ROBOTICS PROJECT COURSE CDT508/DVA410

PROJECT VASA SYSTEM OVERVIEW

January 23, 2012

Micael Östberg *et al.*
micael.ostberg@baggetorp.com
http://www.projectvasa.com

Supervisor:
Mikael Ekström

Mälardalen University
Västerås, Sweden

Abstract

A robot that is able to control itself solely based upon sensor data is of great value to any field which today uses so called robots controlled by humans. One variant of an underwater robot is being described in this report. It was done as a student project with the goal of competing in the 2012 RoboSub competition in San Diego.

The robot is named Vasa after the famous Swedish warship with the same name which sunk in the year 1628. The robot is designed to be completely autonomous and acts only using its sensors and actuators to complete the tasks given from the RoboSub competition specifications.

In this report the authors have described thoroughly the different parts of how the robot is constructed. It includes the design and construction process for electronics as well as for mechanics. The reports also states on how the robot has been programmed with handling all the data and logic to fulfill the goals of the competition.

Document version

Version	Date	Note
1.0	2012-01-23	Initial release

Table of Contents

1 Introduction **7**

2 Method **9**

3 The competition **10**
3.1 Overview . . . 10
3.2 Static judging . . . 10
3.3 Missions . . . 10
3.3.1 The gate . . . 10
3.3.2 The path . . . 11
3.3.3 Training . . . 11
3.3.4 Obstacle course . . . 12
3.3.5 Gladiator rings . . . 12
3.3.6 Kill Caesar . . . 13
3.3.7 Feed emperor grapes . . . 13
3.3.8 Laurel wreath . . . 14

4 Mechanics **15**
4.1 Introduction . . . 15
4.2 Method . . . 15
4.3 Old designs . . . 15
4.3.1 First design . . . 15
4.3.2 Second design . . . 15
4.4 Frame . . . 16
4.4.1 Overview . . . 16
4.4.2 The frame . . . 16
4.4.3 Mounts . . . 17
4.4.4 Camera mounts . . . 17
4.4.5 Thruster mounts front and back . . . 17
4.4.6 Thruster mounts middle . . . 17
4.4.7 Thruster mounts horizontal . . . 18
4.4.8 Tube mounts . . . 18
4.4.9 IMU mounts . . . 19
4.4.10 Gripper mounts . . . 19
4.5 Thruster configuration . . . 20
4.6 Watertight compartments . . . 22
4.6.1 Overview . . . 22
4.6.2 Tubes . . . 22
4.6.3 Camera housing . . . 23
4.6.4 IMU box . . . 24
4.7 Electronics support system . . . 24
4.7.1 Overview . . . 24
4.7.2 ESS structure . . . 24
4.7.3 Cooling fans . . . 25
4.8 External connectors . . . 26
4.8.1 Overview . . . 26
4.8.2 Front hull connectors . . . 26
4.8.3 Rear hull connectors . . . 26
4.8.4 External component connectors . . . 27
4.9 Kill switch and mission switch . . . 27

4.10 Equipment . . . 27
4.10.1 Grippers . . . 27
4.10.2 Torpedoes . . . 27
4.10.3 Markers . . . 28
4.11 Testing the robot . . . 30
4.11.1 First assembly of Vasa . . . 30
4.11.2 First pool test . . . 30

5 Electronics **31**
5.1 Introduction . . . 31
5.2 Method . . . 31
5.3 Background . . . 33
5.4 Battery . . . 33
5.5 Sensors and cameras . . . 33
5.6 Shark board . . . 33
5.6.1 Backbone . . . 33
5.6.2 Router card . . . 35
5.6.3 GPIO card . . . 35
5.6.4 Solenoid card . . . 36
5.6.5 Sonar card . . . 36
5.7 Power distribution board . . . 38
5.7.1 General functionality . . . 38
5.7.2 Power requirements and load handling . . . 39
5.7.3 Kill switch, emergency stop . . . 39
5.7.4 Misc and layout . . . 40
5.8 Motor controller . . . 40
5.9 System verification . . . 41

6 Software **42**
6.1 Introduction . . . 42
6.2 Vision System . . . 44
6.2.1 Cameras . . . 45
6.2.2 OpenCV . . . 50
6.3 Main CPU . . . 52
6.3.1 High level CPU introduction . . . 52
6.3.2 Choice of hardware . . . 52
6.3.3 Design of high level CPU . . . 52
6.3.4 Threaded system . . . 54
6.3.5 State machine design . . . 55
6.4 Router board . . . 56
6.4.1 The communication protocol . . . 56
6.5 Sensor board . . . 57
6.5.1 The pressure sensor . . . 57
6.5.2 The IMU . . . 57
6.6 Power distribution board . . . 59
6.6.1 Design of the Power distribution board . . . 59
6.6.2 Mounted LCD screen . . . 59
6.6.3 Flow of events . . . 59
6.7 Motor controller . . . 61
6.7.1 Output signals for controlling the motors . . . 61
6.7.2 Communication test . . . 61
6.7.3 Balancing Vasa . . . 62

6.7.4 Depth control . . . 62
6.7.5 Roll control . . . 62
6.7.6 Pitch control . . . 62
6.7.7 Yaw control . . . 63
6.8 PD-controller . . . 64

7 Results **67**
7.1 Mechanics results . . . 67
7.2 Electronics results . . . 67
7.3 Software results . . . 67

8 Discussion and conclusion **69**

9 Acknowlegments **70**

References **72**

Appendix A Software API **73**
A.1 Vision API . . . 73
A.2 Statemachine API . . . 74
A.3 Motorcontrol API . . . 76
A.4 Sensor API . . . 77

Appendix B CAN-messages **79**

Appendix C Electronic schematics **80**

Appendix D Electronics test protocols **105**
D.1 Unit Power distribution . . . 105
D.1.1 U-P-001 . . . 105
D.1.2 U-P-002 . . . 105
D.1.3 U-P-003 . . . 106
D.1.4 U-P-004 . . . 107
D.1.5 U-P-005 . . . 107
D.2 Unit Router . . . 108
D.2.1 U-R-001 . . . 108
D.2.2 U-R-002 . . . 109
D.2.3 U-R-003 . . . 109
D.3 Unit Sensor . . . 110
D.3.1 U-S-001 . . . 110
D.3.2 U-S-002 . . . 111
D.3.3 U-S-003 . . . 111
D.3.4 U-S-004 . . . 112
D.3.5 U-S-005 . . . 113
D.4 Unit Shark Board . . . 113
D.4.1 U-M-001 . . . 113
D.4.2 U-M-002 . . . 114
D.4.3 U-M-003 . . . 115
D.4.4 U-M-004 . . . 116
D.4.5 U-M-005 . . . 116
D.4.6 U-M-006 . . . 117
D.4.7 U-M-007 . . . 118
D.5 Unit SONAR . . . 118

D.5.1 U-S1-001 . . . 118
D.5.2 U-S1-002 . . . 119
D.5.3 U-S1-003 . . . 120

Appendix E Mechanics exploded views **121**

Appendix F Thruster configuration **126**

Appendix G Mechanical drawings **129**

Appendix H User guides **153**
H.1 Electrical . . . 153
H.2 **Camera tutorial** . . . 156
H.2.1 Coriander . . . 156
H.2.2 Capture a frame with libdc1394 . . . 160
H.2.3 Conversion to OpenCV . . . 160
H.3 Mechanical Guide . . . 162
H.3.1 Step 1: Attach thruster to mount . . . 162
H.3.2 Step 2: Assemble camera . . . 162
H.3.3 Step 3: Attach to the frame . . . 162
H.3.4 Step 4: Attach thruster to mount . . . 162
H.3.5 Step 5: Attach tube mounts to frame . . . 168
H.3.6 Step 6: Assemble the IMU-box . . . 169
H.3.7 Step 7: Assemble the middle camera . . . 171
H.3.8 Step 8: Attach gripper mount . . . 173
H.3.9 Step 9: Attach thruster to mount . . . 174
H.3.10 Step 10: Attach excenter locks to the frame . . . 176
H.3.11 Step 11: Assemble the tubes . . . 177
H.3.12 Step 12: Attach thruster to mount . . . 186

Appendix I Retailers and sponsors **188**

Appendix J Bill of materials **189**

1 Introduction

Underwater robots offers endless of possibilities in the field of naval exploration, warfare and even navigation. This is because the robots are without any direct control by a human being. But as ROV's (Remotely Operated Vehicles) are getting more and more common for naval operations, there is also the constant need for a human to control its every motions. As opposite to ROV's there is also the less commonly known autonomous kind of underwater robot systems, called AUV (Autonomous Underwater Vehicle). Vasa is an AUV since it is designed to completely operate on its own, without any external contact or communication.

There are several scenarios where an AUV would be useful, especially in hazardous environments and under dangerous operations. These underwater operations could be:

- **Detection of underwater mines**
- **Construction and reparation under water**
- **Exploration of underwater caves**
- **Environmental sanitation**
- **Surveillance of coasts**
- **Rescue missions**

These are only a few prospective operations mentioned, the list can be made much longer.

Figure 1: VASA

With this in mind, Vasa is designed to be modular in the fields of mechanics, electronics and software design. With a modular design it is easier to expand the intent of usage for the robot as it is very easy to construct and add new modules to the Vasa.

The team is well confident that Vasa robot will be attractive in terms of having a well functioning starting platform that a company or a person easily can evolve to fit its own special needs.

The team consisted of eleven master students studying at MDH (Mälardalen University) which all came from different backgrounds and specialization areas of expertise. Vasa was constructed after a previous AUV project (RALF II) at Mälardalen University. As the team felt that the older platform had flaws in terms of modularity and expandability, the team constructed a new model with just these parts as the core motivator. The team hopes that the Vasa will be worked on by new students in the future and will take on the advantage of all its features and open possibilities.

The project was financed by MDH and various sponsors. All throughout the project this has limited the purchases and pushed the students to find the best solution to the lowest cost.

The students involved developed the robot Vasa with the initial field of usage to be a competitor for the 15th annual RoboSub competition. This competition is held by the AUVSI foundation and the competition itself will be in San Diego, July 2012. The competition is designed to be simple yet challenging. The competition is designed as a student competition so that students from all around the world are able to compete with other students and compare each others constructions at a live competition.

This report provides an extensive overview and abstraction as how Vasa is designed and constructed by the students taking the course CDT508 at Mälardalen University in fall 2012.

Mechanics group members:

- Rickard Linder (Mechanics team leader)
- Micael Östberg (Project leader)
- Anton Widenius
- Ammar Ismail

Electronics group members:

- Johnny Holmström (Electronics team leader)
- Athar Ahmed
- Rafat Ghanim
- Ejaz Ui Haq

Software group members:

- Lars Lagerholm (Software team leader)
- Peter Wåhlin
- Mingli Wu

To read the project blog, access more pictures, watch movies and for asking questions visit Project Vasa's homepage: http://www.projectvasa.com

2 Method

One of the earliest things that the developing team behind Vasa did was to first evaluate what each team member had interest in and what they wanted to specialize themselves in while constructing the robot Vasa. This division was successful and the team was fully able to choose which kind of field they most likely wanted to work with throughout the project course.

The team was divided into the following subgroups:

- Mechanics
- Electronics
- Software

Each subgroup had to choose one group leader that was suppose to be in charge of keeping the group together and synchronizing the workload between the group members and always pushing forward towards the common goal. The group leader was also in charge of writing more specific documentation that would only affect the specific group, this could for example be project plans so the group could easily see what has been done and what still needed development. This was the main roll of the group leaders as well as communicating more closely with the team leader. Each group had different kinds of protocols for internal communication between the members. An example of this was that the electronics group had weekly meetings so that everybody was up to speed to what was being done and needed to be done. This can be compared to the software group which did not have any kind of regular meetings, this was mainly because of the smaller size of the software group. So this group only talked over things when specific news needed to be shared among the members.

The project leader was the one that always tried to keep track of all the work that was done throughout the project course. One of the biggest role that the project leader had during the project was to keep the groups synchronized as well as communicating with the supervisors/customers. The project leader had regular meetings with the group leaders, this was to get an update in time plan/workload and internal communication between students and group. As well as these tasks that were stoved upon the project leader there is also worth mentioning that the leader did general documentation that concerned the whole project. This was often done with the help of all or the individual group leaders.

Most of the work took place at Mälardalen University during day hours and mostly full time. This timetables changed throughout the course as some students went down halftime and also some other students worked sometimes more than full time. Mälardalen University provided the team full access to the school workshop equipment. With this equipment the team was able to effectively construct prototype circuit boards as well as quickly mill out CAD drawings directly in aluminum or plastics. In order to synchronize all the work that was being done the team used Subversion (SVN). With this helpful tool the team could easily share documents and files between each other in a very fast and reliable way.

3 The competition

3.1 Overview

Although Vasa has been designed with modularity in mind so it can be used in various situations after this project ends, the end goal of this project is to compete in the International RoboSub Competition in San Diego 2012.

RoboSub is an international competition where AUVs (Autonomous Underwater Vehicle) from all over the world compete. The participants are all students from either universities or high schools and the 2011 competition attracted students from USA, Canada, Japan, Iceland, India and China, making it a truly international competition. As quoted on the RoboSub website [10]:

"The goal of this competition is to advance the development of Autonomous Underwater Vehicles (AUVs) by challenging a new generation of engineers to perform realistic missions in an underwater environment. This event also serves to foster ties between young engineers and the organizations developing AUV technologies."

It is indeed a great opportunity for the team members to evolve their engineering skills and at the same time make new connections to other like-minded students and people from different organizations.

The competition itself consists of several parts and all of them are scored and summed up to a total score that crowns the winning team. Some parts are subjective and judged such as craftsmanship, video, journal paper and website while others are scored based on pure performance, i.e. the missions the robot should perform. The competition has had different themes throughout the years, naming each mission related to that theme. This year they have chosen a gladiator theme.

3.2 Static judging

The parts of the competition which are judged and scored all give different points and although the missions will give the teams most of the points, the static judging can be the difference between getting the winners glory or not. These are the parts that will be scored:

- Total weight of the robot
- Utility of website
- Technical merit (from journal paper)
- Written style (from journal paper)
- Technical accomplishments (from static judging)
- Craftsmanship (from static judging)
- Team uniform (from static judging)
- Team Video
- Discretionary static points (awarded after static judging)

As can be seen there are a lot of factors that has to be considered when entering this competition.

3.3 Missions

3.3.1 The gate

The first mission is a relatively simple one. The robot has to pass through a green gate [Figure 2] with the size of 1.8x1.2m made out of 5cm PVC pipes. This gate is also used in the qualifying rounds where the robots have to pass through it to be able to enter the semi-final round.

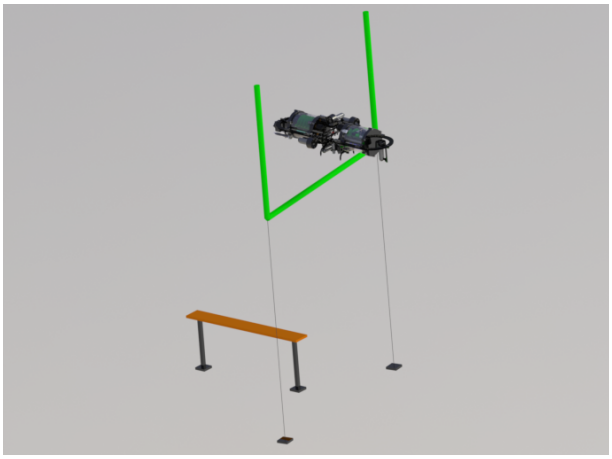

Figure 2: The first mission and qualifying rounds.

3.3.2 The path

Throughout the entire course there will be a total of six path segments placed. These segments are all placed after the current task and points to the next task. The path segments are 1.2m long and 15cm wide in a bright orange color and one can be seen in [Figure 3]. The segments will be placed around 0.3-0.6m above the bottom of the arena.

3.3.3 Training

The training mission [Figure 3] has 3 buoys with red, green and yellow colors, all 23cm in diameter. The goal of this mission is to touch 2 of the buoys in the correct order with the AUV. The judges will announce what colors and in what order to be touched just before the round starts. The buoys are all placed within a 91x91x240cm segment 2.1m above the arena bottom. Each buoy is separated by 1.2m and can be placed at different heights inside the box.

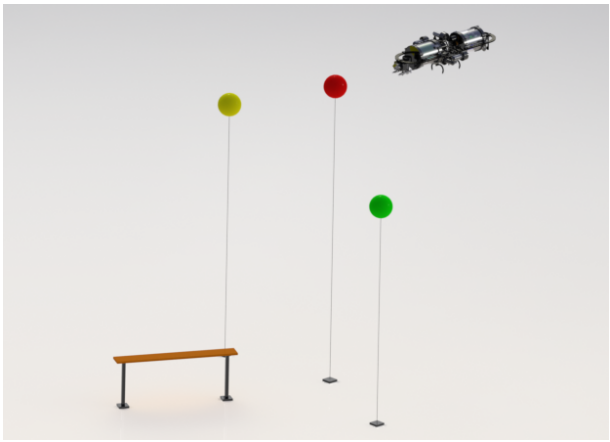

Figure 3: Vasa together with the buoys and a path segment.

3.3.4 Obstacle course

There will be three obstacle gates [Figure 4] placed in the arena which of the robots have to pass through them of them. They are "U"-shaped gates with two sections removed made out of 5cm PVC pipes in green color. To get full points for this mission the robot must pass inside the vertical segment and ½ or more of its height below the virtual top line.

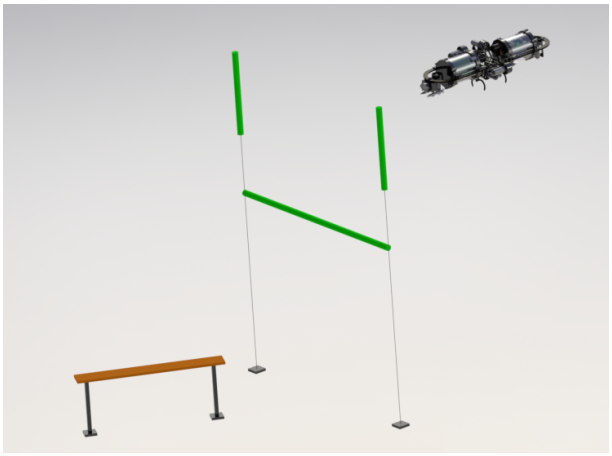

Figure 4: The obstacle course.

3.3.5 Gladiator rings

The robot should drop two markers inside two out of four bins [Figure 5]. The bins have a unique picture inside them, two swords and two shields so the robot can distinguish them. The judges will tell which marker should be put in which bin just before the round starts. Maximum points are awarded when the correct marker is placed entirely within the right bin.

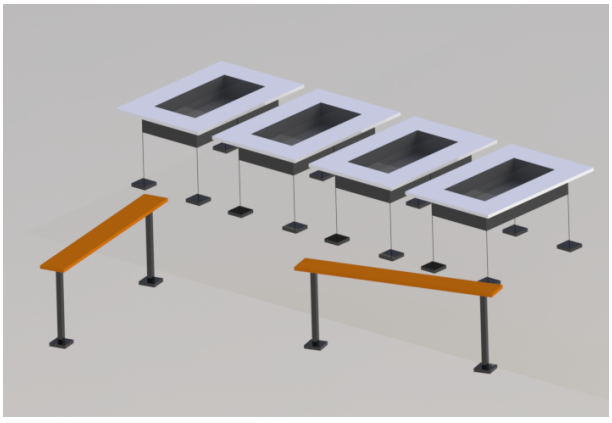

Figure 5: The bins that the markers should be dropped in.

3.3.6 Kill Caesar

Kill Caesar is the mission where the robot should fire torpedoes through specific holes placed in a board [Figure 6]. The board is 61x61cm large and red on one side, blue on the other. The board has two holes in it, one small and one large. Maximum points are awarded when shooting the red marked torpedo through the small hole on the red side and the blue marked torpedo in the small hole on the blue side.

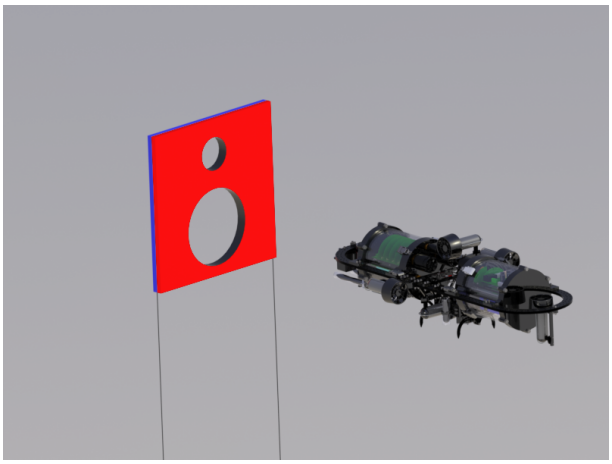

Figure 6: The board with holes which the torpedoes should be fired through.

3.3.7 Feed emperor grapes

This mission is a new mission for this year [Figure 7]. It is a type of manipulation task, where the robot should move a small red pipe, one horizontal and one vertical. Both the pipes are placed on a tab and should be moved either up for the horizontal or right for the vertical. The tabs are located on a yellow board, 1.2x1.2m in size.

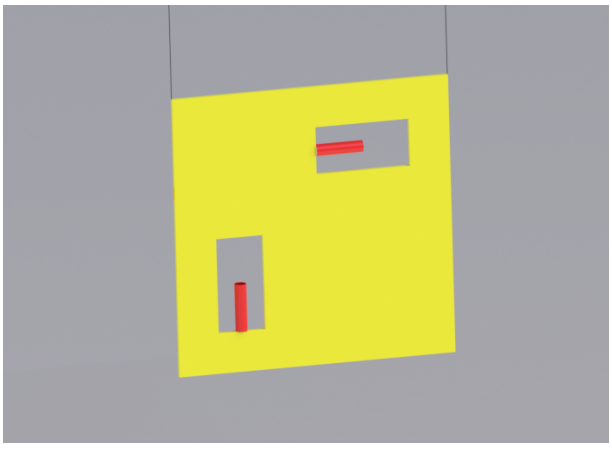

Figure 7: The feed emperor grapes mission.

3.3.8 Laurel wreath

This is the last parts of the round and can be considered the most complicated as well. It consists of two assemblies of the same look and size [Figure 8]. They both have an acoustic pinger, but only one of the pingers will be active at the same time. Above this pinger is the laurel wreath which is a PVC pipe of the size 7.6x91cm large, trimmed to be slightly negative buoyant. The pinger and pipe is both located near the bottom and above them floating on the surface is an octagon representing the emperors palace. The mission can be completed in several ways. The first is for the robot to simply surface within any octagon. To get more points the robot can localize the active pinger, grab the pipe and then surface in the octagon above. When also the pipe is released once surfaced, more points is awarded. To get maximum points, the robot should localize the correct pinger, grab the pipe, the team asks for a switch of the active pinger, the robot moves to that pinger, releases the pipe in that stand and finally surfaces within the octagon above.

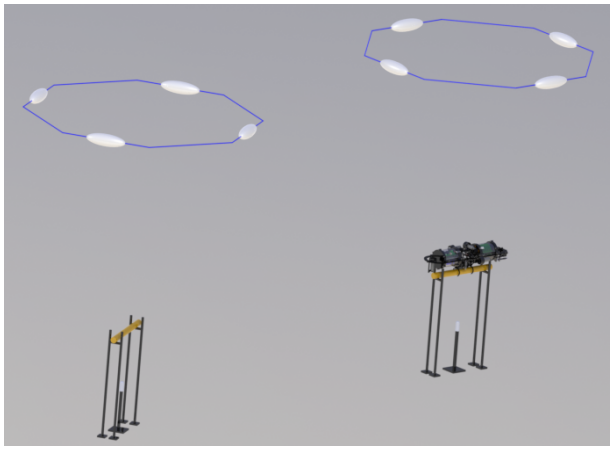

Figure 8: The acoustic pinger with pipe and octagon.

4 Mechanics

4.1 Introduction

The mechanics of Vasa is designed to be very modular. The design simplifies service and maintenance and the robot can be configured for different applications. The materials used in the robot are mostly aluminium alloy (EN AW-5754), polyoxymethylene (POM-H) and polymethylmethacrylate (PMMA). The alloy is used because it is relatively lightweight, water resistant and is easy to process (like CNC-milling etc.). Since the surface of the polished aluminum get scratches really easy it is anodizde the to get a robust and good looking surface.

4.2 Method

The mechanics group contained of four students with one team leader responsible to follow the team progress and to communicate with the project leader. Almost every day the group members worked 8 hours. The mechanics group divided the team members tasks to reach the milestone goals throughout the project. The group had many discussions throughout the project regarding the design, and the design phase was relatively long to ensure that everything would work togehter.

The first milestone was to design the robot from scratch by using SolidWorks CAD software. Each member was responsible for different parts, but the whole team made the important design descisions throug meetings and discussions. Once the design had been done, many of the aluminium parts were cut by water jet. The next step was to further mill the parts in the university CNC machine. The team used EdgeCam software in order to generate CNC-code for the machine. Some parts needed to be modified by hand after the CNC milling to match the parts designed in SolidWorks.

The team members were responsible to contact different companies to acquire the aluminum, anodize the parts, get connectors etc. The last and final stage of the mechanics was the testing. For the mechanical team, one crucial test is to see wether the compartments are watertight. This test was done at a swimming hall located near the university. Other tests have also been done throughout the project, such as CAD-simulations, electronic rack functionality, etc.

The mechanics group has been communicating with the electronics group throughout the project to agree about the PCBs size and design to ensure proper interacion between the meachanics and the electronics.

4.3 Old designs

Before the design was finished, the mechanics team did several different designs. The first idea was to have a single hull construction to minimize the number of waterproof compartments. This design also allows easy access for the electronics team to add and remove cables between different components.

4.3.1 First design

The first design [Figure 9] was the single hull construction. The bad things with this design are that it is hard to add new equipment in the center of the robot. Since both cameras are located in the front of the robot, it is hard to keep all equipment close to the cameras, which means that the aiming will be harder. Because it is possible to solve those problems with a new design, the mechanics team made a new design.

4.3.2 Second design

[Figure 10] shows the second design. This design was the first design that was the real foundation to the actual robot. In this design, a two hull construction was designed. That means that equipment easily can be added and placed at a good position. The two hulls also make it possible to separate sensitive electronics from interference, often produced from high current components.

Figure 9: First design.

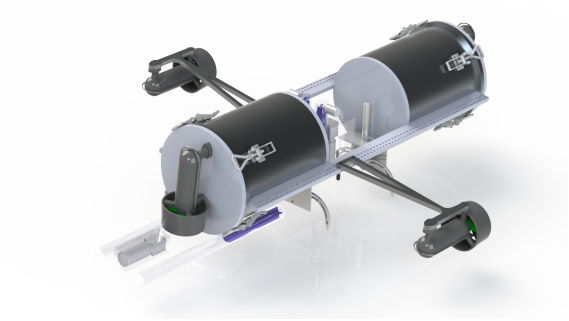

Figure 10: Second design.

4.4 Frame

4.4.1 Overview

The main idea behind the frame is that it should be modular and easy to make to add and replace parts. That is why there are many holes in the frame, which makes it easy to attach new mounts and parts to the robot. All mounts are attached by screws and nuts, which is a robust and easy way of adding equipment.

4.4.2 The frame

The first idea was to keep all electronics in one single hull to minimize the number of cable connections in and out from the main hull. That also gives easy access to the electronics, since external connectors between different hulls are not needed. That idea was scrapped quite fast since it was desired to have the IMU, markers, grippers, camera and thruster in the middle of the robot. One other thing is the heat produced in the hull and also the electrical interference. That is why a two hull construction was chosen.

The main parts of the two hull construction is the frame and the tubes. The frame [Figure 11] is the part where all mounts are attached and where the tubes are sliding when the they are inserted or removed from its position in the frame.

To determine the size of the frame, all the parts and equipment of the robot must be roughly designed. During the project, the frame was extended a few times to fit all the parts in the frame.

All the holes in the frame were designed to use M5 screws because they are quite small and strong enough. Most of the mounts used in the robot are about 10mm wide, wich fits well with the dimensions of the M5 screws with washers.

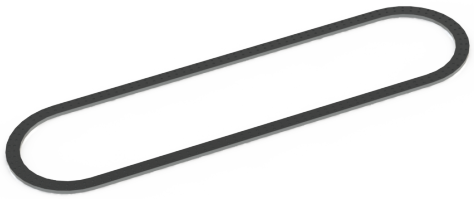

Figure 11: The frame.

4.4.3 Mounts

All the equipments and parts used in the robot are attached to the frame using mounts. Because the frame is modular, it is easier to design optimal mounts for the specific application. All mounts are fastened with screws, washers and nuts, except for the tubes. The tubes are attached with compression spring latches, which makes it easy to remove the tubes (including the electronics) very fast without any tools.

4.4.4 Camera mounts

The forward facing camera has no mounts, it is screwed directly to the frame. The holes in the frame determines where all mounting holes are placed in the housing. Also the camera housing is where the screws for the front thruster mount will screw into.

For the downward facing camera, the housing is equipped with mounting ears. Here there are separate mounts [Figure 12] used to attach the camera housing to the frame. The mounts are designed to keep the camera near the middle of the thruster and gripper, making it possible to mount the camera in between. By having the mounting ears integrated in the housing, it is possible to attach more equipment close to the camera. The design is kept as simple as possible and is influenced by the gripper mount, maximizing space close to the lids (to have as much space as possible for underwater connectors to the lids).

4.4.5 Thruster mounts front and back

These thruster mounts [Figure 13] are designed to keep the thruster´s propeller at the right height (in the middle of the tubes) to follow the "optimal" thruster configuration and keeping the robot to be balanced. The design is based on having mounts to the thruster and at the same time mounts it to the frame. By making a "L" shape of the mounts, it fulfilled our goals.

4.4.6 Thruster mounts middle

With respect to make it as easy as possible to manufacture these mounts, the mounts were separated into two different parts that are screwed together. The piece attached to the thruster is a kind of "adapter plate". This is designed to fit the contours of the thruster. The position of this mount makes the thruster´s propeller sit in the right position, in the center of the robot.

The other part of the mount [Figure 14], attaching to the frame, is designed to be strong and esthetically pleasing. The width of the mount is determined by two M5 screws with washers. One extra rod was added to reinforce the mount.

Figure 12: Camera mount for the downward facing camera.

Figure 13: Thruster mount for front- and back thruster.

4.4.7 Thruster mounts horizontal

Since the idea is to keep the thruster in the middle of the robot, the mounts [Figure 15] were designed from 20mm aluminum and then slided onto the frame, so the mount is attached from top and bottom to the frame, with the frame in the middle. That makes the design very stable and most important, in the center of the robot. Because the thruster is screwed to the mount, that also affected the design.

4.4.8 Tube mounts

The tube mounts are separated into two different parts, one on the flange and one attached to the frame [Figure 16]. Also the frame is used to guide the tube when it is sliding in or out from its position in the robot.

The flanges are designed to hold all the weight from the tube and transfer weight to the frame. One pocket is created in the flange to attach the tube to the frame.

The mounts mounted to the frame holds the tubes in place and compression spring latches is used to prevent the tube from sliding out. The mounts used in the middle of the robot have countersunk holes to make space for the compression spring latches.

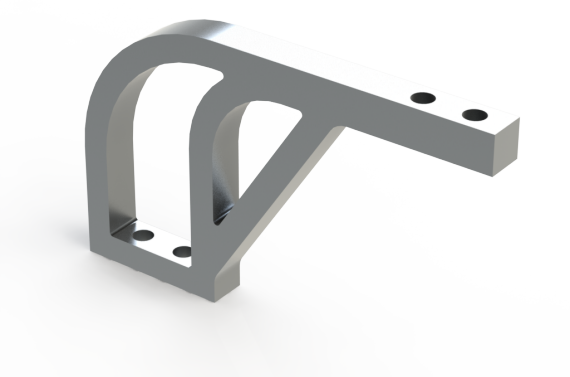

Figure 14: Thruster mount center.

Figure 15: Thruster mount horizontal.

4.4.9 IMU mounts

For the IMU, the main issue was to keep it in the center of the robot. The mounts [Figure 17] then keeps the IMU housing in the middle of the robot.

The mounts it selves are designed in a really simple way, one rod from one side of the robot to the other side. In the middle there are holes were the IMU housing can be attached, which is dependent on the housing it selves to keep the accelerometer in exact center of the robot.

4.4.10 Gripper mounts

To be able to attach the gripper to the frame, gripper mounts [Figure 18] were needed. The main idea is to keep as much space as possible in front of the lids so the gripper mounts were designed to be as small as possible, but still strong enough to pick up heavy objects with the gripper. For the design, it looks much as the design used for the camera mounts.

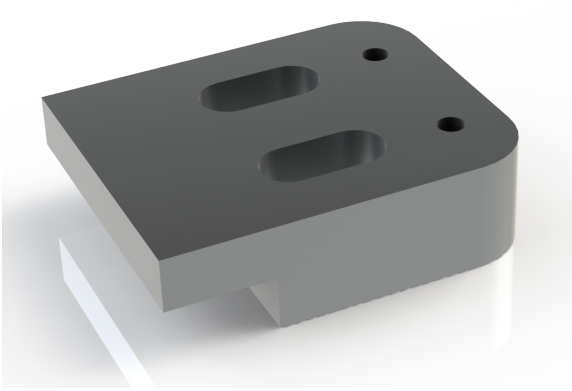

Figure 16: Short tube mount.

Figure 17: Accelerometer mount.

4.5 Thruster configuration

To be able to achieve all possible motions of the robot, a six thruster configuration was chosen. They are positioned in a way that enables the robot to move in six degrees of freedom:

The three linear motions are

- **Heave** - Vertical (up/down) motion
- **Sway** - Lateral (side-to-side) motion
- **Surge** - Longitudinal (font/back) motion

and the three rotational motions are

- **Yaw** - Rotation around vertical (up/down) axis
- **Pitch** - Rotation around lateral (side-to-side) axis
- **Roll** - Rotation around longitudinal (font/back) axis

When configured in this way, the robot has the ability to move in the best possible way to its target, no matter of its current position.

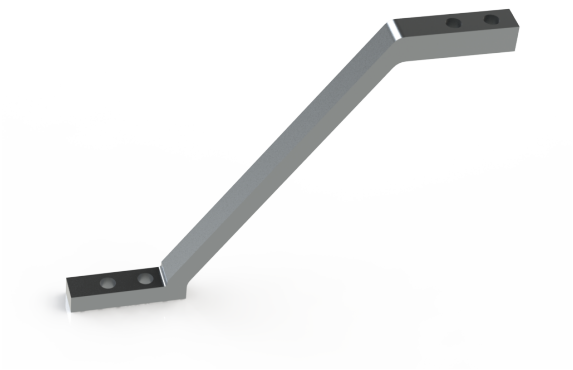

Figure 18: Gripper mount.

When having six thrusters, they can be positioned in a number of ways to achieve the six degrees of freedom. The way chosen [Figure 19] is a variation of the cubic formation [13] that enables the robot to maintain its modular design together with the two main hull configuration and optimal sensor, gripper and marker positions.

All the thruster pairs have one singe intersection point and this point is moved slightly forward from the center of the frame. This is to be able to fit the downward facing camera and grippers in their near optimal positions. Although still, as described in 4.6.4, in the intersection point of the thrusters is the IMU-unit located to ensure optimal conditions of the accelerometer and gyroscope data.

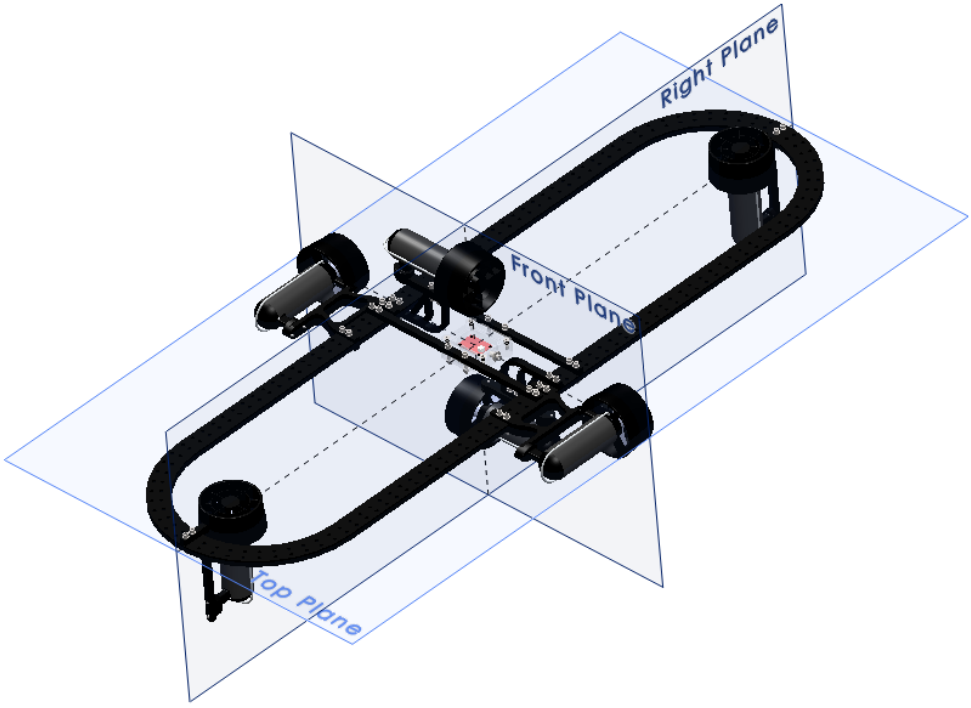

Figure 19: Thruster configuration.

4.6 Watertight compartments

4.6.1 Overview

Since this is an underwater robot, it has to have some watertight compartments containing all the electronics used in the robot. Vasa currently consists of three different types of watertight compartments; main hulls, camera housings and IMU-box.

At first there was a lot of discussions about having a single hull containing all the electronics. Having a single hull configuration would simplify the construction and minimize the number of places where leakage could occur. However, it would make optimal positioning of thrusters, cameras, grippers etc. a lot more complicated. It would also be harder to isolate sensitive electronics from electric and magnetic interference. These two factors is the very reason to why Vasa has several hulls instead of one single hull.

