro is orientated at a Pitch of 45 degrees. Three measurements were done where Zebro is lifted upwards. The results of this measurement are shown in Figure 4.6.

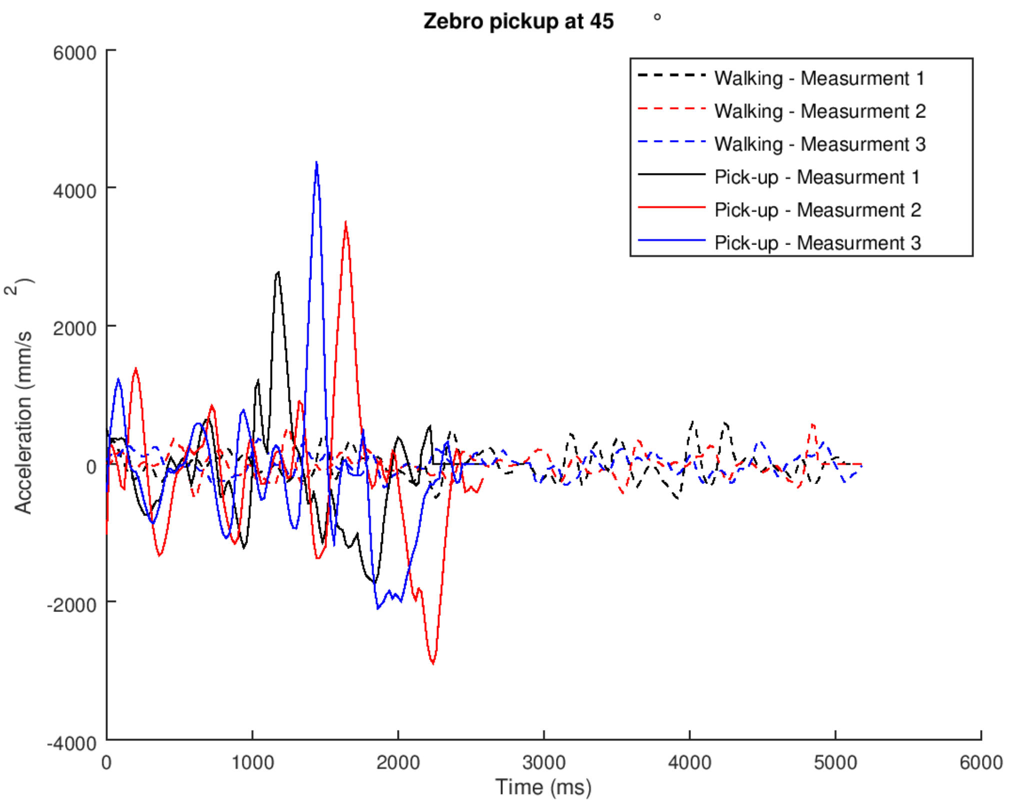


Figure 4.6: Zebro walking and lifted in random direction

Conclusion

From this test can be seen that between the acceleration measured during walking and the acceleration during lifting the values are of different magnitudes. Between these two cases, walking and lifting, a clear limit can be set. This limit can be chosen between 1500 and 3000. From the measurements can also be seen that a limit in z-direction will always be reached by picking it up. For this reason only the limit will be set on z-axis to chose if Zebro is lifted or not. This limit is set as a variable in the code, because the Zebro is not walking yet. The way Zebro will be walk and the simulated way could be different and thus new tests needs to be done before use. A recommendation therefore is to do the test with the accelerometer again while Zebro is able to walk and inspect if the limit is still valid. The limit used in this design is 1800. In this case it is assumed that Zebro is walking on a smooth surface. When Zebro is walking on rough terrain, where it moves a lot in the other directions, this can give different values compared to the normal walking acceleration. For this reason the limit should be a bit on the high side of the possible values to cope with this problem. There is not a clear limit that can be set in the gliding case. The gliding and pick-up difference is so small that this direction of movement will be ignored.

4.6.2 Gyroscope accuracy test

The gyroscope is tested as follows. The gyroscope is held in different angles for the Pitch and the Roll, with 10 degrees between each measurement, where the angle is measured with the help of a protractor triangle. The measurement is done two times for both Pitch and Roll for which the results are shown in Table 4.4.

Table 4.4: Gyro accuracy test

Real	**Pitch 1**	**Pitch 2**	**Roll 1**	**Roll 2**
-90.0	-90.6	-91.0	-90.2	-90.2
-80.0	-80.0	-79.8	-79.8	-79.8
-70.0	-71.2	-70.4	-70.3	-70.1
-60.0	-60.2	-61.0	-60.8	-58.9
-50.0	-50.7	-50.3	-50.2	-49.2
-40.0	-38.9	-39.3	-40.5	-39.5
-30.0	-29.2	-29.4	-29.9	-29.0
-20.0	-19.5	-18.9	-20.2	-19.4
-10.0	-10.2	-10.9	-9.2	-8.9
0.0	0.5	0.4	0.5	0.6
10.0	9.6	8.5	10.0	9.7
20.0	19.8	19.6	20.5	19.9
30.0	30.5	29.6	30.4	30.8
40.0	40.5	39.4	40.2	40.3
50.0	50.8	50.0	50.3	51.7
60.0	60.7	60.3	60.5	61.7
70.0	71.4	71.8	69.9	71.1
80.0	81.7	81.6	80.5	81.1
90.0	91.0	91.8	90.7	91.3

With this measurement it is easy to make some measurement errors by reading off the protractor triangle, this can be with an accuracy of 1 degree. On top of that it is difficult to put the gyroscope in exactly the right angle. A table is used and expected to be horizontal, but as can be seen on the measurements the gyroscope measures approximately 0.6 degrees. So by this measurement we can conclude that the gyroscope has an accuracy of approximately 2 to 3 degrees and thus meets CR-4.1. This accuracy should be taken into account by decisions into boundary conditions, such as too steep.

4.6.3 Conclusion

According to the measurements the gyroscope has an accuracy of approximately 2 to 3 degrees. This is within the chapter requirement of 5 degrees (CR-4.1). For this reason the sensor is good enough to use in our case.

4.7 Conclusion

For the orientation a gyroscope is used, where for the acceleration an accelerometer is used. The combination is integrated in the MPU-6050, which will be used (D-1 and S-6). From the tests both CR-4.1 and CR-4.2 are met with this sensor. Also with this sensor the Zebro will be able to detect whether it is lifted (CR-4.3), when it is on a too steep surface (CR-4.4) or should ignore obstacle measurements. This part only needs data gathered from the Observation Module (S-2). From the tests done with this sensor it can be concluded that it is suitable for this application.

Chapter 5

Additional sensors

In this chapter some additional sensors are explained. These sensors can be helpful in more specific situations for collecting data from its surroundings and not only for self protection purposes. The additional sensors implemented are external temperature sensors, a light intensity sensor and an humidity sensor. In each section the targets of each of the sensor, how the specific sensor is chosen and how it is implemented is discussed.

5.1 Requirements

In this chapter requirements S-2 and S-6 and criteria C-1, C-4 and C-5 from chapter 3 have to be taken into account. On top of that some extra chapter requirements were set, namely:

CR-5.1 external temperature has to be measured with a minimal precision of 5 degrees celsius

CR-5.2 must be able to make a distinction between day and night

CR-5.3 should be able to collect data about the humidity of its environment

5.2 External temperature

One set of the additional sensors are the temperature sensors with which the external temperature can be measured. These sensors can be helpful for data collecting of the surroundings, but also can this information be important for the functioning of other sensors, such as for example an ultrasound sensor where the propagation speed of sound is dependent of temperature. Besides that, the external temperature can be used for safety purposes, such as for example knowing if there is fire somewhere nearby.

5.2.1 Sensors

For measuring the external temperature an accuracy of 5 degrees celsius is required (CR-5.1), because this gives a maximal deviation of 1% in the data of the distance sensor. One solution would be an integrated infrared camera [22]. The data coming from this would also be usefull for other applications, but the high costs makes it not interesting. Another solution is to use a simple IC temperature sensor. They are cheap and have simple linear read-out. Also the component itself is small in form factor. Even simpler is the Resistance Temperature Detector (RTD), which effectively is a temperature dependent resistor. Making a voltage divider with and additional resistor would make it work, but it is not practical to calibrate for multiple issues.

Finally the MCP9701-E/TO [23] IC temperature sensor is chosen, because of its price and simplicity (S-6, D-1). This sensor is connected to an ADC of the PCB. The voltage measured is transformed with the formula of Equation 5.1:

$$A_{in} \times \frac{V_{cc}}{T_c \times 1023} - \frac{V_0}{T_c} = T_{out} \tag{5.1}$$

With A_{in} the 10-bit ADC readout value. $V_{cc} = 5.0$ V the supply voltage, $T_c = 0.0195$ volt per degree celsius the voltage to temperature coefficient and $V_0 = 0.400$ V the sensor output at 0 V.