4.6.2 Tubes

The two main hulls contains most of the electronics for the robot. One contains the backbone with its attached cards and the mini-ITX board and the other tube contains the power board and the motor controller. This way all the sensitive electronics is placed in the first tube and all the high voltage and high current electronics in the other tube, separating them from electromagnetical interference.

The two main tubes [Figure 20] concists each of five major parts; two flanges, two lids and one plastic tube. Both the flanges are attached to the plastic tube on each end using an ms-polymer based adhesive (Tekton MS-Polymer) that ensures a waterproof connection while still being flexible. Since aluminum and plastic reacts different on pressure and temperature changes a 1 mm space in every direction has been left between the flanges and the tube to allow the connection to flex. The flanges have lids connected to them using either screws and bolts on one side or compression spring latches on the other side. To waterproof the lid-flange connections, o-rings are used.

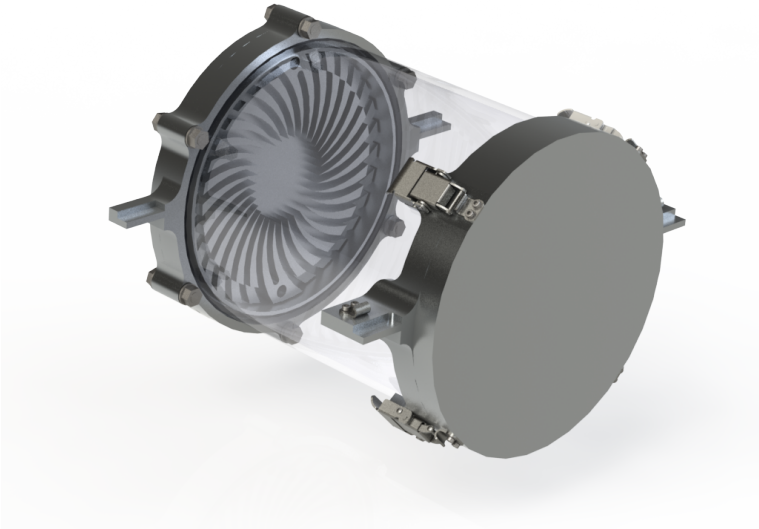

Figure 20: One of the main tubes, including compression spring latches and screws.

Lids

There are two types of lids used. The first ([Figure 113], [Appendix G]) has integrated cooling fins which helps cooling the inside electronics and it is designed so that the cooling fan sucks the cold air from the lid out into the tube. There are also two guiding holes to which the electronic support system (4.7) slides into. Six M8

clearance holes are also present for attaching the it to the flange. Also in the lid is an o-ring track to keep the o-ring.

The other lid ([Figure 114], [Appendix G]) is where all the external connectors (4.8) of the hulls are placed and the ESS (4.7) is screwed onto. This lid has on the inside mounting holes for the ESS and also an o-ring track. On the outside there are holes for the compression spring latch clips. Also in one of these lids is the pressure sensor mounted.

Flanges

Just like with the lids, there are two types of flanges used. The first one ([Figure 111], [Appendix G]) is connected to the lid with M8 clearance holes in it, and has the corresponding holes on it. A track is made on the inside to fit the plastic tube, with enough clearance to allow flexibility. On the outside of the flange there is a mounts for the frame so that when the whole tube is mounted in the frame, these mounts rests on the frame and at the same time locks the tube into place.

The other flange ([Figure 112], [Appendix G]) is the thicker of the two flanges. This is to make enough room for attaching the compression spring latches at the correct distance so that enough pressure is applied to compress the o-ring when attached to the lid. Just like the other flange there is a track for the plastic tube and mounts for the frame. These mounts however have clips attached to them that connects the compression spring latch that is mounted on the frame. When this compression spring latch is locked the whole tube is firmly locked into position in the frame and cannot move.

4.6.3 Camera housing

The camera housing is designed for the Point Grey Dragonfly2 camera including the optics. The housing [Figure 21] can be splitted into two parts, to make it possible to reach and adjust the camera.

To waterproof the camera housing, two o-rings are used. The walls of the camera housing has to be thick enough to support the o-rings. The camera housing is designed to have enough space to fit the camera, that standard dimensions of o-rings can be used, the connectors fits and that the housing should be able to be mounted easily to the frame. It is also enough space to adjust the camera with spacers, that the angle of the camera can easily be adjusted.

Because the optics used has a specific shape, the camera housing was designed after that. Since the optics has an oval shape, and we want the camera housing as small as possible, the camera housing was also designed as an oval shape.

Screws are used to attach the three different parts to make a complete camera housing (lense, front body and rear body). Because screws are used, it is fast and easy to reach the camera and also replacing the lens if it is damaged.

The cameras positions makes the them see forward and down. The forward facing camera is used for navigation, aim the torpedos and see the buoys. The camera facing down is used to aim the markers and see the orange path markers to be able to navigate between the missions.

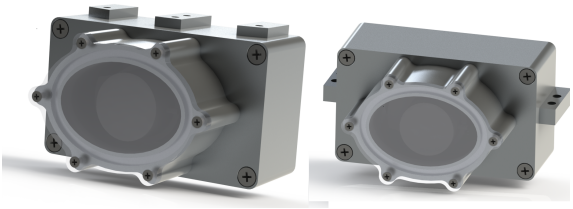

Figure 21: Camera housing.

4.6.4 IMU box

The IMU box [Figure 22] contains a Sparkfun Razor 9 DOF IMU and is specifically designed for it. It consist of two separate parts, a lid and the housing, manufactured with clear PMMA plastic. To waterproof the box, an o-ring is used between the two parts.

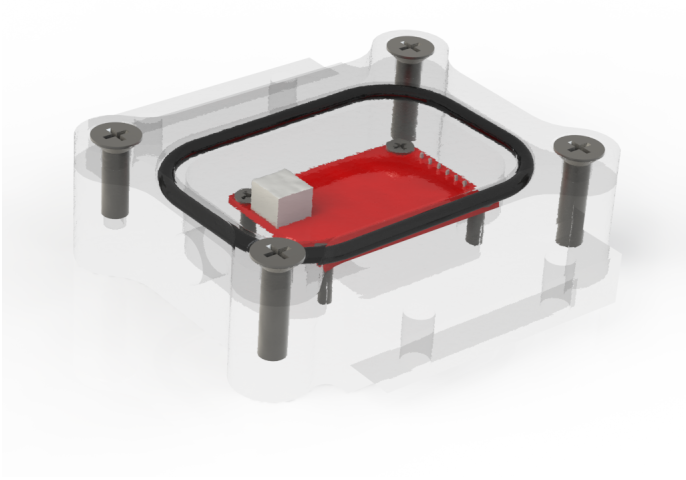

Figure 22: The IMU box including the IMU-unit.

Inside the housing is mounting holes specifically designed to fit the IMU, and they are positioned in such a way that it makes the center of the IMU-unit to be in the exact center of the thrusters intersection point. This is to ensure an optimal position that enables reading and using the accelerometer and gyroscope data without the need of any transformation of the data.

Since all the rotational movements intersects in the same point and having the IMU positioned in that spot, all the calculations that has to be done by the programmers to move the robot in the desired motions is considerably easier to perform, compared to an off-centered IMU and non-intersecting thruster configuration.

4.7 Electronics support system

4.7.1 Overview

The Electronics Support System (ESS) [Figure 23] is CNC milled from polyoxymethylene (POM-H), also known as Delrin [17], and is responsible for keeping all the electronics and inner components in place in the hulls. It is designed in a way that allows for basically an unlimited number of configurations of how things can be mounted. The ESS is constantly mounted to the inner lid where all the cables for the electronics go through. This eliminates the need to disconnect any cables on the inside when the ESS is removed from the hulls, as seen in [Figure 24].

4.7.2 ESS structure

The ESS concists of two circular plates at each end which are connected to each other with rods. These rods are mounted to grooves in the circular plates and acts both as the supporting structure and as mounting points. The grooves in the circular plates span over 83% of the faces of the plates which gives a lot of flexibility as the rods can be moved along the grooves to the optimal position. On top of the mounting rods smaller aluminium mounting plates are mounted, enabling the mounted components to be moved in a direction perpendicular to the mounting rods. More rods can easily be added if there is a need for more mounting points or additional

Figure 23: Electronics support system.

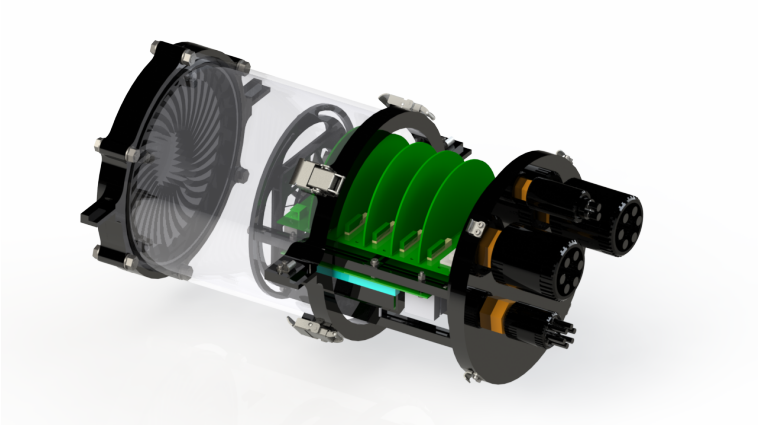

Figure 24: Hull and ESS with electronics.

components. On the outer end of the ESS two tabs mates with the outer lids, keeping the ESS in place during operation.

4.7.3 Cooling fans

On the outer ends of each ESS a Schythe SY1212SL12H [18] cooling fan supplies the electronics with air cooled by the cooling fins on the inside of the outer lids. The fans are permanently mounted to the ESS and comes out with it as it is removed from the hull.

4.8 External connectors

4.8.1 Overview

To be able to interact with the environment many signals needs to go in to and out from the hulls. All external connectors are made by Seacon [15] and the cables are spliced together with solder, heat shrink tube, silicon and vulcanizing tape. There are many free slots in the split connectors for future use.

4.8.2 Front hull connectors

On the front hull there are four separate connectors. All are split connectors which means that many male connectors can be connected to the same female chassis connector, see [Figure 25]

Figure 25: Seacon split connector.

- Connector 1: 24 signals, 4 signals/cable. This connector connects both hulls to each other. One cable is used for power, one for ground and one for the CAN bus.
- Connector 2: 24 signals, 4 signals/cable. This connector connects the front hull to the IMU and future external components such as sonar.
- Connector 3: 12 signals, 6 signals/cable. This connector connects both cameras to the mini-ITX in the front hull.
- Connector 4: 12 signals, 2 signals/cable. This connects the solenoids to the future solenoid card in the front hull.

4.8.3 Rear hull connectors

The rear hull uses two separate split connectors.

- Connector 1: 24 signals, 4 signals/cable. This connector connects both hulls to each other. One cable is used for power, one for ground and one for the CAN bus.
- Connector 2: 12 signals, 2 signals/cable. This connector connects the rear hull to the thrusters.

4.8.4 External component connectors

The camera housings use the same detachable solution as the hulls, but with dry-mateable connectors. This means that the connectors can not be detached under water as the other connectors. These connectors are smaller and used in the camera housings due to space constraints.

4.9 Kill switch and mission switch

The purpose of the kill switch is to shut down the thrusters and stop the robot. The mission switch will provide the starting signal to start the mission. The switches are located on the frame and work by using a magnet and a corresponding hall effect sensor. When the switch is moved the change in the magnetic field in sensed by the hall effect sensor connected to the power board.

4.10 Equipment

4.10.1 Grippers

The idea of the gripper design is to pick up the PVC pipe of the competition but it can also be used to work as a general gripper to pick up other objects. The re are two identical gripper mechanicsms, see [Figure 26]. They are placed in the center of the robot to match the shape of the pipe.

The basics of the design were to manufacture two claws with an appropriate diameter connected together to one rod. Each gripper mechanism is powered by a solenoid.

The the parts in the gripper mechanicsm are milled down to 5mm thickness from the original 10mm to keep the weight and dimensions down. The natural position of the gripper is open and the system only consumes power when is is closed.

Figure 26: Down Gripper.

4.10.2 Torpedoes

The basic idea of the torpedo launcher is to fire at targets in the competition. The robot has two torpedoes placed close to the front camera to simplify the aiming [Figure 27]. The torpedoes are designed to be manu-

factured from plastics to keep the weight down. Four fins keeps the torpedo stable [Figure 28]. The torpedoes are mounted inside plastic tubes with a simple launching mechanism triggered by solenoids.

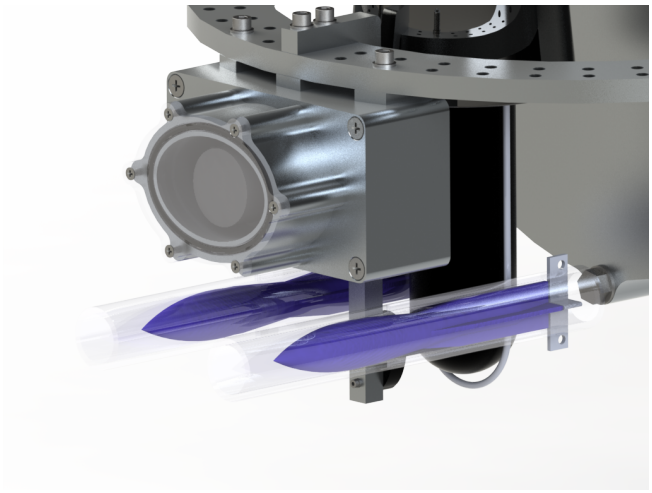

Figure 27: Placement of torpedoes.

Figure 28: Torpedo and tube.

The four fins are attached to the base that is connected directly to the springs that provides the launching force. The part that pushes the torpedo forward during the launch are connected to the solenoid to release the base with the help of springs. Simulations to the torpedo was done in SolidWorks to optimize the design to minimize drag, see [Figure 29].

4.10.3 Markers

In the competition, the markers are supposed to be dropped in specific bins . The idea of this design is to mount the two markers close to the downward facing camera in the center of the robot [Figure 30] at the same line,

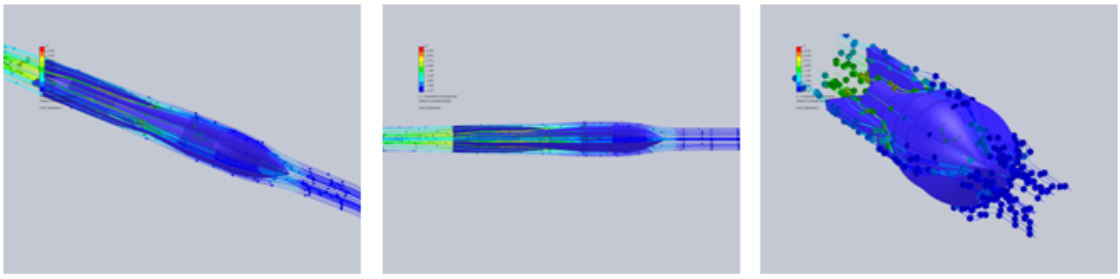

Figure 29: Torpedo flow tests.

for accuracy purposes. The markers are designed to be manufactured from plastic with four fins to keep them stable. Inside the marker in front side there is some weight placed to move downward towards the destination and don't float.

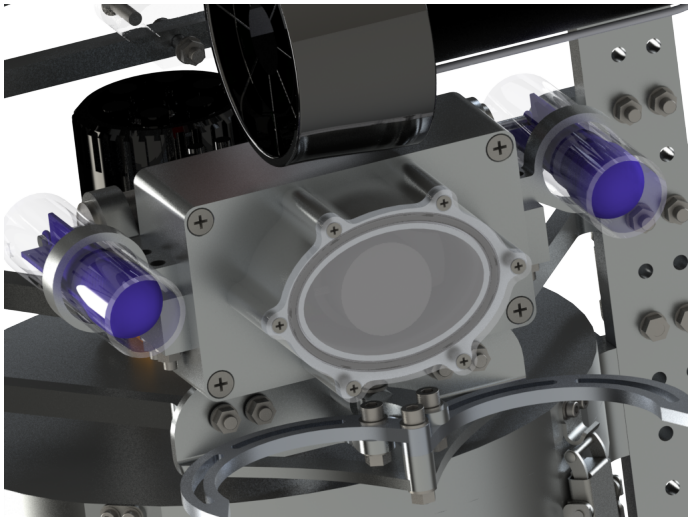

Figure 30: Markers mounting.

The marker placed inside the plastic tubes are attached by a circular hook to the solenoid shaft [Figure 31]. Both markers and the solenoids attached to aluminum mounting connected directly to the camera house.

Figure 31: The marker.

4.11 Testing the robot

4.11.1 First assembly of Vasa

The first test of the robot was the first assembly. All parts of the robot are designed and assembled in SolidWorks and should therefore fit together. The assembly of the manufactured parts is thus an important test.

When all parts were mounted, all parts fitted as expected. Since the screws used to attach mounts to the frame was M5 and the holes are 5,5 mm, there is free play to adjust angles on mounts and thrusters.

4.11.2 First pool test

When the robot was assembled after anodizing, it was time for the first pool test to examine if the design was waterproof, see [Figure 32]. The robot had no electronics inside since this test is just to see if the construction is waterproof. The robot was tested at Kristiansborgsbadet, depth 3,8 meter. Vasa was put at the bottom of the pool for 35 minutes. During the test, the robot was moved around, all to simulate the motions of the robot in the future.

The test was really successful, no leakage at all. All compartments that should be waterproof, such as the two tubes, two camera boxes and IMU-box, was waterproof.

As the o-rings are placed as they are in the construction, the force compressing the o-rings is increased with increasing depth. That means that the pressure of the water will make Vasa´s o-rings seal more and more the deeper it goes.

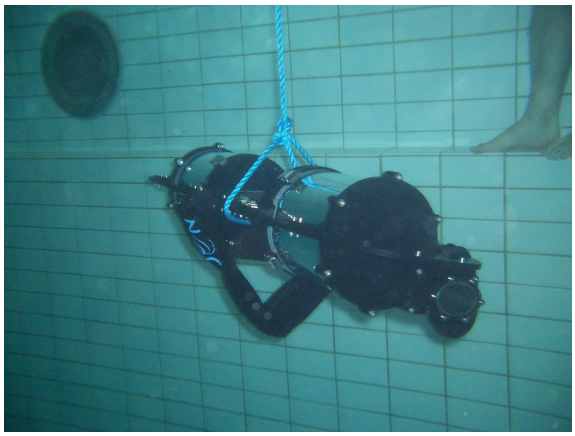

Figure 32: First underwater test.

5 Electronics

5.1 Introduction

The electronics part in Vasa has to support the mechanical functionality as well as to create a platform for the software controlling the robot. The system was decided to be made modular with as few cables as possible, as cables were considered one of the weak point of the construction. A modular design will give Vasa a system that is easy to modify and repair. One part or subpart can be changed without affecting the other parts. Several designs were proposed and the final version includes ideas from all of them. The electronics in Vasa consist of five parts who are all connected according to [Figure 33].The Backbone is a board that works as the physical layer of CAN and power. Connected to this board are the smaller dedicated nodes. The backbone and dedicated nodes are gathered under a collected name: Sharkboard [Figure 34]. To enable the Mini-ITX communication with all nodes in the system a CAN-controller or router is installed as one of these smaller dedicated nodes. The router handles USB to CAN communication. Other nodes connected to this backbone are GPIO card, SONAR card and a solenoid driver card for torpedoes and markers. The motor-controller makes decision for motor speed and direction to change the direction of the robot and its speed in the water. These decisions are based on orders from the Mini-ITX and sensor data from nodes in the Sharkboard.In order for the electronics system to be powered there was a need for a board handling, monitoring and distributing power. The power board contains fuses and relays to cut power if voltage is outside of normal operating levels. In the figure of connections [Figure 33] there is a number next to a wire, these number are marked on the cables in the prototype. The indication of wires are defined in the form of TYPEOFCABLE INDICATOR:FROM - TO for instance could a power cable to a camera be called PWR 010 : ITX - CAM.

5.2 Method

The electronics group consisted of 4 members, one of these members were a team leader. The team leader worked to synchronize work in this group to other groups and look over all work being performed in the electronics group. Once a week a meeting took place were everyone described the upcoming week and what they did during the last week. During this meeting necessary problems were discussed and planing was edited. The group worked according to a GANT schedule were every work task were given as a responsibility to one person in the group. During testing everyone had to write a test protocol before the practical testing, this gave good results to enforce that all team member had some time to reflect on the testing and its requirements.

The first milestone was to find previous reserch and use parts that were fited for our project. Ideas and deigns were integrated to fit our needs. This phase would then be integreated with milestone two, which was

aimed for designing all new cards to get a operational system. Milestone three was designed to add functionality needed for the competion.

Due to communication difficulty all communication by companies and sponsors, technichal advisment were handled by the team leader.

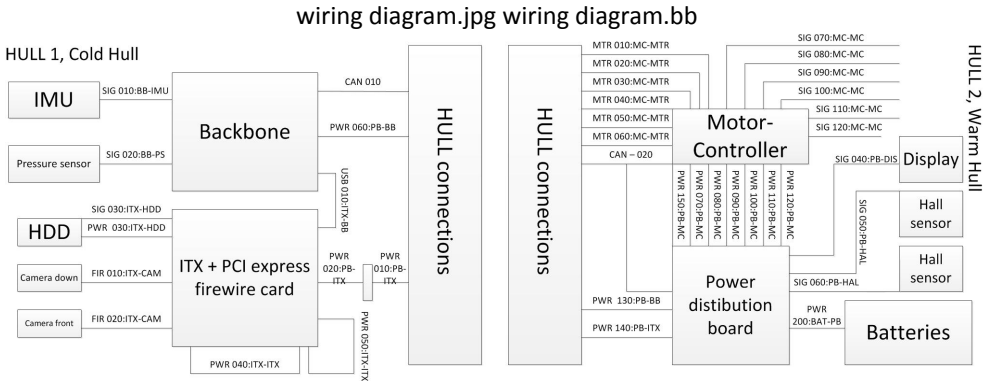

Figure 33: The overall design of the system incuding internal wiring.

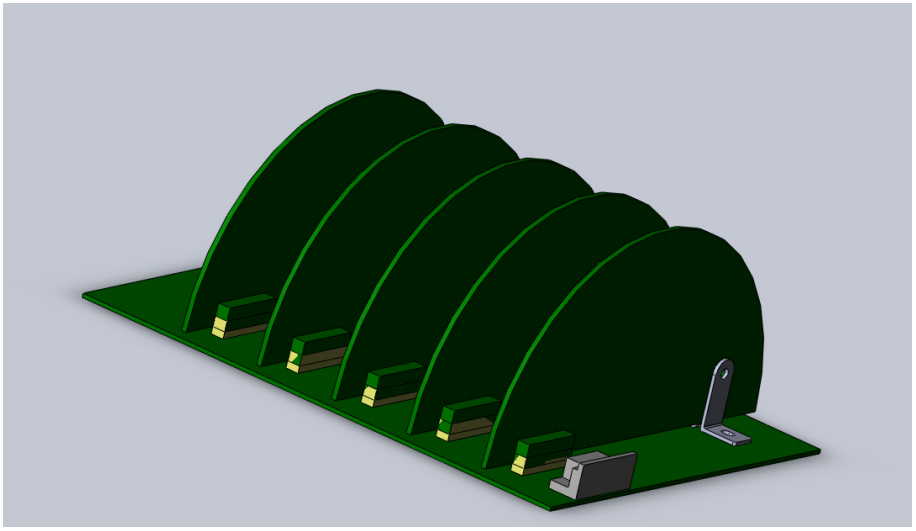

Figure 34: A CAD representation of the Shark-board.

5.3 Background

At MDH (Mälardalen University) the AUV prototype (RALFII) [9] gave Vasa some help. During the evaluation of the RALFII, the CAN-bus, USB to CAN Router card and the GPIO card were selected as parts to be reused. The motor-controller were also selected but extensive work would be needed to redesign this board to fit the new design. All cards were to be modified to fit the new design to some extent. Documentation from this project was limited and considerable experimentation was necessary to find the functionality of the RALFII robot and the requirements of hardware like motors. The same electrical construction has been used on all cards. If a card in this report is mentioned to have CAN it has a CAN interface based on the RALFII design. The main reason for selecting CAN as the primary communication interface was of the already existing documented design from the RALFII project. According to this documentation the communication worked without errors and it used 3% [9] of the total capacity, that left room for further development without changing design.

5.4 Battery

To feed the system there is a need of batteries. This system uses a LiPo battery of 16Ah with 5 cells. This means that the system will operate on voltage of 18 Volts - 21 Volts, never exced 21Volts. 8Ah is recomended as a minimum size of battery.

5.5 Sensors and cameras

Vasa is equipped with 2 sensors and 2 cameras. The sensors are off-the-shelf and consists of one inertial measurement units (IMU) and one pressure sensor. The pressure sensor is designed for a great depth and have been modified with a amplifier to make it operational at a shallow depth. The IMU are of type, the SEN-10736, Razor inertial measurement unit from Sparkfun Electronics [6]. This is used to keep the robot balanced and to get its heading. The IMU operates on 3.3 volts and is therefore connected to the 5 volt sensor card via a logic level converter [7] located on the back of the IMU. The connector on the IMU is set up for both 5 volt and 3.3 volt communication by bypassing the logic level converter. This enables easy IMU debugging and calibration with a computer through a 3.3 volt serial-to-usb converter. The cameras will handle the observation of the surrounding world and be the primary sensing. For more information on the cameras and set up of cameras see 6.2 , for pressure sensor see 6.5.1 and for a description of the software concerning the IMU, see 6.5.2.

5.6 Shark board

A board for different dedicated nodes were created to function as communications bus and power supply. This board gives the advantage of a modular design without the use of cables but rather slot (headers). This will sustain the nodes with power of 3.3V, 5V, -5V and battery voltage. The different power and communication are as follows in columns [Figure 134]. The base of this modoule is called backbone and the units inserted are called cards. a module like this is refered to as board. backbone and cards are collectivly refered to as shark-board [Figure 35].

5.6.1 Backbone

For the Sharkboard portion of our design, two aspects of backbone were considered: the power handling and the signal handling. The main change in the power handling is the addition of voltage regulators. For the signal handling, a digital signal is passed and distributed physically through connectors and will be interpreted by a logic controller (Router and Sensor boards).

Power Handling
The battery of the system has an output of 18 - 21 volts and the design for the RALFII [9] routes the voltage directly to each board. As the new design will be modular and compact, the Buck topology were used so the backbone will produce three levels of voltage (5, 3.3 and -5 volts), and passed directly from the battery through

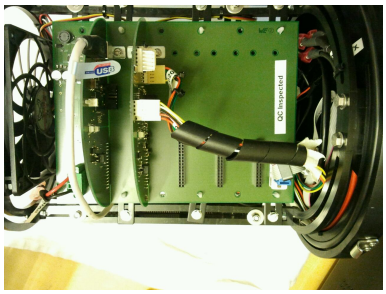

Figure 35: Topview of Shark board, Backbone and Cards.

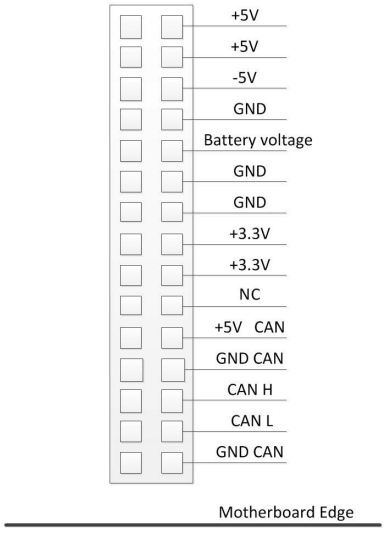

Figure 36: Topview of backbone header.

connectors. To accomplish this, voltage regulators were included in the backbone. Since a voltage regulator is an electrical regulator designed to automatically maintain a constant voltage, it can be used to regulate the voltage coming into the boards. The components that were used to accomplish this task are ADP2302_2303 [5] (3.3 and 5 volts) and LM2611(-5 volts), getting an output current 2 and 0.25 ampere respectively.

Signal Handling

The digital signal is transferred through a physical media, within connectors. The Sharkboard will not interpret this digital signal, instead passed it from/into connectors.

Noise due to the switching frequency of the switching voltage regulators may occur. An unfiltered output may cause glitches in digital circuits and this can be suppressed with capacitors and other filtering circuitry in the output stage. With a switched voltage regulator the switching frequency can be chosen to keep the noise out of the circuits working frequency band. Different values were chosen and types of the capacitors, the values are chosen so that less ripples occur. The inductors are chosen for filtering and getting a smooth output with low noise, the inductors values were picked depending on the manufacturer of the voltage regulators to fulfill the requirement and get smoother outputs [8].

5.6.2 Router card

The router card [Figure 37] is developed to establish a communication between CAN and MiniITX. The major communication inside AUV is held by CAN protocol so the Router card is designed to relay information between CAN and Mini-ITX. The Router card has the main functionalities of USB serial communication between MiniITX and CAN also Serial communication for PC monitoring. The AT90CAN128 microcontroller is used in this card, this has a built in capability for CAN support. The serial communication is done by using FTDI232R chip that is a USB to serial UART interface. To support CAN protocol,the router card were equipped with a TJA1040 high speed CAN transceiver. MiniITX is connected to the router card directly through standard USB connector. This interface support data communication and also as a source of power that is used for powering up the microcontroller and FTDI232R chip. The microcontroller is connected to a 16MHz external oscillator. The card is divided into two sides, one USB/UART side and one dedicated for the CAN bus part. The CAN circuitry side of the card is isolated from the rest of the components in the card using two optocouplers.These are connected to the transmitter and receiver side from CAN transceiver to microcontroller. The card have TJA1040 CAN transceiver that supports ISO 11898,which is a serial communication technology used to interfacing the physical layer and CAN protocol. It provides the facility of differential transmitting and receiving capability. The CAN constitute of four transmission peripherals, CAN low, CAN high, voltage and ground. An 1500 Ohms termination resistance is connected between inputs of CAN high and CAN low to the router card. A choke is connected after the resistor which is used to reduce noise from signal input to the card. The use of connector which would connect the router card to the backbone, special consideration taken for having a firm connection. The card was fit with a zenerdiod which is used to keep the system safe from high voltage spikes. For easy programming the board is fitted with a JTAG connector. In general the FTDI and microcontroller get power from Mini-ITX through USB. The Mini-ITX communicate with microcontroller using FTDI232R and the Microcontroller communicate to CAN using TJA 1040 CAN transceiver.

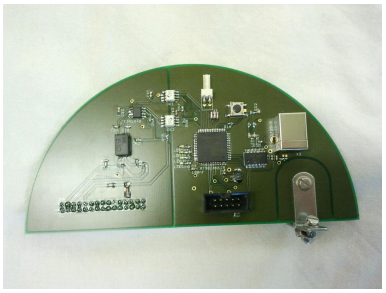

Figure 37: Router card.

5.6.3 GPIO card

GPIO card [Figure 38] is used for interfacing sensors to the rest of the system. As the AUV needs sensor input to get all the required information regarding pressure and balance. Currently the GPIO card is providing interface to measure the pressure and IMU (Inertial Measurement Unit). Comparing the design between GPIO card and the Router card, one can easily see big similarities between the two. The main difference is that the GPIO card has some additional peripherals for different sensors connectivity. The I/O that are available is : Analogue, Interrupt, PWM and UART. Currently the GPIO card is only using one UART and one Analogue input pin.The rest of the I/O pins are reserved for future use. The GPIO card receives data from the two sensors, IMU and Pressure sensor. The IMU sends data to card through UART1 port and pressure sensor sends data from Analogue1 port. The pressure sensor is powered by 5V and-5V and sends data to microcontroller via analog port. The IMU is powered by 5V and delivers information through UART1 port. The GPIO card is connected to main board, which provides the physical layers that connects CAN from main board and power input to the GPIO card. The voltage provided to the sensor card via backbone is 5V, 3V and -5V. Filters are applied on the GPIO at the input of power

tracks and they are protected by a Zenerdiode for addition overvoltage protection. The 3 volt power supplied is added for the use for future functionality.

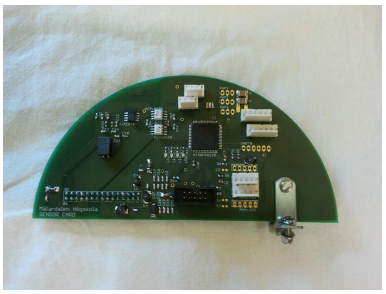

Figure 38: GPIO card.

5.6.4 Solenoid card

The Solenoids are powered by the Texas instrument DRV104 driver. The drivers are set to a fixed PWM at 50KHz with a 70% duty cycle. The Solenoid card [Figure 39] is equipped with the CAN interface and can receive its orders from the ITX by CAN. The card is powered by a AT90CAN128 and it has 6 enable signals to the 6 different driver circuits. Each circuit works as a master running it operation without the interference of the other circuits. If the DRV104 driver is overheated or if there is a over-current there will be a red low current led connected to the status flag that will turn on and the chip resets. The solenoids are powered with battery voltage. Solenoids are a inductive load and protection diodes are added to the card. The DRV104 is limited to 1.2Ampere current.

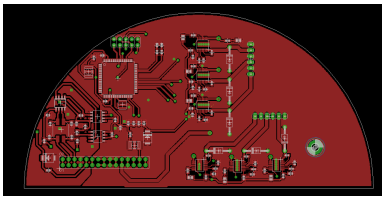

Figure 39: Solenoid card.

5.6.5 Sonar card

The active sonar is the sound ranging sensor device which can measure the distance, time or speed of sound in the medium. This active sonar which the team is going to design on the basis of past research work done in our university and todays researching on implementation and optimization of design module which will help us in future to take a view of environment inside ocean or a specific water pool. The team is seeking the simple design which will full fill our basic requirement of obstacle detection underwater. For a ready made solution, we also researched on Maxbotix active sonar products which is mostly for air and integrated circuit based solutions from the company 'Total Ultrasonic Solution' which is a one of the solution in terms of auto frequency tracking concept, but both of the ready made design solutions have limitations in terms of driving voltage of pinger and have difficulty to change some parameter in terms of both hardware and software, so the team switched to make our own compatible design of software and hardware which is discuss in this report.

The ideal features sonar design for Vasa is consist of dual functionality of passive and active sonar with only one transducer built on smart module of size 1'' x 1'' x 1'' of sealed box unit consist of CAN based microcontroller which also have to select the mode passive or active and then select range or other functionality, which depend

upon the complexity and simplicity of the programming code. The module also consists of temperature sensor which updates the parameter of constant speed of sound in the water.

Above features based sonar is the challenge to design so, it is divided into two parts. The first part is the simple basic design of sonar and the second part is the optimization with addition of more features in first part. The final block diagram of features based optimized sonar is shown in below[Figure 40].

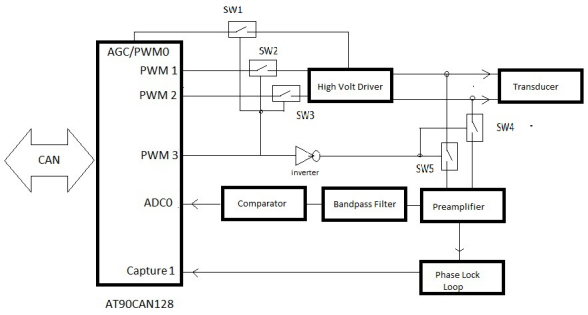

Figure 40: Sonar Block Diagram

The block diagram of above feature sonar is complex and depend upon the timely operation to control the SW1 ,SW2 , SW3, SW4 and SW5 switch from software to direct the pinger and receiver and calculate the distance from the obstacle.

For the sake of simplicity, the design is keep simple in the beginning until the team get the results then go for optimization of design and use of analog switching to control the single transducer to transmit as well as receiver to receive. The simple basic block diagram design compare to sonar complex block diagram is shown in [Figure 41].

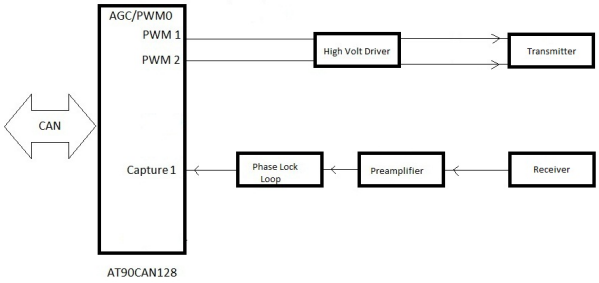

Figure 41: Basic Block Diagram

The above figure shown consists of three basic parts.

- The pinger (transmitter).
- The receiver.
- Main controller(AT90CAN128).

The pinger is driven from 5volt to 100volt with frequency of around 41Khz .The High voltage mosfet driver is used to drive this pinger .The second part is the receiver in which a operational amplefier is used as a preamplifier and then that amplifier signal pass through the PLL (phase Lock Loop), and that signal went to microcontroller for further calculations by software and send message to CAN bus with appropriate data value. Following are the preliminary Test necessary for full fill the basic active sonar requirements.

1. Power of the pinger transmitter.
2. Sensitivity of the receiver.
3. Frequency to pass (solutions).
4. Collecting the echo signal coming from receiver side to capture by microcontroller.
5. The software to calculate and pre scales the coming data for measuring the distance.

5.7 Power distribution board

This is a board required to stabilize the power and protect the electronics from over-current and transient voltage. It should control auto cut of for dangerous low voltage, If Lithium Polymer Batteries (LiPo) drops to low heat will increase rapidly [14]. It controls the emergency stop for motors and load leveling parallel batteries. This board should also have CAN connection and a Liquid Cristal Display (LCD) screen. To make this board the power consumption data was needed, as the data-sheet of the motors stated a maximum continuous current of 4.2 ampere and bursts at 5.8 ampere and the Mini - ITX Power Supply Unit (PSU) stated a maximum current from the PSU to be at 15 ampere [4] the current was expected to be high. This data however did not specify conditions for these current values. Worst case power consumption could be estimated but not with reliable numbers. Two test were constructed one for the motors and one for the MINI -ITX set up in expected conditions for our robot. These test gave a worst case current of 40 ampere at 20volts (800W) but the test also showed us that the motors would overload at 20V. For more information the tests are documented under U-P-001 and U-P-002 in Appendices D. The power distribution board can be seen in figure [42].

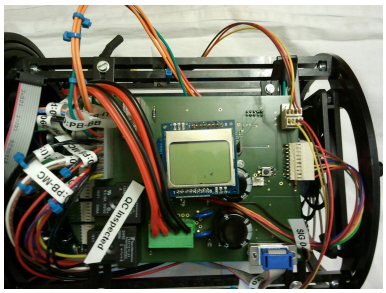

Figure 42: Power distribution board.

5.7.1 General functionality

The design of the power distribution board has one or two connected batteries. The current will pass two large fuses directly after entering the board, then it will pass electrolytic capacitors of different sizes.Lastly it will reach the relay and fuses. All output pins are controllable by relays with exception of the Mini-ITX power. The Mini-ITX is controlled by the Backbone power. If the backbone power turns off, the Mini-ITX will also turn itself off. The reason why the ITX has a control-signal from the Backbone and it is not stopped directly by relay is to ensure a software shutoff, the PSU will signal ITX and after 1minute shutdown the ITX by force [16]. This time frame for shutdown is user definable by the PSU. The micro controller of the board will handle CAN communication, measuring of voltage aswell as handeling the LCD and emergency switches for stopping

motors. Stopping of the motors can be done by the relays as well as stopping the rest of the system individually or together. The width of the tracks were selected according to David, L's paper PCB Design tutorial [4].Thou the tracks were are not dimensioned for continuous run at worst case current rating. The board is designed for GLCD5110 LCD screen for its low price, it can show voltage in batteries both as Voltage and as percentage and priority messages like critical low power. If needed, other screens using a SPI based communication can be used.

5.7.2 Power requirements and load handling

The load leveling of the batteries were considered according to the article by Schmode, P. and Welsh, J. Design of Lithium, Primary Battery Comprising Several Parallel String for High, Pulse Current Loads [14]. By this approach a diode is put in series with the batteries in each string to prevent reversed current, then the negative poles are set to a common signal. The diodes are not included in the power distribution board and must be added separately to each battery pack if this set up is required. The solution should draw power from the battery with highest potential, caution is required to this approach due to the fact that the solution have been not tested. The solution should be tested before implementation. Another system that was selected because of its simplicity is that strings are directly connected at both poles and then they can never be separated. The batteries should be of same type, brand, size, cell count and so on. This is important when they are starting to become worn out. The electrical system in Vasa is constructed to use a 5 cell LiPo battery or batteries with at least 60Ampere continuous current per string and a voltage of 17 - 21 volts. If the current goes below 15V relays will cut power to the system because the voltage is not enough to power the relay coils on the power board.There has been implemented a software limit to 18.5V as a lower limit for cutting the power. At this point the motors, motor-controller and backbone (Including CAN communications) stops. Everything except the power board logic and LCD will be turned of at this point. Another method evaluated was to use some kind of potentiometer in series with the batteries to manipulate the relation of voltage and internal resistance to draw equal power from each string in-dependable on the voltage of each [2]. The later method was unpractical for implementation. According to articles of Lithium Polymer Battery in Warm Environment [11] and Design of Lithium, Primary Battery Comprising Several Parallel Strings for High, Pulse Current Loads [14] the temperature of batteries can be as high as 80 degree Celsius without problems.

5.7.3 Kill switch, emergency stop

For the system to function safely a kill switch is needed. The system uses a magnet on the outside of the hull from which a hall sensor reads the value of the magnetic field.The output is a analog voltage that varies depending of the intensity of the magnetic field. The sensor to be used is the 22L Allegro hall effect sensor and it will give a value from 0V to 5V. When no magnet is close to the hall sensor the value is 2.5v. For the system to be on the value will be close to zero volts. A micro controller reads this value and outputs a high signal to keep the motors running, if logic would lose power or if the kill switch is trigged the signal would go low and the motors should stop. Motors are stopped by the use of relays. All current supplied to the motors have to pass trough two relays. It has to be mentioned that no problems with the relays were detected during testing but for increasing security and creating redundancy a CAN message is sent to the motor controller to stop the motors when the kill switch is activated. This CAN message is sent at the same time as the relay stops the power to the motors. This message works as a backup in case of a catastrophic failure in the relays. There is also a motor stop command issued by the motor-controller it self if the CAN system stops working. If there is no communication for 5 seconds the CAN is assumed to be non working and the motors will stop. The selected relays has a operating temperature of -55 to 85 degree Celsius and a lifespan of over 10 million operating cycles. At current requirments there is no need to improve the design, the power board supports the necessary functionality of the robot.

5.7.4 Misc and layout

The Power board is controlled by a atMega90CAN, this micro-controller is installed according to a guide from Atmel [1]. The microcontroller is power by a switching voltage regulator at 5 volts. From this micro-controller two control signals opens the relays (One signal for motors and one for backbone and motor-controller) these signals are active high for security reasons, If the power of the power board fails (including kill switch) the robot will stop instead of continue uncontrollably. There are three GPIO LED's on the top side of the board, one of them is a high power led intended to be used in event of a priority message on the LCD screen. There is also a connection to three analog GPIO connected to this micro-controller, one analog for temperature sensor and one kill switch for a hall sensor. For flexibility a digital GPIO was added but without any intension. There is also a signal from the battery divided by 10 connected to the analog port. This signal can be used to show the battery voltage this is inaccurate of 0.4 volts after voltage divider this gives a voltage span inaccuracy of 4V in the battery, this must be measured and compensated for in software. To the display interface there is a SPI port with complimentary digital-ports for control signals. The display has voltage divider for running the display on 3.3v But if necessary this voltage divider may be bypassed during soldering if necessary. One final port was added to connect a cooling fan of the warm compartment of the Vasa, this port will deliver 12v from a linear voltage regulator of Low Dropout voltage (LDO) type to the fan. For further documentations of the testing of power distribution board see unit testing U-P-003, U-P-004, U-P-005 in Appendices D.

5.8 Motor controller

The IC based solution of the motor driver used in Vasa has six H-Bridge drivers of type POLOLU VNH5019 for six thrusters. The H-Bridge drivers are connected to the AT90CAN128 microcontroller which controls the direction, speed and break of motors using digital logic signals. The communication between microcontroller and Mini-ITX is held by CAN protocol. Each H-bridge has battery and ground connections as well as control signals from microcontroller. The output to the driver is connected to thruster. The signals run through optocouplers to prevent noice from h-bridges disturbing the microcontroller.
Motor controller board [Figure 42] is divided into two cards. The card with microcontroller contains a voltage regulator, CAN circuitry and JTAG Interface. The microcontroller and CAN configuration in motor controller board is similar as it is in other cards (i.e. Router). The card with microcontroller connects all control signals for the motors and VCC via a 20pin connector. Each thruster would have separate control signals so in the card there are seperate physical connections for each thruster.
The second card of motor controller drivers contains the optocouplers that filters the input control signals to the h-bridges. The output from optocouplers is the PWM and direction signals that are connected to each H-Bridge driver ICs separately. The voltage regulator in the card provides VCC and ground to the drivers.

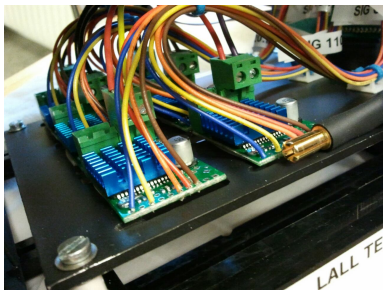

Figure 43: Motor controller.

5.9 System verification

System was verified according to a pre-defined testing protocol in unit testing. This testing is made to verified the functionality of the system before final assembly. Testing is divided into testing of prototype during soldering, for instance could the USB communication be tested before the micro-controller was soldered to a card. When a card has finished assembly it is tested individually, for instance to see that it can be programed with blinking LED's. Then cards in a group like the Sharkboard and nodes are tested, finally several boards like Sharkboard and power distribution board is tested as a whole. This unit testing can be over-viewed in the assembled test cases in the figure [44]. More detailed information of the unit testing can be found in the separated files named U - (board indicator) - (test number) like U-P-001 in Appendices D.

Pass
Fail
No change

Test case ID	Test case description	Result(Pass/Fail)
U-P-001	Finding motor electrical current.	No change
U-P-002	Finding ITX electrical current .	No change
U-P-003	Programmability of power distribution board.	
U-P-004	Emergency stop functionality in power distribution board.	
U-P-005	Functionality of CAN and LCD In Power distribution board.	
U-R-001	Test serial communication in Router.	
U-R-002	Programmability of Router.	
U-R-003	Test CAN communication in Router.	
U-S-001	Verify power levels in GPIO.	
U-S-002	Programmability of GPIO.	
U-S-003	Test CAN communication of GPIO.	
U-S-004	Communication test of GPIO.	
U-S-005	Testing of new software in Old sensor card.	
U-M-001	No-Load voltage and full-load voltage of the voltage regulator in Backbone.	
U-M-002	Test Voltages on Connectors in Backbone.	
U-M-003	Test Voltages on V-Regs in Backbone.	
U-M-004	Heating Test of V-Regs in Backbone.	
U-M-005	Output Current with and without Load in Backbone.	
U-M-006	Voltage Stability in Backbone.	
U-M-007	Interference test in Backbone.	
U-S1-001	Test of power peripherals in Old SONAR card.	
U-S1-002	Testing of Sonar card.	
U-S1-003	Test of Sonar card at pool.	

Figure 44: Overview of electronics testing.

6 Software

6.1 Introduction

This part of the report presents the software in Vasa. The system is built upon both new and old software. For the new functionalities new software has been written but certain parts that are inherited from an older software design. This is due to the fact that several parts of the old software architecture was robust and also

because the programming team saved many working hours by not "reinventing the wheel". Parts that worked and was sufficient were retained, while other parts simply were removed or reprogrammed for better efficiency.

The new functionalities that has been implemented are:

- Extended mobility
- Vision
- Tools for outside purpose
- Monitoring of the power distribution
- Dead mans grip

Extended mobility Increased movement ability results in greater mobility

Vision For the vision system an open source computer vision library called OpenCV has been used. It provides all the necessary tools for a basic vision system. Another library that was used was the libdc1394 library used for controlling firewire cameras.

The first thing that the vision system does is to grab a frame from the camera using the libdc1394 library and then converts it to a OpenCV frame so the vision algorithms provided can be applied.

Tools for outside purpose These tools are equipment that can interact with the outer environment. The things that are planned to be implemented on the robot are gripper, solenoids, markers and torpedoes as mentioned earlier in the report. These tools will be used for solving various tasks in the competition. This part of the robot is still under development.

The computational power is distributed by dividing the software into two parts, high level and low level architecture. The high level architecture runs on the main CPU while the low level architecture runs on the micro controllers in the distributed system. The architecture of the high level control is a behavioral based model. It takes input data from sensors and cameras and based on this it will decide which mission to run. When the robot shall move, the high level control will send a setpoint to the low level control that will take the robot to the setpoint by manipulating the motors.

The communication between the microcontrollers and the CPU is done via a CAN-bus (Controller Area Network). CAN is implemented because of its broadcast characteristics. Each node that is connected to the CAN-bus can at any time read from the bus and use any message that is currently on it. The system can also prioritize messages through the CAN-bus. This is a great advantage and can give a more efficient system.

The CAN is implemented in ADA. To assure a failsafe system an "I am alive"-algorithm is implemented. For each alive-message coming from the motor controller board, the CPU replies back that message. This is done to make sure that the nodes are functionally operating in the system. If the link appears to be broken, the motor controller board is deactivated (dead mans grip) so it does not keep actuating the motors. The periodicity of the test is 2 seconds. First the sensor board transmit a CAN message (ID 35) that the high level controller receives and acknowledge by replying with another CAN message (ID 36).

Vasa is equipped with a pressure sensor and an IMU. The data from these units are sent over the CAN-bus which makes it possible for the CPU and the motor controller board to read the sensor data at the same time. The pressure sensor informs the robot about its depth. The value is converted into meters.

A mission is constructed so that it can only be done once. Each mission has its own state for solving the tasks. This state will activate the different behaviors to accomplish the goal of the mission.

6.2 Vision System

The overall design of the vision system is to detect different kinds of object with a camera. To complete the missions Vasa has two angles that needs to be covered. One camera is pointing downwards and one pointing forward, shown in [Figure 45]. For the implementation of the vision software, OpenCV is used. OpenCV is an open source based library for real-time vision based computer programming. By utilizing this, the development time was greatly reduced as there was no need for developing own algorithms for image manipulations. But in order to utilize OpenCV in a easy way some sort of operating system was a criteria.

The team decided to have as good and powerful system as possible but also as small and having as little heat dissipation as possible. Consequently a mini-ITX board with an Intel Core i3 processor was chosen. This fully utilizes the possibilities and advantages of an up to date computer with a regular operating system without having restrictions on performance and reliability.

The vision software is divided into two parts. One part of the software covers the front vision camera and the other covers the bottom vision camera. The dividing of the parts were done by creating a multi-threaded system within our i3 processor. The processor and its operating system divides the calculations between its cores, thus lowering the stress on a single core.

The shapes and positioning of the obstacles and objects in the missions are known. It is therefore possible to determine how the detection of them will be handled. By knowing how objects look, approximations can be made and code can be reused for detecting several objects. For example: when detecting buoys, it is possible to use the same code as for detecting the shape of the letter 'O', as these objects are similar in shape.

Most of the object in the missions are defined by a specific color. Knowing this makes it easy to filter out the specific wanted colors by the use of OpenCV's built in functions.

Frontal Vision: Detects obstacles in front of the AUV.
Bottom Vision: Detects obstacles directly under the AUV.

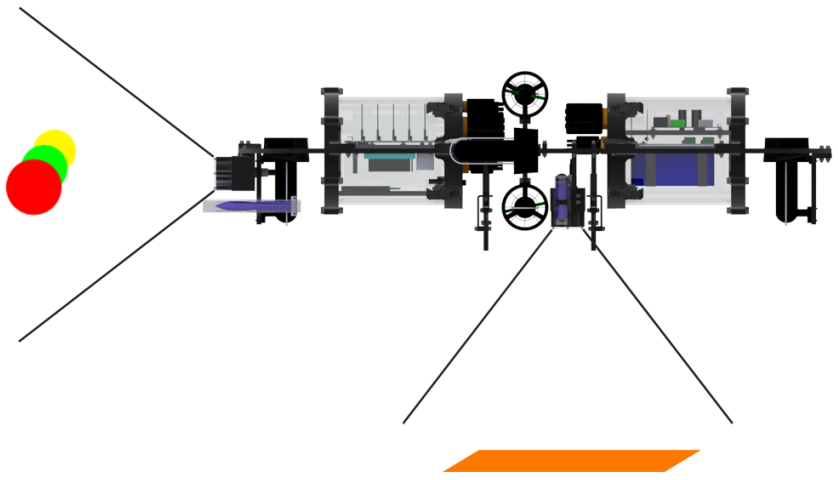

Figure 45: Camera positioning and field of view.

6.2.1 Cameras

Introduction: This section of the report presents an essential part of the vision system: the cameras. There are several aspects that must be considered when choosing cameras for an AUV. Under the water the light conditions will vary with depth since the amount of light will be reduced. Objects contrasts is also reduced with increased distance between the object and the observer. This makes it especially difficult to detect colors underwater. The reduction of light is not evenly spread across the spectrum, different colors with different wavelength is absorbed in different quantities. The red color will diffuse much faster than blue which is on the other side of the spectrum. Paper [19] discusses color registration of underwater images.

Another fact is that the refraction of the light will result in distorted images. Not only the cameras must be picked with certain demands but also the associated optics. With suitable optics, the cameras will be effective and provide the vision system with a much better image.

Under the water illumination is limited and the level of light is unpredictable. At one moment the sun might shine and everything is very bright and seconds after a cloud can block the sunlight resulting in low light condition. A good thing would therefore be the ability to adjust the amount of light being transmitted to the camera sensor. An auto iris lens provides just that. It is a lens in which the aperture automatically opens or closes to maintain proper light levels. For an even better low light performance an aspherical lens can be used. It is formed to collect maximal amount of incoming light to the camera sensor.

In addition to this, a better image can be obtained by adjustable image acquisition and processing the image. It is though very power and time consuming for the CPU to handle. To achieve the best image acquisition and the processing of the image without affecting the system, the camera board should be able to adjust image acquisition parameters and have its own processing of the image before sending it to the computer. Pre-processing on the camera board can be used to adjust various parameters of the camera as well as the properties of the image to maintain the best possible image. This will spare the main CPU from a lot of processing and the vision system will be provided with a good image. A good image acquisition can be achieved by adjusting parameters such as gain, exposure and shutter speed.

Dragonfly2

Vasa is equipped with two Dragonfly2 From Point Gray Research Inc., USA. It is a Firewire IEEE-1394 board level camera designed for imaging product development. It is constructed to provide full control and maximum flexibility for digital imaging. It has many functionalities, such as pre-processing, support for DC auto-iris and auto adjustment of image acquisition parameters along with sufficient and good documentation.

Camera specification

Specification	**DR2-HICOL-CS**
Dimensions	63.5mm x 50.8mm x 13.5mm (bare board w/o lens holder)
Mass	25 grams (bare board w/o lens holder)
Lens mount	CS-mount
Imaging Sensor	Sony 1/3" progressive scan CCD
Sensor Model	ICX204 (1024x768)
Sensor Pixel Size	4.65um x 4.65um
Video Data Output	8, 16 and 24-bit digital data
Interfaces	6-pin IEEE-1394 for camera control and video data transmission. 4 general-purpose digital input/output (GPIO) pins.
Power Requirements	8<30V, <2W
Gain	0 dB to 24 dB
Shutter	0.01ms to 66.63ms @ 15 FPS
Gamma	0.50 to 4.00
Signal To Noise Ratio	Greater than 60 dB @ 15 FPS
Camera Specification	IIDC1394-based Digital Camera Specification v1.31
Operating Temperature	0 to 45 Degrees Celsius
Operating Relative Humidity	20 to 80%(no condensation)

Image Acquisition

For fast calculation of image acquisitions parameters and pre-processing on the camera, the board is equipped with a FPGA from Xilinx. It can calculate and adjust the image acquisition parameters and process the images before sending them to the computer.Some of the cameras image acquisitions features are listed below.

- Auto Exposure
- Auto Gain
- Automatic shutter control
- Frame Rate Control
- Pixel Binning and Region of Interest

The auto exposure gives the camera the possibility to automatically control shutter and gain. By using auto exposure a specific average image intensity will be achieved. In other words, this will save processing time and still provide a good image to the vision system.

Pre-processing

Some of the pre-processing features are listed below.

- Color and gray scale Conversion
- Lookup Table (LUT)
- Control of Gamma
- Control of Saturation
- Control of Sharpness
- Control of White balance (Auto)

By controlling these parameters it is possible to get a good image despite bad light conditions.

Camera and Device Control

The Dragonfly2 also has some hardware features that can be used for different purposes. It has a temperature sensor for measuring the operating temperature and it can also measure the voltage level. These two functionalities can be used to monitor the status of the camera and according to the output stress the camera more or not.

The camera also has a 8-pin GPIO connector on the board which for example can be used to control a strobe. When using the pins as inputs the incoming voltage should be either 3.3 or 5 volts. For more information see [12] technical reference manual.

Optics

The optics for an underwater vision system is very crucial. Light is refracted differently under water and the illumination might be insufficient. The optics on Vasa solves one of these issues by having an aspherical lens, an extra large aperture and auto iris. It is ideal in environments and scenarios where the light illumination is limited and the level of light is unpredictable. The "technical guide for lenses" [3] found under appendices gives a good understanding of all crucial factors. The extra **large aperture** allows more light to reach the sensor. An **aspherical lens** collects and transmits more light to the camera sensor and provides better low-light performance. The benefit of using an **auto iris** is that despite low illumination it is still possible to achieve a pleasing image. When the camera detects that it is insufficient illumination, it adjusts the iris and lets more light in. The other way around works similar. If the incoming image is too bright, the camera decreases the iris opening and providing less light to the sensor. This makes Vasa's vision system very flexible and suitable for applications where the light condition varies continuously.

The easiest way to solve the distortion problem caused by the refraction is to do it in software.

Computar - TG3Z2910FCS-IR

The optics that is mounted on the cameras is made by Computar. It is a vari-focal optics which allows the focal length to vary between 2.9mm and 8.2mm. This is preferable when the suitable focal length is unknown for the application. When the cameras are installed in the robot the angle of view and depth of view is easy to adjust. One must remember that the depth of field is influenced by several factors. A smaller iris value closes the aperture which results in a larger depth of field and a camera with high resolution has a larger depth of field.

The optics has an aspherical lens as earlier mentioned. [Figure 46] describes how the lens refracts the light compared to a normal lens.

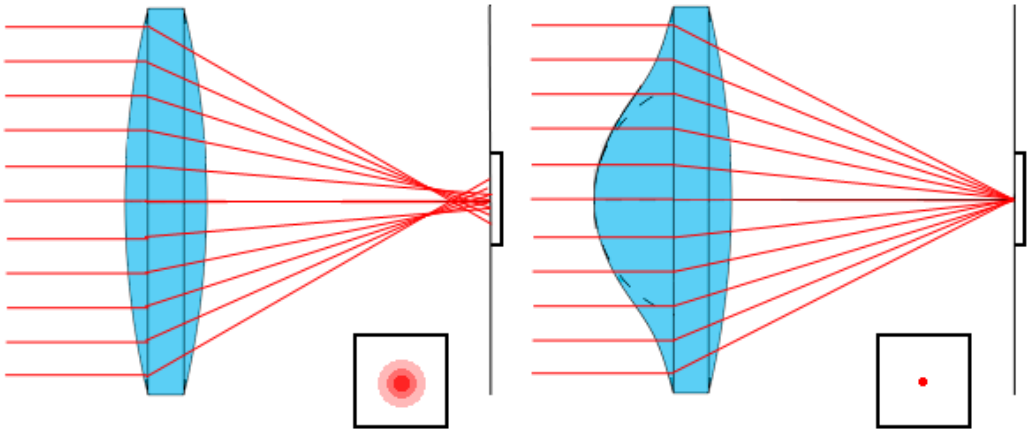

Figure 46: Left: Spherical , Right: Aspherical.

The auto iris is a DC auto iris that is compatible with the Dragonfly2 camera. The optics is connected to the camera board through a 4-pin mini connector. To save space in the camera house, the connector is mounted on the front of the camera board. When the cable is unplugged the iris is fully closed.

The Setup

[Figure 47] displays the camera setup. The two cameras are connected to the PCI-E Firewire 1394a card. The PCI-E card is then inserted to the mini-ITX board.

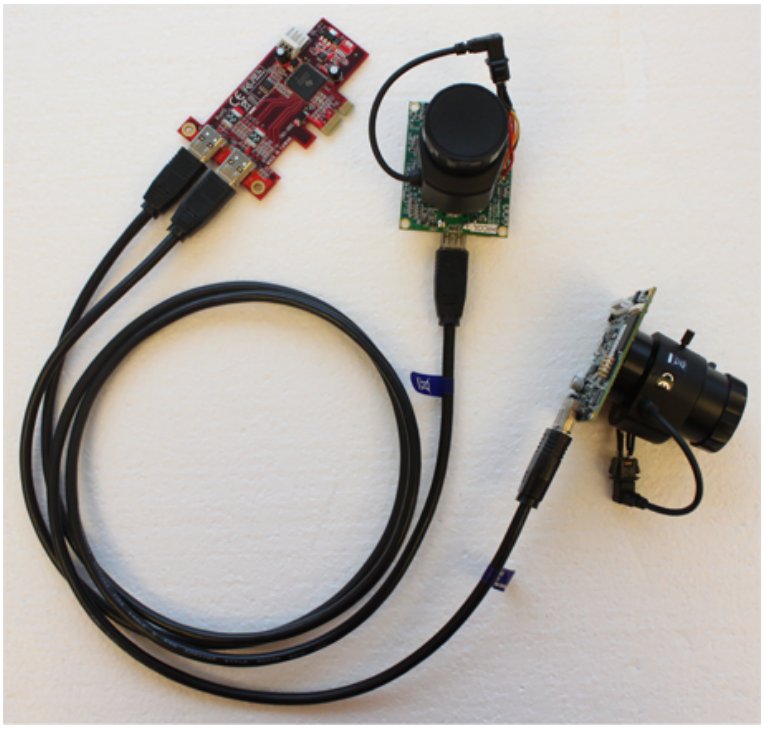

Figure 47: Cameras with optics connected to PCI-E firewire card.

The software setup that is used is mainly based on two libraries; Libdc1394 and OpenCV. The Libdc1394 library is used to operate and communicate with the cameras, it also handles the image acquisition. OpenCV is used for image processing and vision calculations.

6.2.2 OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision, developed by Intel. It is free for use under the open source BSD license. OpenCV helps students and professionals efficiently implement projects and jump-starts research by providing them with a computer vision and machine learning infrastructure.

Image Transformations
In order to track any target based on colors using OpenCV some kind of image transformations need to be done. The vision system does image transformations as following:
After getting one frame from the camera, the vision system converts RGB channels of this frame to HSV by calling the function cvCvtColor. The vision system can choose the color ranges by calling the function cvInRangeS, which means that this frame will be transformed to a binary image in which should contain the target based on the color ranges. In order to reduce unwanted or noise-driven segments, the vision system does morphological transformations on this binary image by calling the function cvMorphologyEx.
In the following section, all image transformations are done in the same way as mentioned above.

Detection of Circles:
For the detection of circles, for each frame, based on the color, after doing image transformations, the vision system pass the transformed binary image as a parameter to the function cvHoughCircles to get circles. Because of noises and interferences, the vision system may get some fake circles. In order to remove all circles which are not expected, some values as parameters should be passed to the function cvHoughCircles, these parameters include the maximum and the minimum radius of circles, and the minimum distance that must exist between two circles. After above steps are done, if there are still more than one circle, it is assumed that the biggest circle is the target.

Detection of Rectangles:
For the detection of rectangles, for each frame, based on the color, after doing image transformations, the vision system tries to find contours on the transformed binary image by calling the function cvFindContours. (According to the document of OpenCV, a contour is a list of points that represents, in one way or another, a curve in an image). Based on contours, the vision system can detect rectangles. In order to find rectangles, Firstly, the vision system locates all quadrilaterals by calling the function cvApproxPoly to find a polygon. Because the targets are rectangles, the vision system only chooses quadrilaterals. Secondly for each quadrilateral, the vision system calculate cosines of all angles between joint edges of this quadrilateral. If cosines of all angles are less than some specific value, for example, 0.3, then it is assumed that this quadrilateral is rectangle. Lastly based on the result of above steps, the vision system calculates center point, width and height by calling the function cvMinAreaRect2. This function will find the minimum rectangle boundary of the contour of the target.

Detection of the "pickup" Object:
The pickup object which is a 0.5 inch PVC pipe [Figure 8]. For the detection of the "pickup" object, for each frame, based on the color, after doing image transformations, the vision system tries to find the contour of the "pickup" object, which is the similar to the Detection of Rectangles. Based on the contours, the vision system use the function cvHoughLines2 to detect lines. At this point, it is assumed that the longest line is supposed to represent the "pickup" object, then vision system calculates the angle between this line and horizontal direction. The mission thread will need this angle and value of center point of this longest line.

Detection of the Kill Caesar:
According to the rules of the competition 2012 the targets should be two rectangles and each contains one big circle and one small circle. one of these rectangular is red,the other is blue [Figure 6].
For the detection of the Kill Caesar, for each frame, based on the color, after doing image transformations, the vision system tries to find contours on the transformed binary image by calling the function cvFindContours. Based on contours, it is assumed that the biggest contour should be that rectangular (the boundary of the Kill

Caesar), and the big circle and the small circle inside that rectangular are the targets, so the vision system removes the rectangular from contours, then the vision system tries to detect two circles among these contours. If more than two circles are found among contours, it is assumed that two biggest circles are the targets, and all other circles are supposed to be noises. Then the vision system check the spatial relationship between two circles. Firstly, two circles should be inside the rectangular, secondly, there should be some distance between these two circles. After checking the spatial relationship, the vision system will determine whether the target is found or not.

Detection of Obstacle Courses:
According to the rules of the competition 2012, Obstacle Courses looks like [Figure 48]

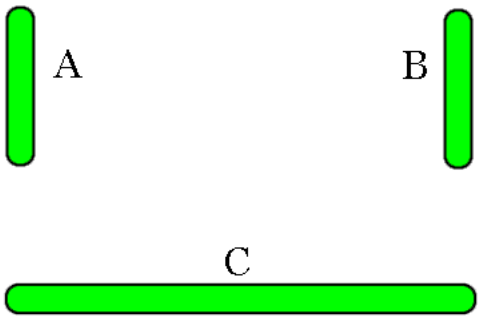

Figure 48: Obstacle courses.

For the detection of Obstacle Courses, for each frame, based on the color, after doing image transformations, the function cvHoughLines2 is used to detect lines, then these lines should be filtered. If the angle between a line and horizontal direction is less than a specific degree (At present, it is 40 degree), this line should be in the horizontal (C) direction, otherwise this line is in the vertical (A or B) direction (See above graph). All horizontal lines are sorted in the ascending order according to the value of coordinate X and these lines are put in the array C. (A, B and C, see above graph). All vertical lines are sorted in the ascending order according to the value of coordinate Y and these lines are put into a temporary array. For the vertical lines, if the value of coordinate X of a line is close enough to the leftmost point of C, this line then should be put in the array A, if the value of coordinate X of a line is close enough to the rightmost point of C, this line will be put in the array B. All other lines will be removed. In each array (A, B and C), there should be at least one line, otherwise it is assumed that Obstacle Courses are not detected.

6.3 Main CPU

6.3.1 High level CPU introduction

The high level CPU in Vasa is the brain of the whole system. This processing unit handles most of all communication that needs to be sent from any point to another. The CPU is handling our sensor data from both cameras and the IMU or pressure sensor. The CPU is mounted onto a mini-ITX board which is used in order to connect our peripheral devices such as Solid State Hard drive (SSD), WiFi antennas or cameras by the use of PCI-Express firewire module. The CAN-router card is also connected here by the use of USB interface. This is the communication port that the CPU uses to communicate through the CAN bus. With all of this information, Vasa is able to orient itself in its environment and do the necessary calculations to know what to do next. This also has the control of our mission protocol that is defined by the RoboSub 2012 competition.

6.3.2 Choice of hardware

The hardware used for the high level CPU is a mini-ITX from ASUS. The model is P8H67-I and it has been equipped with a core i3 2100T processor from Intel. Together with this the system has 8GB DDR3 1333MHZ as well as a Corsair 60GB SSD hard drive. The reason because Vasa were given the main CPU to be a well known and tested system was mainly because of well proven stability. With this type of equipment we can utilize already proven software and hardware that is compliant to a normal computer. Because for operating system in our high level CPU we are actually using a Linux based desktop server OS called Ubuntu with Xfce graphical interface. This choice of hardware has been crucial in order to fully develop our system. Especially important has been the use of OpenCV that have been talked about earlier. This has reduced the development time for the whole vision part greatly.

6.3.3 Design of high level CPU

The design of the main CPU can be said to be divided into five different groups. Each of these part has a vital role in making the program run. An architecture overview is shown in [Figure 49].

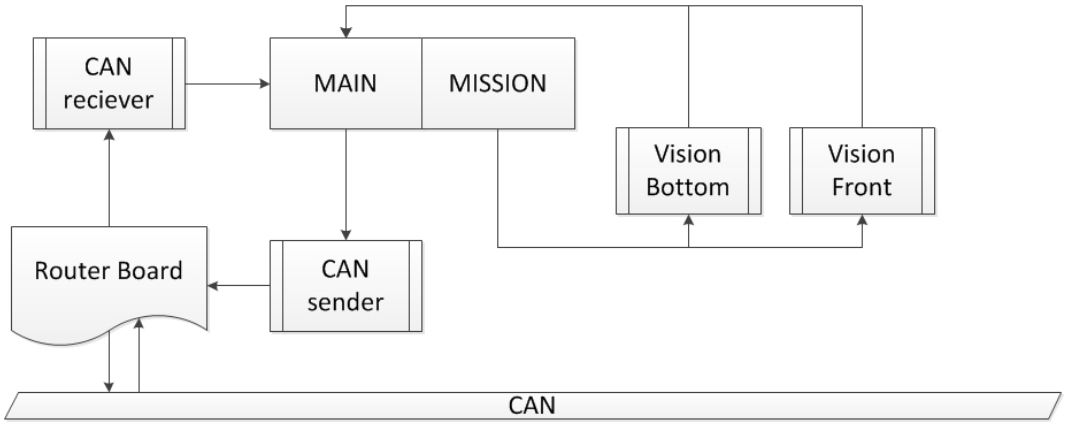

Figure 49: High Level CPU Architecture

CanSender: This part handles all the outgoing messages from our program and sends it to the router board which will redirect it to the CAN bus. This part will also check that our program will not send the same message that was the last message that was sent out to the CAN bus. This is to minimize potential and unnecessary flooding of the CAN bus.

CanReciever: The CanReciever part of the high level CPU handles all of the incoming messages from the CAN bus. It will though filter out the messages that are of no interest to the CPU. This is to minimize extra communication that might occur when doing our mission based protocol.

VisionFront / VisionBottom: VisionFront is handling all our vision specific algorithms that is used for detection of obstacles for the camera that is pointed forward. This parts only has information about algorithms that are of interest for seeing forward. It is this part that handles our OpenCV specific function calls and the vision detection algorithms that are based solenly on the RoboSub 2012 competition specific detection targets. As this part only handles the objects in front of the robot there will be no computing power doing unnecessary work for objects that is unimportant for the specific camera. The choosing of which kind of object the camera is suppose to look for is decided by the mission protocol that will be described below. This was a design choice to minimize the CPU usage by only looking for objects that we care about looking for.

The VisionBottom is keeping track of the detections for the camera that is positioned under our robot. This camera looks for objects below the robot. And this can for example be a path marker rectangle that tells us in which direction we should move. The VisionBottom part works in the same way as the VisionFront. This with respect to that it does not care about objects in front of the robot and only care about the things below it. In general both of these parts handle their own specific task depending on what the mission protocol decides and how it should work. So the two vision parts are directly controlled by the mission. And the mission will tell each vision part what kind of object it wants it to look for.

Main/Mission: The mission handler and also called the main part of the system is keeping all the other parts together. This involves keeping track of all the data coming from both of the vision parts and its respective cameras. And as described before, this is the kind of mission data specified by the Robosub 2012 competition. The main part handles the communication towards all of the other parts. So this is actually the part that controls all of the mission specific functionality and outgoing communicational protocols. When the main wants to send an outgoing command it will send it to the other respective part that will take care of it. This can be for example a message that the mission wants to send to the engines. Then it will send that specific message to the canSender part of the system which in itself will redirect it out towards the physical CAN-bus.

The Main and Mission part of the ITX is merged as one part. Even though it might be assumed to be two different parts. This is because the main part handles all incoming data from the rest of the system. While as the Mission part runs after the Main and do the necessary calculations in order to fulfill our designated tasks. So the mission will always be run after the main has gotten the latest updated information from all of the AUVs sensors. This is a design in order to minimize the number of shared memory variables that might cause locking and in worst case deadlock. This specific problem will be addressed more in the section 'Threading'.

The mission handling of our robot is consisting of a state machine that keeps track of what to do, when to do it, and how to do it. It will utilize the information from the sensors in our AUV by always having the latest most significant data at its disposal. And by having this data and knowing what to do at the specific times the robot easily knows what the outcome will be. More detailed of protocol and handling of the state machine is talked about in the 'State machine' section.

6.3.4 Threaded system

All of the above parts in our high level CPU has been divided into threads. With this the CPU has been able to fully utilize all of the processors many cores. As the i3 processor the CPU have is divided into four logical cores (two physical, with two logical each with use of hyper-threading technology). So software wise the CPU have four different cores that the CPU is now able to fully utilize workload over all of the cores. In order to communicate between our different threads the CPU are using message queues that are built in with our thread library POSIX. With these communication protocol the processor has an easy time redirecting information between the different nodes within the high level processing unit. By the use of message queues the high level CPU does not get any kind of variables that are allowed to be used simultaneously by different threads. As with queues a thread sends one message over to the other thread in terms of communicating with themselves. So the CPU will have no chances of getting corrupt readings while trying to access shared variables at the same time. And by trying to solve that shared variable problem a solution would have been to use semaphores which could have caused deadlocks in our system in the worst case. And that would cause our system to be inoperable. But all of that problems are nonexistent just because of the design the high level CPU has today with the use of message queues for terms of communication.

The message queues are built up so that it will use the technology of polling, if there is no message it will just continue with its normal run. The main thread in our system has one message queue that all of the other parts in the system sends to. So when any part wants to redirect information towards the main or the mission protocol it will just need to send an appropriate message structure over to the main by the use of the main queues message slot. same method is used when the mission wants to communicate with the other parts of our system. It is just a decision of choosing the right message queue to send it to the right thread.

6.3.5 State machine design

The high level CPU or the ITX board as it is also referred too, has a mission protocol in order to keep track of what to do in order to fulfill the specific tasks that are specified by the RoboSub 2012 competition. In order to fully keep track of all kinds of missions that will be carried out, Vasa has been implemented with a state machine. This state machine works with all of the sensor data and actuators. The state machine are able to fully control and have control of everything that is of interest for the AUV in order to complete each individual mission. As well as knowing what to do when each mission has been completed. The state machine is shown in [Figure 50].

The state machine is constructed into the following different states:

1. Start state: Set starting depths and move forward.
2. Path state: Look for a Path marker with the bottom camera.
3. Gate State: Look for the gate and align itself so it can go through it.
4. Path state: Look for a Path marker with the bottom camera.
5. Buoy state: Look for the buoy of the specific color and touch the buoy. When touched it shall look for a rectangle of the same color as the buoy, then touch that one as well.
6. Path state: Look for a Path marker with the bottom camera.
7. Bin State: Look for symbols with bottom camera so it can drop marker successfully.
8. Shooting state:Look for the object so it can shoot torpedoes through specified holes.
9. Pickup state: Listen to sonar frequency and locate the tube to pick up.