5.2.2 Implementation

Three temperature sensors were placed on the outside of the enclosure of the observation module: one at the front and the other two, one at each side. Three sensors are chosen, because this is needed for making the system more robust (criterion C-4) and this covers most directions. There will be no temperature sensor on the backside of the enclosure, because there it faces to another module and thus not provide relevant information.

Making the system robust

If one of the sensors is malfunctioning, this will give an extreme temperature, namely -20.5 or 235.9 degrees, since A_{in} will be at maximum or minimum value. This will change the average temperature of the three sensors enough to notice by the software. There is expected that Zebro works in temperatures between the 5 and 40 degrees celsius. If the average of the three sensors becomes outside this range, there will be checked if actually one of the sensors data deviates too much, more than x degrees celsius, from the other two sensor data. If this is not the case the temperature measured will be send to the toplevel of the observation Module. But if one of the sensors data deviates more than x degrees celsius from one of the other sensor data, it can be concluded that one of the temperature sensors is malfunctioning and this will be sent to the toplevel of the Observation Module (requirement S-4). After the same results of a malfunctioning sensor is found several times consecutively, action should be taken to check the sensors.

Determine value x

As mentioned before a limit value x is needed, so this one has to be determined. Because the extreme in minus direction is much smaller than in positive direction, two different values of x were chosen in this two cases, namely x_negative and x_positive. The datasheet [23] states the sensor has an accuracy of ± 4 $°C$, thus the maximum the sensor could differ from each other is 8 $°C$. For this reason both x_negative and x_positive should be above four. There can be a situation in which one of the sensors will be heated more than the other two (criterion C-5). In this case the average can become above 40 degrees and there will be tested if one sensor is malfunctioning. For this reason the value of x_positive should be set high to prevent the system from thinking one sensor is malfunctioning, without this is even the case. Because of the great values of the extremes the difference in malfunctioning and functioning sensors can easily be found. for this reason the x_negative cannot be above the 17.5 degrees (5+20.5-8) and the x_positive not above 187.9 (235.9-40-8). For the reasons the value of x_negative is set on 17 degrees an the value of x_positive is set on 100 degrees.

5.2.3 Applications

The following applications can be done with the data from the external temperature sensors.

Data collecting

The data collected from the three temperature sensors were translated into the temperature as calculated with Equation 5.1. The average of this data is calculated and sent to the toplevel of Zebro, so other sensors can make use of this data.

Fire

The other application with external temperature data is sensing fire. If the average temperature of the sensors is above a certain predetermined value, an error bit will become high and will communicate with the Zebro toplevel to say it should walk into another direction (requirement S-5).

5.3 Light intensity

Another additional sensor is the light intensity sensor, this sensor can be used to see the difference between day and night and also between a bright day or a gloomy day. This information will be sent to the toplevel of Zebro (criterion C-6) and to the sensors that need to know if it is day or night for example a camera without infrared/night view, or lights that need to be turned on or off can make use of this info.

5.3.1 Sensors

For sensing the light intensity several types of sensors can be used, such as an IC like the TSL257-LF [24]. It outputs a voltage according to an incoming irradiance. It will however saturate with indoor lighting and thus cannot be applied. A simpler solution is a voltage divider made of a LDR in series with a resistor of the LDRs characteristic resistance. The voltage can be read-out with an ADC. It is also the low-cost.

Finally a LDR is chosen, because only the distinction between 3 states (night, gloomy day, day) should be extracted from the sensor data. Besides its capability it is also cheap (criterion C-4).

5.3.2 Implementation

A voltage divider is created using an LDR in series with a resistor of the LDR its characteristic resistance. The voltage in between the resistances is readout by an 10-bit ADC. The data coming from the 10-bit ADC is directly used to determine the state.

The LDR is placed on top of the enclosure. In this way the light can be received in a perfect way without any other part of the observation module creating a shade on the sensor. The data about the situation; day, night, gloomy day will be send to the toplevel of Zebro. The distinction is based on threshold values are based on the values found Figure 5.1.

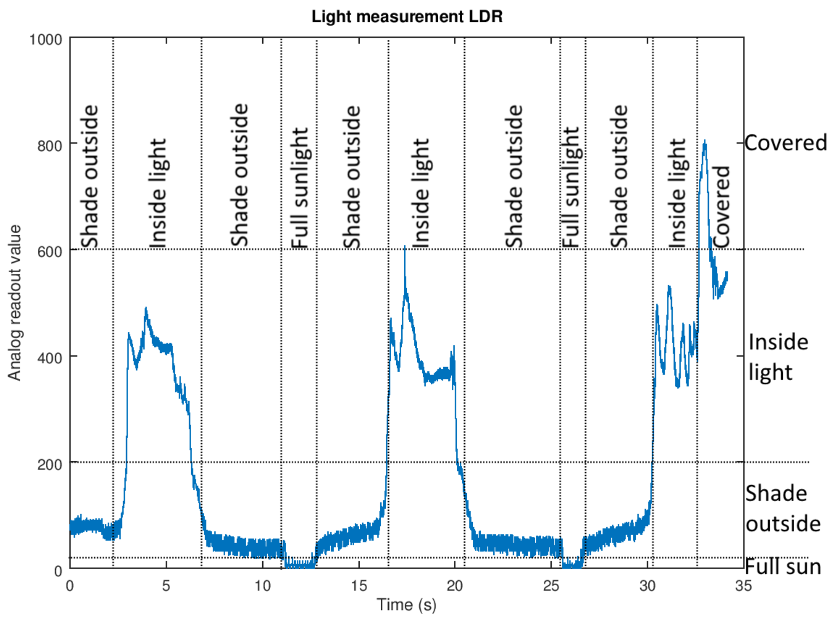


Figure 5.1: Plot of the LDR readout voltage over time, walking inside en outside on a sunny day.

The figure is made by reading the ADC when walking in and out of a building. Inside the lights were on and outside was first shaded by the building and then full sun. In the figure the transitions are denoted.

From the figure can be concluded what values belong to which state. Assuming inside lighting corresponds to a gloomy day it would be: everything below 200 day, everything below 600 a gloomy day and everything above night.

5.4 Humidity

A third additional sensor is the humidity sensor, that is able to measure the relative humidity in the air (CR-5.3). This data is mostly meant for data collection, which can for example be used for other sensors that are influenced by humidity. This data is available to the toplevel of Zebro and will be available for other sensors.

5.4.1 Sensors

There are several techniques to implement a humidity sensor, which were discussed by Z.M. Rittersma [25]. From this review became clear that any technique meet the requirements, but it should have integrated technologies to compensate for errors introduced by the technique.

The SHT21 [26] is chosen, because it features a simple readout and is cheap (criterion C-4), that keeps the module price low. The sensor features self-calibration via an internal heater and internal temperature sensor.

5.4.2 Implementation

The humidity sensor is placed within the enclosure at the top edge of it. In this enclosure one or more little holes will be made, so the humidity sensor can sense its surrounding in a correct way. The data of the humidity sensor is in the form of the relative humidity compared with the air temperature.

The sensor is addressed via I^2C and it outputs a 12 bit value that can be converted to a relative humidity (RH) via Equation 5.2.

$$RH = -6 + 125 \times \frac{A_{in}}{2^{16}} \tag{5.2}$$

5.5 Applications of sensors

The described sensors came from the idea that Zebro could come in handy when used in agricultural context. In agricultural context the following applications come in handy.

5.5.1 Rain detection

With the combination of the sensors rain could be measured. If the relative humidity rises to 100 $\%$, the temperature drops a bit and light intensity tells it is night or a gloomy day then it is likely that it rains.

5.5.2 Water

The detection is quite the same as the rain detection, but the changes are within a shorter time. Most likely it is too late to warn the top-level to go back, but for the time being an alarm could go off to request help from passengers. This application is not yet implemented.

5.5.3 Tropical day

With the combination of the external temperature and the light sensor it is possible to detect with more certainty that it is a sunny or even tropical day.

5.6 Conclusion

With the help of a temperature sensor the external temperature can be measured with an accuracy of 4 degrees (CR-5.1). With the light intensity sensor, Zebro is able to no the difference between day and night (CR-5.2) and is able to know how humid its environment is (CR-5.3).

Chapter 6

Power and safety

This chapter discusses the power consumption, power considerations, possible solutions and the power solution finally used for the submodule. Three different systems are evaluated (BT-2).

6.1 Power consumption

One thing that has to be known before designing the power protection system of the Observation Module (requirements EM-6, BT-4 and S-3) is the power required for a correct functioning of the module (requirement EM-5). This can be determined by collecting the maximum currents and voltages needed of each part of the Observation module that is used in the module. In Table 6.1 an overview of all the parts of the Observation module and corresponding maximum currents and voltages are shown.

Table 6.1: All parts in module with corresponding current and voltage

Part	**Current (mA)**	**Voltage (V)**
Accelerometer/gyroscope (MPU-6050/9150) [16, 17]	3.9	3.3
Atmega2560 [27]	200	5
Distance sensor [28]	20	5
Humidity sensor [26]	0.33	3.3
LEDs (24x) [29] [30]	1440	5
Laser [31]	25	5
Raspberry Pi Zero W [32]	150	5
Camera [33]	250	-
Servo [34]	400	5
Speaker [35]	50(@6V)	5
Temperature sensor [23]	12	5
cliff sensor (2x) [36]	80	5
light sensor [37]	75	5
Needed:	2700	5

According to Table 6.1 a power supply of 5 V @ 2.7 A is required. But this current will not be used in in practice, because not all parts will be used at its maximum currents at the same time. Whilst booting the module the current draw will mostly be determined by the Atmega 2560 and Raspberry Pi Zero that are powering on. The LEDs and servo are turned off whilst booting, so a peak current of approximately 0.4 A is needed during booting. On top of that under working conditions the LEDs will never use their maximum current, because they are software limited to a maximum brightness and only a certain colors will be used. To know how much current is exactly needed under working conditions, the following test is done.

6.1.1 Current test

With this test all the parts of the Observation Module were connected to an Arduino Mega 2560. With the help of two multi-meters, one in current and one in voltage mode, the current and voltage needed can be measured. The test setup is shown in Figure 6.1. The current sensor is put in maximum hold mode. In this way the maximum current used during the measurement will be shown on the screen.

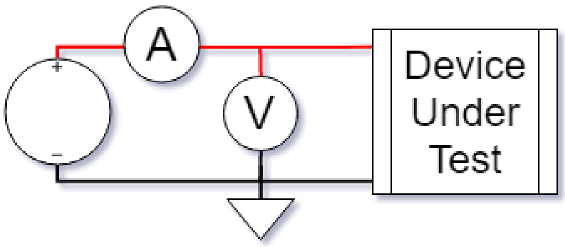


Figure 6.1: Setup of the current test

Results

After switching the system on and off for several times and letting the system do things simultaneously a maximum current of 0.392 A is found @ approximately 5 V (5.202 V). This measurement is done without the use of the Raspberry Pi and everything connected to this, such as the Pi camera and the laser. After the part of the Raspberry Pi was ready for testing this is done in the same way. This gives a maximum current of 0.886 A @ approximately 5 V (5.170 V).

Conclusion

From this test can be concluded a maximum current needed for the system will become 0.886 A. From this can be concluded that a solution with 5 V @ 1 A is enough for the system to work properly.

6.2 Temperature

To protect the Observation Module from overheating and malfunctioning, the internal temperature has to be measured (S-3). The internal temperature should not rise above the lowest maximum operating temperature of all the components that are enclosed in the same area as the powering part. If the internal temperature comes close to this operating temperature, the components that produce a significant amount of heat should be shut down to cool the internal system.

6.2.1 Implementation

In the first place a temperature sensor is needed. There are many different temperature sensors. The quality of the temperature sensor is not needed at very high standards, its accuracy and precision should be about 5 degrees Celsius. The precision should not become a too high value, because this makes the temperature range to work is smaller (max working temperature-precision). This will cause the system to switch on and off frequently, which can give problems. The precision should not have to be to a small value, because this accuracy is not needed to know whether overheating occurs. For this reason a precision of 5 degrees celsius is chosen. For this reason the same sensor can be used as the one used for measuring the external temperature, as is described in chapter 5. This is the MCP9701-E/TO [23] and is chosen because of their price and simple analog readout.

After some testing and research it seems to be that the servo can draw a significant amount of current and thereby produces indirectly the most heat. An error bit (explained in subsection 7.1.1) will be set high (requirement S-5) if the temperature is above a certain level. The PCB will react on this error bit and stops sending a PWM signal to the servo after which the system will cool down. This level will be determined by the lowest maximum operating temperature of the devices in the enclosure of the Observation module. If the temperature still rises after disabling the servo, other modules like the Raspberry Pi and everything connected to it will be shut down to decrease the power used any further. In Table 6.2 an overview of all the internal components with its temperature function ranges is shown.

Table 6.2: Internal component temperature function ranges

Part	**Minimum(°C)**	**Maximum (°C)**
Accelerometer/gyroscope (MPU-6050/9150) [16, 17]	-40	85
Atmega 2560 [27]	-55	125
Humidity sensor [26]	-50	125
Raspberry Pi Zero [32]	0	70
Servo [34]	0	55
Speaker [35]	-30	70
Temperature sensor [23]	-40	150
Needed:	0	55

From the table can be concluded that a maximum temperature of 55 °C should be used as a limit for internal overheat. From this an extra safety of 4 degrees is taken into account, because the precision of the temperature sensor chosen is 4 degrees [23]. When the temperature becomes higher than 51 degrees Celsius the error bit becomes high. When the temperature becomes below 41 degrees Celsius the error bit will become low. This range of 10 degrees Celsius is taken into account to prevent the system from flipping on and of if it is around the limit. This maximum and minimum limit are put as variables in the code, so these can be changed if new components with lower maximum working temperature are placed in the same enclosed area.

6.3 Requirements

The requirements for designing the power and safety part that has to be taken into account from chapter 3 are EM-4, EM-5, EM-6, BT-4, S-1, S-3, S-6 and criteria C-1 and C-4. On top of that some criteria are added that are specific for this chapter and can be derived form the information of section 6.1:

CR 7.1 It should be able to deliver 5 V @ 1 A from a 4 S LI-Po battery, thus 12.8 V-16.8 V with a cell voltage 3.2 V to 4.2 V (3.7 V nominal).

CR-6.2 Should be able to control temperature and protect system by switching off above 55 °C.

6.4 Buck converter and switch

One possible solution for powering the Observation Module is a buck converter with a separate switch. The general idea of this, is that the buck converter will take care of the right voltage for the $S\&S$ module (CR-6.1). The switch will take care of the safety part of the input, namely when the input current or the internal temperature becomes too high (called over-current and internal overheat from now on) the switch cuts off the power supply from the Observation Module (requirement S-3). In this way no more heat will be produced by the components of the module. The only part that should not be powered off is the PCB, because this is the only part that is able to regulate the switch and communicate with the toplevel of Zebro (requirement S-7). For this design also a current sensing part is needed.

6.4.1 Design

Switch

The switch can be created with the help of a transistor or by using a relay. When a transistor is used, it can be regulated with a signal coming from the PCB. When using a relay there are some types that are able to measure the voltage. If a certain limit is reached the relay will turn on or off dependent on the type. When an over-current or internal overheat occurs, the switch will take care of switching off the power supply from the rest of the system, so the module will not damage other parts.

Buck converter

Buck converter circuits can be bought online. This components are expected to be reliable enough for use, so it is not needed to create a self-designed system for it.

Current measurement

For doing the current measurement a separate circuit can be made or a complete component can be bought online. Such a circuit can be for example as shown in Figure 6.2.

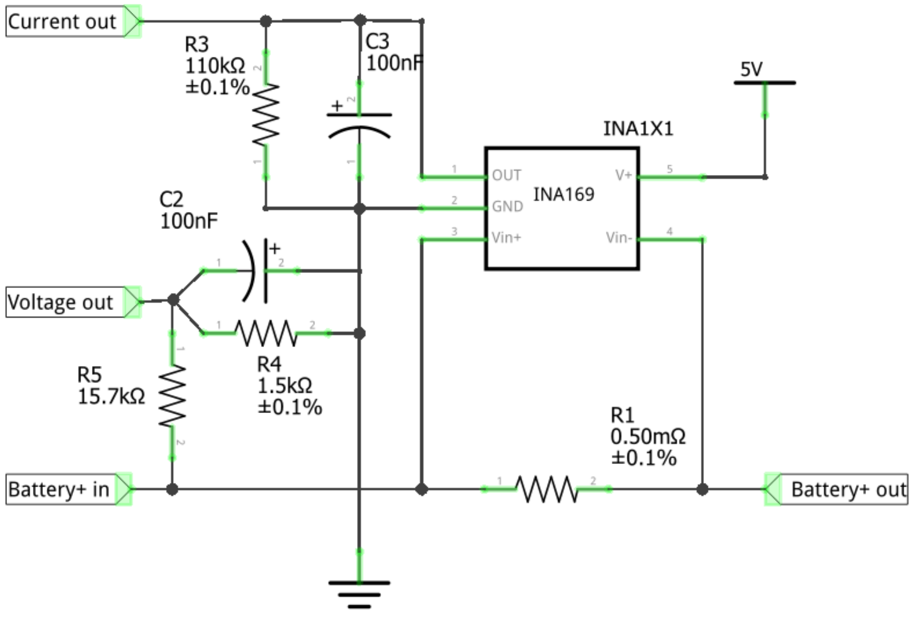


Figure 6.2: Current sensor circuit (with voltage divider) [38]

In this figure the in- and output are in line with the supply line of the PCB. The voltage difference over resistor $R1$ is measured and then converted to a current using the $INA169$ [39]. By changing the value of resistor $R3$ the output voltage range can be regulated.

In Figure 6.3 the global design of the buck converter with switch is shown.