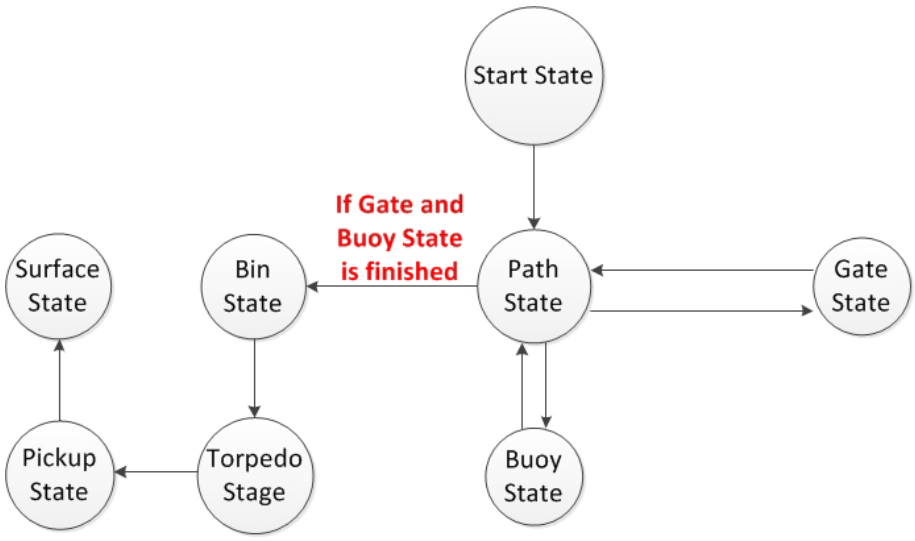

Figure 50: State machine.

6.4 Router board

In this section, High-Level CPU or Main CPU means the ITX board.
Low-Level Microcontrollers mean Sensor board, Motor board, Power board and so on.
The Router Board works like a bridge to route messages between the Main CPU and Low-Level Microcontrollers.
Software for the Router Board can be divided into 2 parts:
1. Serial communication between the Router board and the Main CPU.
2. CAN bus communication between the Router board and Low-Level Microcontrollers.

When the router board receives a message from the Low-Level Microcontrollers via the CAN bus, it translates this into a packet with a predefined format based on a protocol (explained in the next section). Then the packet is sent to the Main CPU via the serial port.

When the router board receives a packet from the Main CPU via the serial port, it translates this packet into a CAN bus message based on a protocol (explained in the next section). The message is then sent to the Low-Level Micro controllers via the CAN bus.

6.4.1 The communication protocol

For the communication between the Main CPU and Low-Level Microcontrollers, the length of a message should be from 5 to 13 bytes. A Message consists of two parts; one part is the header which includes Message type (1 byte), ID (2 bytes), Data length (1 byte) and Checksum (1 byte). The other part is the data part (see [Figure 51]). The header of a message always has 5 bytes, and the data part of a message can be from 0 to 8 bytes.

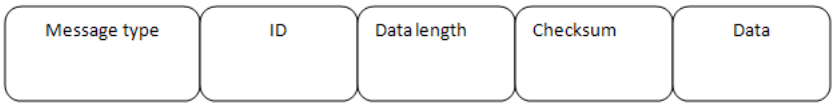

Figure 51: Message Format

Message Type (1 byte): If it is 0, this is a CAN message, if it is 3, this is a handshaking message.
ID (2 bytes): If this message is CAN bus message, this field is the ID of the CAN message, otherwise, this field should be 0.
Data Length (1 byte): The length of the data part (not including the header).
Checksum (1 byte): The checksum byte is calculated as bitwise Exclusive-OR (XOR) from all bytes of Data part (not including the header).
Data(from 0 to 8 types): The data part of a message.

6.5 Sensor board

The sensor board receives data from two nodes:

- **Pressure sensor**
- **IMU (containing accelerometer, gyroscope and magnetometer)**

The data from each sensor is separately received and handled on the board. Once the data has been processed and converted it is distributed over the CAN bus.
Once in a while the board tests the communication to the CPU.

6.5.1 The pressure sensor

The sensor data from the pressure sensor is used to determine the depth of the robot. The processing of the incoming data is explained here.

The data is received through channel 0 (ADC) on the microcontroller and the reference voltage used is 3.3 volts. The pressure sensor is adapted to be used in very deep waters (over 4000 meters) which makes it hard to read the small output that occurs just 15 meters underwater. Therefore the sensor values are amplified before it is received by the microcontroller.

The signal is converted so that a voltage of 3.3 volts is equivalent to a depth of 15 meters.

After the microcontroller has received the analog signal, the data is converted to a 10-bit digital value. A digital value of 1023 is representing 15 meters depth.

Data from the sensor is read each 25 milliseconds and stored in an accumulator which calculates the average value. The resulting value is then divided into two bytes and sent over the CAN bus.

Msg ID	DLC	Byte 1	Byte 2
51	2	3	255

An example of a CAN-message from the pressure sensor process.

6.5.2 The IMU

The Inertial Measurement Unit (IMU) uses serial communication to transmit data to the sensor board. The unit sends a string of data containing information regarding the yaw, the roll and the pitch of the robot every 25 milliseconds. The different values are separated by comma.

The string also contains characters that indicate the start and the end of the string.

!ANG	96.865,	-23.847,	15.902	<r> <n>
START	YAW	PITCH	ROLL	END

Example of a string from the IMU

The string is stored in a circular buffer each time the sensor board receives a character. The buffer is evaluated each 25 milliseconds to determine if it contains valuable information. That is if the start (!) and end (<n>) is found which indicates that there is a new string to extract. If the main program does not find a start and an end, the string is discarded and the buffer will be checked again after 25 milliseconds.

Once a new string is found the parts separated by the comma is identified. The three separated strings are then converted into unsigned 10-bit integers and distributed over the CAN bus. The 10-bit is translated into an unsigned value according to the following table.

Signed Value (Deg)	Unsigned Value (Deg)	10 Bit Value
0	0	0
180	180	512
-1	359	1024

The CAN message 71 refers to the IMU sensor values. In the message byte 1 and 2 represent the heading, byte 3 and 4 the pitch and finally byte 5 and 6 the roll. See example of a IMU message below.

Msg ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
71	6	2	0	2	0	2	0

6.6 Power distribution board

The overall idea of the power distribution board is that it handles power and can actively monitor the input voltage and be able to regulate power to the rest of the system.

6.6.1 Design of the Power distribution board

The board is designed to be able to monitor and stabilize the current and voltage that will be fed to our whole system. The software part of this is being done by the use the micro controller AT90CAN from Atmel. This micro controller will monitor the input voltage to our system by constantly reading a input pin that has the input voltage on. This pin though have been scaled down 10 times so that it doesn't feed the micro controller with a too high voltage thus breaking it. This voltage will be converted by the micro controller and handled as a DC voltage or in software it will look like an integer. The micro controller will then do specific tasks depending on what kind of module is connected and what its values are. As mentioned this board does not only handle the downscaled Vcc voltage, but it also handles a series of analog GPIO pins (General Purpose Input Output). These pins can be use to connect any kind of sensor or module that sends back analog values. The final design should handle two analog switches and one temperature sensor. It will also include one output that is designated for controlling a fan.

6.6.2 Mounted LCD screen

The powerboard is equipped with a 84 x 48 pixel LCD screen from InexGlobal. This display is used for outputting information to the user. The display will output the values of the voltage the powerboard is supplied with. And with some margin of error it will also display the status in percent. The percentage shows how fully loaded the batteries are. As well as the status of the batteries, the LCD also shows which state the switches are currently in.

[Figure 52] illustrates an output of the LCD during operational voltage supplied and [Figure 53] shows a display during too low operational voltage supplied

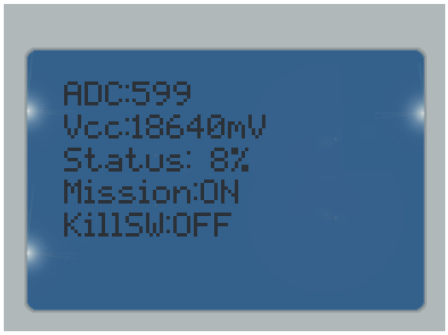

Figure 52: Illustration of possible output from LCD during normal runtime.

6.6.3 Flow of events

The powerboard first checks the supply voltage. Converts it to a value that it can be handled by the micro-controller. So in this case this is done by converting the signal into an integer number with the use of the microcontrollers built in Analog-digital converter (ADC). This value is then compared to a predefined maximum and minimum value. If any of these cases are true, that is if the input voltage is higher or lower then what we define as operating voltage. Then the powerboard will lock itself into a never ending state that will handle this conditions. In order to get out of this state then the board needs to be manually reset.

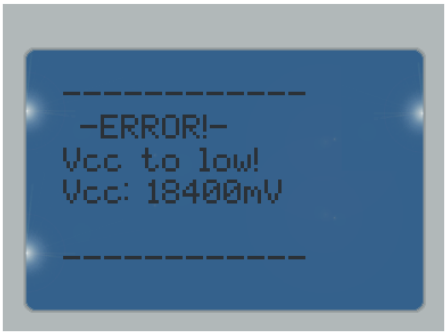

Figure 53: Illustration of possible output from LCD during a undervoltage feed.

What the board should do in in the event of a voltage lower or higher than the operating range is the following:

1. Cut power to both the engines as well as the rest of the system. This will trigger the ITX to turn itself of automatically.
2. Send a CAN message to the motorcontroller to turn of the motors. This only acts as a failsafe in case the relays does not work.
3. . Print error message on LCD screen.
4. Wait until user triggers a reset (reset button or changing of batteries).

Flow of events during normal operational runtime:

1. Get and calculate the input analog pins.
2. Check so that the input voltage is within acceptable limits.
3. Compare the values for the two switches and handle them in case of a change of state in a switch.
4. In case of a switch has have a change of state it will send an appropriate CAN message to the high level CPU.
5. Write on LCD screen the ADC value, Vcc input in millivolt and a percentage of battery level. The LCD will also show information about the state of the switches.

6.7 Motor controller

The motor controller board receives data from two nodes:

- **The sensor board**
- **The high level CPU**

The data received from the sensor board is originally from the IMU and the pressure sensor. The purpose of gathering data directly from the sensor board is so that the robot can keep balance without involving the main CPU. This results in a more efficient system.

The high level control sends data (set point) telling how the robot should orientate and move itself. This information is acquired by the vision system and the sensor board.

With the sensor data and the set point the robot executes three PD controllers controlling yaw, pitch and roll respectively. After the PD controller the motor controller board generates output signals and actuates the motors.

6.7.1 Output signals for controlling the motors

Each one of the motor is controlled by three types of signals. One of the signals is a PWM signal that controls the speed of the motor. The other two signals are controlling the direction of the rotation.

Motor Num	Timer	PWM Mode	Resulotion	Freq. (KHz)	Duty C. Register	Direction bit 1	Direction bit 2
1	3	P.C.	10-bit	7.82	OCR3A	PORTE.bit1	PORTE.bit2
2	3	P.C.	10-bit	7.82	OCR3B	PORTE.bit6	PORTE.bit7
3	3	P.C.	10-bit	7.82	OCR3C	PORTE.bit2	PORTE.bit3
4	1	P.C.	10-bit	7.82	OCR1A	PORTE.bit3	PORTE.bit4
5	1	P.C.	10-bit	7.82	OCR1B	PORTE.bit4	PORTE.bit5
6	1	P.C.	10-bit	7.82	OCR1C	PORTE.bit5	PORTE.bit6

P.C. = Phase Correct

To control the rotation of the motors the two direction bits must be set in a certain combination or the motors will shut down. A motor will go forward if direction bit 2 is set to be true and direction bit 1 set to be false. To make a motor go backwards it must be the opposite, direction bit 2 must be set to true and direction bit 1 must be set to false.

6.7.2 Communication test

Just like the sensor board the motor controller board has a communication test. It tests the communication link to the high level control on a periodic basis. It has the same benefits as the communication test in the sensor board. If the communication test fails three times, a function named “dead man grip” will shut down the motors.

6.7.3 Balancing Vasa

When controlling the balance of the robot the current sensor values from the IMU are compared to a specific set point. All the set points are set by the high level CPU in real time (CAN message 71), except for the set points for the roll control and the pitch control. These two set points are always set to zero because desired value is always the same.

ID	Description
71	**Bytes 1 and 2**: Yaw value (10-bit) A value of 0 represents a 0 degrees deviation from north. A value of 1023 represents a 359 degrees deviation from north. **Bytes 3 and 4**: Pitch value (10-bit) A value of 0 represents a 0 degrees deviation. A value of 1023 represents a 359 degrees deviation. **Bytes 5 and 6**: Roll value (10-bit) Same conversion as for the pitch.

Yaw, Pitch and Roll values from the sensor board to the motor controller board

6.7.4 Depth control

The objective for this control algorithm is to maintain the right depth of the robot according to the set point received from the high level controller. The depth controller has a periodicity of 240 milliseconds and a 10-bit resolution. Every 240 milliseconds the depth controller expects to receive new data from the pressure sensor informing of the current depth. After receiving a new depth value the depth controller will actuate the motors to go to or keep the right depth. A 10-bit resolution is used due to the fact that the pressure sensor represents the depth with 10 bits which is the same amount of bits used for controlling the PWM signals for the motors.

ID	Description
51	Bytes 1 and 2: Pressure sensor value (10-bit) A value of 0 represents a depth of 0 meters. A value of 1023 represents a depth of 15 meters.
52	Bytes 1 and 2: Depth set point (10-bit) A value of 0 represents a depth of 0 meters (surface). A value of 1023 represents a depth of 15 meters (maximum depth).

Message 51 from the sensor board and message 52 from the high level CPU to the depth control algorithm

6.7.5 Roll control

The objective for this controller is to maintain the balance of the robot by minimizing the rolling. The structure of the controller is similar to the depth controller. It has a periodicity of 240 milliseconds, a 10-bit resolution and use the same PD algorithm. Just like the depth controller the roll controller receives data from the high level control and the sensor board. The difference between the two algorithms is that the set point variable in the roll controller is static and permanently has the value 0. This is due to the fact that the desirable roll angle is 0 degrees.

6.7.6 Pitch control

The pitch controller is balancing the pitch of the robot. The pitch set point is always zero since the robot should be horizontal to the force of gravity. When controlling the pitch one must remember that the depth regulation is related to the same motors. The result from the PD algorithm is converted to a 10-bit integer and added to the output from the depth controller. This will result in the right correction of the motor speed. Just like the other balancing controllers it has a periodicity of 240 milliseconds, a 10-bit resolution and uses the PD controller.

6.7.7 Yaw control

The purpose of this algorithm is to keep the robots heading corresponding to the set point received from the high level control. It works similar to the other control algorithms. It has a periodicity of 240 milliseconds, a 10-bit resolution and uses the PD controller.

ID	Description
74	Yaw set point **Bytes 1 and 2**: Represents the linear speed (0 to 1023) **Byte 3**: Represents the direction (0 = forward, 1 = backward, 2 = right, 3 = left) **Bytes 4 and 5**: Represents the yaw set point (0 = 0 deg, 1023 = 359 deg)

CAN messages from the high level CPU to the yaw control algorithm

6.8 PD-controller

A PD controller was chosen because it satisfies the demands of the control system and due to the fact that a PID controller is in this case unnecessary complex. The PID controller does not benefit the controller when the gravity force is very low (neutral buoyancy). By using a PD controller the integration calculations are excluded and the system will be more efficient.

The PD controller gets input from two nodes; A set point from the high level CPU and current sensor data from the sensor board. This is shown in [Figure 54]. By comparing the inputs a final adjustment value is achieved and used for controlling the motors.

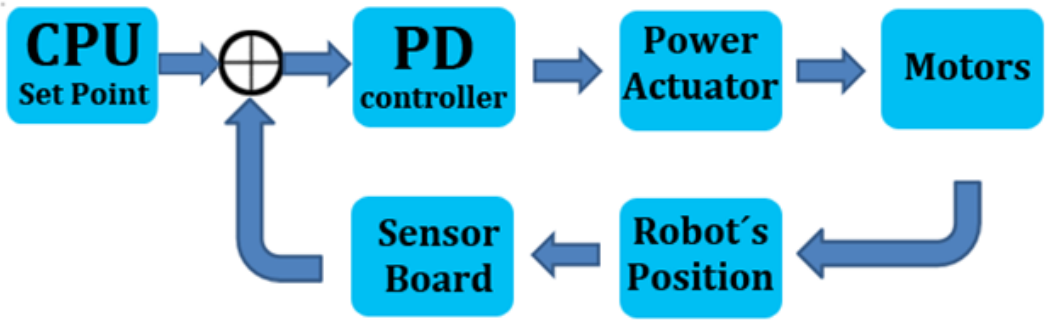

Figure 54: Control loop for the PD controller

Parameter	Value = 0	Value = 1023
Yaw	$0\,^{\circ}$	$359\,^{\circ}$
Pitch	$0\,^{\circ}$	$359\,^{\circ}$
Roll	$0\,^{\circ}$	$359\,^{\circ}$
Depth	0 meters	15 meters

Each parameter is represented by a 10-bit string.

The equations used for the depth and pitch control are explained below. One big advantage with the system is that the K values in the PD controller are adjustable from the high level CPU during run-time. This means that the system can update the P and D constants without reprogramming the motor controller board.

ID	Description
54	**Byte 1**: New value for DepthKp (Kp constant for PD) **Byte 2**: NOT USED **Byte 3**: New value for DepthKd (Kd constant for PD)
55	**Byte 1**: Stored value for DepthKp (Kp constant for PD) **Byte 2**: NOT USED **Byte 3**: Stored value for DepthKd (Kd constant for PD)
56	**Byte 1**: New value for RollKp (Kp constant for PD) **Byte 2**: NOT USED **Byte 3**: New value for RollKp (Kd constant for PD)
57	**Byte 1**: Stored value for RollKp (Kp constant for PD) **Byte 2**: NOT USED **Byte 3**: Stored value for RollKp (Kd constant for PD)
75	**Byte 1**: New value for YawKp (Kp constant for PD) **Byte 2**: NOT USED **Byte 3**: New value for YawKd (Kd constant for PD)
76	**Byte 1**: Stored value for YawKp (Kp constant for PD) **Byte 2**: NOT USED **Byte 3**: Stored value for YawKd (Kd constant for PD)

CAN messages for tuning the PD controller

The PD-controller uses the equations shown below.

The depth equations:

$$D_{error} = D_{setpoint} - D_{sensorvalue}$$
$$D_p = K_p * D_{error}$$
$$D_d = (D_{error} - D_{previous}) * K_d$$
$$D_{output} = D_p + D_d$$

The pitch equations:

$$P_{error} = P_{setpoint} - P_{sensorvalue}$$
$$P_p = K_p * P_{error}$$
$$P_d = (P_{error} - P_{previous}) * K_d$$
$$P_{output} = P_p + P_d$$

These results are then fused together to achieve the correct correction values for actuating the motors. The depth and the pitch is corrected by motor 1 and 2.

Pitch and Depth fusion (Final output to the motors):

$$Output = D_{output} + P_{output}$$

The roll and yaw correction have no motor correlation and can be corrected separately. The correction of the roll angle is made with motor 5 and 6 while the yaw is corrected by motor 3 and 4.

The roll equations:

$$R_{error} = R_{setpoint} - R_{sensorvalue}$$
$$R_p = K_p * R_{error}$$
$$R_d = (R_{error} - R_{previous}) * K_d$$
$$R_{output} = R_p + R_d$$

The yaw equations:

$$Y_{error} = Y_{setpoint} - Y_{sensorvalue}$$
$$Y_p = Y_p * Y_{error}$$
$$Y_d = (Y_{error} - Y_{previous}) * K_d$$
$$Y_{output} = Y_p + Y_d$$

The output from each part is the final value that is used to control the motors and keep balance.

7 Results

7.1 Mechanics results

The members in the mechanical group has successfully constructed and finalized a good starting platform for the robot. This framework has made it easy to continue to work on and add new features and modifications due to its high level of modularity. The main reason for the success in creating the framework was because of a successful transition from a thoroughly designed SolidWorks CAD model into a real life robot platform. The computerized model have been changed several times, both small and as well sometimes big changes for optimization. These changes though is hard to notice in the finalized model, due to a well thought system in the design stage. With the finalized model, Vasa have successfully been implemented with six degrees of freedom. This effects the robot in such a way that the robot is able to move in all possible desired motions and axis. A test showed that the hulls were watertight and can act as worthy containers for the inner electronics. Inside the two constructed hulls, one internal rack for each separate tube have been manufactured. These tubes fits very well with the constructed internal electronics.

7.2 Electronics results

The internal team which was in charge of developing and constructing the onboard electronics equipment in Vasa have managed to reconstruct previously designed electronic boards. These boards were given a new customized layout to fully fit within the mechanical design. These boards are mounted on the Sharkboard which serves as a common interface. The Sharkboard has been successfully designed by the group and serves as a medium for communication between cards as well as power to run the modules mounted on the Sharkboard. All of the designed boards have a built in communication interface which have been tested with a successful outcome. In addition to these boards, there has also been a construction of a power distribution board. This was designed in order to keep the system safe from unwanted currents that could potentially harm the sensitive electronics. The power distribution board has gotten a good foundation to work on and is currently working as intended by the use of successful monitoring of voltages. This board also handles an LCD display for displaying information and serves as a one way communication from Vasa towards an observer. The power distribution board handles the interface for two kinds of switches that will later be mounted on the mechanical frame of Vasa. As the power distribution board also has a connection to the CAN bus, it will then pass on information to the high level CPU. This has been tested successfully so that we know that all messaging and communication between nodes are working as intended. There have been problems during development of the electronics, as with almost everything of this kind of advanced project, but the finalized products are working as they were designed and intended for. This has been crucial, otherwise the overall design of the electronics would have to be redesigned. The overall mounting of the designed electronics into the internal rackets in Vasa has been tricky to do and get it there, but ones its there, it fits very well and is really compact and tidy.

7.3 Software results

The group that was responsible of the software of Vasa has been doing extensive unit testing on all newly designed components through out the project . This was done to minimize problems that might occur when the system later on was assembled. The major reason for this have been because of software being designed simultaneously as the electronics. It was also needed to have a well thought out structure for the software design for each separate unit. A good achievement of the software team was the integration of the router-, GPIO-, and power distribution board. These boards have a great design to fully work together as a unit which communicates through our CAN bus system that had already been constructed and designed in beforehand the project started. In order for Vasa to be fully reliable on completing the competition requirements, the robot needed to have a vision system. This was implemented successfully with the help of OpenCV. The overall development time has been quite short for the customized vision algorithms that will be used for detecting the different kinds of obstacles in the RoboSub competition. The vision system was implemented with DragonFly2 cameras with good results. With the use of a mini ITX motherboard the advantages of both a Ubuntu operating

system and well documented other third party software could be utilized. This ITX board have been a vital part in the system as this serves as the main CPU in Vasa. A complete framework has been designed and developed for modularity for easier implementation for future additions of more modules attached to Vasa.

8 Discussion and conclusion

Throughout the project there has been a lot of different ideas and possible solutions for them. Many of these ideas have been discussed thoroughly to be able to successfully optimize the workload within the project. Early in the project there was a decision to have a mini-ITX as the sole use as the main processing unit in Vasa. This was a pretty early and as well a very vital good choice that the team made. Even though the mechanical design of the robot needed to be larger in order to fit the processor, the group handling the software got less restrictions and could utilize pre written third party software. The team could boldly say that without the use of this ITX processing platform, there would be a big chance that a successful vision handling software would never have been completed. This is mainly because of the extensive ability to successfully implement camera functionality with the use of OpenCV.

The whole mechanics team has put much effort into creating the mechanical parts of the robot. The mechanical group has spent a long time developing. As of this, the robot has not been completely constructed with all the initial features that were originally planned for. But one should not completely see this as negative, because of the long time spent during the first half of the project time helped the mechanical group into optimizing and realizing what the robot needed design wise. The major drawback of the fact that the robot has not been completely tested with all features connected to it has been because of the late arrival of our connectors. Without these cables Vasa is unable to control anything that is outside of the tubes, for example motors and tube-to-tube communication. Though the communication between all of the nodes have been successfully tested in a dry dock platform with the tubes open and just using normal communication and power cables. With this dry dock, the software could fully test individual tasks directly on the electronics without having the need of a functional watertight robotic mechanical design. While doing these testings it could easily be seen that the robot had an autonomous behavior in the way it controls the cameras and doing appropriate actions depending on what the robot gets as feedback from the vision algorithms.

The electronics have been working hard to create a functional system and even though the finalized system has some problems, the overall electronics is working as intended. There has been one team member that have really put effort into creating a stable working system and the whole team has him to thank for having a functioning electronic system. The circuit boards has the initial functionality of the early design, though there have been some changes and ideas throughout the project course. These changes have been for instance removing the 12 volt line that was originally planned to be in the Sharkboard. In the finalized design there is no power line supplying 12 volt, there is only 3.3 and 5 volts. Another important factor is that there have been an self designed power distribution board that will supply the whole electronic equipment with built in safety protocols handled by micro controllers. The electronics and the software groups have been working close together in order to fully benefit from both teams testing protocols and to verify that all circuits and programs are working as intended. Another thing that has has some negative influence on the electronics is that there was a failure in creating a new motor controller board and because of that, the team had to reconstruct a preexisting motor controller developed earlier by the previous years underwater robot project.

The overall software development have progressed extraordinary, with a lot of progress in many fields and subjects. The subgroup worked hard on developing both a functional main processing unit that could handle all the mission specific data as well as keeping track of all incoming data from sensors and cameras. The developing of this main processing unit has progressed very well thanks to the ability to use third party software within a fully operational Ubuntu operating system. With the use of this the software team could utilize the help of OpenCV, which has been explain before. This reduced developing time for vision specific algorithms severely. The team estimated the developing time down to around three weeks for the entire vision specific tasks and algorithms.This can be compared to previous years when they were not possible to use the OpenCV or similar preexisting functionality and thus they never got any kind of vision detection throughout their entire project. Another very helpful resource was that the software group had a well documented and extensive code base to start working with. This code base were from previous project Ralf2. This code came with well functional behavior of the system that were easily to continue on with new functionality and modularity. One specially great feature of this was that it had a fully functional and well documented CAN-bus that this years Vasa software have used extensively.

9 Acknowlegments

In order to make this project a reality we have received great support. ÅF deserves a our thanks for their generous sponsorship of monetary means that have been really helpful for completing our project. We would like to thank Würth Elektronik for their technical support and sponsorship in giving us passive components, printed circuit boards, connectors and mechanical fasteners. Stainless Steel Welding HB, Torshälla for helping us produce our mechanical parts and sponsored us with material. We give thanks to Preciform AB for making the robot anodized and to Gullbergs Marina AB for gluing assistance. We also thank MDH and its personnel for their support, in particular the personnel at the University workshop, Mikael Ekström, Fredrik Ekstrand, Martin Ekström, Giacomo Spampinato, Lars Asplund, Carl Ahlberg, Bengt Erik Gustavsson, Henrik Lekryd and Göran Svensson.

References

[1] Atmel. Avr hardware design considerations. www.atmel.com/atmel/acrobat/doc2521.pdf, 2003. [Online; accessed10-January-2012].

[2] Han-Sik Ban, Jeong-Min Lee, Hyung-Soo Mok, and Gyu-Ha Choe. Load sharing improvement in parallel-operated lead acid batteries. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, volume 2, pages 1026 --1031 vol.2, 2001.

[3] Computar. Technical guide for lenses. http://computarganz.com/misc/Computar_Lens_Tech_0208.pdf, 2012. [Online; accessed 9-January-2012].

[4] J. David, L. Pcb design tutorial. http://alternatezone.com/electronics/pcbdesign.htm, 2004. [Online; accessed11-January-2012].

[5] Analog Devices. Voltage regulator. http://megads.com/analog-devices/adp2302-datasheet.html, 2012. [Online; accessed12-January-2012].

[6] Sparkfun Electronics. Imu manufacturer. http://www.sparkfun.com/products/10736, 2012. [Online; accessed11-January-2012].

[7] Sparkfun Electronics. Logic lecer converter manufacturer. http://www.sparkfun.com/products/8745, 2012. [Online; accessed11-January-2012].

[8] Würth Elektronik. Shielded power inductor. http://katalog.we-online.de/kataloge/eisos?language=en&rubrik=355/356/357/388&suchwort=we%20pd, 2012. [Online; accessed12-January-2012].

[9] J Eriksson. Robotics project ralfii. http://www.robotik.se/Ralf, 2011. [Online; accessed11-January-2012].

[10] AUVSI Foundation. Robosub official website. http://www.auvsifoundation.org/auvsi/foundation/competitions/robosub/, 2012. [Online; accessed 8-January-2012].

[11] D. Geoffroy, D. Pomerleau, M. Parent, R. Rouillard, Y. Choquette, and D. Brouillette. Lithium polymer battery in warm environment. In *Telecommunications Energy Conference, 2000. INTELEC. Twenty-second International*, pages 202 --209, 2000.

[12] Point Gray. Technical reference manual. http://www.ptgrey.com/support/downloads/index.asp/, 2012. [Online; accessed13-January-2012].

[13] F. Pierrot, M. Benoit, and P. Dauchez. Optimal thruster configuration for omni-directional underwater vehicles. samos: a pythagorean solution. In *OCEANS '98 Conference Proceedings*, volume 2, pages 655 --659 vol.2, sep-1 oct 1998.

[14] P. Schmode and J. Welsh. Design of lithium, primary battery comprising several parallel strings for high pulse current loads. In *Power Sources Symposium, 1992., IEEE 35th International*, pages 1 --3, jun 1992.

[15] Seacon. Underwater connector manufacturer. http://seaconworldwide.com/, 2012. [Online; accessed12-January-2012].

[16] Logic supply. M3-atx automotive dc-dc power supply, 125 w. http://www.logicsupply.com/products/m3_atx, 2012. [Online; accessed11-January-2012].

[17] Nordbergs Tekniska. Plastics supplier. http://www.nordbergstekniska.se/Product_Details.aspx?ID=1, 2012. [Online; accessed 9-January-2012].

[18] Scythe USA. cooling fan manufacturer. http://www.scythe-usa.com/product/acc/066/sy1212sl12_detail.html, 2012. [Online; accessed10-January-2012].

[19] A. Yamashita, M. Fujii, and T. Kaneko. Color registration of underwater images for underwater sensing with consideration of light attenuation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4570 --4575, april 2007.

A Software API

A.1 Vision API

Functioncalls for determining which kind of vision algorithm should be used in order to find different objects.

```
int detect_red_circles (void *org_frame
        ,circle_info *obj_info
        ,int camera);
int detect_blue_circles (void *org_frame
        ,circle_info *obj_info
        ,int camera);
int detect_green_circles (void *org_frame
        ,circle_info *obj_info
        ,int camera);
```

Listing 1: Function calls that will detect specific colored circle objects in an frame

detect_red_circles, detect_blue_circles, detect_green_circles	
void* org_frame	A pointer to the original frame that should be used for looking for circles
circle_info *obj_info	The structure that will get the information about the detected circle.
int camera	Integer that defines which cameras it is that should be used for detecting circles.
Return value:	Returns TRUE on a succesfull finding of a circle, otherwise FALSE.

```
int detect_color_gate(void *org_frame
      ,gate_info *gateinfop
      ,int which);
```

Listing 2: Function call that will detect gate objects in an frame

detect_color_gates	
void* org_frame	A pointer to the original frame that should be used for looking for circles
gate_info *obj_info	The structure that will get the information about the detected gate.
int which	Integer that defines which cameras it is that should be used for detecting circles.
Return value:	Returns TRUE on a succesfull finding of a circle, otherwise FALSE.

```
int detect_color_pickup(void *org_frame
      ,pickup_info *obj_info
      ,int camera);
```

Listing 3: Function call that will detect pickup objects in an frame

detect_color_pickup	
void* org_frame	A pointer to the original frame that should be used for looking for circles
gate_info *obj_info	The structure that will get the information about the detected pickup object.
int camera	Integer that defines which cameras it is that should be used for detecting circles.
Return value:	Returns TRUE on a succesfull finding of a circle, otherwise FALSE.

```
int detect_color_rectangles(void *org_frame
      ,rect_info *obj_info
      ,int camera);
```

Listing 4: Function call that will detect rectangle objects in an frame

detect_color_rectangles	
void* org_frame	A pointer to the original frame that should be used for looking for circles
gate_info *obj_info	The structure that will get the information about the detected rectangle object.
int camera	Integer that defines which cameras it is that should be used for detecting circles.
Return value:	Returns TRUE on a succesfull finding of a circle, otherwise FALSE.

```
int detect_blue_targets (void *org_frame
      ,target_info *obj_info
      ,int which);
int detect_red_targets (void *org_frame
      ,target_info *obj_info
      ,int which);
```

Listing 5: Function call that will detect shooting targets for the torpedos in a specific frame

detect_blue_targets, detect_red_targets	
void* org_frame	A pointer to the original frame that should be used for looking for circles
target_info *obj_info	The structure that will get the information about the target object.
int which	Integer that defines which cameras it is that should be used for detecting circles.
Return value:	Returns TRUE on a succesfull finding of a circle, otherwise FALSE.

A.2 Statemachine API

Statemachine specific API for handling operations within the statemachine.

```
int Mission_Handler(mqd_t canSender_id
      ,Msg_type *SendData
      ,mqd_t frontq_id
      ,mqd_t bottomq_id
      ,int *MissionReset)
```

Listing 6: Mission handler that handles our statemachine protocol

Mission_Handler	
mqd_t canSender_id	The parameter for sending to the canSender queue.
Msg_type *SendData	Updating SendData with information for alligning the motors with the pathmarker.
mqd_t frontq_id	The parameter for sending to the VisionFront thread.
mqd_t bottomq_id	The parameter for sending to the VisionBottom thread.
int *MissionReset	Integer pointer for knowing if the statemachine should reset itself
Return value:	Used for future compatibility, currently only returns TRUE

```
int Allign_rect(mqd_t canSender_id
      ,Msg_type *SendData
      ,int *roll_dir
      ,float *new_bearing
      ,int camera)
```

Listing 7: Alligning with a rectangle pathmarker

Allign_rect	
mqd_t canSender_id	The parameter for sending to the canSender queue.
Msg_type *SendData	Updating SendData with information for alligning the motors with the pathmarker.
int *roll_dir	Integer pointer for storing the direction we are swaying.
float *new_bearing	Float pointer which will store the new bearing that the robot should aim for.
int camera	Integer that defines which camera it has been that detected the object.
Return value:	TRUE on successfull allignment, otherwise FALSE

```
int Allign_pickup(mqd_t canSender_id
      ,Msg_type *SendData
      ,float *new_bearing
      ,int *roll_dir
      ,int camera )
```

Listing 8: Alligning with the object to pickup with the grippers

Allign_pickup	
mqd_t canSender_id	The parameter for sending to the canSender queue.
Msg_type *SendData	Updating SendData with information for alligning the motors with the pickup object.
float *new_bearing	Float pointer which will store the new bearing that the robot should aim for.
int *roll_dir	Integer pointer for storing the direction we are swaying.
int camera	Integer that defines which camera it has been that detected the object.
Return value:	TRUE on successfull allignment, otherwise FALSE

```
int Allign_gate(mqd_t canSender_id
      ,Msg_type *SendData
      ,int *roll_dir
      ,float *new_bearing
      ,int camera)
```

Listing 9: Alligning with the gate object so it will stand infront of the gate

Allign_gate	
mqd_t canSender_id	The parameter for sending to the canSender queue.
Msg_type *SendData	Updating SendData with information for alligning the motors with the gate.
int *roll_dir	Integer pointer for storing the direction we are swaying.
float *new_bearing	Float pointer which will store the new bearing that the robot should aim for.
int camera	Integer that defines which camera it has been that detected the object.
Return value:	TRUE on successfull allignment, otherwise FALSE

```
int Torpedo_Positioning(int size
        ,int color
        ,mqd_t canSender_id
        ,Msg_type *SendData,
        int *pressure
        ,int *roll_dir
        ,float *new_bearing)
```

Listing 10: Positioning the robot infront of the torpedo shooting object

Torpedo_Positioning	
int size	Integers that defines which size of the object it should allign itself towards.
int color	Integer that defines which color of the object has found.
mqd_t canSender_id	The parameter for sending to the canSender queue.
Msg_type *SendData	Updating SendData with information for alligning the motors with the gate.
int *pressure	Integer pointer that stores the new pressure value the robot will go to.
int *roll_dir	Integer pointer for storing the direction we are swaying.
float *new_bearing	Float pointer which will store the new bearing that the robot should aim for.
Return value:	TRUE on successfull allignment, otherwise FALSE

```
int Align_buoys(mqd_t canSender_id
        ,Msg_type *SendData
        ,int *roll_dir
        ,float *new_bearing
        ,int camera)
```

Listing 11: Positioning the robot infront of the buoy of interest

Torpedo_Positioning	
mqd_t canSender_id	The parameter for sending to the canSender queue.
Msg_type *SendData	Updating SendData with information for alligning the motors with the gate.
int *roll_dir	Integer pointer for storing the direction we are swaying.
float *new_bearing	Float pointer which will store the new bearing that the robot should aim for.
int camera	Integer that defines which camera it has been that detected the object.
Return value:	TRUE on successfull allignment, otherwise FALSE

A.3 Motorcontrol API

The following API is for controlling the motors from the Main system.

```
void Start_Controllers(Msg_type* SendData
        ,mqd_t canSender_id)
```

Listing 12: Starting the motorcontroller

Start_Controllers	
Msg_type *SendData	Updating SendData with information for starting motorcontrollers
mqd_t canSender_id	The parameter for sending to the canSender queue.
Return value:	void

```
void Stop_Controllers(Msg_type* SendData
        ,mqd_t canSender_id)
```

Listing 13: Stopping the motorcontroller

Stop_Controllers	
Msg_type *SendData	Updating SendData with information for turning of the motorcontrollers
mqd_t canSender_id	The parameter for sending to the canSender queue.
Return value:	void

```
void Yaw_Set(Msg_type *SendData
      ,unsigned short speed
      ,unsigned char direction
      ,unsigned short angle)
```

Listing 14: Setting a new YAW value for the motors

Yaw_Set	
Msg_type *SendData	Updating SendData with information for setting Yaw,speed,direction
unsigned short speed	The desired speed the motors should run at when it is alligned with the angle
unsigned char direction	The direction:BACKWARDS or FORWARD towards the angle
unsigned short angle	The desired angle the motors should align themselves towards
Return value:	void

```
void Sway_Set(Msg_type *SendData
      ,unsigned short speed
      ,unsigned char direction)
```

Listing 15: Setting a direction for sway motion

Sway_Set	
Msg_type *SendData	Updating SendData with information for setting speed and direction
unsigned short speed	The desired speed of the motors in its sway direction
unsigned char direction	The direction:RIGHT or LEFT
Return value:	void

```
void Pressure_Set(Msg_type *SendData
      ,unsigned int depth)
```

Listing 16: Set the motors to go down to a certain depth

Pressure_Set	
Msg_type *SendData	Updating SendData with information for setting depth parameters
unsigned int depth	The integer depth value the robots should go down to
Return value:	void

A.4 Sensor API

API for accessing the latest sensordata saved within the Main CPU.

```
struct curr_lowlevel_caninput_data
{
  int Pitch_Val;
  int Roll_Val;
  int Yaw_Val;
  int Pressure_Val;
  int Passive_Sonar_Val;
  int Active_Sonar_Val;
};


/*Global variable of CAN-bus sensor value */
struct curr_lowlevel_caninput_data curr_candata;
```

Listing 17: Global structures that keeps track of all the latest input data from the sensors connected to the CAN-bus

curr_candata	
int Pitch_Val	Stores the latest value of the pitch angle
int Roll_Val	Stores the latest value of the rolling angle
int Yaw_Val	Stores the latest value of the Yaw
int Pressure_Val	Stores the latest value of pressure sensor
int Passive_Sonar_Val	Stores the latest value coming from the passive sonar
int Active_Sonar_Val	Stores the latest value coming from the active sonar

```
struct curr_Vision_Findings_struct
{
  int Vis_Bottom_Type;
  pickup_info Vis_Bottom_Pickup;
  rect_info Vis_Bottom_Rectangle;
  circle_info Vis_Bottom_Red_Circle;
  circle_info Vis_Bottom_Blue_Circle;
  circle_info Vis_Bottom_Green_Circle;
  target_info Vis_Bottom_Torpedo_Blue;
  target_info Vis_Bottom_Torpedo_Red;
  gate_info Vis_Bottom_Gate;

  int Vis_Front_Type;
  pickup_info Vis_Front_Pickup;
  rect_info Vis_Front_Rectangle;
  circle_info Vis_Front_Red_Circle;
  circle_info Vis_Front_Blue_Circle;
  circle_info Vis_Front_Green_Circle;
  target_info Vis_Front_Torpedo_Blue;
  target_info Vis_Front_Torpedo_Red;
  gate_info Vis_Front_Gate;

};

/*Global variable of Vision findings information*/
struct curr_Vision_Findings_struct curr_Vision_Findings;
```

Listing 18: Global structures that keeps track of all the latest input data from the cameras

curr_Vision_Findings	
int Vis_Bottom_Type	Integer which points to the latest type of object that the bottom camera has detected
pickup_info Vis_Bottom_Pickup	Structure with latest data about pickup located by bottom camera object
rect_info Vis_Bottom_Rectangle	Structure with latest data about rectangle object located by bottom camera object
circle_info Vis_Bottom_Red_Circle	Structure with latest data about red circle object located by bottom camera object
circle_info Vis_Bottom_Blue_Circle	Structure with latest data about blue circle object located by bottom camera object
circle_info Vis_Bottom_Green_Circle	Structure with latest data about green circle object located by bottom camera object
target_info Vis_Bottom_Torpedo_Blue	Structure with latest data about blue shooting object located by bottom camera object
target_info Vis_Bottom_Torpedo_Red	Structure with latest data about red shooting object located by bottom camera object
gate_info Vis_Bottom_Gate	Structure with latest data about gate object located by bottom camera object
int Vis_Front_Type	Integer which points to the latest type of object that the front camera has detected
pickup_info Vis_Front_Pickup	Structure with latest data about pickup object located by front camera object
rect_info Vis_Front_Rectangle	Structure with latest data about rectangle object located by front camera object
circle_info Vis_Front_Red_Circle	Structure with latest data about red circle object located by front camera object
circle_info Vis_Front_Blue_Circle	Structure with latest data about blue circle object located by front camera object
circle_info Vis_Front_ Green_Circle	Structure with latest data about green circle object located by front camera object
target_info Vis_Front_Torpedo_Blue	Structure with latest data about blue shooting objectl located by front camera object
target_info Vis_Front_Torpedo_Red	Structure with latest data about red shooting object located by front camera object
gate_info Vis_Front_Gate	Structure with latest data about gate object located by front camera object

B CAN-messages

The CAN-messages shown below are distributed by the high-level CPU.

Message name	ID	Data bytes	Source	Periodicity(ms)
Communication Test Response	16	8	Main CPU	1920
Start Controllers	18	1	Main CPU	Aperiodic
Pressure Sensor Value	51	2	Sensor Board	240
Depth Set Point	52	2	Main CPU	Aperiodic
Update Depth Control Constants	54	2	Main CPU	Aperiodic
Update Pitch Control Constants	58	2	Main CPU	Aperiodic
Update Roll Control Constants	56	2	Main CPU	Aperiodic
Pitch, Roll and Yaw Values	71	6	Sensor Board	240
Yaw Set Point	74	5	Main CPU	Aperiodic
Update Yaw Control Constans	76	2	Main CPU	Aperiodic
Movement Set Point	42	3	Main CPU	Aperiodic
Torpedo Activation	20	1	Main CPU	Aperiodic
Gripper Activation	22	1	Main CPU	Aperiodic
Marker Activation	24	1	Main CPU	Aperiodic
Passive Sonar	33	2	Sonar Board	240
Active Sonar	35	2	Sonar Board	240
ShutdownMainCpu	13	1	Powerboard	Aperiodic
ShutdownMainCpu Response	14	1	Main CPU	Aperiodic
Kill switch	23	1	Powerboard	Aperiodic
Mission Switch	25	1	Powerboard	Aperiodic

The CAN-messages shown below are distributed by the low-level micro controllers.

Message name	ID	Data bytes	Source	Periodicity(ms)
Speed for motor 1 and 2	11	3	Motor Controller	240
Speed for motor 3 and 4	31	3	Motor Controller	240
Speed for motor 5 and 6	41	3	Motor Controller	240
Communication Test Request	15	8	Motor Controller	1920
Reply Depth Control constants	55	2	Motor Controller	20 000
Reply Roll Control Constants	57	2	Motor Controller	20 000
Reply Yaw Control Constants	75	2	Motor Controller	20 000

C Electronic schematics

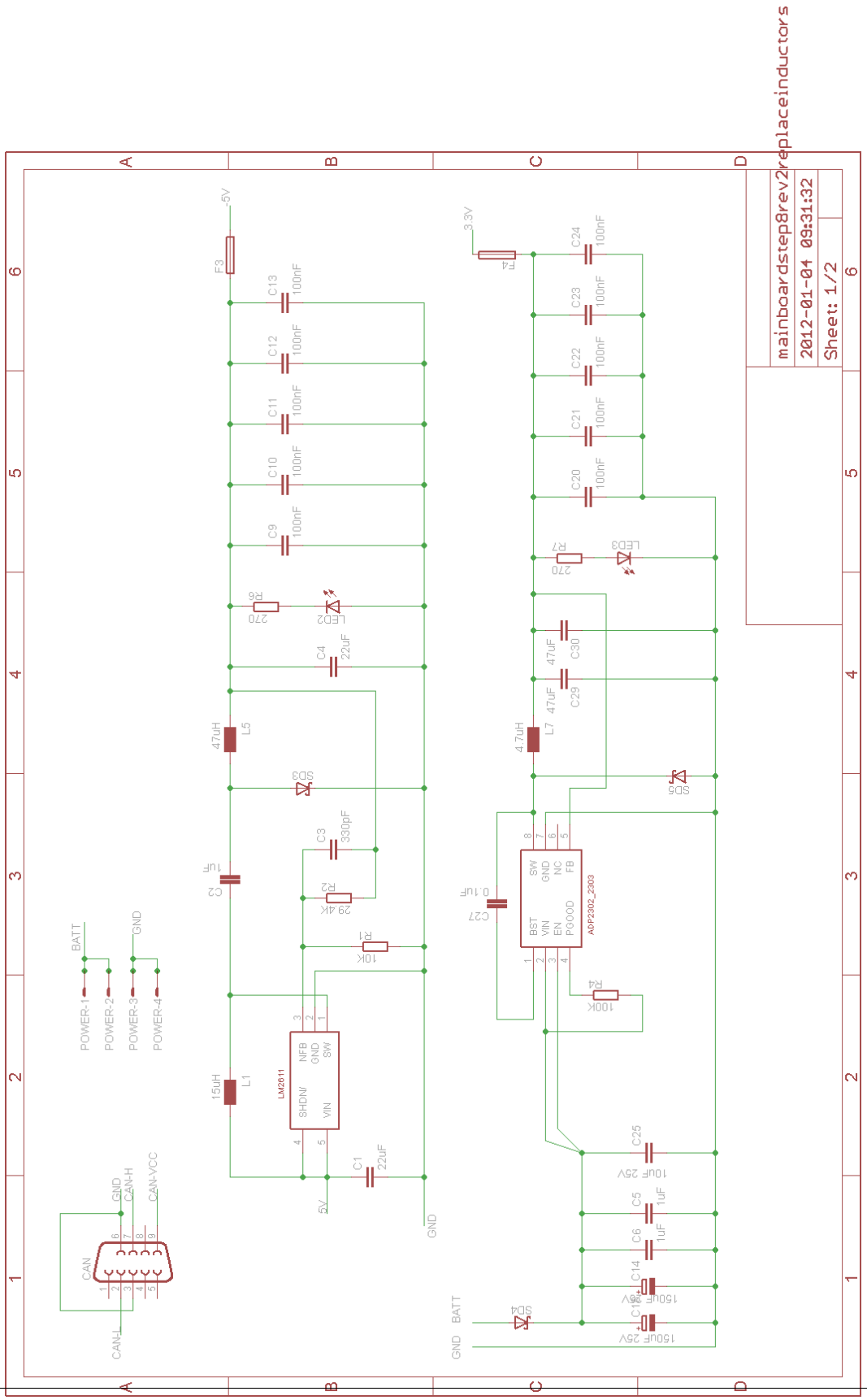

Figure 55: Backbone: Connectors, 3.3v and -5V.

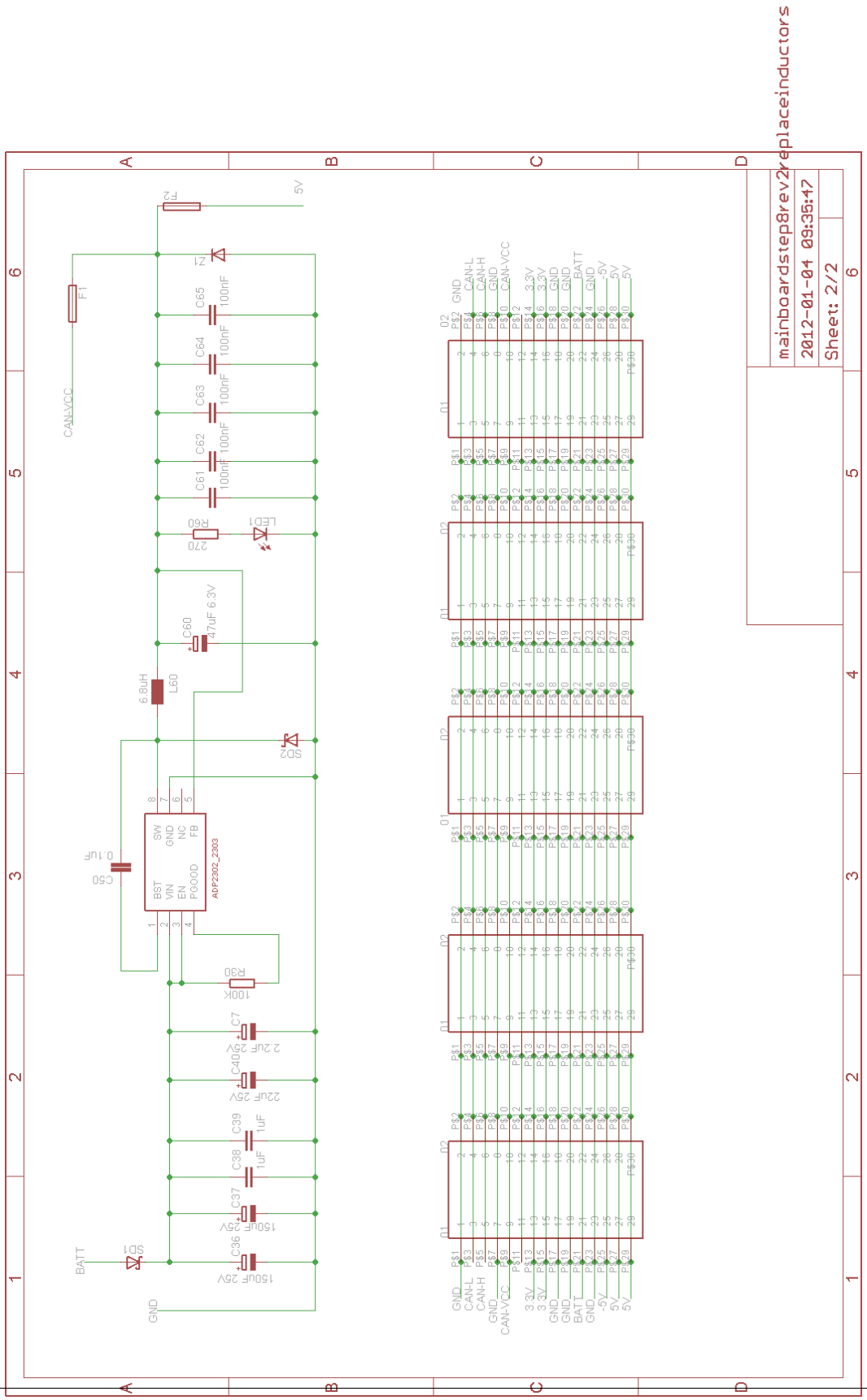

Figure 56: Backbone: 5V and header slots.

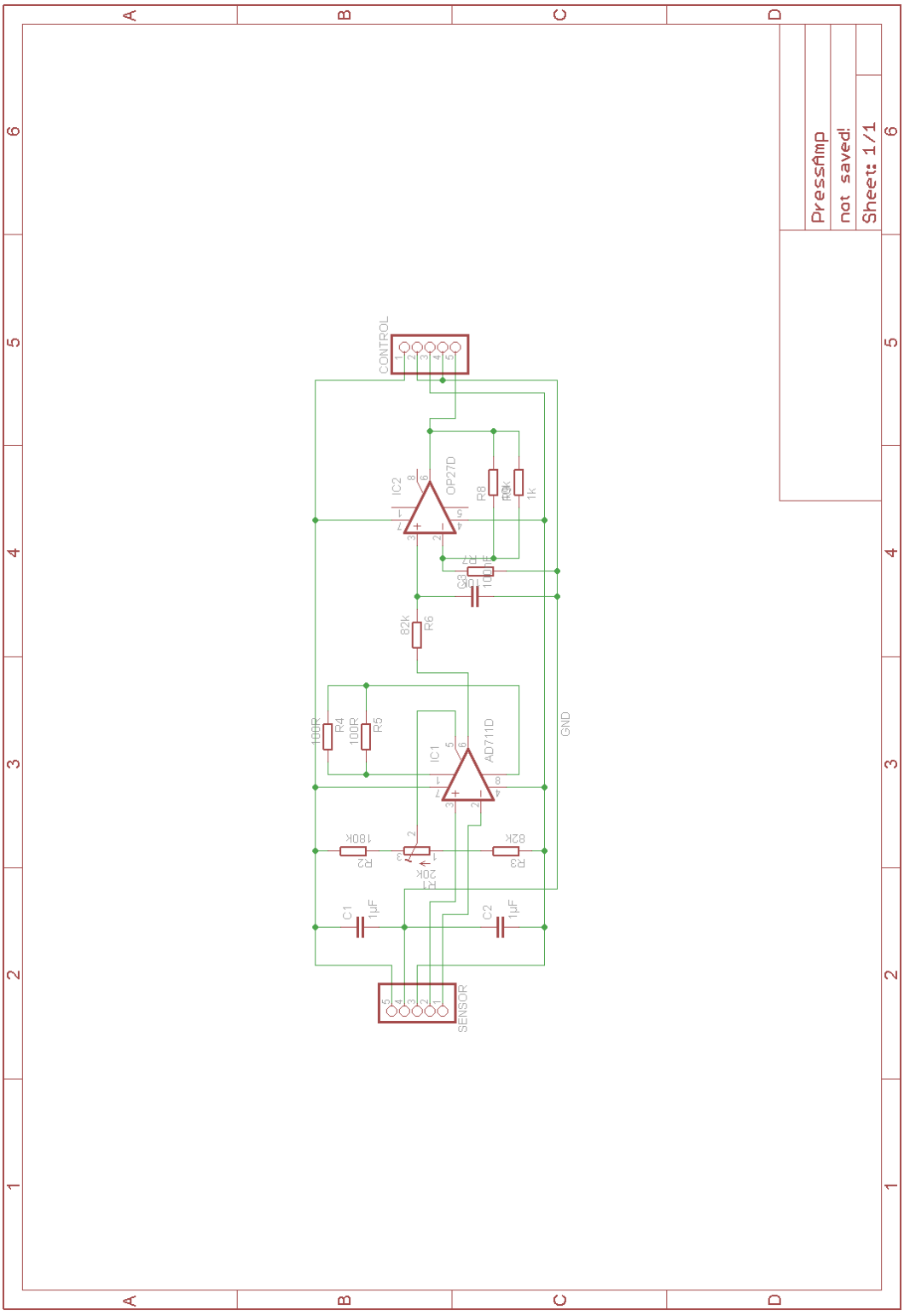

Figure 57: Pressure amplifier

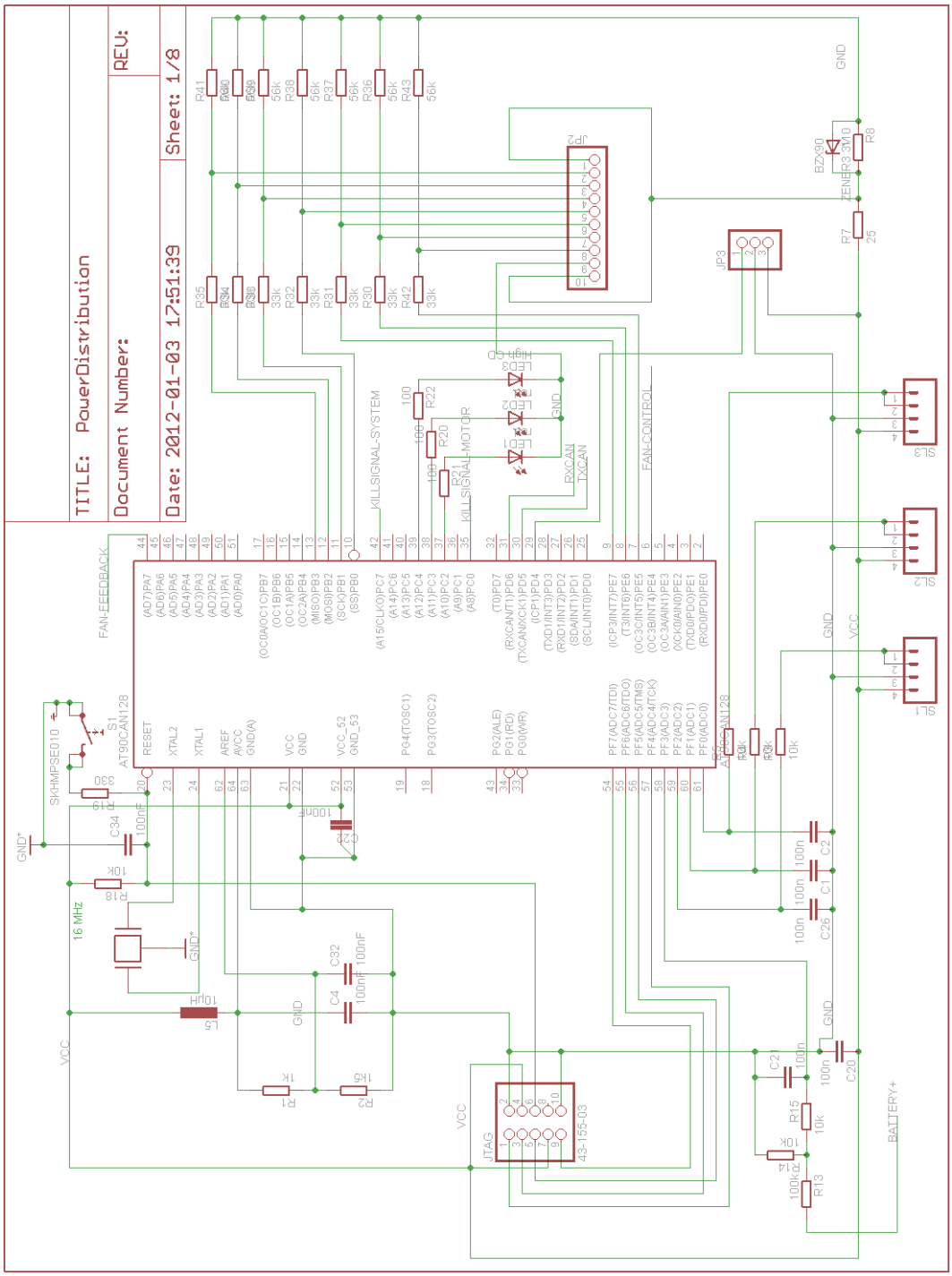

Figure 58: Powerboard: Microcontroller set up.

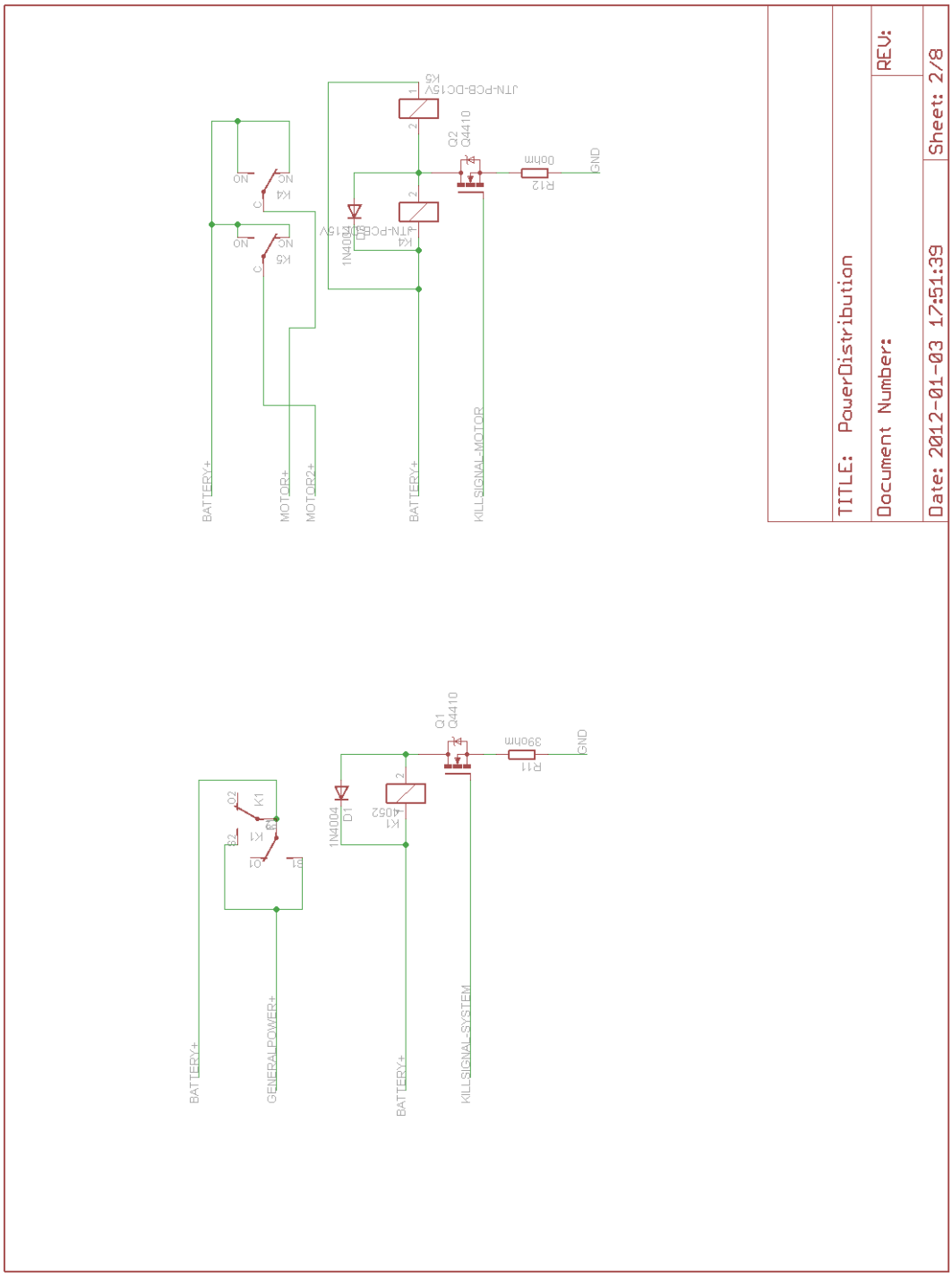

Figure 59: Powerboard: Relays.

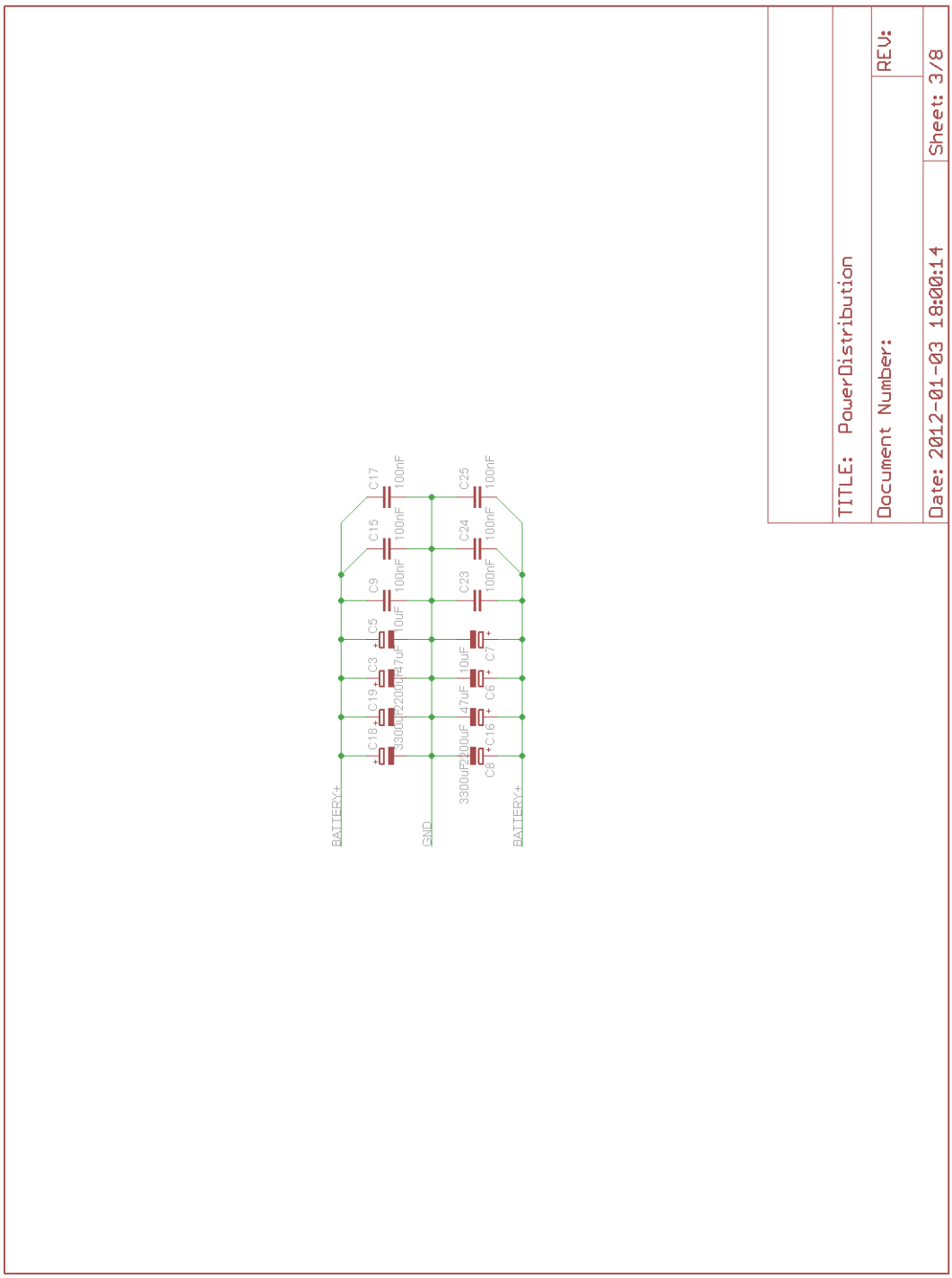

Figure 60: Powerboard: Filter capacitors.

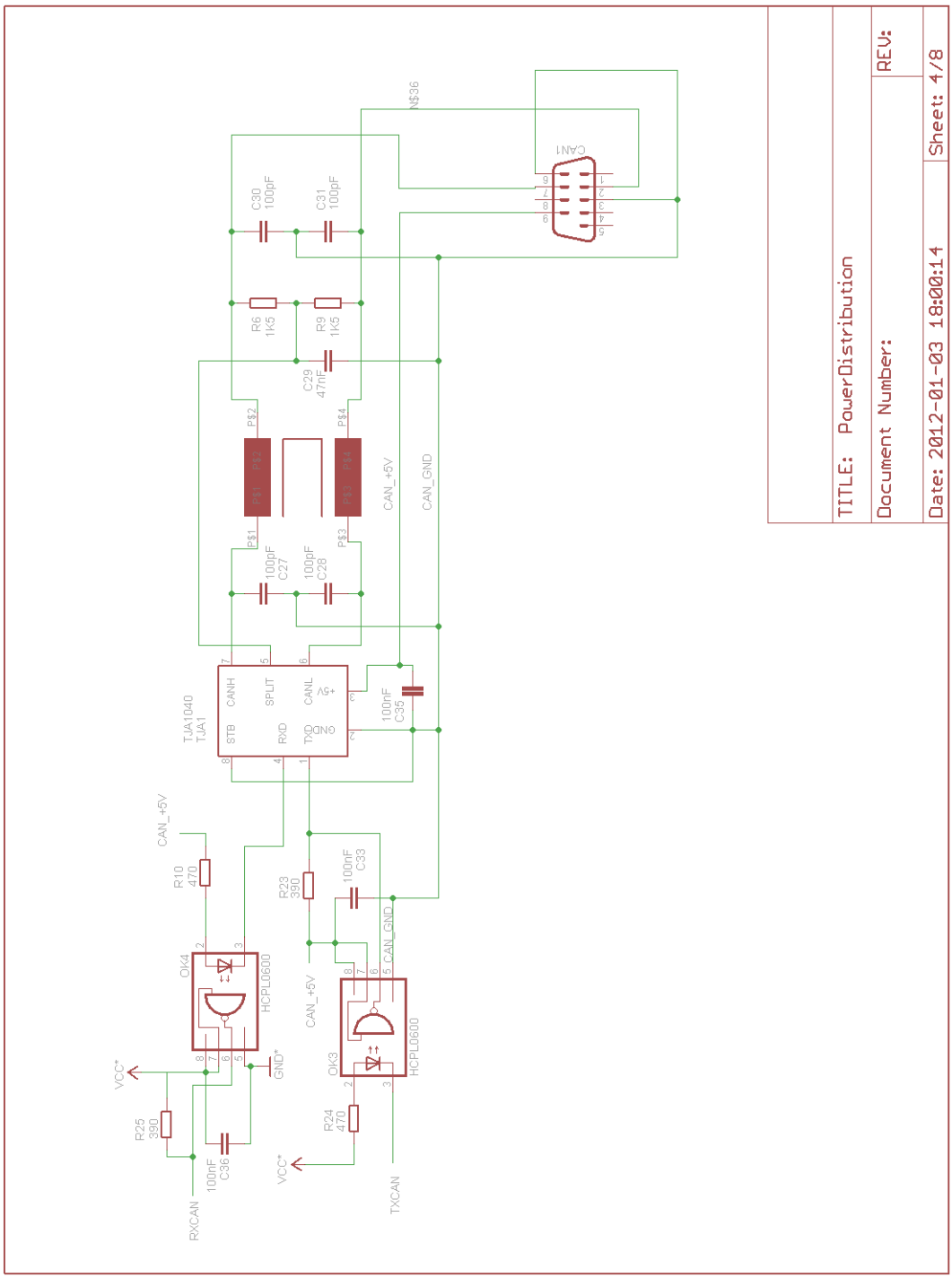

Figure 61: Powerboard: CAN.

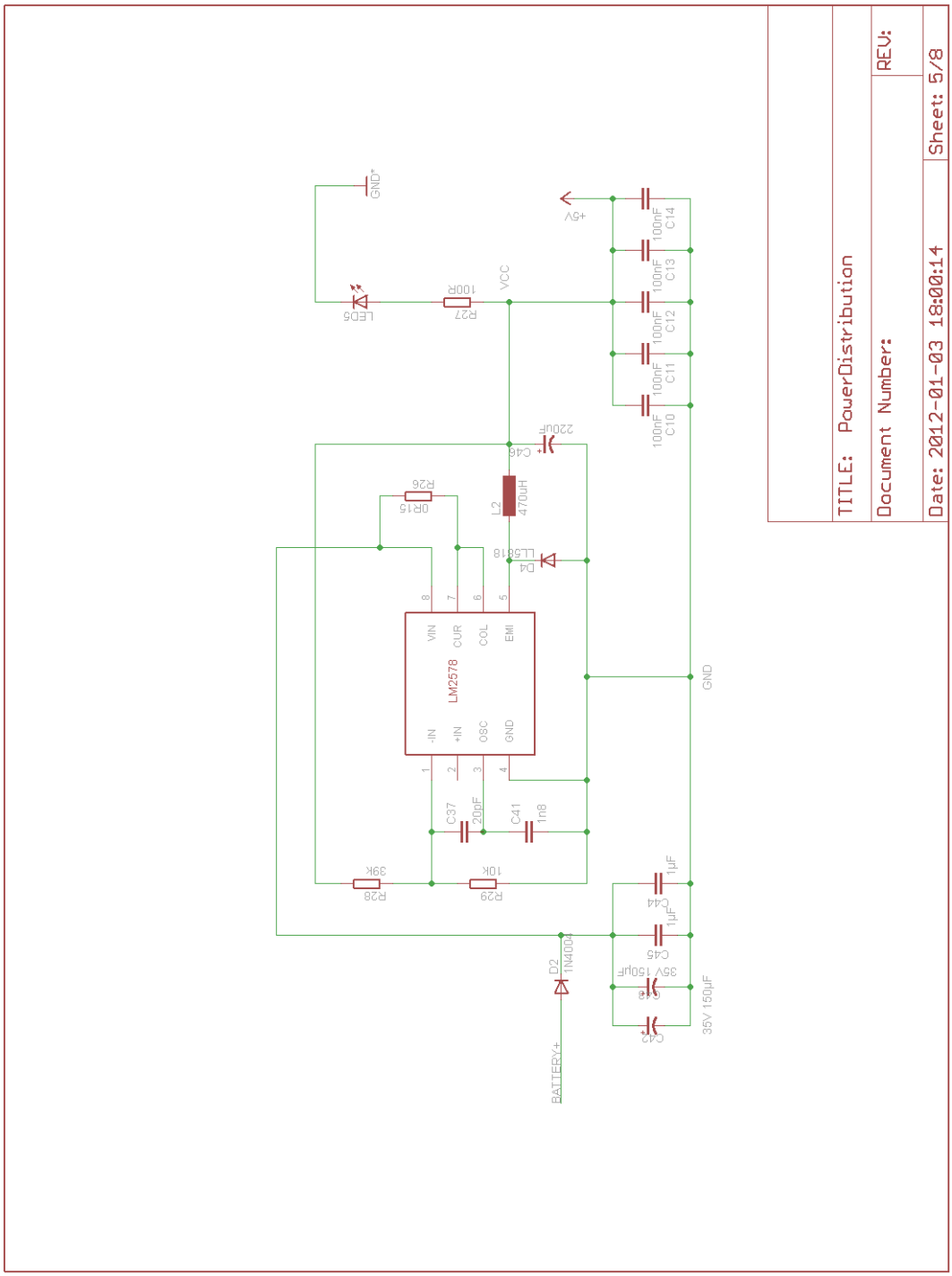

Figure 62: Powerboard: Voltage regulator logic power.