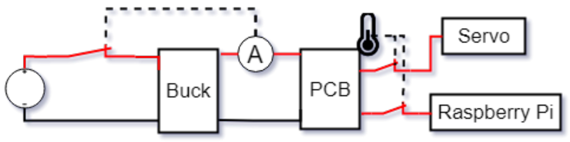


Figure 6.3: Design of buck converter and switch

6.4.2 Advantages and disadvantages

There are some advantages and disadvantages for using this design. The advantages are:

+ The voltage and current can be easily regulated by the buck converter (CR-6.1 and criterion C-4)
+ The whole system can be disconnected from the power supply (requirement S-3)
+ Communication with the top level of Zebro is still possible after an internal overheat of the system (S-7)
+ Current can be monitored with a current sensing circuit (S-3)

The disadvantages are:

- Separate hardware components, that makes it more difficult and expensive for mass production (criterion C-4)

6.5 LDO regulator

Another possible solution is a LDO regulator, Low-dropout regulator. An LDO regulator can regulate the output voltage by a range input voltages (CR-6.1). Some of these regulators are also equipped with current and overheat protection. This means that if the temperature of the LDO regulator itself will become above a certain device dependent limit it will drop its voltage to stop producing more heat. Also if over-current of the regulator occurs it will drop its voltage. In this way the power supply will be ”switched off” when over-current or overheat of the device occurs ((S-3).

6.5.1 Design

An LDO regulator is one component that can be bought as a whole, equipped with overheat and over-current protection. The only part that has not been taken into account with this solution is the internal overheat problem. As explained in subsection 6.2.1 the servo only needs to be switched off by internal overheat. The servo can be turned off by stopping to send a PWM signal to it, which can be done by software. So if internal overheat occurs, the module will stop sending a PWM signal to the servo. After this less current is needed, which in turn causes the LDO regulator to produce less heat. If the module is cooled down enough the PWM signal can be send out again. If it is not enough to cool the system down by shutting down the servo, the Raspberry Pi can also be switched off.

6.5.2 Measurements

A LDO regulator of 5 V @ 3 A is tested. It is powered by an adjustable voltage source. The LDO regulator is attached to an heat sink to make it possible to go to the device its limits without it switched off by overheat, before proper measurements were done. The test setup can be seen in Figure 6.4, were the load resistance can be regulated. By changing the load, both the output current and voltage of the LDO regulator will change. In this way can be seen if the device works properly. In Figure 6.5 the test results are shown. In this figure an arrow is placed at the highest possible current for that voltage.

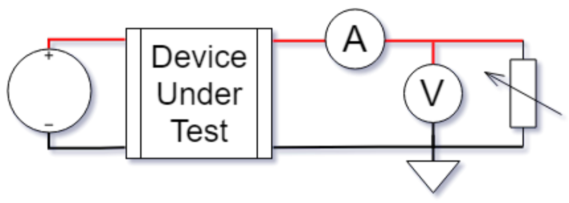


Figure 6.4: Setup for testing the LDO/Buck converter

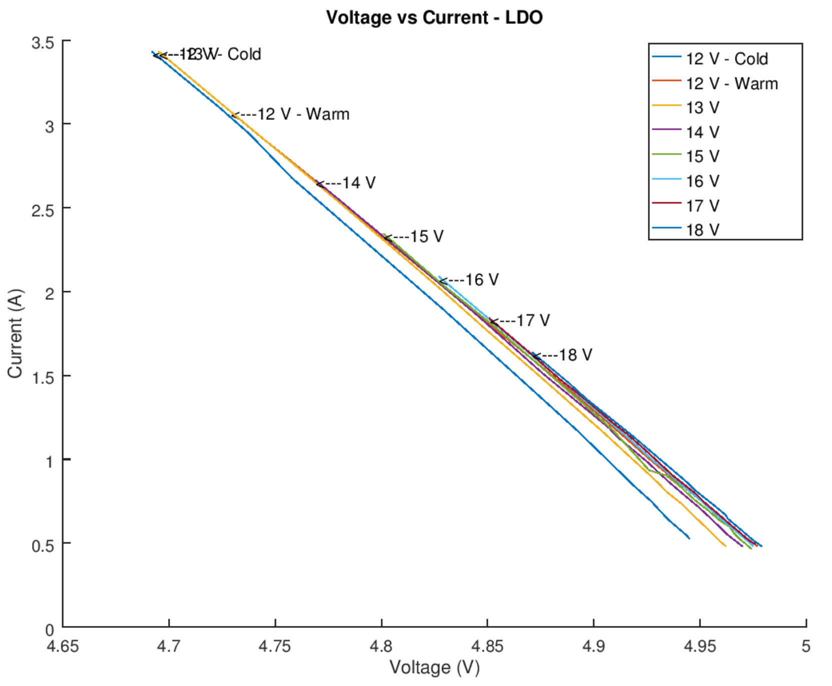


Figure 6.5: Voltage and current vs load LDO

Also a temperature measurement is done, to know how fast the device heats. In Figure 6.6 the test results of the temperature of the device is shown over time. In this figure can be seen that the LDO saturates at around 65 $°C$.

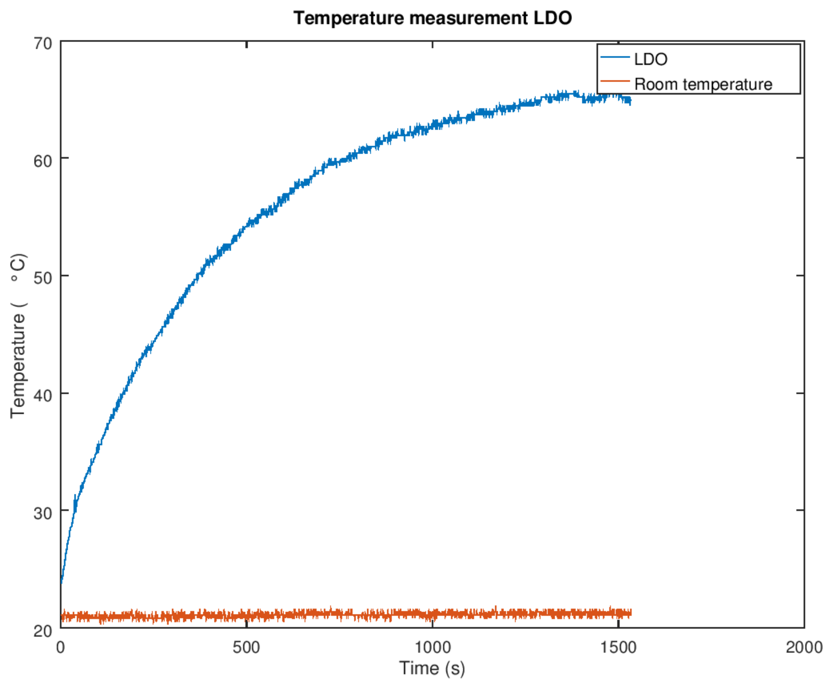


Figure 6.6: Temperature over time at 5 V / 1 A LDO

6.5.3 Advantages and disadvantages

There are some advantages and disadvantages for using this design. The advantages are:

+ One hardware component, easy to reproduce (criterion C-4)
+ Communication with the top level of Zebro is still possible after an internal overheat of the system (S-7)
+ The voltage and current can easily be regulated by the LDO (CR-6.1)
+ Current can be monitored by the LDO component itself (S-3 and criterion C-4)

The disadvantages are:

- Communication with Zebro top level not possible, while switched off by over-current.
- Only a part of the system can be disconnected when overheating occurs
- LDO produces a lot of heat

6.6 Buck converter

After some more research another solution was found, namely a buck converter with current protection integrated (S-1, S-3, criterion C-4). This component will be able to protect the system from over-current and provide the system of the right voltage and current supply.

6.6.1 Design

For this design only one hardware component is needed (Criterion D-1). This component takes care of the over-current protection and providing the system of the right power and voltage. The only thing the hardware component is not capable of, is internal overheat protection. It is the same as the LDO regulator mentioned in section 6.5, that is able to measure its own overheating, but is not able to regulate the internal temperature of the whole Observation module. Just as the LDO regulator design, the Buck converter solution will only switch off the PWM signal of the servo and if that does not work also the Raspberry Pi with everything connected to it, to cool the system by internal overheat. The temperature is measured with the same temperature sensor.

6.6.2 Measurements

A buck converter of 5 V @ 1 A is tested in the same way as the LDO regulator is tested. The test setup can be seen in Figure 6.4, where the load is also adjustable. In Figure 6.7 the test results are shown.