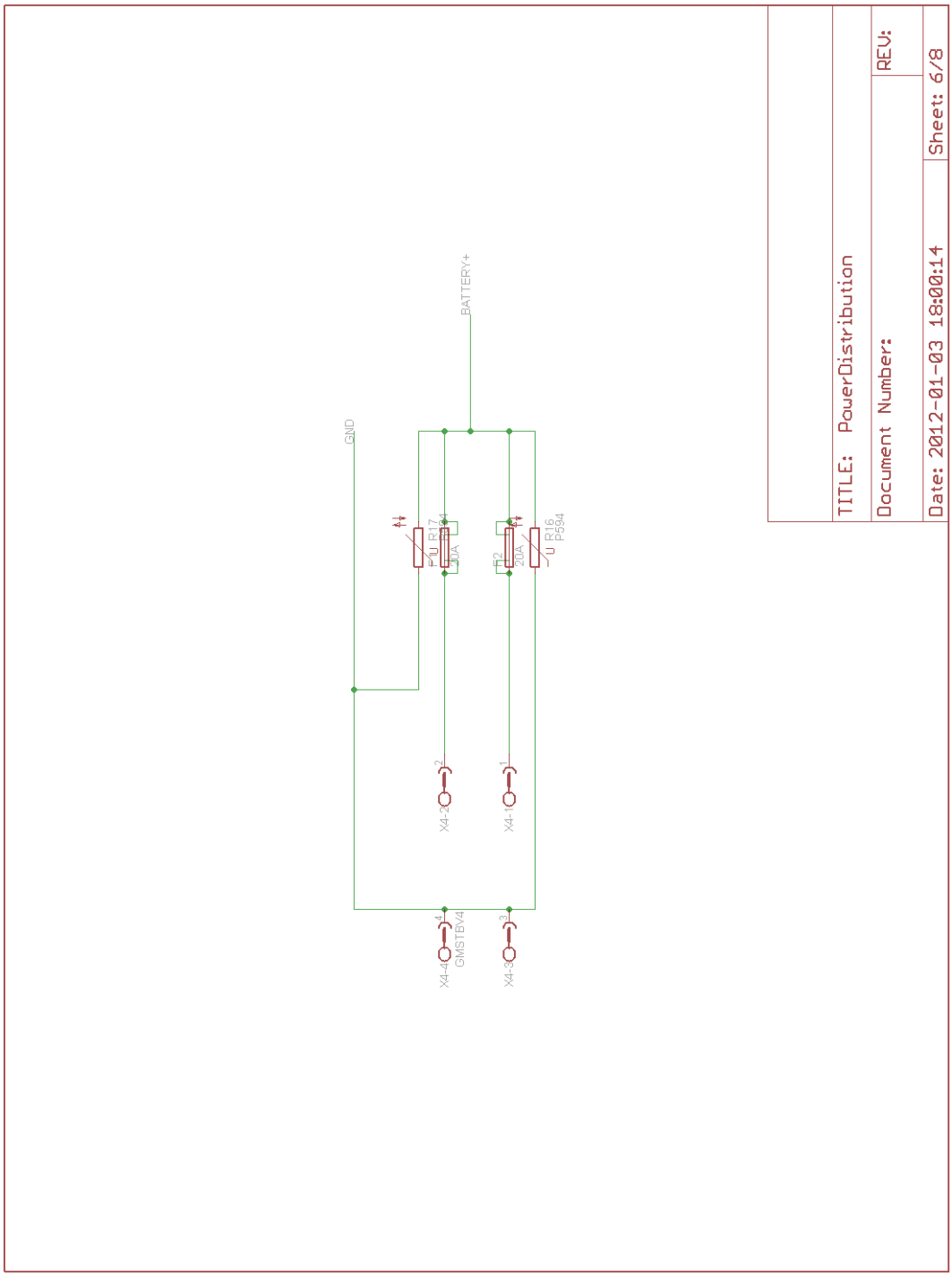

Figure 63: Powerboard: Main power connectors.

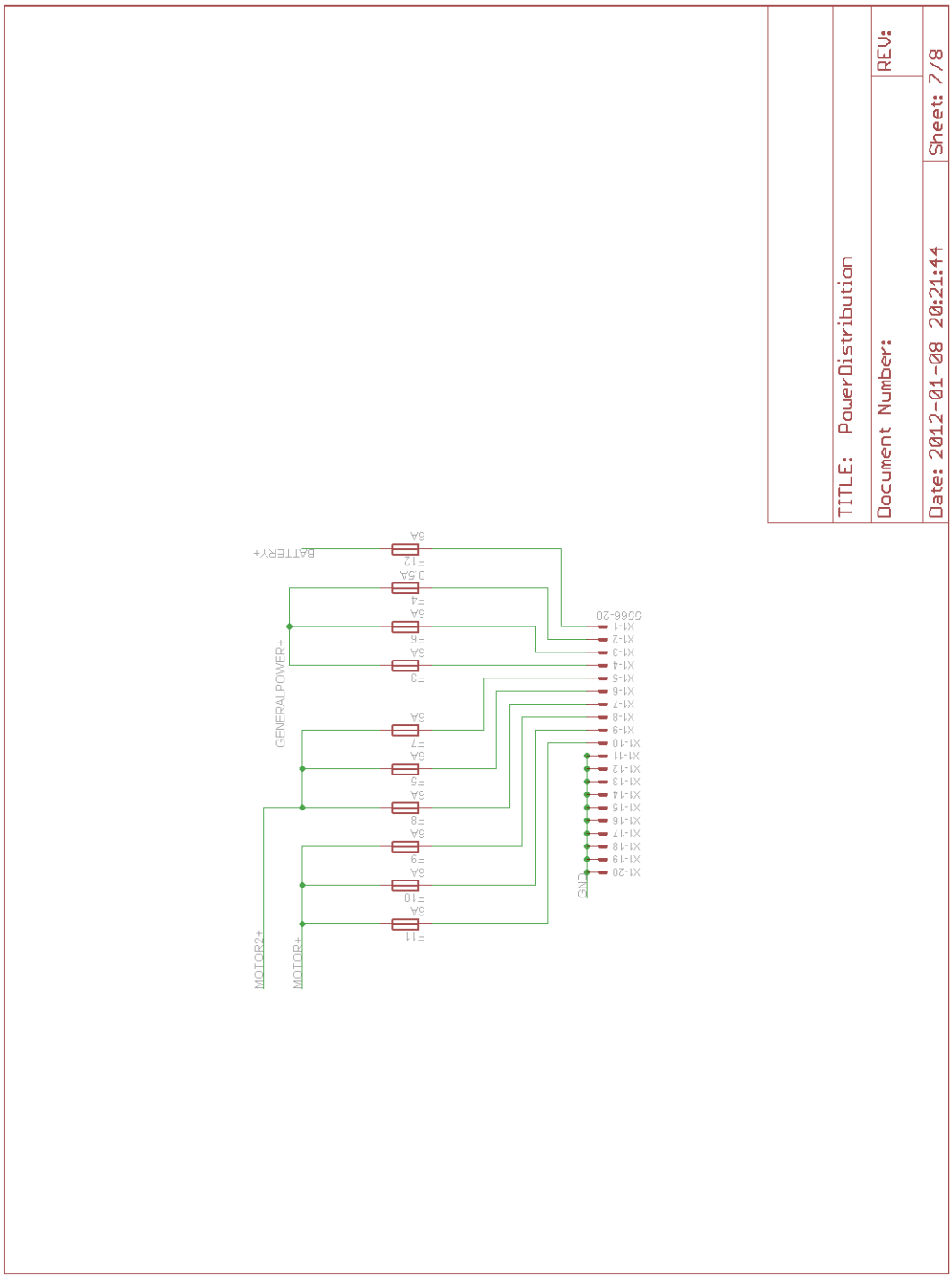

Figure 64: Powerboard: Output fuses.

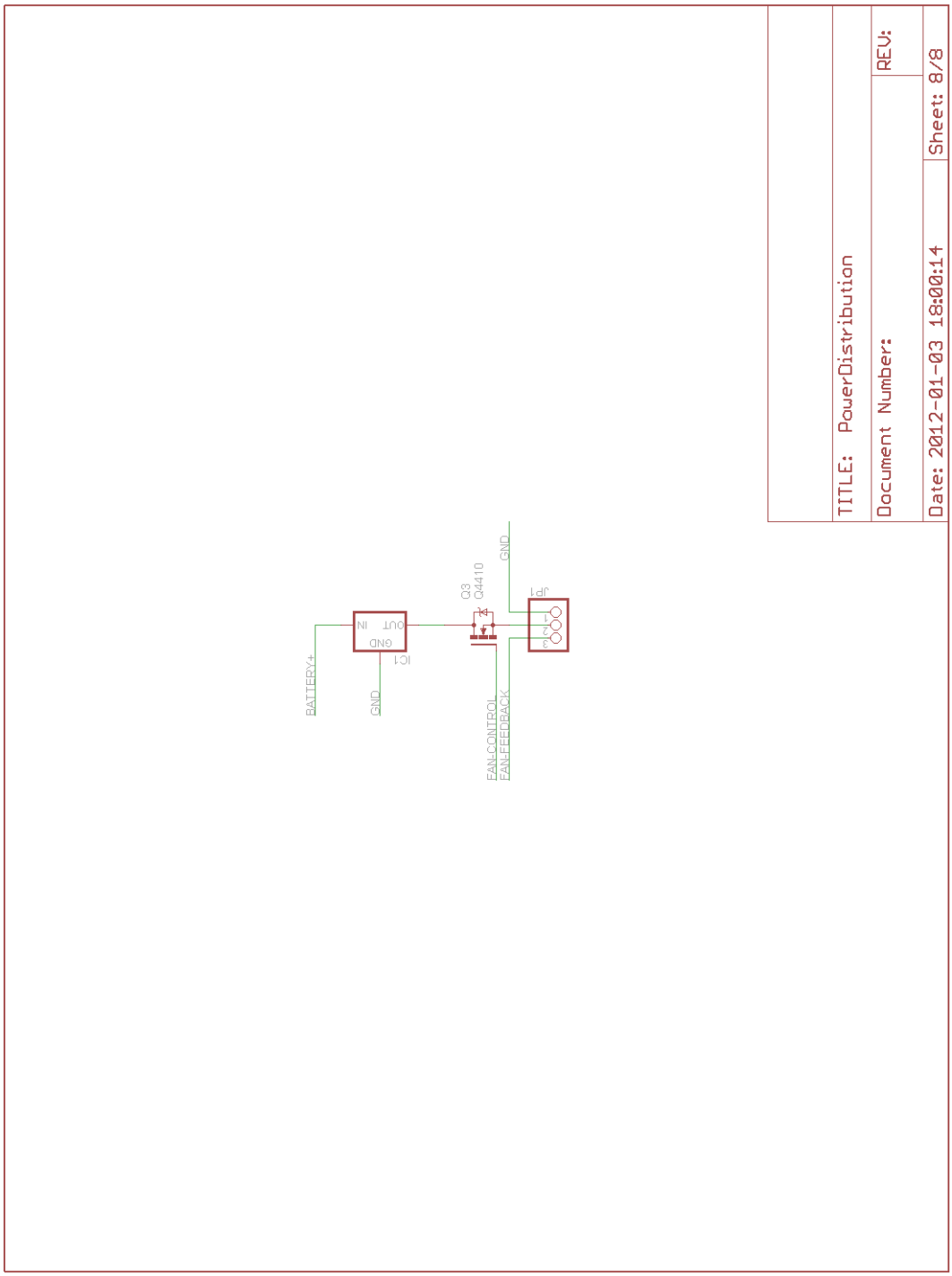

Figure 65: Powerboard: Fan control.

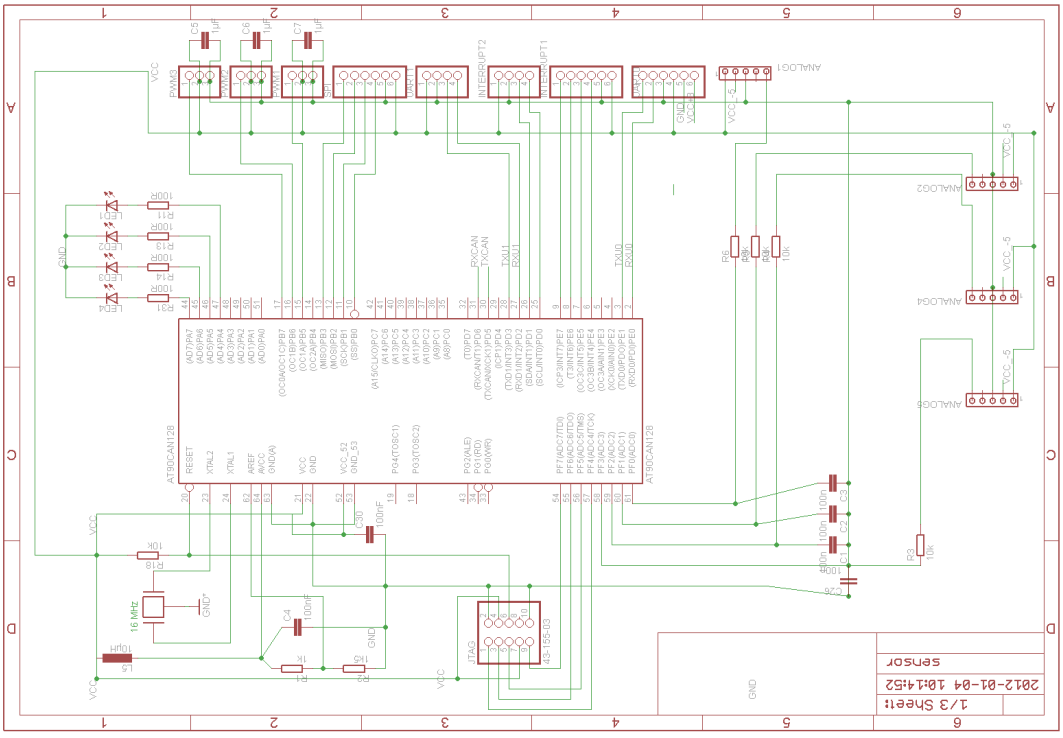

Figure 66: GPIO card: Microcontroller set up.

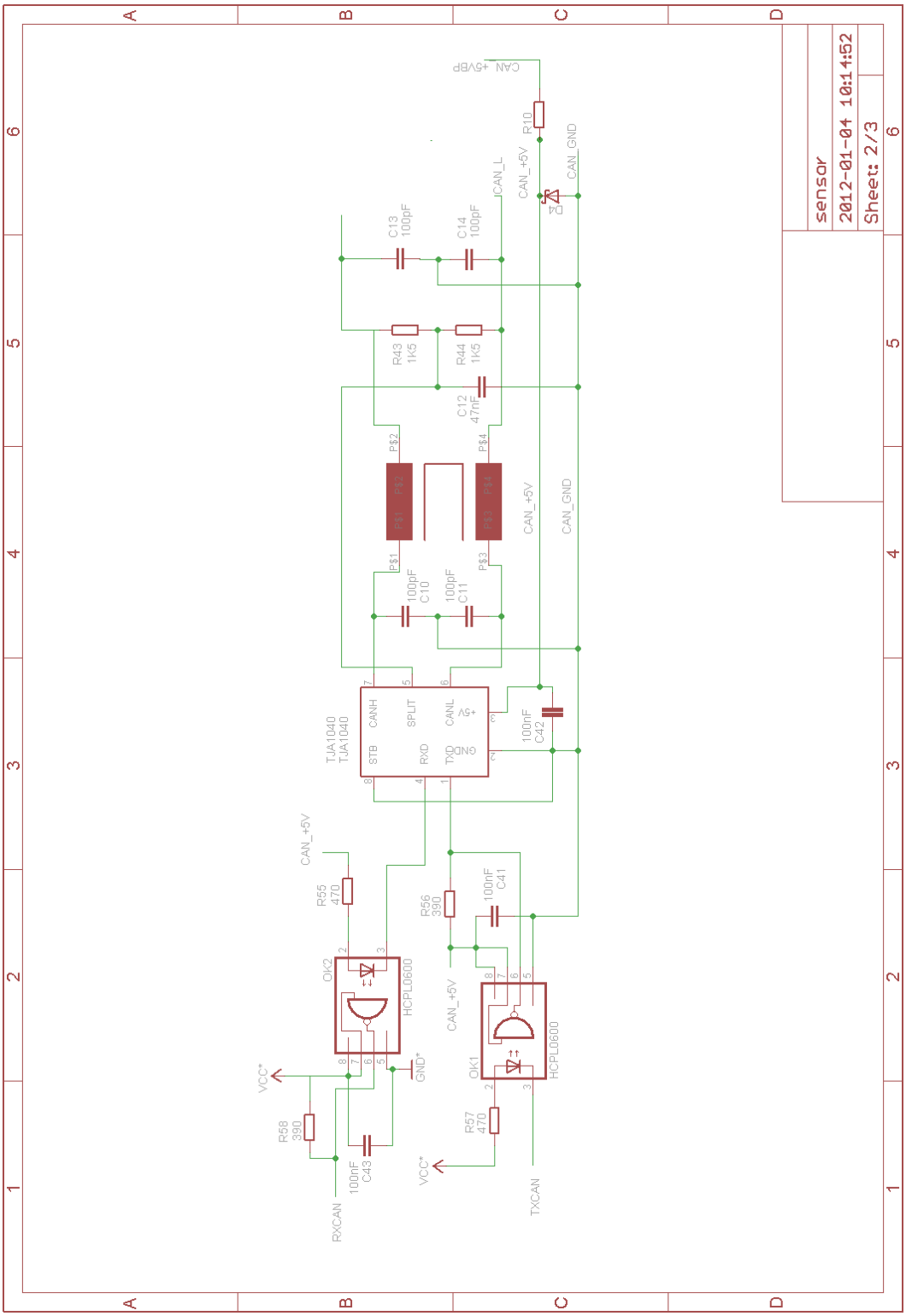

Figure 67: GPIO card: CAN.

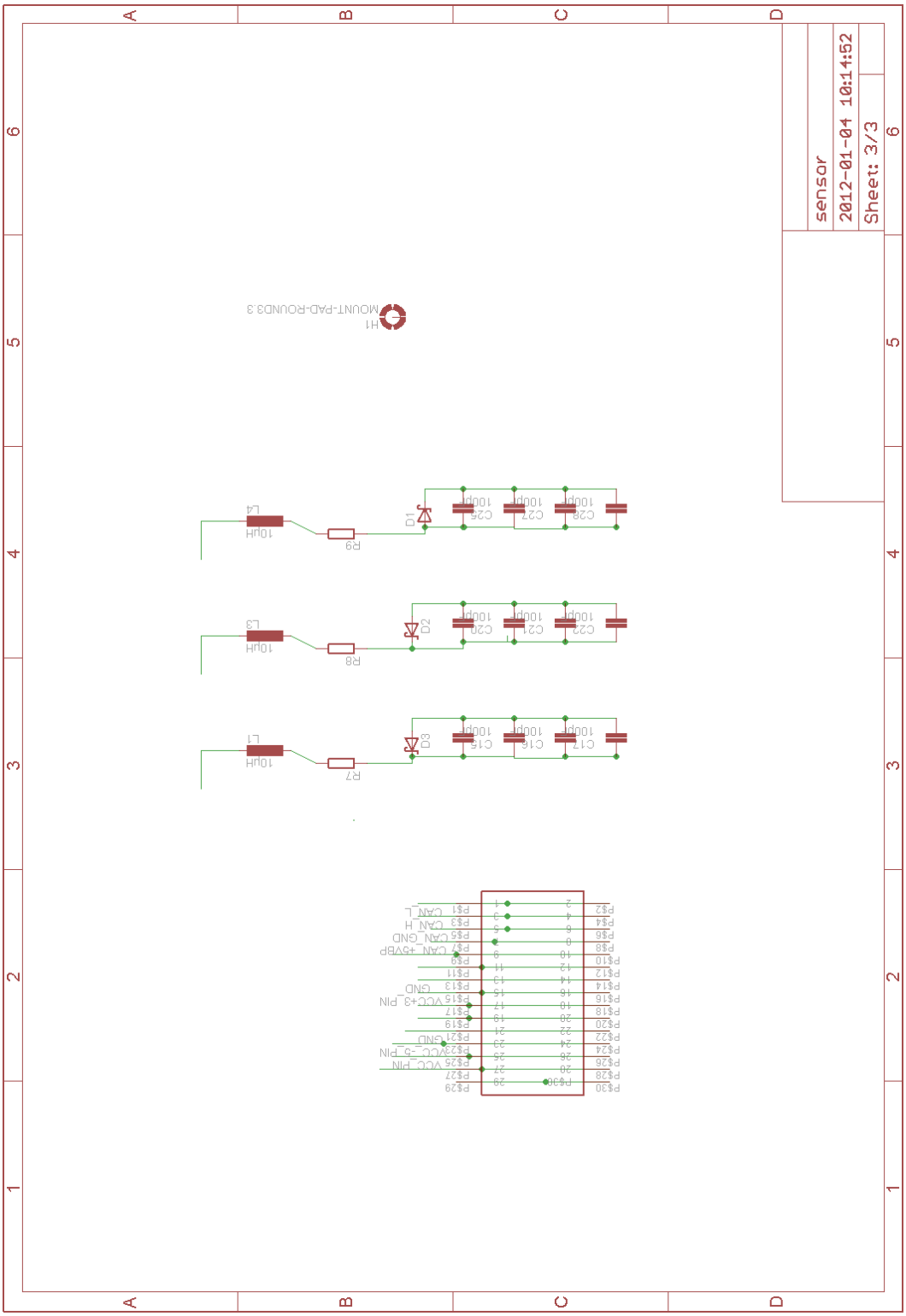

Figure 68: GPIO card: Power connector and filter.

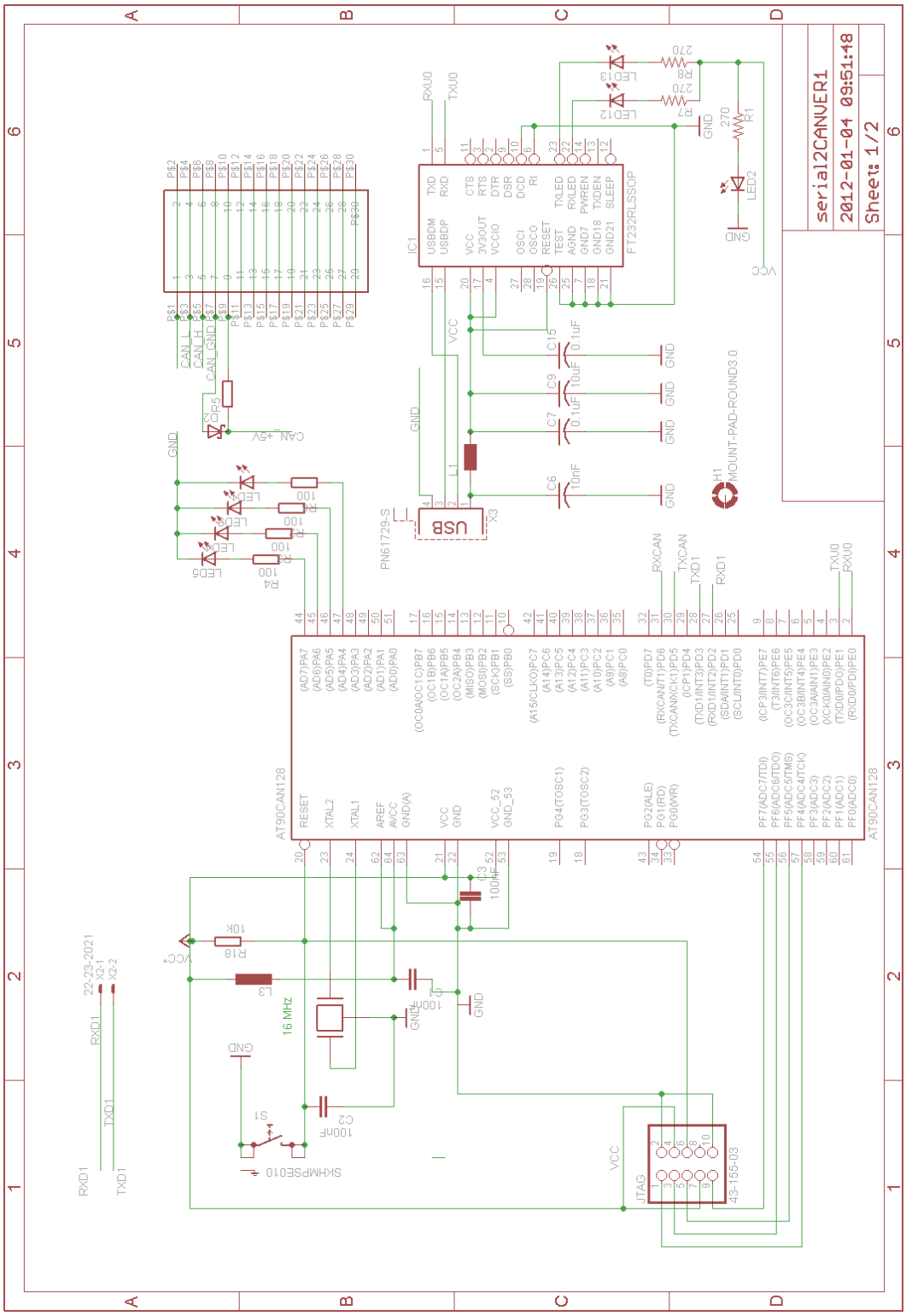

Figure 69: USB to CAN Router card: CAN not included.

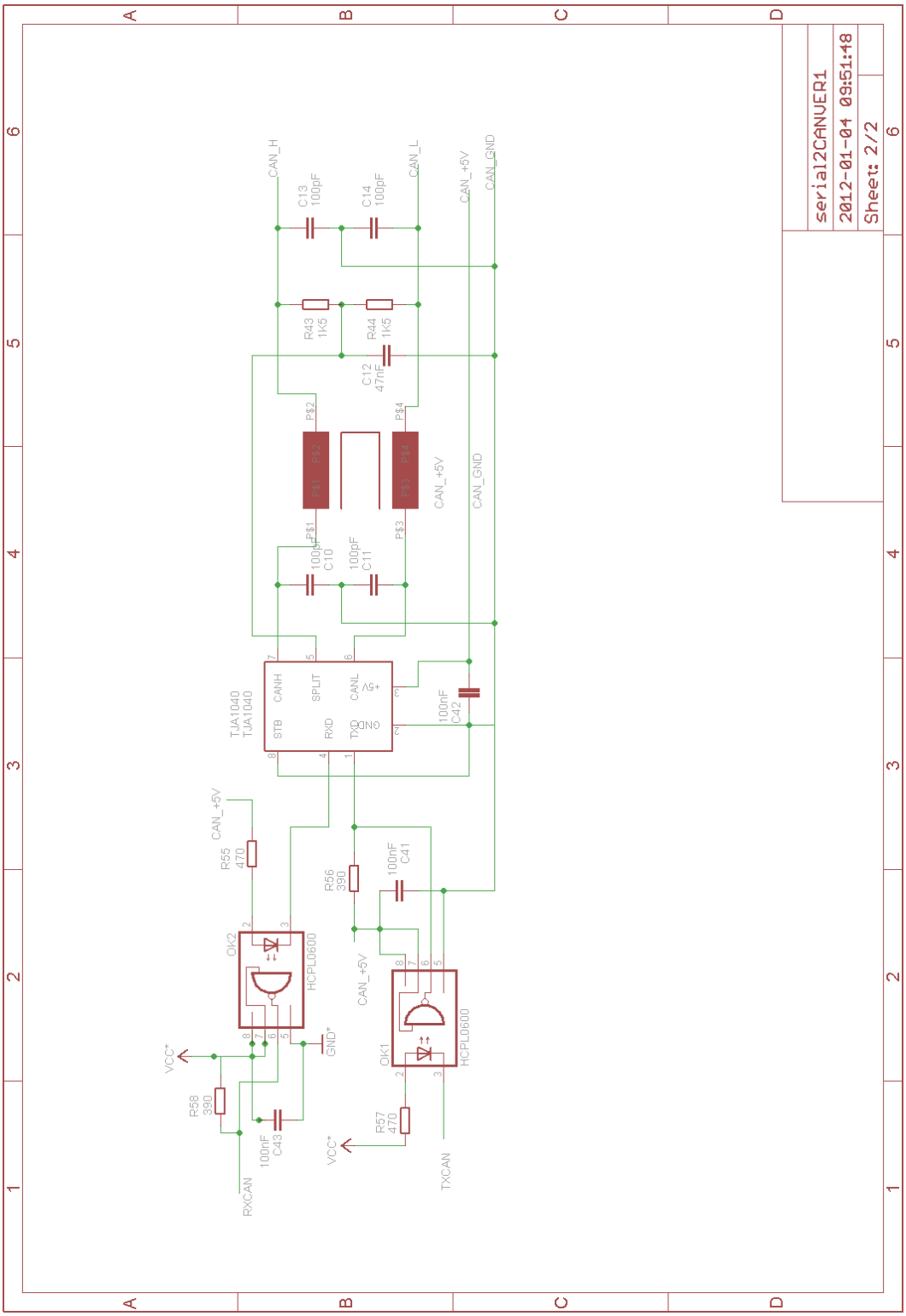

Figure 70: USB to CAN Router card: CAN.

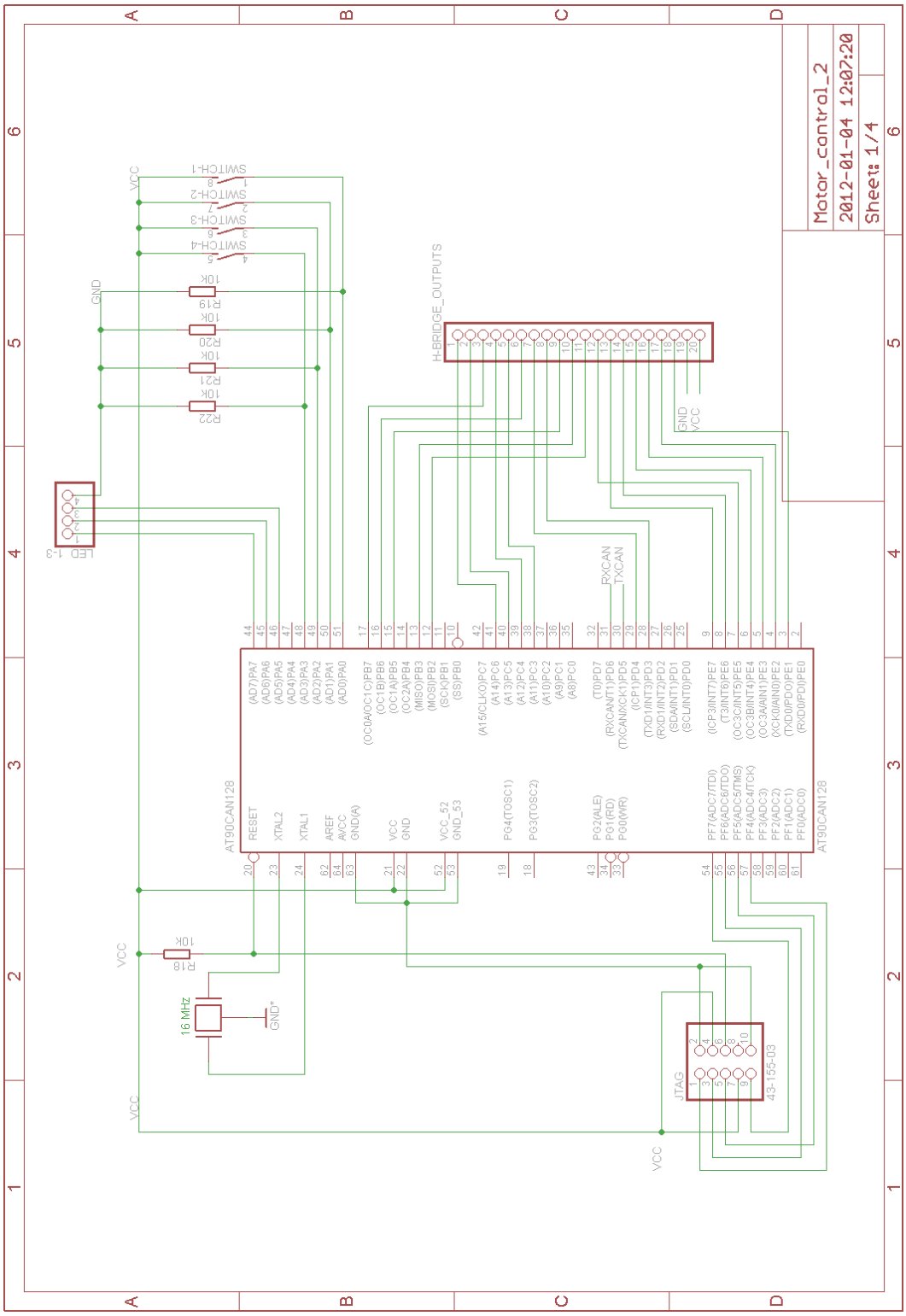

Figure 71: Motorcontroller: Microcontroller set up.

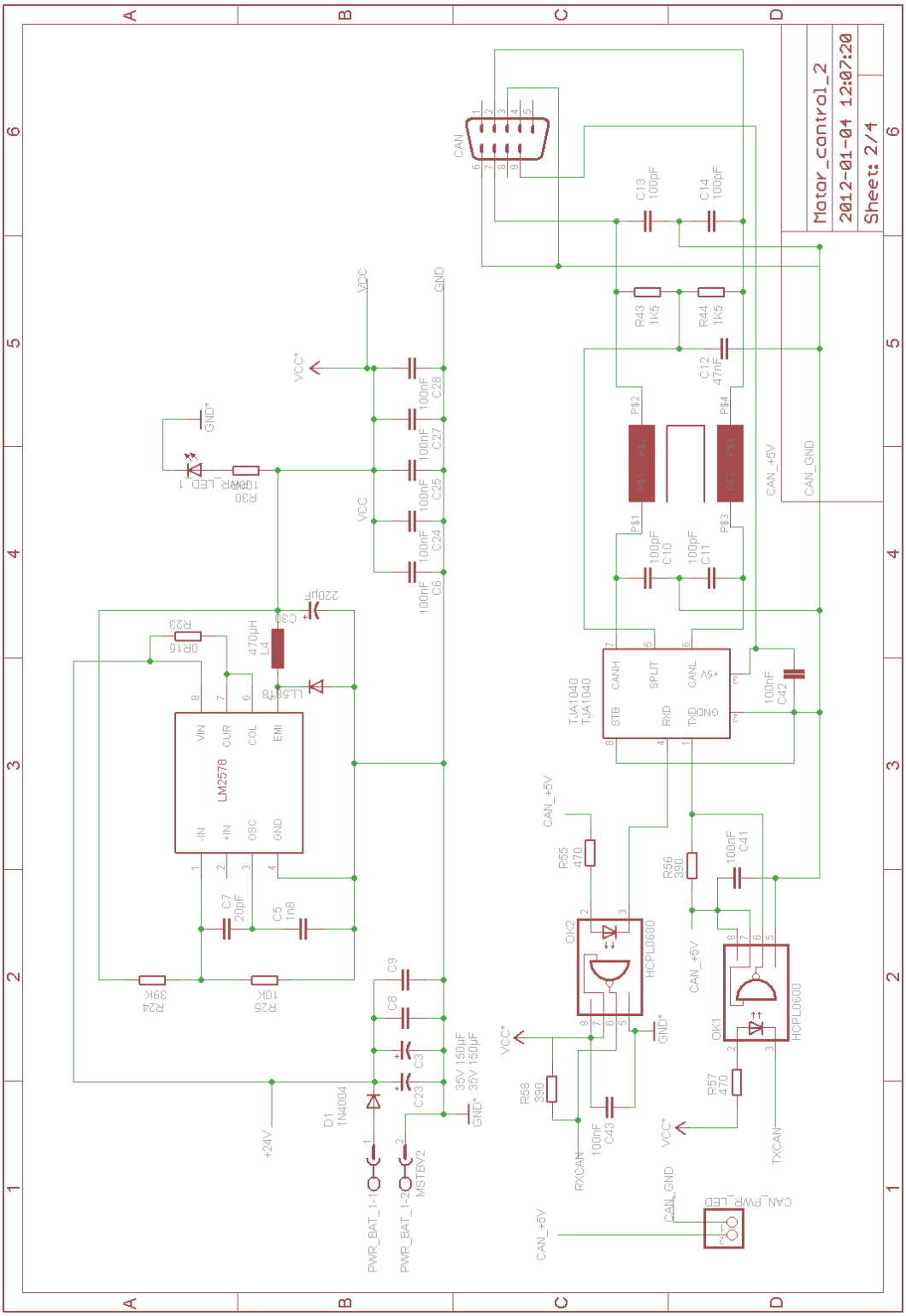

Figure 72: Motorcontroller: Voltage regulator logic and CAN.

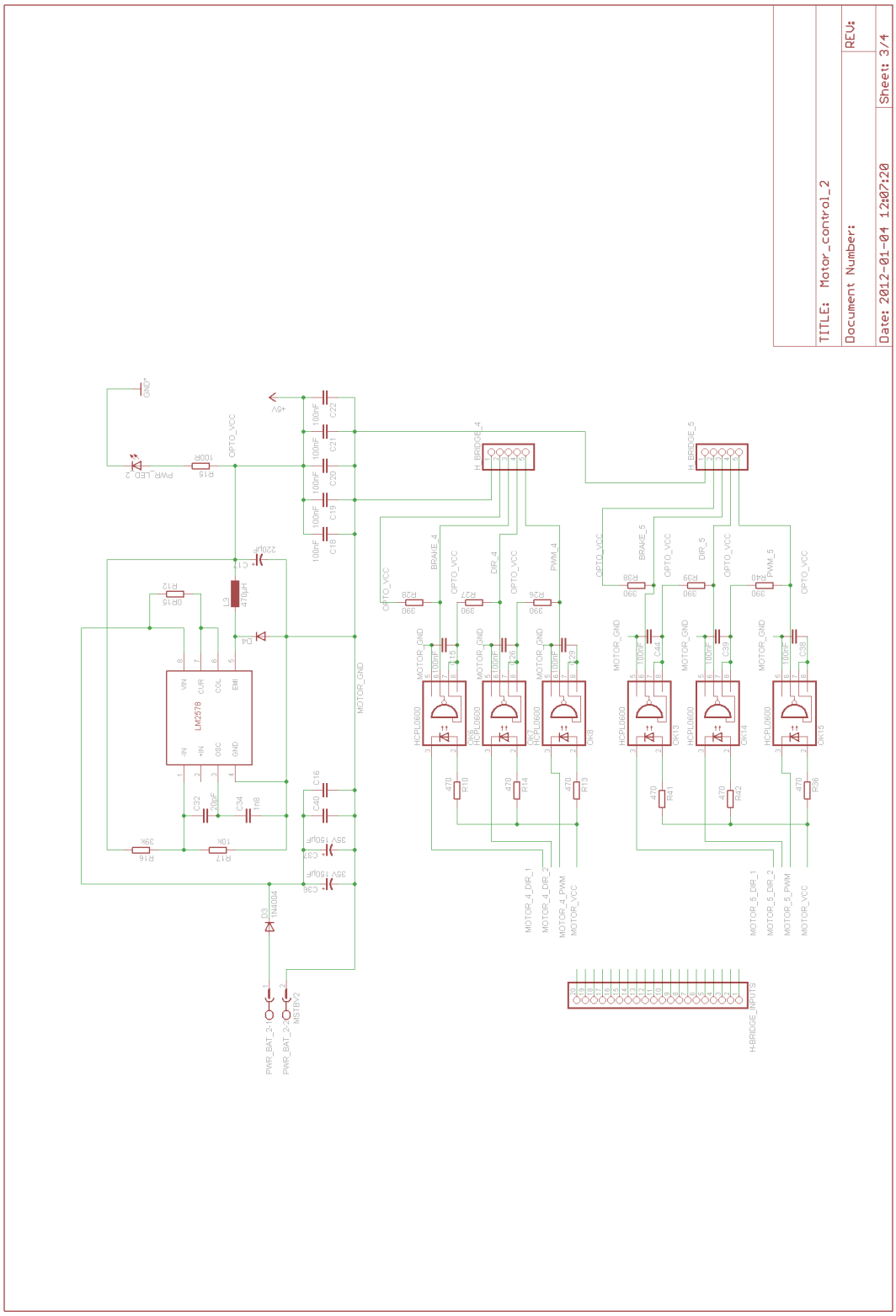

Figure 73: Motorcontroller: Voltage regulator for h-bridges and optisolators.