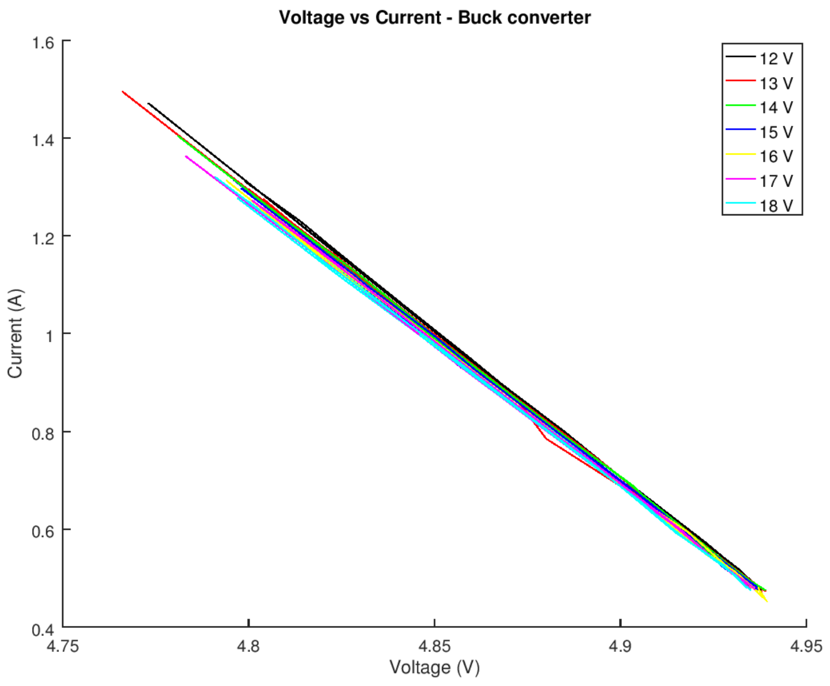


Figure 6.7: Voltage and current vs load, Buck

Also a temperature measurement is done for the buck converter to know how fast it heats up. In Figure 6.8 the test results of the temperature is shown over time.

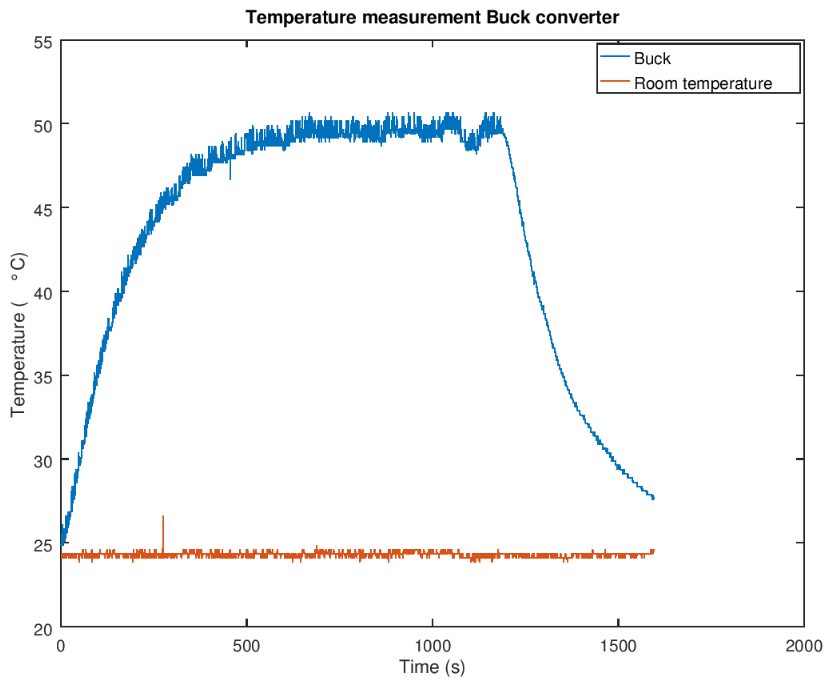


Figure 6.8: Temperature over time at 5 V / 1 A, Buck

6.6.3 Advantages and disadvantages

There are some advantages and disadvantages for using this design. The advantages are:

+ One hardware component (criterion C-4)
+ The voltage and current can be easily regulated by the buck converter (CR-6.1)
+ Current can be monitored by the component itself (criterion 4.4 and S-3)
+ Communication with the top level of Zebro is still possible after an internal overheat of the system (S-7)
+ Temperature is reasonable under load

The disadvantages are:

- Communication with Zebro top level not possible after switching of by over-current.
- Only parts of the system can be disconnected when a too high internal temperature is detected

6.7 Final design power and safety part

For the final design the buck converter is used, because this is easier for mass-production than the buck converter with switch and it produces less heat than the LDO regulator. One of the disadvantages of the buck converter was that the communication with the toplevel of Zebro is disconnected after switching off by over-current. The question is, is this really a problem? When disconnecting the communication, the toplevel knows something is wrong. This can have two reasons; something is wrong with the communication or the Observation Module is shut down, because of a defect or over-current. So the toplevel will try to reach the Observation module several times, if this does not work it know something is defect and the module should be inspected.

The other disadvantage is that only parts of the system will be disconnected when overheating is detected. This should not be seen as a disadvantage, because in this way the other data of the sensors, can still be collected and transmitted to the toplevel. The PCB will send the toplevel a high error bit, so it knows the servo and/or the Raspberry Pi is off, because of overheat.

Conclusively, this final design as shown in Figure 6.9, is a good design for our purposes. With this design the Observation Module can work safely (EM-6 and BT-4) and has enough power to function properly (S-1), with the manufacturing kept in mind (criterion D-1).

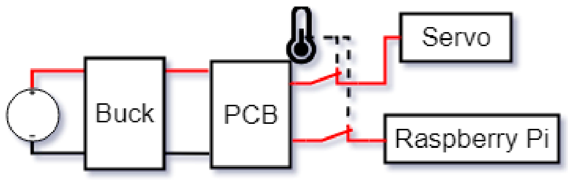


Figure 6.9: Final design for power and safety of Observation Module

Chapter 7

Software

With the subdivision of the Observation Module to the $S\&S$ submodule also came the task of integrating all different software of the submodules into one software code. The first decision to be made was the hardware platform, what became the Arduino [40] platform, because of its cheap boards, availability in the research team, vast amount of example code and libraries and simple IDE.

7.1 Toplevel

The top-level is supposed to be as short a possible, since it only connects code. However, there is still a structure in it, as can be seen in Figure 7.1. In the inclusions there is a special file named $PINout.h$. This is the file in which the overview of pins is shown. It is managed by the maker of the PCB [19] to keep integration on PCB easy.

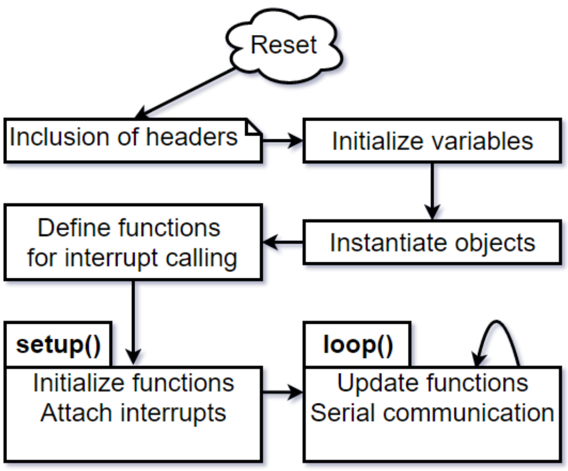

Reset
Inclusion of headers
Initialize variables
Define functions for interrupt calling
Instantiate objects
setup()
Initialize functions Attach interrupts
loop()
Update functions Serial communication

Figure 7.1: Overview of the Software

The header files that are included are as follows:

- BuzzerClass.h - Control of the piezo speaker. Got heartbeat and alarm mode
- CliffClass.h - Detection of cliff
- CONFIG_class.h - Configurator for the other classes
- DEBUG_class.h - Controls the output to the serial monitor of the classes
- DistanceClass.h - Controls the servo + sonar for object detection
- GyroAccel_class - Uses the MPU-6050 to detect too steep surfaces and pick-ups

- LEDringClass.h - Controls the LEDring
- PINout.h - Pinout map to the Arduino
- SHT2xClass.h - Readout of the humidity sensor
- SimpleSensorsClass.h - Readout of the temperature sensors and LDR
- Main.ino - The top-level of the Observation module

7.1.1 Error byte

The most important variable in the Observation Module is the ”Error_Byte” variable. This variable tells whether problems/ errors have occurred and this will directly trigger some functions, like the buzzer. Each bit in the byte has a different meaning, which is as follows:

Table 7.1: Error_Byte outline

#Bit	Meaning
7	-
6	-
5	-
4	Cliff at the right of Zebro
3	Cliff at the left of Zebro
2	Zebro is on a too steep surface
1	Zebro is being lifted or fell off something
0	The Module has detected an internal overheat

If an error pops up the Module will try to contact the Zebro top-level at once via the ZebroBus.

7.1.2 Classes

One of the requirements made to the other submodules was that their code should work in classes. This is done so functions and variables would not have duplicate names as it makes integrating easier. Within their header files the following things should be specified in comments:

- [FILENAME].h
- Author(s)
- Date of last edit
- Build according to (library):
  - (link to github)
  - Author
  - Code date
  - Licence of the library
- Calling functions:
  - In variable specifications: [CLASSNAME] foo([GLOBAL_VARIABLES]);
    - Other global variables that need instantiating (preferably none)
  - In $void\ setup()\{\}$: foo.Initialize();
    - Other things that need initialization (preferably none)
    - Interrupt calls
  - In $void\ loop()\{\}$: foo.UpdateValues([OTHERS_VARIABLES]);
    - Other functions that can be called in the main loop (preferably none)

- List of public variables with their respective meaning
- Used libraries
- Used definitions
- To do's

The code should be implemented according to this comment structure.

foo.Initialize()

In the initialization all ports required for the class should be set, as all object instantiated. As last function the $UpdateValues()$ function should be called to test its own sensor. For debug purposes on start up the following structure was used, with the serial info in between.

- Serial.println("→ [CLASSNAME]");
- Serial.println("\t[ACTION1]");
- [ACTION1];
- Serial.println("\t[ACTION2]");
- [ACTION2];
- ...
- Serial.println("\t[Check [CLASSNAME]]");
- UpdateValues();

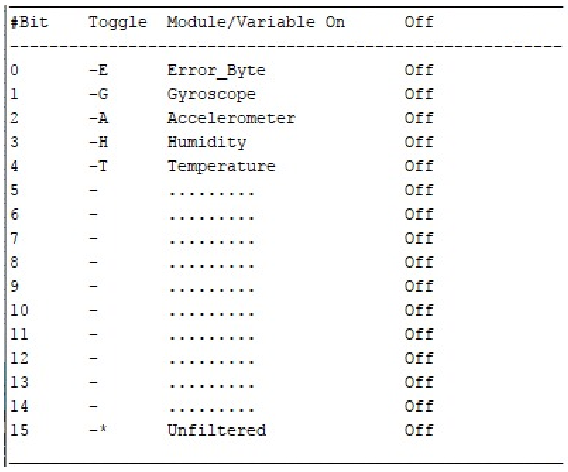
With this structure defects can be found in the early stage, as it can determine were code breaks.

foo.UpdateValues()

In this functions the class should do its task where it is meant for. One of the main rules is that no blocking elements like $delay([TIME\_IN\_MS])$ are allowed, to speed up the code. Delays should be tracked by keeping track of time between function calls and if needed converted to interrupts.

7.1.3 DEBUG

One of the special classes is the $DEBUG$ class. This class can be called on the $Serial$ port by sending "-". It will send back a list of possible commands to enable certain classes their intermediate variables and outputs (C-2, S-4, S-7). In Figure 7.2 an example is showed.



```
#Bit    Toggle  Module/Variable On      Off
---------------------------------------------------------
0       -E      Error_Byte              Off
1       -G      Gyroscope               Off
2       -A      Accelerometer           Off
3       -H      Humidity                Off
4       -T      Temperature             Off
5       -       .........               Off
6       -       .........               Off
7       -       .........               Off
8       -       .........               Off
9       -       .........               Off
10      -       .........               Off
11      -       .........               Off
12      -       .........               Off
13      -       .........               Off
14      -       .........               Off
15      -*      Unfiltered              Off
```

Figure 7.2: Debug list of the Observation Module

The $\#Bit$ value is used in the code to check whether this class should output its value. This is done with the member function DEBUG.isset([$\#Bit$]), which returns a Boolean, where true means that output is allowed.

7.1.4 CONFIG

The CONFIG class is still being developed, but the idea is that all ”#DEFINE” statements will be transferred to this class and that it can be changed via the serial port and eventually the ZebroBus (C-2, S-7).

7.2 ZebroBus

To connect the module to the Zebro top-level the ZebroBus [41] has been developed. It is based on the working principles of I^2C, but can work over other protocols, for example it has already been implemented using Bluetooth. Its main features are virtual registers and unicast, multicast and reserved addresses. With these features several architectures are possible, for example: all sensors can be directly addressed by the Zebro top-level .

The structure used by the Observation module can be seen in Figure 7.3. In this the sensor is not directly addressable by the Zebro top-level , but is first processed by the module itself.

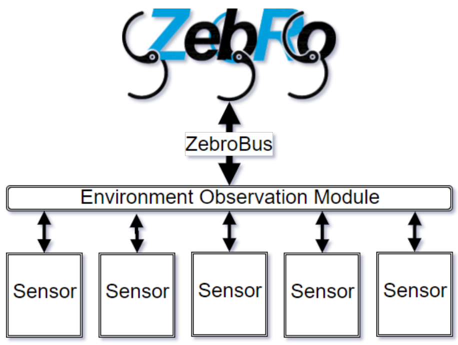


Figure 7.3: Overview of the Environment Observation Module with the ZebroBus

The Module will present its Error_Byte to the ZebroBus, as it will present other functional data to it like temperature and humidity. The outputs of the sensors discussed before will be available on the ZebroBus (C-4, S-5 and S-7). The integration is still in progress.

7.2.1 Software I2C

With the development of the code a problem raised: the Arduino platform only supports one hardware port for I^2C. Two of the sensors (the MPU-6050 and the SHT21) were interfaced with I^2C as well and thus some of the code had to be changed to comply with the ZebroBus standard. With the use of the ”SoftwareWire” library [42] it was possible to simply change ”Wire.” to ”MyWire.”, which was declared earlier in the code, except for the MPU-6050 code. This one had more options for different kinds of I^2C implementations and thus needed a thorough look-through. With the help of declaring ”MyWire” $extern$ all problems where solved.

7.3 Conclusion

The software used in the Observation Module is bound to some design rules. With this structure it is tried to keep the code as fast as possible, as to keep it convenient. With the Module top-level code the following requirements are met: C-2, C-4, S-2, S-4, S-5 and S-7. Besides that it supports criteria C-1 and D-1.

Chapter 8

Integration and Discussion

In this chapter can be read how the submodules of the Observation module are integrated and how it functions. Also can be read if the results are as expected and how this can be improved.

8.1 Integration

After the sensors were tested individually and they met the requirement, they were integrated in the Modules top-level software. A class specific for the sensor is written according to the structure given in subsection 7.1.2. The sensors are then retested to check for faults in the code. After that it is integrated in the top-level and added to the DEBUG class. From the tests can be concluded that the sensor is fully functional it so it can be integrated in the enclosure [19].

8.2 Testing and results

After integration of the code the top-level is tested. The code is tested on performance by measuring the execution times of the $foo.UpdateValues()$ member functions and tested on functionality by displaying the sensor output data via the DEBUG class. If data is not correct, the code will be reintegrated.

Current execution time results are displayed in Table 8.1.

Table 8.1: Execution times of sensor classes

Class name	**Execution time (ms)**
GyroAccel	22
SHT2x	< 1
SimpleSensors	< 1
Distance	< 1
LEDring	< 3
Buzzer	< 1
Cliff	< 1

Via this method it was possible to see that the SHT2x had an execution time of around 120 ms. After further investigation it was found that the sensor blocked the code whilst measuring. After switching this to a non-blocking measurement the execution speed was less than 1 ms. This is however not the update time, but only the time needed to check the sensor for available data.

8.3 Discussion

Via this testing method it is possible to guarantee that the sensors work together in the same way as they did individually (BT-1). For this it is necessary that the initial code works properly. During this project the Zebro was not yet walking and thus not all code could be optimized before integration. With the integrated sensors almost all $S\&S$ requirements were met. Only S-7 is not yet reached because of problems with the ZebroBus. In the beginning it was said it was possible to propose a ZebroBus specification working on another protocol than I^2C. Later on this was recalled and it was required to work on I^2C. This required the work as described in subsection 7.2.1. An overview of the used materials and their respective prices is shown in Appendix A.

Chapter 9

Conclusion

The Observation Module created is more or less ready, to be placed on Zebro. With this Module Zebro is able to do things as was required in chapter 3.

The system is able to protect itself against damage from over-current or internal overheat (EM-6). For this the internal temperature is measured (EM-4), and the current, voltage and power flow of the system is controlled by a buck converter (EM-5). Thus the general requirements are met.

By making the choices of the sensors first several sensor were researched or tested (BT-2). With the individual tests the performance was measured of the sensors, after which the Module was tested as a whole. The performance was recorded and improved (BT-1). BT-4 is met by meeting the requirements for EM-6.

The observation module is able to work properly without needing any information from other parts of Zebro (C-1). Criterion C-2 is partially met with the $S\&S$ submodule; the control & status interface is made. Criterion C-4 is still being integrated and is currently not yet met. With the use of the ”too-steep” and the ”pick-up” detectors a contribution is deliverd to criterion C-5. During the selection of technology and sensors D-1 was kept in mind.

From the Sensor and Safety requirements only one is not yet met. Requirement S-7 will be met when the ZebroBus communication is functional within the Module.

With only the ZebroBus not implemented it can be concluded that the Sensor and Safety module is successful, but needs some further work. Other future work is also recommended.

Chapter 10

Recommendations and future work

In this chapter the recommendations to the Zebro team are described. These recommendations are extra design tips to create extra features for Zebro. The first feature will take little time to implement and has a high priority to do so. The other three features are just ideas to expand the Zebro Observation Module and need some more time to design and implement. In this chapter the target, design and possible implementation of the features were described. Finally some recommendations are given that are recommended to be done before this Observation Module can function in the right way on a functioning and walking Zebro.