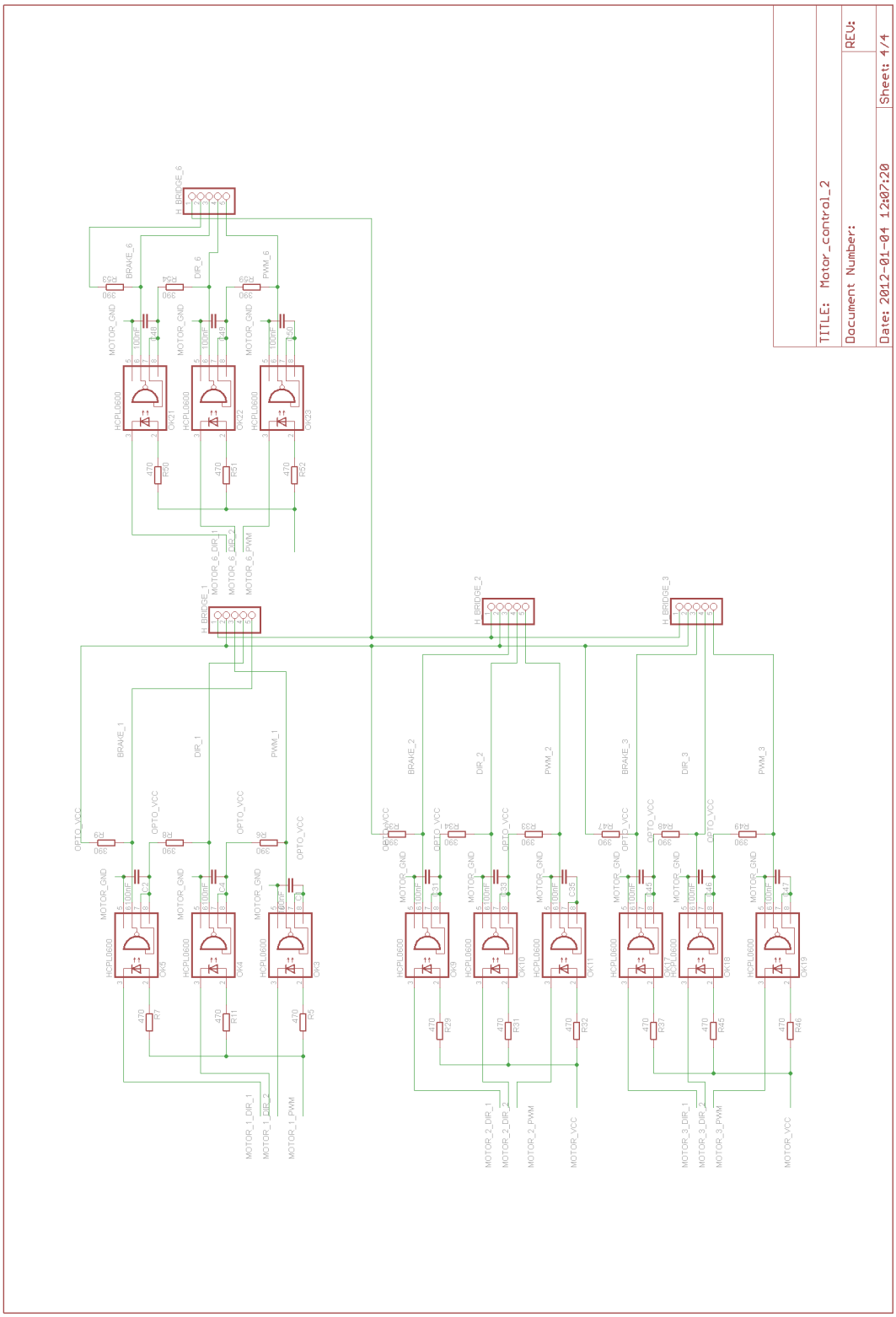

Figure 74: Motorcontroller: Optoisolators.

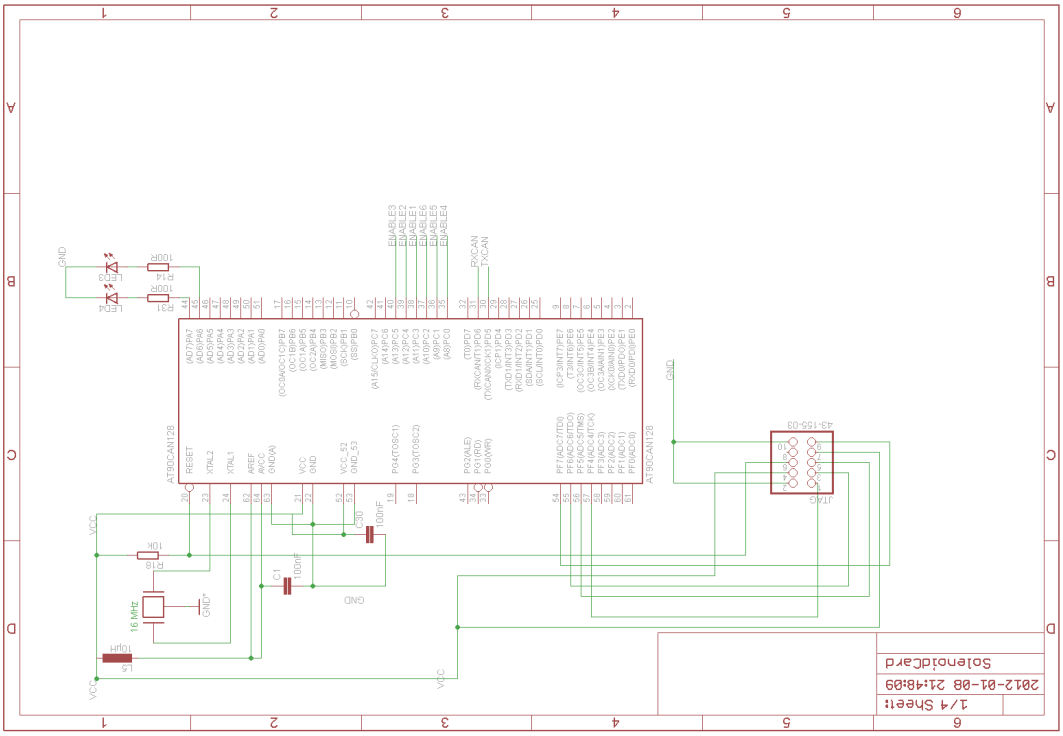

Figure 75: Solenoid card: Microcontroller set up.

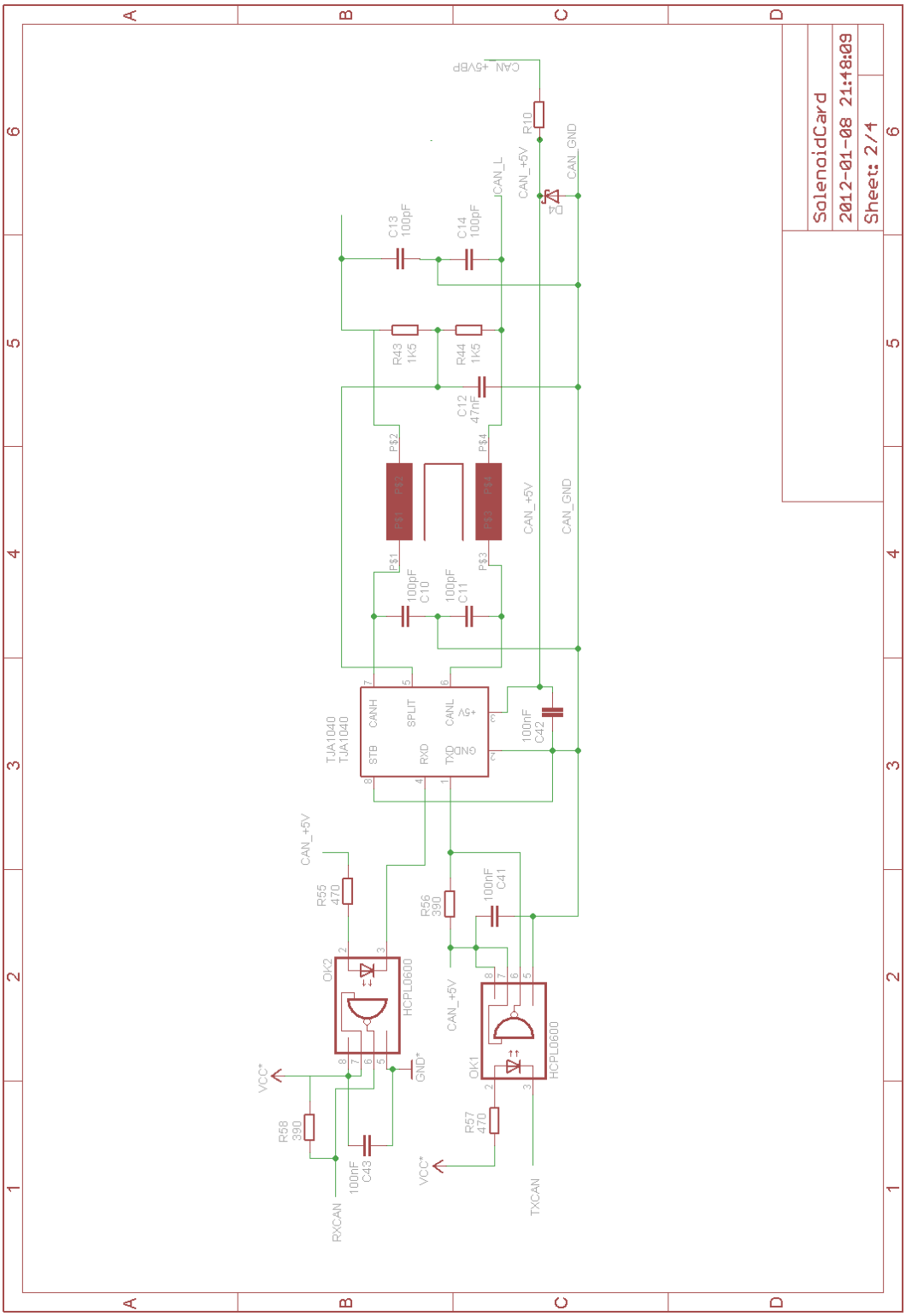

Figure 76: Solenoid card: CAN.

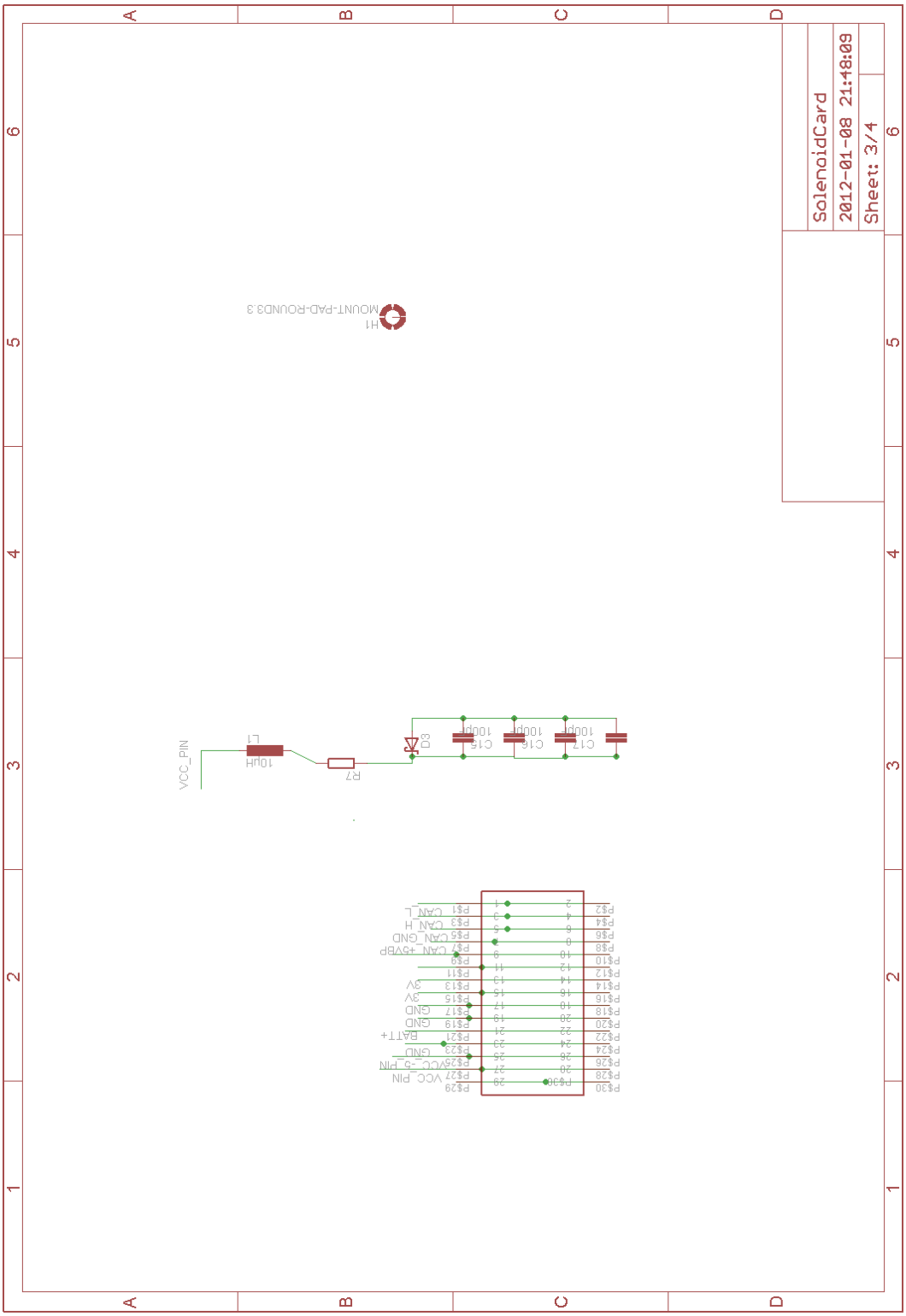

Figure 77: Solenoid card: Power connector and filter.

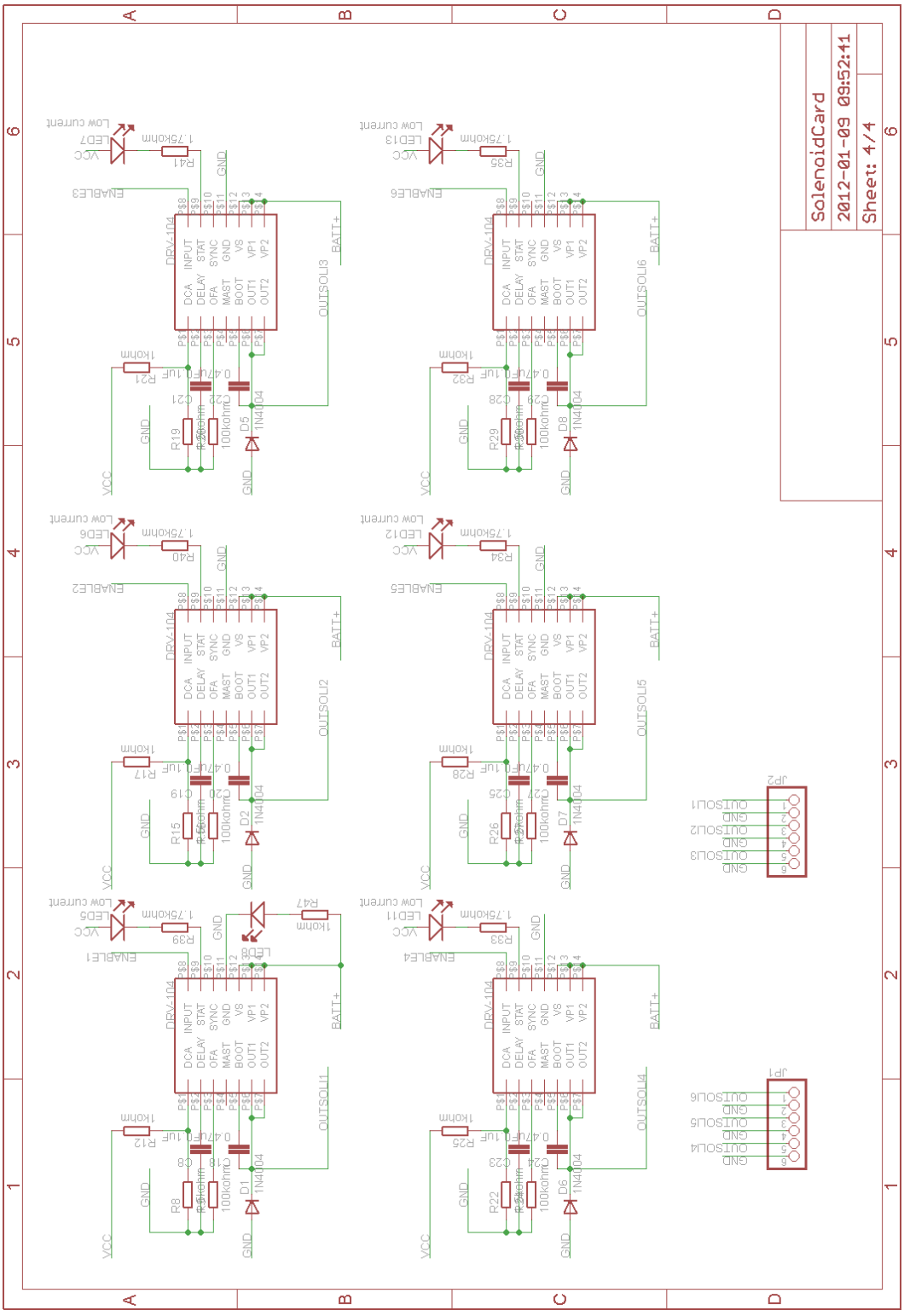

Figure 78: Solenoid card: Solenoid High frequency drivers.

D Electronics test protocols

D.1 Unit Power distribution

D.1.1 U-P-001

Purpose of The Test Case Document
Finding the electrical current used by the motors on the VASA robot. This is useful information in the construction of the power management system in the robots electrical construction. The document will be shared by supervisor and the development team.

Description
This is a informative test it has no parameters for passing or failing. The involved participants in the task are Johnny Holmström and Anton Widenius. The test will put one motor in water connected to a source of voltage and a device for reading current.

Resources
Johnny Holmström: Documentation, Electronics-group head Anton Widenius – Mechanical. Both of the participants will act as developers during the task case.

Preconditions
A motor of type Seabotix is positioned in water in a fixed location connected to a power source of variable voltage. This is a informative test it has no parameters for passing or failing.

Post Conditions
No changes, same as preconditions.

Flow of Events
The motor is held still in the water and the voltage was increased by one volt starting at 12volts and ending at 19 volts. For each voltage level the motor is still then moved forward and backwards. The still motor current is noted as Ampere Still and highest current level of the moving motor is noted as Ampere Move. The moving motor is only slightly effected. The maximum continuous current for the motors were specified to 4.2A witch would give us a maximum voltage of 17V to run continues on. The Vasa system of 6 motors would utilize 30 Ampere in a worst case scenario including margin for variation in current.

Voltage(V) Ampere Move(A) Ampere Still(A) 12 2,8 2,5 13 3,0 2,7 14 3,3 3,0 15 3,5 3,4 16 3,8 3,6 17 4,0 3,9 18 4,3 4,2 19 4,4 4,4

Inclusion/Exclusion Points
There will be no testing in moving water or with a continuous moving motor. This is a informative test it has no parameters for passing or failing.

Special Requirements
No special requirements

D.1.2 U-P-002

Purpose of The Test Case Document
Finding the electrical current used by the ITX on the VASA robot. This is useful information in the construction of the power management system in the robots electrical construction. The document will be shared by supervisor and the development team.

Description
This is a informative test it has no parameters for passing or failing. The involved participants in the task are

Johnny Holmström and Lars lagerholm.

Resources
Johnny Holmström: Documentation, Electronics-group head Lars Lagerholm – software Both of the participants will act as developers during the task case.

Preconditions
A ITX from the VASA is positioned on a bench connected to a power source of variable voltage. The ITX was set up using two USB cameras and two HIDś. Powering the ITX was a special PSU, changing this PSU changes the current usage, see appendix A. Heavy vision algorithms were run on both cameras, utilizing 90% CPU power. This is a informative test it has no parameters for passing or failing.

Post Conditions
No changes, same as preconditions. Flow of Events Current ranged depending on voltage given to the PSU. Current can be assumed to be approximation of 4 Ampere. This is with some margin.
Voltage(V) Current(A) 13 3.5 21 2.3

Inclusion/Exclusion Points
This is a informative test it has no parameters for passing or failing.

Special Requirements
No special requirements

D.1.3 U-P-003

Purpose of The Test Case Document
Showing the verification of the power distribution boards programmability.

Description
The involved participants in the task are Johnny Holmström and Lars lagerholm. The test will program the micro-controller to turn on a LED when the input voltage to the board reaches a certain value.

Resources
Johnny Holmström: Documentation, Electronics-group head Lars Lagerholm – software Both of the participants will act as developers during the task case.

Preconditions
The Power distribution board will be connected to a power source with variable voltage. The input voltage is set to 21V and a General purpose LED is on.

Post Conditions
Input voltage will be 12V.

Flow of Events
The micro-controller is programed to read the analog port from battery (deamplified by voltage divider). Then if the voltage drops below a certain voltage, in this case 16V the led will turn of. The voltage is turn down from 21V to 12V and the LED should go of at 16V.

Inclusion/Exclusion Points
There is no parameters for this category.

Special Requirements

No special requirements

D.1.4 U-P-004

Purpose of The Test Case Document
Documenting the relay functionality on the power distribution board.

Description
The involved participants in the task are Johnny Holmström and Lars lagerholm. The test will turn all relays on and of on the power distribution board when a signal by a magnet is given.

Resources
Johnny Holmström: Documentation, Electronics-group head Lars Lagerholm – software Both of the participants will act as developers during the task case.

Preconditions
The power distribution board Is set up at the minimum voltage of a 5cell battery, 18.5V. A HID is connected to the board in form of a Hall effect sensor to a analog port. The micro-controller is programmed with a test program. Relays are of at the start of the test.

Post Conditions
At the end of the test are the relays on.

Flow of Events
A magnet is put at the hall sensor and the relays are turned on. The voltage after the relays is measured compared to ground. Voltage should be close to 0V before relay turn on and close to 18.5V when the relays are on.

Inclusion/Exclusion Points
The magnet poles must be turned correctly compare to the side of hall sensor.

Special Requirements
No special requirements

D.1.5 U-P-005

Purpose of The Test Case Document
This will document the functionality of CAN Bus and LCD in power distribution board.

Description
The involved participants in the task are Johnny Holmström and Lars Lagerholm. The power distribution board will receive a message that is printed on LCD.

Resources
Johnny Holmström: Documentation, Electronics-group head Lars Lagerholm – software Both of the participants will act as developers during the task case.

Preconditions
The power distribution board will be set up to show the voltage of 21V(the voltage being feed to the system) on the LCD. The LCD is a GLCD5110 84x84 pixel display.

Post Conditions
The LCD will display the input voltage of 21V and the message “HELLO”.

Flow of Events
A message is sent over CAN bus to the power distribution board. The message is “HELLO” and is displayed on the LCD.

Inclusion/Exclusion Points
No special requirements.

Special Requirements
No special requirements.

D.2 Unit Router

D.2.1 U-R-001

Purpose of The Test Case Document
To verify the serial communication from the router board is important as PC communication. The document will be shared by supervisor and the development team.

Description
The serial part of the router card consists of FTD232 IC and is connected through a USB connector to communicate outside. The test will conducted by Mingli and Ejaz. And in the test the power supply to the part and the working of FTD232 will be checked. Ejaz Ul Haq and Mingli Wu would be involved in this test.

Resources
Mingli: To upload the software on the IC and check the return values and verifies the proper response of the card. Ejaz: Connection between the card and the PC.

Preconditions
The Serial part must be soldered well. The software must be compiled and running. The backbone is tested. And router board is connected.

Post Conditions
The FTD232 would be programmed.

Flow of Events
The software would be compiled. The cable would be connected to the FTD232 through USB connector. The LED is blinked and the power input is checked. The software is uploaded and checked if returns correct value. The reference voltage around JTAG is verified.

Inclusion/Exclusion Points
No inclusive points involved

Special Requirements
No special requirements.

D.2.2 U-R-002

Purpose of The Test Case Document
Testing if the microcontroller of router card is working properly. This information is required to as microcontroller drives the card. The document will be shared by supervisor and the development team.

Description
In this test case we will check the working of AT90CAN128 microcontroller if we can program it. And the reference voltage around JTAG is correct. Ejaz Ul Haq and Mingli Wu would be involved in this test.

Resources
1 Mingli Wu – Software. 2 Ejaz Ul Haq- Electronic Both act as developer for the test case.

Preconditions
JTAG must be attached to microcontroller. The software code must be compiled. The main backbone is developed and has all it's testing done and router card is attached.

Post Conditions
The microcontroller is programmed.

Flow of Events
Connect the router card to backbone card. We start with connecting the cable to the JTAG and checking around its reference voltage, the supplied voltage is 5V. Program it from PC. Check the LCDs.

Inclusion/Exclusion Points
No inclusive points involved.

Special Requirements
No special requirements.

D.2.3 U-R-003

Purpose of The Test Case Document
The testing CAN communication in router board is important as it is used to communicate signal inside the card. The document will be shared by supervisor and the development team.

Description
CAN is the source of communication to the board so is an important part in the router card and in the testing of the CAN part we test to assure whether we are able to send and receive the data on CAN successfully. CAN bus is isolated from the rest of the board and have its own power supply and signal lines so we will test it separately for its power and signal constraints.

Resources
Mingli Wu - Software Ejaz Ul Haq – documentation, Electronics Mingli Wu is responsible for the software uploading and Ejaz Ul Haq for physical connections.

Preconditions
CAN part must design, print in circuit board and soldered well. The main backbone is developed and has all it's testing done. The Router card is connected to the main board and have powered and connected to the main PC.

Post Conditions
No change, same as in preconditions.

Flow of Events
The router card is attached to the main board. The main board power have supplied by battery power. The main board is connected with CAN. The PC is attached via router card. The LCDs turn on. The message on CAN is bounced.

Inclusion/Exclusion Points
We will use all test cases from the router board, as all the routing of messages will be done using router board.

Special Requirements
No special requirement.

D.3 Unit Sensor

D.3.1 U-S-001

Purpose of The Test Case Document
Have information of the testing voltage input inside card. The document will be shared by supervisor and the development team.

Description
We required 5V, 3V,-5V and 5V CAN power input to our board and ground polygon inside the board so in the test we will check if all power input has the right voltage value. Ejaz Ul Haq and Ather Nadeem would be involved in this test.

Resources
Ather Nadeem- Electronic Ejaz Ul Haq- Electronic Both act as developer for the test case.

Preconditions
The main backbone is developed and has all it's testing done. The sensor card has connection.

Post Conditions
No change. Same as above.

Flow of Events
The power will be tested by inserting card to the shark board and using multimeter the values of voltage input to the card is verified. Supplied voltage actual voltage 5V -5V CAN 5V CAN Gnd 3V Gnd

Inclusion/Exclusion Points
No exclusive test involved.

Special Requirements
No special requirements.

D.3.2 U-S-002

Purpose of The Test Case Document
Testing if the microcontroller of sensor board is working properly. This information is required to as microcontroller drives the card. The document will be shared by supervisor and the development team.

Description
In this test case we will check the working of AT90CAN128 microcontroller if we can program it. And the reference voltage around JTAG is correct. Ejaz Ul Haq and Mingli Wu would be involved in this test.

Resources
1 Mingli Wu – Software. 2 Ejaz Ul Haq- Electronic Both act as developer for the test case.

Preconditions
JTAG must be attached to microcontroller. The software code must be compiled. The main backbone is developed and has all it's testing done and sensor card is attached.

Post Conditions
The microcontroller is programmed.

Flow of Events
Connect the sensor card to backbone card. We start with connecting the cable to the JTAG and checking around its reference voltage, the supplied voltage is 5V and the actual voltage is. Program it from PC. Check the LCDs.

Inclusion/Exclusion Points
No inclusive points involved.

Special Requirements
No special requirements.

D.3.3 U-S-003

Purpose of The Test Case Document
The testing CAN communication in sensor board is important as it is used to communicate signal inside the card. The document will be shared by supervisor and the development team.

Description
CAN is the source of communication to the board so is an important part in the sensor board and in the testing of the CAN part we test to assure whether we are able to send and receive the data on CAN successfully. CAN bus is isolated from the rest of the board and have its own power supply and signal lines so we will test it separately for its power and signal constraints.

Resources
Mingli Wu - Software Ejaz Ul Haq – documentation, Electronics Mingli Wu is responsible for the software uploading and Ejaz Ul Haq for physical connections.

Preconditions
CAN part must design, print in circuit board and soldered well. The main backbone is developed and has all it's testing done. The Router card is completed and all it's testing is done. The Router card is connected to the main board and have powered and connected to the main PC.

Post Conditions
No change, same as in preconditions.

Flow of Events
The sensor board and router board is attached to the main board. The main board power have supplied by battery power. The main board is connected with CAN. The PC is attached via router card. The LCDs turn on. The message on CAN is bounced.

Inclusion/Exclusion Points
We will use all test cases from the router board, as all the routing of messages will be done using router board.

Special Requirements
No special requirement.

D.3.4 U-S-004

Purpose of The Test Case Document
Communication between Sensor card, Router card and main board. This is important to send the information and data from sensors to the PC. The document will be shared by supervisor and the development team.

Description
The CAN communication from sensors to the PC through Sensor card and Router card will be tested. Ejaz Ul Haq, Ather Nadeem & Mingli would be involved in this test.

Resources
Mingli Wu - Software. Ather Nadeem - Electronics. Ejaz UL Haq - Electronics. All work as developers.

Preconditions
CAN part must design, print in circuit board and soldered well. The main backbone is developed and has all it's testing done. The Router card is completed and all it's testing is done. The Router card is connected to the main board and have powered and connected to the main PC. The sensors are in working condition.

Post Conditions
No changes. Same as preconditions.

Flow of Events
The main board will have all the power and data connection. The router card will connected to main board. Sensor card will connected to main board. PC will connect through Router card. IMU and Pressure sensor will attach to sensor card. PC sends command to microcontroller to read the IMU and pressure sensor value, and read it.

Inclusion/Exclusion Points
The CAN test and Microcontroller test will also conducted.

Special Requirements
No special requirement.

D.3.5 U-S-005

Purpose of The Test Case Document
The changes in software are important to test and verify if working. The document will be shared by supervisor and the development team.

Description
The software group did some improvements in the software of sensor card and since the new sensor card is not ready, the testing is done at old sensor card and checking if all the functions of microcontroller works.

Resources
1 Mingli Wu - Software 2 Ejaz Ul Haq – documentation, Electronics 3 Mingli Wu is responsible for the software uploading and Ejaz Ul Haq for physical connections.

Preconditions
Router card is working and we have a suitable alternate of backbone and the sensor card is working.

Post Conditions
New software is worked in microcontroller.

Flow of Events
The dragon is attached to the JTAG of sensor card. The dragon is connected to PC. The microcontroller is programed. The router card is connected to the CAN connector. The dragon is disconnected. The router card is connected to PC and the sensor card is attached to CAN. The IMU is connected to sensor card and the input signal to the sensor card is checked at PC.

Inclusion/Exclusion Points
U-S-003, U-S-004.

Special Requirements
No special requirement

D.4 Unit Shark Board

D.4.1 U-M-001

Purpose of The Test Case Document
The purpose of the test case is to determine the no-load voltage of the voltage regulators, the full-load voltage of the voltage regulators. Measure the output voltage of the voltage regulator when there are no electronic devices attached. Conclude this is the no-load voltage. Connect all the boards that the voltage regulator must power to the voltage regulators. Turn on the boards. Now measure the voltage regulators. Conclude that this is the full-load voltage. Use for example a full-load voltage of 11 volts.

Resources
We have to test the currents across the boards (Router, Sensor etc.) when they connected to the backbone. Test board by board so we could see if there will be Fault or problem with the high current, short circuit or open circuit. Rafat A.Ghanim: Electronics-group Johnny Holmström: Documentation, Electronics-group head

Preconditions
We must fulfill the current limitation required for the boards (Router, Sensor etc.)

Post Conditions
We should verify that all input currents are OK on the voltage regulators.

Flow of Events
1. Measure the output voltage of the voltage regulator when there are no boards attached. Conclude this is the no-load voltage. Use for example a no-load voltage of 12 volts. 2. Connect all the boards that the voltage regulators must power them to the voltage regulators through the connectors when there is no power in the shark board. Turn on the shark board. Now measure each voltage regulator's voltage. Conclude that this is the full-load voltage. Use for example a full-load voltage of 11 volts. 3. Calculate the change in voltage. Subtract the no-load voltage obtained in Step 1 from the full-load voltage in Step 2. Conclude for example that the change in voltage is 1 volt, since 12 minus 1 is 11. 4. Calculate the load regulation. Divide the change in voltage obtained in the previous step by the full-load voltage. For this example, calculate that the load regulation is 0.091 volts per volt, since 1 divided 11 is 0.091. 5. Calculate the percent load regulation. Multiply the load regulation in Step 4 by 100 percent. Conclude that the percent load regulation is 9.1 percent, since 100 multiplied by 0.091 is 9.1 6. Load regulation (percent) = 100 x (voltage no load --- voltage full load) divided by voltage full load.

Inclusion/Exclusion Points
We should check the currents to boards from the voltage regulators. We should check the ripple level of the output voltages.

Special Requirements
Voltage regulators are devices that are used to keep the voltage constant under varying load conditions. Most manufacturers of voltage regulators provide a load regulation specification. That load regulation specification is calculated for a specified full load. The manufacturer might specify that a 12-voltage regulator has a 0.1 percent load regulation for a load current from 0 milliamperes to 300 milliamperes. A load regulation specification of 0.1 percent means the manufacturer guarantees that the output voltage of the regulator will not drop below 11.98 volts as long as the current required from the regulator is less than 300 milliamperes. The full-load voltage of 11.98 volts is calculated by substituting 0.001 (0.1 percent) and 12 volts into the calculation and then solving for the full-load voltage.

D.4.2 U-M-002

Purpose of The Test Case Document
The purpose of the test case is to verify the Voltages across the connectors, input and output voltages.

Description
We should test the input voltages from the supply (battery) across the input pins of the voltage regulators.
Resources
We have to test the boards (Router, Sensor etc.) when they connected to the shark board, power and communication. Test board by board so we could see if there will be Fault or problem with the high current, short circuit or open circuit. Rafat A.Ghanim: Electronics-group Johnny Holmström: Documentation, Electronics-group head

Preconditions
We must fulfill the voltages routes required for the boards (Router, Sensor etc.) Fault regions, are regions in which failures could happen independently (i.e. without influencing other fault regions).

Post Conditions
We should verify that all input voltages are OK on the voltage connectors (no loss in the soldering). If this failed

then we should check the voltage level or the connectors.

Flow of Events
We first supply the Shark board with power using DC power supply. Start measuring the input voltages using Multi-meter, measuring the output voltages. We increase the voltage level from 5V to 20V to see that the output voltages are working properly and stable.

Inclusion/Exclusion Points
We should check the voltages to boards on the voltage regulators.

Special Requirements
No special Requirements

D.4.3 U-M-003

Purpose of The Test Case Document
The purpose of the test case is to verify the Voltages across the Voltage regulators, input and output voltages.

Description
We should test the input voltages from the connectors across the input pins of the voltage regulators, test the voltages across the output pins of the voltage regulators. Note: All tests should be with no load and with full load.

Resources
We have to test the voltages across the cards (Router, Sensor etc.) when they connected to the backbone. Test card by card so we could see if there will be Fault or problem with the high current, short circuit or open circuit. Rafat A.Ghanim: Electronics-group Johnny Holmström: Documentation, Electronics-group head

Preconditions
We must fulfill the voltages routes required for the cards (Router, Sensor etc.) Fault regions, are regions in which failures could happen independently (i.e. without influencing other fault regions).

Post Conditions
We should verify that all input voltages are OK on the voltage regulators (no loss in the soldering).The voltage regulators (5V,3.3V) have the heat sink underneath, so we should isolate them.

Flow of Events
We first Turn the Shark board on in which the voltage regulators operate, supply with power using DC power supply. Start measuring the input voltages using Multi-meter, measuring the output voltages. The voltage shown should be within a few percent of the device's rated voltage, If it reads zero or some value other than the regulator's rated voltage, the regulator is bad. We increase the voltage level from 5V to 20V to see that the output voltages are working properly and stable. We measure the voltages across the output pins of the voltage regulators.

Inclusion/Exclusion Points
We should check the voltages to boards on the voltage regulators. We should check the ripple level of the output voltages. We should check the heating of the voltage regulators.