10.1 Controlling the sound

As can be read in ”Zebro is lifted” in section 4.4.1 Zebro sets an error bit (explained in subsection 7.1.1) high after it is lifted. With this a speaker goes off, which can be switched off by pressing a button, after which the error bit gets low. This is not really sufficient if someone that lifts Zebro can switch off the sound by him/herself. Also it will become very unpractical if Zebro walks and screams at the same time with someone chasing after it to stop the sound. For this a solution was conceived, that was not implemented yet, because the Deci Zebro was not walking at this time.

10.1.1 Design and implementation

The solution for this problem is that Zebro is able to shut off its own sound by knowing it is put down again. When Zebro is walking a certain power on the legs of Zebro can be measured. When this power is measured the top-level of Zebro can send a signal to the Observation Module to shut off the sound. This is also a good solution for the problem of faults in the ”zebro lifted” part of the system. When Zebro thinks it is lifted, but it actually is not, the power of the legs will measured. If it is above a certain threshold the top-level will send a signal to the Observation Module to shut off the sound. This will be done this fast, that only a little beep will be heard. This design can also be used when Zebro has fallen, for example when Zebro lays somewhere where it cannot be seen, on its back or somewhere he cannot reach the ground with its legs. If it gets in such a position the sound will attract attention and hopefully someone will help it. In the mean time Zebro has to wait until it can reach the ground with its legs again.

10.1.2 Conclusion

This part of the design has to be implemented by the designers of the Zebro top-level . The Observation Module is ready for receiving a signal that can shut off the sound. This solution is both for a Zebro that is lifted and one that is fallen. Also in an emergency situation where it cannot safe itself, for example where it fell on its back or if someone walks with Zebro in his/her hands this solution can be really useful.

10.2 Sound of its surrounding

Another extra feature proposed for the Deci Zebro, is to sense its surrounding with hearing. This means that it will able to listen in which direction the noise is coming from and decide by itself if it walks in the direction of the noise or to walk to a more silent place. In this way the Zebro team will be able to give each Zebro its own character.

10.2.1 Design and implementation

For this the following design is conceived. Four microphones will be placed on the enclosure, each in one direction as can be seen in Figure 10.1.

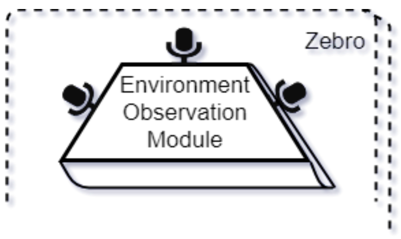


Figure 10.1: Design monitoring sound of its environment

For this application it is recommended to use unidirectional microphones, because the distance between the microphones on the enclosure will be so small that the data otherwise will not give any measurable difference in noise. By using the data coming from the microphones it is possible to detect if there is something making noise and from which directions it comes from. This information can then be send to the top-level Zebro to decide which direction Zebro will walk to.

10.2.2 Conclusion

This design needs to be elaborated more and the theory lined out has to be rechecked. This can give Zebro extra features and can create different characters for each Zebro. Creating this feature needs some more time and maybe it will be implemented by someone from the Zebro team. The disadvantage of this feature can be that it makes Zebro too intelligent for what it was intended for.

10.3 Feelers

The Zebro design is inspired by looking at insects. Insects are able to feel their surrounding by using their feelers. This brings us to the last extra feature proposed for the Deci Zebro. With the current Observation Module Zebro is already able to know if something is in front of it and will be able to determine the size of the object. By giving Zebro its own feelers, it will also be able to feel if the object will move away by pushing it or feel the objects characteristics. People could argue why Zebro needs to be able to walk against objects in the first place, when it is able to walk around it. The answer to this is that each Zebro can have its own character. Also by developing Zebro, it might be able to learn from things done before, so this feature can contribute to both this character and learning possibility.

10.3.1 Current situation

At first glance it seems to be unnecessary to give Zebro some extra feelers to touch the objects detected, because currently there is also a solution available at Zebro. At this time, when Zebro wants to know if a object can move, it can walk against the object and measure the power of the legs to know if the object will move or not. But it can be unwise to do this, because in this way Zebro or one of its components can be damaged if it underestimates the danger of the object.

10.3.2 Static design

One of the proposed solutions for this is a static design. This will just be two static cylinders mounted on the front of Zebro that will reach further than the legs of Zebro. These ”feelers” will be able to touch the object and try to move it before Zebro itself will walk against it. On top of this, the feelers could be fitted with some extra pressure sensors, so the pressure put on the object can be measured. If this becomes above a certain limit, Zebro can assume that the object will not move, so it can decide to walk in another direction. The design is showed in Figure 10.2.

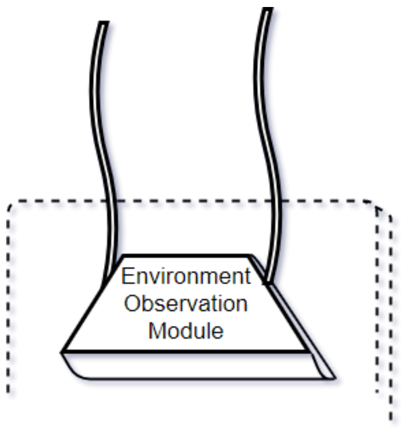


Figure 10.2: Static design of feelers

10.3.3 Dynamic design

Another, more natural and dynamic solution is to design two linear actuators on top of the Observation Module. In this way the feelers can be controlled and used when desired. On the ends of these ”feelers” pressure sensors can be placed to measure the pressure used to move the object. Another way of doing this is by measuring the power of both the linear actuators. If the value of the pressure exceeds a certain level, Zebro can assume that it will not move away and can walk another direction. The design is showed in Figure 10.3.

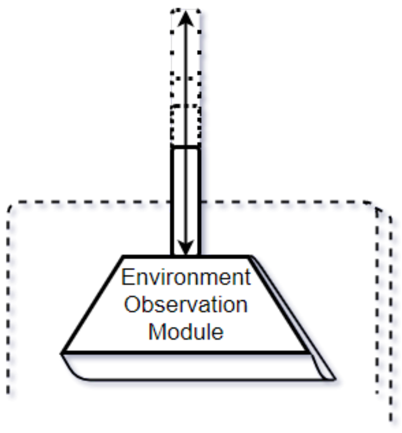


Figure 10.3: Dynamic design of feelers

10.3.4 Conclusion

There are several solutions to give Zebro the ability to move away an object. These designs have each their advantages and disadvantages. The main disadvantage of the static and dynamic design is that they both work on only one place in a fixed direction and on a fixed height. But no matter which design will be used, both need some more time spend on the design and research of their working principle.

10.4 Zebro is moving

From the data coming from the gyroscope and the accelerometer can be determined whether Zebro is moving or standing still. This is not implemented yet, because this data was not needed for the current sensors and Deci Zebro is not able to walk at this time.

10.4.1 Implementation

By monitoring the acceleration in the z-direction and the gyroscope, it can be determined if Zebro is walking. This can be done by characterizing the walking motion of Zebro and compare that with the current situation. If it does not comply with current situation and there are no changes it can be concluded that Zebro is standing still. When the current data does not comply with the characterized walking and Zebro is not standing still an error has arose.

10.4.2 Conclusion

Before it is able to implement this, some measurements have to be done on a walking Zebro. After this research has been done, can be concluded if this will work or not. If the principle can work it can be useful information for other sensors that will be added in the future and are dependent of movement.

10.5 Recommendations before use

There are also some main recommendations that the Zebro team needs to do before it is ready for use.

- The MPU has to be calibrated before use. For which the code is already available.
- The offsets of the MPU has to be filled in the code, this can be measured when the chip is placed in Zebro.
- Some data of the accelerometer has to be gathered while Zebro is walking on both a flat ground and a rough ground, to find a new limit for ’Zebro is lifted’ code.
- It needs to be tested at what slope Zebro is able to walk and this variable has to be adapted in the ’too steep’ code. This value should below 40 degrees.
- When some devices are adapted or new sensors were added to the Observation Module and is placed inside the enclosure. There needs to be checked if the Maximal internal temperature is good enough for this device to work properly or that this should be changed.
- Check if the switch is in the right position to enable the buzzer.

Appendices

Appendix A

Bill of Materials (BOM)

This is the BOM of the whole Observation module. Be aware that the given prices are only indicative. These prices may change over time and are dependent on the supplier used. Any discounts that the TU Delft may have have not been taken into account.

Table A.1: Bill of Materials (BOM)

Component	Part number	#	€/ 10	€/ 100	Reseller NL	€/ 100	Reseller World
Buck Converter	173010578	1	8.32	6.70	Farnell	0.50	AliExpress (alternative)
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	Sharp GP2Y0A21YK0F	2	8.66	7.18	Farnell	3.45	AliExpress
Laser	KY-008	1	1,95	1,95	hobbyelec-tronica	0.67	AliExpress
Pi Camera V2		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	EuroCircuits	0.66	ALLPCB
Plastic Enclosure		1	13.14	11.68	3Dhubs	11.10	3Dhubs (China)
			130.37	111.68		71.59	

Bibliography

[1] T. D. R. Institute, “Mission and vision,” p. Front page, May 2017, accessed on 16-06-2017. [Online]. Available: https://tudelftroboticsinstitute.nl/about-us/mission-and-vision

[2] D. B. Chris Verhoeven and E. Hakkennes, *Research and Development of Environment Observation Module for Nano Zebro*, 2017th ed., Zebro Project research team, HB 18.130, Mekelweg 4, 2628 CD, Delft, Zuid-Holland, The Netherlands, January 2017.

[3] C.-Y. Chong and S. P. Kumar, “Sensor networks: evolution, opportunities, and challenges,” *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug 2003.

[4] L. Slats, “Building thriving communities: To realize the things network’s mission of building a global, open and crowdsourced internet of things network,” Master Thesis, Delft University of Technology, Prometheusplein 1, 2628 ZC Delft, Zuid-Holland, The Netherlands, October 2016, accessed on 16-09-2017. [Online]. Available: http://resolver.tudelft.nl/uuid:3ee2b9ba-8c08-4b4e-9d64-b985aeeeb288

[5] A. Ollero, P. J. Marron, M. Bernard, J. Lepley, M. la Civita, E. de Andres, and L. van Hoesel, “Aware: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned aerial vehicles,” in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, Sept 2007, pp. 1–6.

[6] D. Piyabongkarn, R. Rajamani, and M. Greminger, “The development of a mems gyroscope for absolute angle measurement,” *IEEE Transactions on Control Systems Technology*, vol. 13, no. 2, pp. 185–195, March 2005.

[7] S. Pala, M. Çiçek, and K. Azgin, “A lorentz force mems magnetometer,” in *2016 IEEE SENSORS*, Oct 2016, pp. 1–3.

[8] H. Chen, D. Sun, J. Yang, and W. Shang, “Orientation correction based monocular slam for a mobile robot,” in *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, July 2008, pp. 1378–1383.

[9] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (slam) problem,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun 2001.

[10] M. Switch, *Rolling Ball Tilt Switch*, November 2006, accessed on 29-05-2017. [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/010/132/original/MS-100906.pdf

[11] J. Titus, L. Tetrault, and J. Peters, “Tilt switch array for electronic orientation detection,” Sep. 21 1999, uS Patent 5,955,713. [Online]. Available: https://www.google.com/patents/US5955713

[12] R. Goodrich, “Accelerometers: What they are & how they work,” LiveScience, October 2013, accessed on 16-04-2017. [Online]. Available: https://www.livescience.com/40102-accelerometers.html

[13] G. Pang and H. Liu, "Evaluation of a low-cost mems accelerometer for distance measurement," *Journal of Intelligent and Robotic Systems*, vol. 30, no. 3, pp. 249–265, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1008113324758

[14] B. Li, B. Harvey, and T. Gallagher, "Using barometers to determine the height for indoor positioning," in *International Conference on Indoor Positioning and Indoor Navigation*, Oct 2013, pp. 1–7.

[15] A. M. Dujon, R. T. Lindstrom, and G. C. Hays, "The accuracy of fastloc-gps locations and implications for animal tracking," *Methods in Ecology and Evolution*, vol. 5, no. 11, pp. 1162–1169, 2014.

[16] InvenSense, *MPU-9150 Product Specification Revision 4.3*, InvenSense Inc., 1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A., September 2013, accessed on 28-04-2017. [Online]. Available: https://www.invensense.com/wp-content/uploads/2015/02/MPU-9150-Datasheet.pdf

[17] ——, *MPU-6000 and MPU-6050 Product Specification Revision 3.4*, InvenSense Inc., 1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A., August 2013, accessed on 28-04-2017. [Online]. Available: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[18] J. Rowberg, "I2c device library collection for avr/arduino or other c++-based mcus," March 2017, accessed on 03-05-2017. [Online]. Available: https://github.com/jrowberg/i2cdevlib

[19] L. De Herdt and J. M. Buis, "Obstacle and cliff detection for robotics applications using miniaturized sonar and ir distance triangulation," June 2017, submodule of the Environment Observation Module.

[20] O. Oosterlee and S. Peterse, "Obstacle detection using laser triangulation and optical flow," June 2017, submodule of the Environment Observation Module.

[21] Arduino, "Mpu-6050 accelerometer + gyro," Arduino AG, accessed on 13-05-2017. [Online]. Available: https://playground.arduino.cc/Main/MPU-6050

[22] Webmaster, "Lwir uncooled thermal camera core products," FLIR® Systems, Inc., FLIR Systems, Inc. 27700 SW Parkway Avenue Wilsonville, OR 97070, 2017, accessed on 17-09-2017. [Online]. Available: http://www.flir.com/cores/display/?id=51981

[23] M. T. Inc., *MCP9700/MCP9700A/MCP9701/MCP9701A Data Sheet*, Microchip Technology Inc., 2355 West Chandler Blvd., Chandler, AZ 85224-6199, U.S.A., March 2014, accessed on 03-05-2017. [Online]. Available: http://www.farnell.com/datasheets/2124381.pdf?_ga=2.16331317.2031955116.1497367993-338716377.1479406068

[24] AMS, *TSL257-LF*, AMS, July 2006, accessed on 17-05-2017. [Online]. Available: http://www.farnell.com/datasheets/2258619.pdf?_ga=2.246455428.1613771824.1497799559-2050599603.1492887917

[25] Z. Rittersma, "Recent achievements in miniaturised humidity sensors—a review of transduction techniques," *Sensors and Actuators A: Physical*, vol. 96, no. 2, pp. 196 – 210, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924424701007889

[26] SENSIRION, *Datasheet SHT21*, SENSIRION AG, Laubisruetistr. 50, CH-8712 Staefa ZH, Switzerland, October 2011, accessed on 10-05-2017. [Online]. Available: https://anel-elektronik.de/SITE/produkte/sensor_1/Datasheet%20SHT21.pdf

[27] A. Corporation, *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V*, Atmel Corporation, 1600 Technology Drive, San Jose, CA 95110 USA, February 2014, accessed on 15-05-2017. [Online]. Available: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

[28] C. T. S. Bhd, *Product User's Manual - HC-SR04 Ultrasonic Sensor*, Cytron Technologies Sdn. Bhd, No. 16, Jalan Industri Ringan Permatang Tinggi 2,Kawasan Industri Ringan Permatang Tinggi, 14100 Simpang Ampat,Penang, Malaysia, May 2013, accessed on 02-05-2017. [Online]. Available: https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit

[29] Kuohuai, *WS2812B*, Shenzhen Worldsemi Technology Co. Ltd., Shenzhen, China (Mainland), 2013, accessed on 17-05-2017. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

[30] P. Burgess, *Powering NeoPixels*, Adafruit, June 2017, accessed on 17-05-2017. [Online]. Available: https://learn.adafruit.com/adafruit-neopixel-uberguide/power

[31] K. E. B.V, “Laser diode - 5mw 650nm rood,” Kiwi Electronics B.V, accessed on 25-05-2017. [Online]. Available: https://www.kiwi-electronics.nl/laser-diode-5mw-650nm-red?search=ADA-1054

[32] R. P. FOUNDATION, “What are the power requirements?” RASPBERRY PI FOUNDATION, accessed on 24-05-2017. [Online]. Available: https://www.raspberrypi.org/help/faqs/#Power

[33] ——, “How much power does the camera module use?” RASPBERRY PI FOUNDATION, accessed on 24-05-2017. [Online]. Available: https://www.raspberrypi.org/help/faqs/#cameraPower

[34] TowerPro, *MG90S*, TowerPro, 2014, accessed on 15-05-2017. [Online]. Available: http://www.towerpro.com.tw/product/mg90s-3/

[35] pro signal, *Electromechanical Audio Transducer 6V*, Premier Farnell, April 2012, accessed on 05-05-2017. [Online]. Available: http://www.farnell.com/datasheets/1563662.pdf?_ga=2.186989096.1030272784.1497289930-101769311.1476733867

[36] S. Corporation, *GP2Y0A21YK0F*, SHARP Corporation, January 2006, accessed on 04-05-2017. [Online]. Available: http://www.sharp.co.jp/products/device/doc/opto/gp2y0a21yk_e.pdf

[37] Sunrom, *Light Dependent Resistor - LDR*, Sunrom Technologies, Ahmedabad, Gujarat, India, July 2008, accessed on 05-05-2017. [Online]. Available: http://kennarar.vma.is/thor/v2011/vgr402/ldr.pdf

[38] B. T. V. Faris Elghlan, Sander van Leeuwen and Rico Tubbing, *Project AIR*. Delft University of Technology, Januari 2017.

[39] T. Instruments, *INA1x9 High-Side Measurement Current Shunt Monitor*, Texas Instruments Incorporated, Post Office Box 655303, Dallas, Texas 75265, February 2017, accessed on 17-05-2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/ina139.pdf

[40] Arduino, “Arduino - home,” Arduino AG, accessed on 15-05-2017. [Online]. Available: https://www.arduino.cc/

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

[42] T. P. Raul, Tod E. Kurt, “Creates a software i2c/twi bus on every pins,” May 2017, accessed on 25-05-2017. [Online]. Available: https://github.com/Testato/SoftwareWire