Special Requirements
No Special Requirements

D.4.4 U-M-004

Purpose of The Test Case Document
The purpose of the test case is to verify the heating across the voltage regulators.

Description
We should check the heating radiating from the voltage regulators. Note: All tests should be with no load and with full load.

Resources
We have to test the cards (Router, Sensor etc.) when they connected to the backbone, if they radiate some heating. Test card by card so we could see if there will be Fault or problem with the high current, short circuit or open circuit. Rafat A.Ghanim: Electronics-group Johnny Holmström: Documentation, Electronics-group head

Preconditions
We must fulfill the heating level required within the range.

Post Conditions
We should verify that all components are OK and loaded so they radiating the maximum level of heating.

Flow of Events
We first supply the Shark board with power using DC power supply. Start measuring the heating using Thermometer or we could sense the heat physically by using finger over the voltage regulators. We increase the voltage level from 5V to 20V to see that the output voltages are working properly and stable. We check the heating on the voltage regulators every 5 minutes for 2 hours, we use finger sense.

Inclusion/Exclusion Points
We should check the heating radiates from other boards and the environment enclosed to the voltage regulators. We check the environment for heat dissipation.

Special Requirements
No special Requirements

D.4.5 U-M-005

Purpose of The Test Case Document
The purpose of the test case is to verify the currents across at the output of the Voltage regulators.

Description
We should test the output currents from the connectors across the output pins of the voltage regulators. Note: All tests should be with no load and with full load.

Resources
We have to test the currents across the boards (Router, Sensor etc.) when they connected to the backbone. Test board by board so we could see if there will be Fault or problem with the high current, short circuit or open circuit. Rafat A.Ghanim: Electronics-group Johnny Holmström: Documentation, Electronics-group head

Preconditions
We must fulfill the current limitation required for the boards (Router, Sensor etc.)

Post Conditions
We should verify that all input currents are OK on the voltage regulators.

Flow of Events
We first supply the Shark board with power using DC power supply. Initialize the bus voltages. Calculate the each bus current. Calculate the each branch current and node voltages starting from far end moving toward the Volt. Start measuring the input voltages using Multi-meter, measuring the output voltages. We increase the voltage level from 5V to 20V to see that the output voltages are working properly and stable. We measure the currents across the output pins of the voltage regulators.

Inclusion/Exclusion Points
We should check the currents to boards from the voltage regulators. We should check the ripple level of the output voltages.

Special Requirements
No special Requirements

D.4.6 U-M-006

Purpose of The Test Case Document
The purpose of the test case is to verify the voltage stability with minimum level of ripple across the outputs of the Voltage regulators.

Description
We should test the output voltage and current ripples from the output pins of the voltage regulators. How could we get minimum curtail load. Note: All tests should be with no load and with full load.

Resources
We have to test the voltage and current ripples across the boards (Router, Sensor etc.) when they connected to the backbone. Test board by board so we could see if there will be Fault or problem with the high current, short circuit or open circuit. Rafat A.Ghanim: Electronics-group Johnny Holmström: Documentation, Electronics-group head

Preconditions
We must fulfill the minimum limitation of ripple effects (Router, Sensor etc.) Fault regions, are regions in which failures could happen independently (i.e. without influencing other fault regions).

Post Conditions
We should verify that all voltage and current ripples are OK on the boards.

Flow of Events
We first supply the Shark board with power using DC power supply. Initialize the bus voltages. Start measuring the input voltages using Multi-meter, measuring the output voltages. We increase the voltage level from 5V to 20V to see that the output voltages are working properly and stable. We measure the voltage and current ripples across the output pins of the voltage regulators, using oscilloscope.

Inclusion/Exclusion Points

We should check the currents to boards from the voltage regulators. We should check the ripple level of the output voltages.

Special Requirements

D.4.7 U-M-007

Purpose of The Test Case Document
The purpose of the test case is to verify the interference in the circuits of the Voltage regulators.

Description
This test is an inadequate because of the invalidity of the approximation of the exponential integral function for the drop voltage of the voltage regulators at large times. The accuracy requirement is based on the voltage and current sensing apparatus.

Resources
Rafat A.Ghanim: Electronics-group Johnny Holmström: Documentation, Electronics-group head

Preconditions
We must fulfill the current limitation required for the boards (Router, Sensor etc.) Fault regions, are regions in which failures could happen independently (i.e. without influencing other fault regions).

Post Conditions
No post condition

Flow of Events
We should turn on all the cards one by one.

Inclusion/Exclusion Points
We should check the currents to boards from the voltage regulators. We should check the ripple level of the output voltages.

Special Requirements
No special Requirements

D.5 Unit SONAR

D.5.1 U-S1-001

Purpose of The Test Case Document
Documentation of voltage levels at each connector in the SONAR card and pinger.

Description
The participant is Ejaz Ul Haq. The test is necessary to verify if other components would run.

Resources
Ejaz Ul Haq- Electronics Group. Would work as a developer.

Preconditions
No Preconditions.

Post Conditions
No post conditions.

Flow of Events
The multimeter is used to find if there is any short circuit in the board. The power supply would be set to 18V battery connects to the sensor card via power board. The sonar card would attach to sensor card and voltage values would be checked at each pin of connector. The required voltage levels are 5V and -5V. The voltage of 9V would apply to pinger and reference voltage would be measured.

Inclusion/Exclusion Points
No inclusive/exclusive point.

Special Requirements
No special requirements.

D.5.2 U-S1-002

Purpose of The Test Case Document
Documentation of the testing of passive SONAR card.

Description
The Test is done by Ejaz Ul Haq, Ather Nadeem and Mingli Wu. This test would be of the input values of SONAR.

Resources
Ejaz Ul Haq – Electronics group. Ather Nadeem- Electronics group. Mingli Wu- Software group. All works as developer.

Preconditions
Software is compiled.

Post Conditions
No post conditioned.

Flow of Events
The PC is connected to Sonar card via router and sensor card. The sonar card is attached to sensor card and pinger would get power on. The four receivers would be attached as water proof in an array of distance 10cm each, and set into water. The pinger is power on and the transmitter would put into water at an appropriate distance to receiver. The input values to the CAN from receiver would be check at minicom.

Inclusion/Exclusion Points
U-R-001, U-R-002, U-R-003, U-S-001, U-S-002, U-S-003, U-S-004, U-S-005.

Special Requirements
No special requirement.

D.5.3 U-S1-003

Purpose of The Test Case Document
Documentation to the testing of passive sonar card in pool.

Description
The Test is done by Ejaz Ul Haq and Mingli Wu and Ather Nadeem. The range of SONAR receivers would be measured.

Resources
Ejaz Ul Haq – Electronics group. Ather Nadeem – Electronics group. Mingli Wu- Software group. All would work as developer.

Preconditions
Software is compiled.

Post Conditions
The distance through passive SONAR would be measured.

Flow of Events
1 The PC is connected to Sonar card via router and sensor card. The sonar card gets power from sensor card and pinger would get power on. The four receivers would be attached as water proof in an array of distance 10cm each, and set into water at the pool. The pinger is power on and the transmitter would put into water in the pool at an appropriate fix point. The receivers would bring from far distance and the values would start measuring. The range of SONAR would be measured as soon as the values get in and the receiver would put all directions and with some height and angles to estimate the exact range of receiver by measuring the distances. The input values to the CAN from receiver and variation by distance would be check at minicom.

Inclusion/Exclusion Points
U-R-001, U-R-002, U-R-003, U-S-001, U-S-002, U-S-003, U-S-004, U-S-005. U-S1-001, U-S1-002

Special Requirements
No special requirement.

E Mechanics exploded views

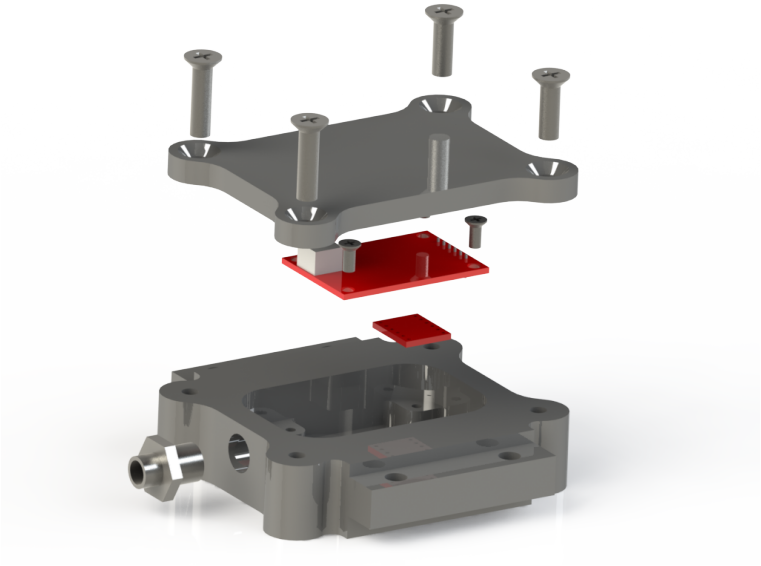

Figure 79: IMU box assembly.

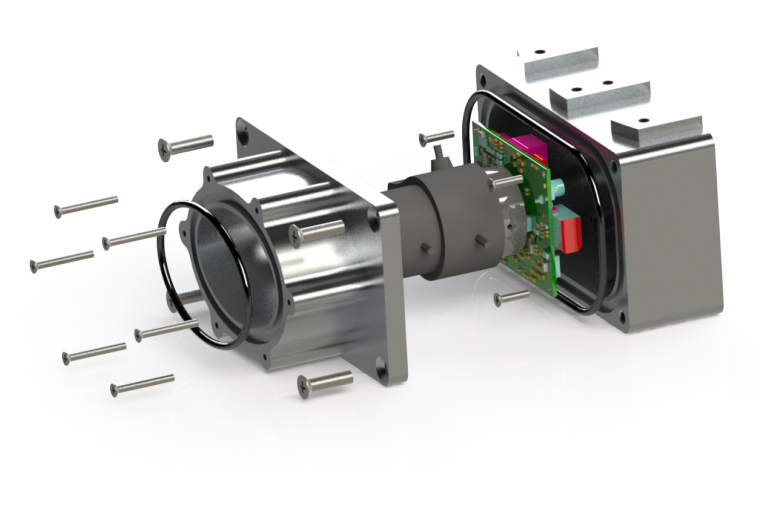

Figure 80: Camera assembly.

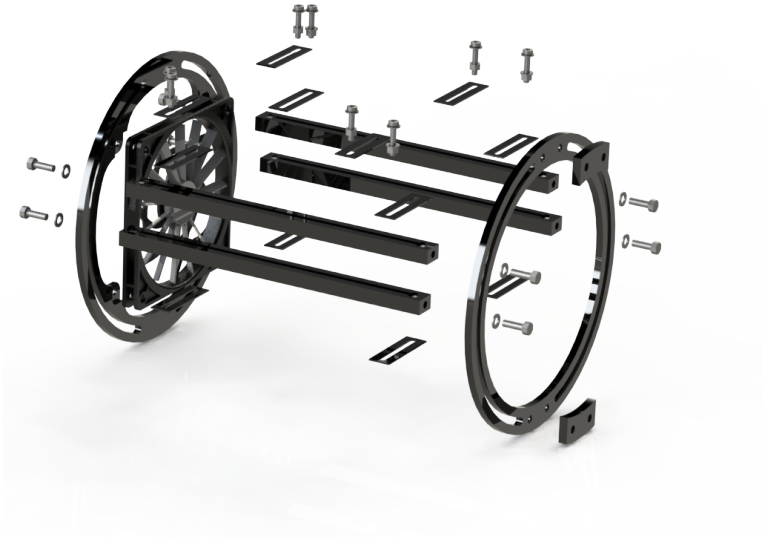

Figure 81: Electronics support system.

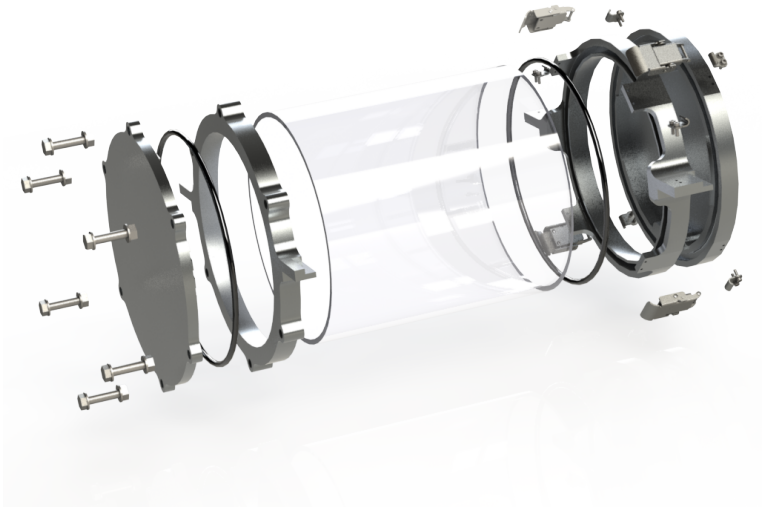

Figure 82: Tube assembly.

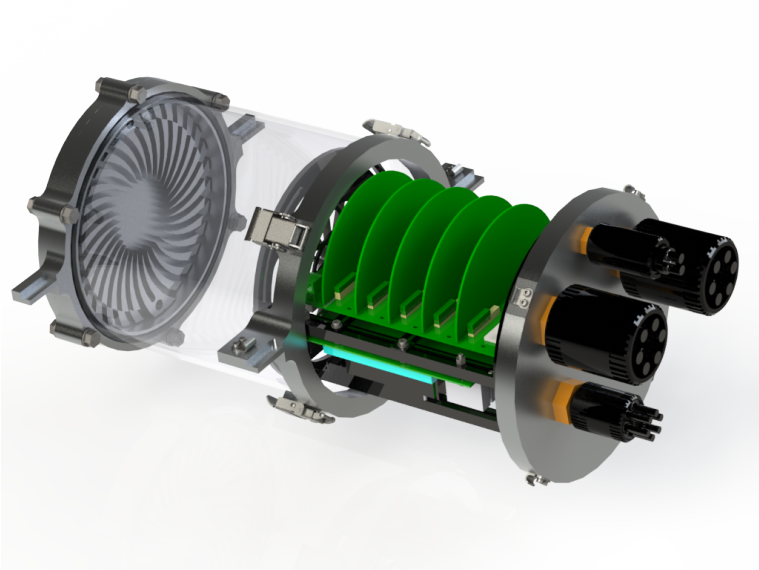

Figure 83: Tube assembly with electronics.

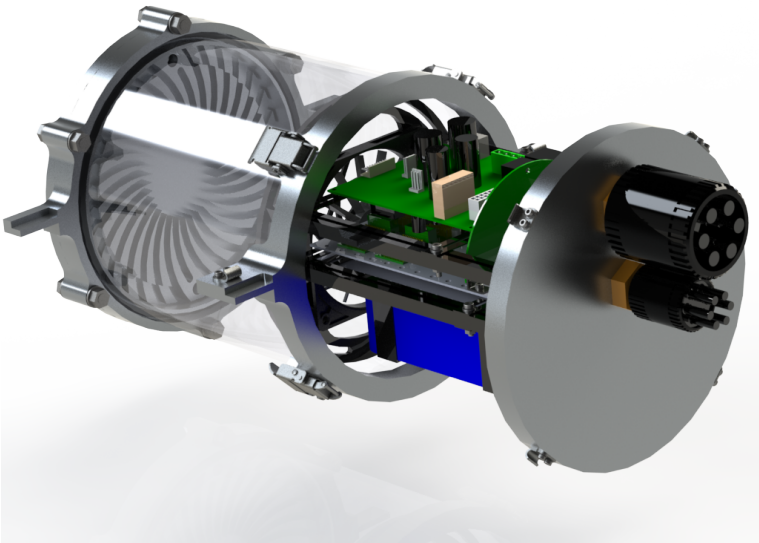

Figure 84: Tube assembly with electronics.

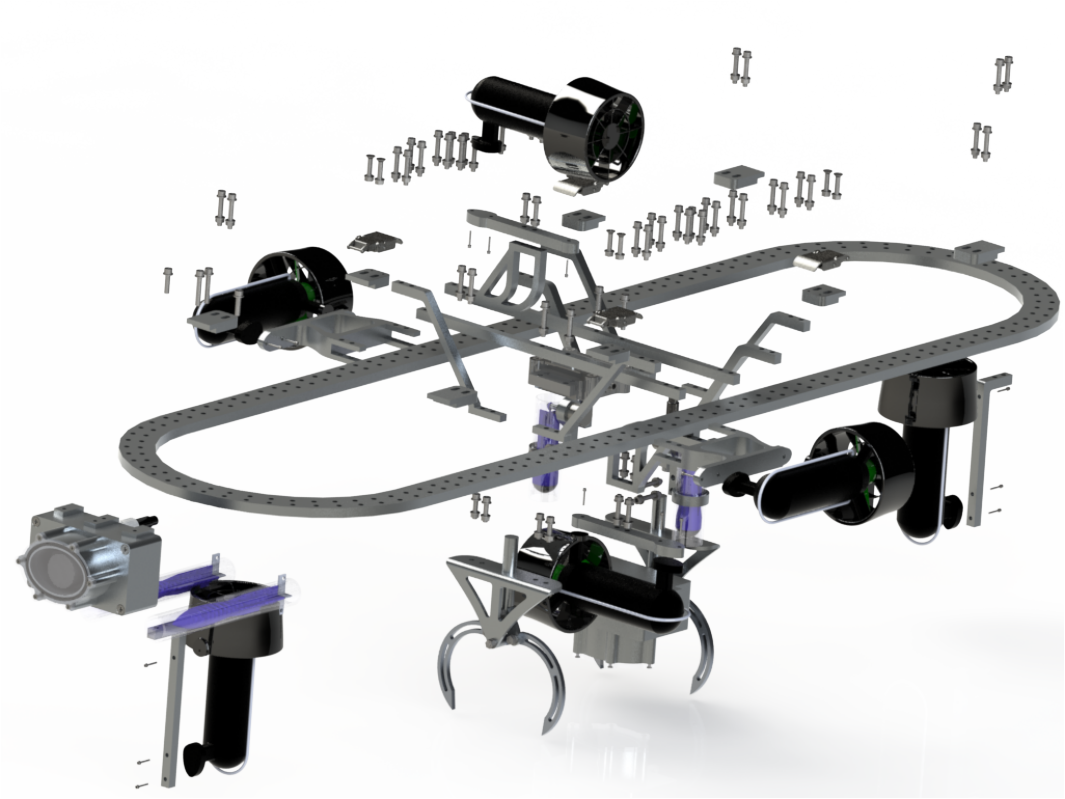

Figure 85: Frame assembly.

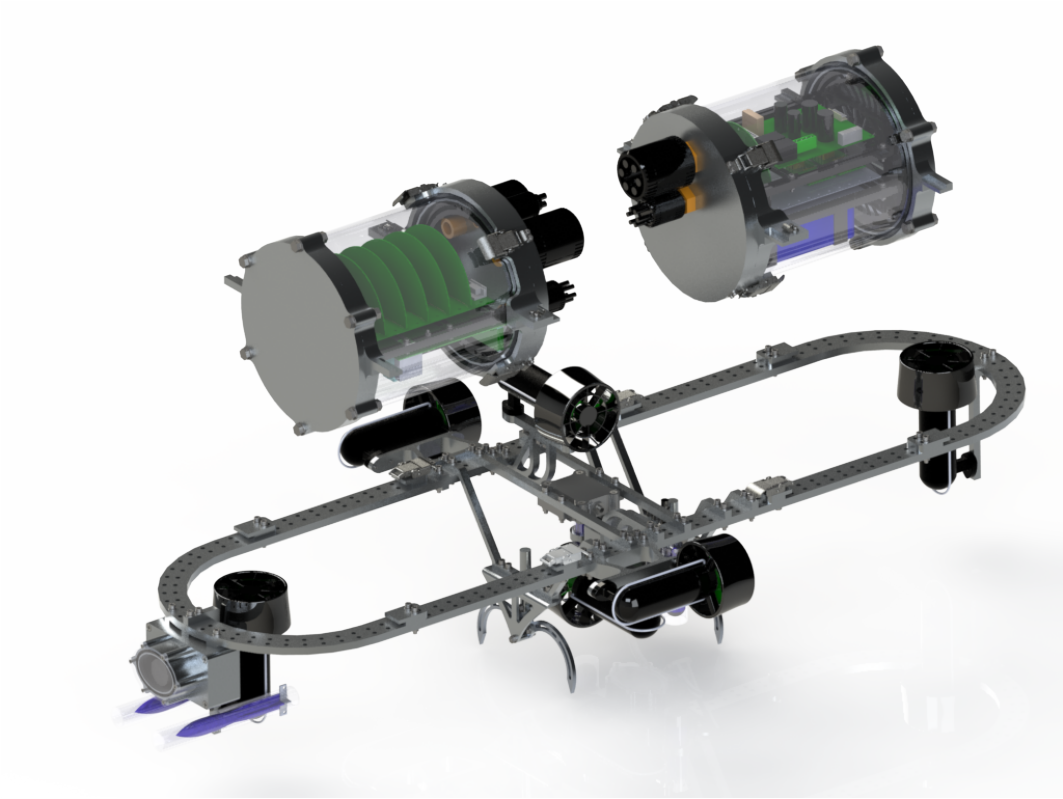

Figure 86: Frame with tubes assembly.

F Thruster configuration

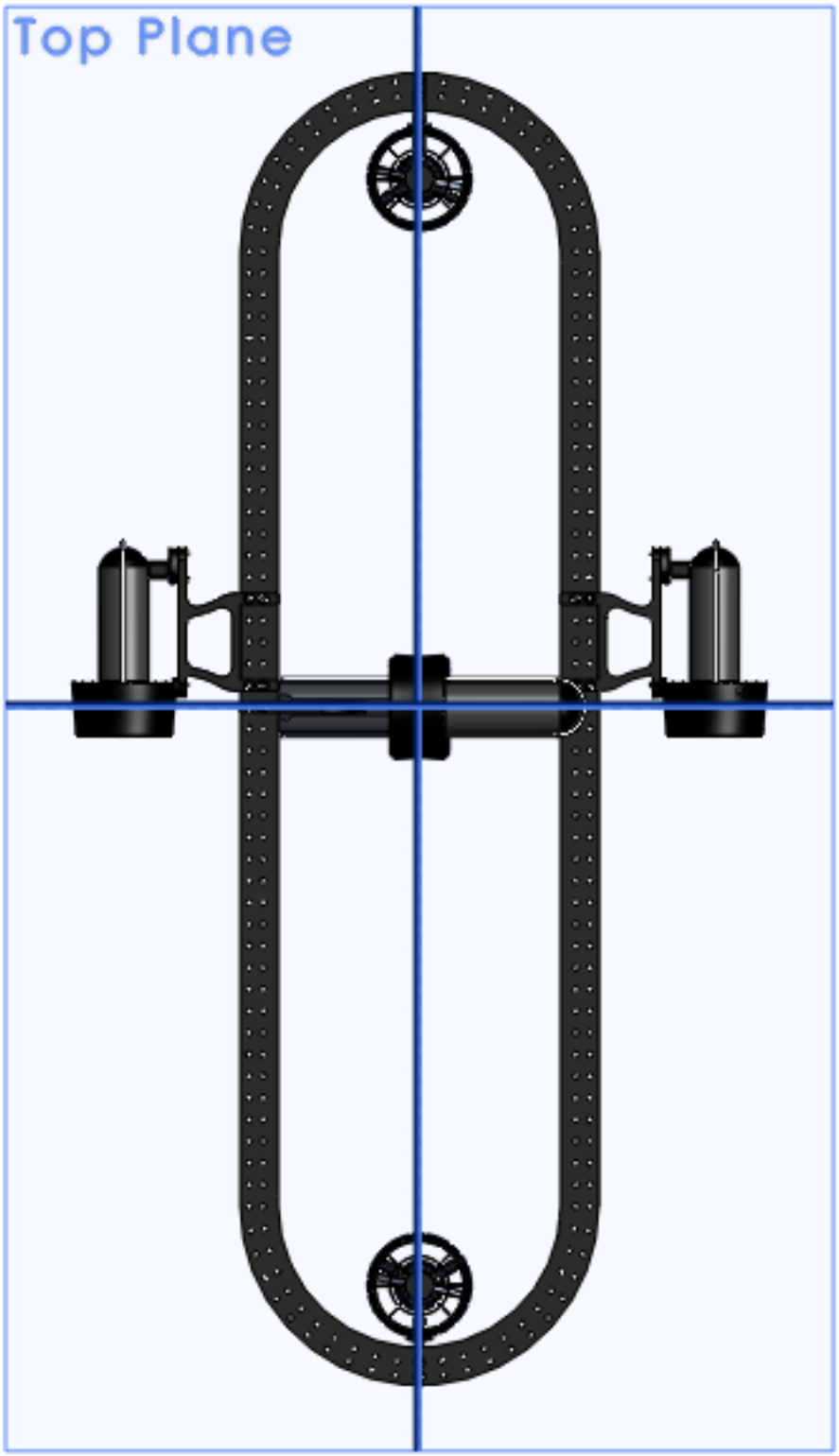

Figure 87: Thruster configuration top view.

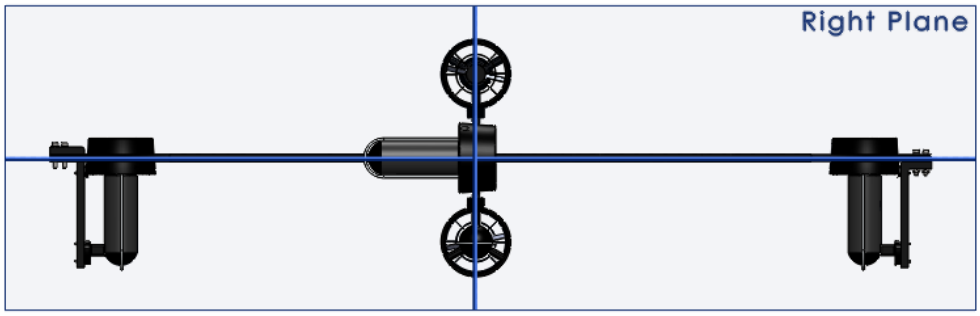

Figure 88: Thruster configuration left view.

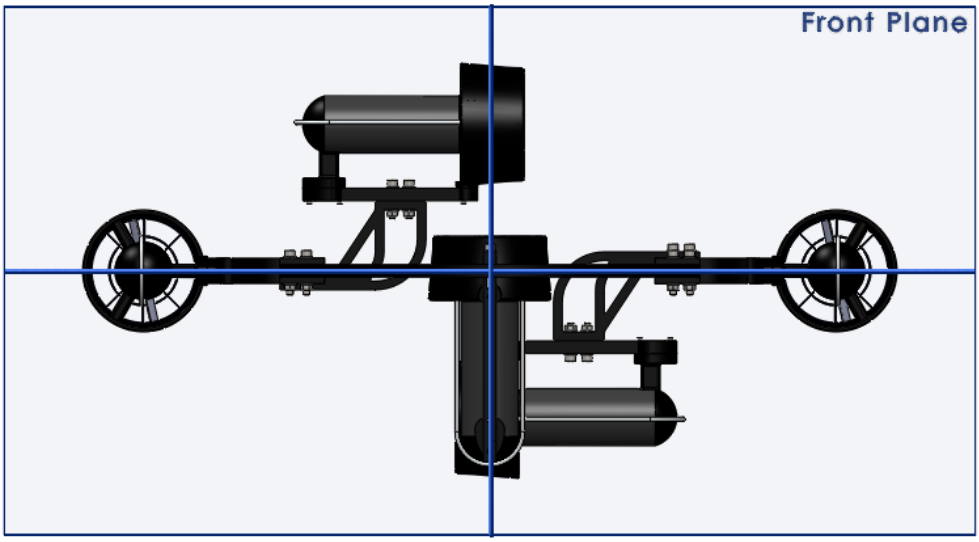

Figure 89: Thruster configuration front view.

G Mechanical drawings

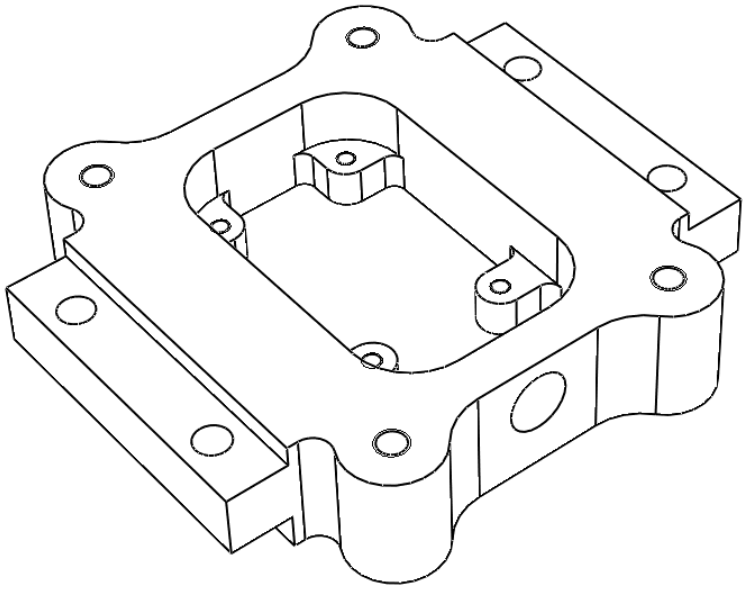

Figure 90: IMU box.

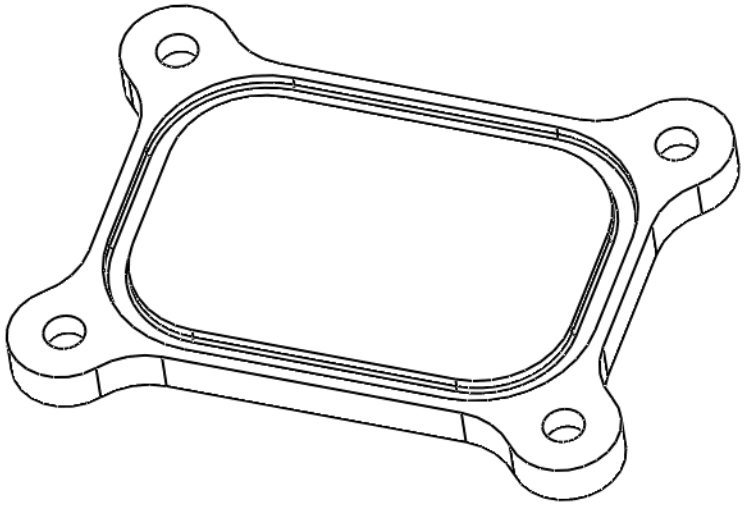

Figure 91: IMU lid.

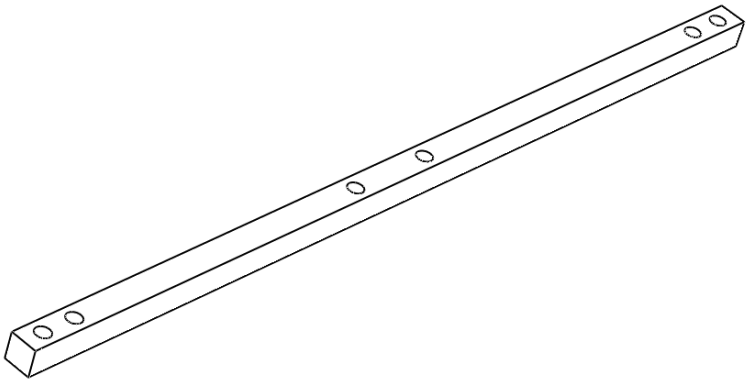

Figure 92: IMU box mount.

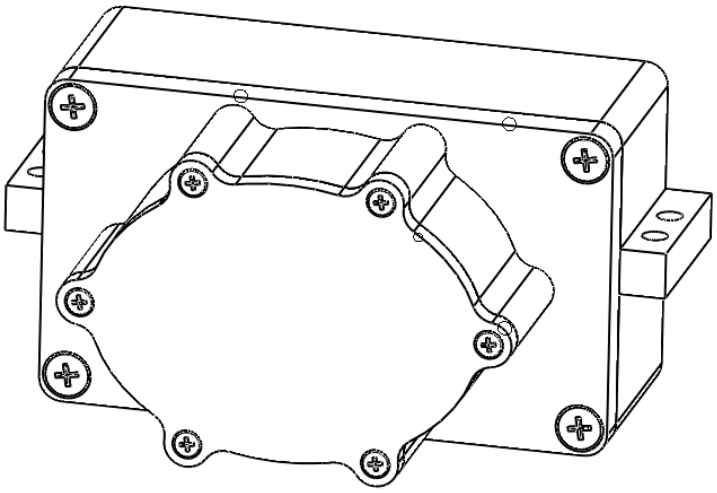

Figure 93: Downward facing camera.

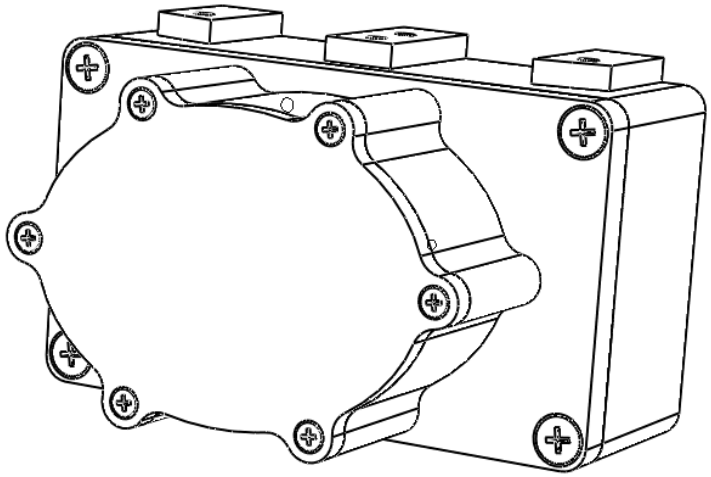

Figure 94: Forward facing camera.

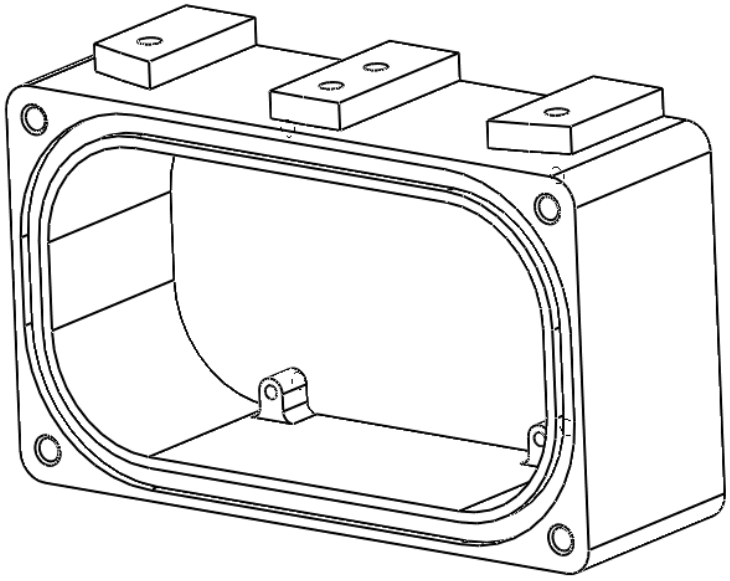

Figure 95: Rear part of the forward facing camera.

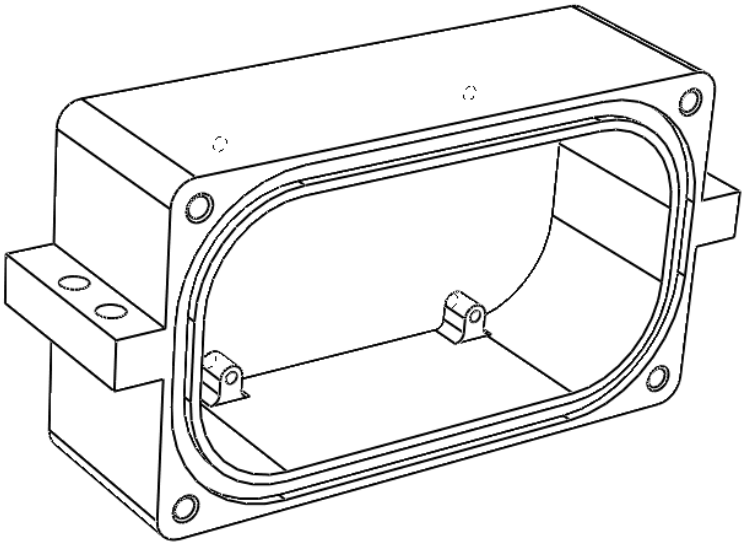

Figure 96: Rear part of the downward facing camera.

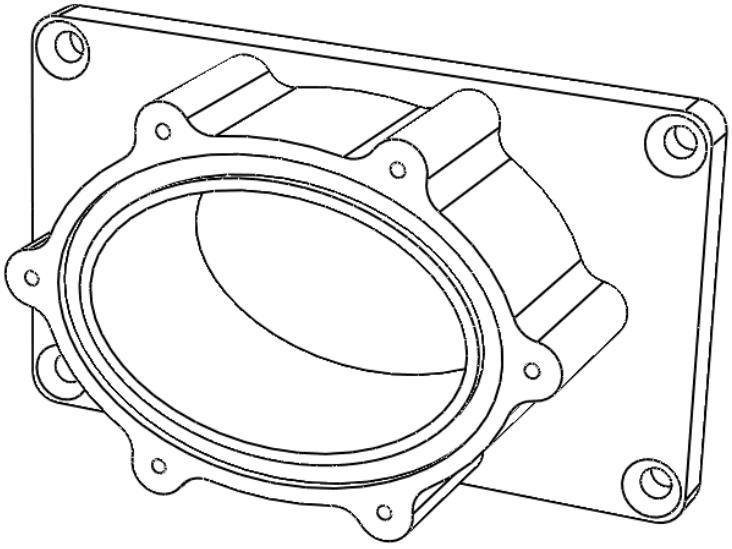

Figure 97: Front body of camera housing. This part is used for both camera boxes.

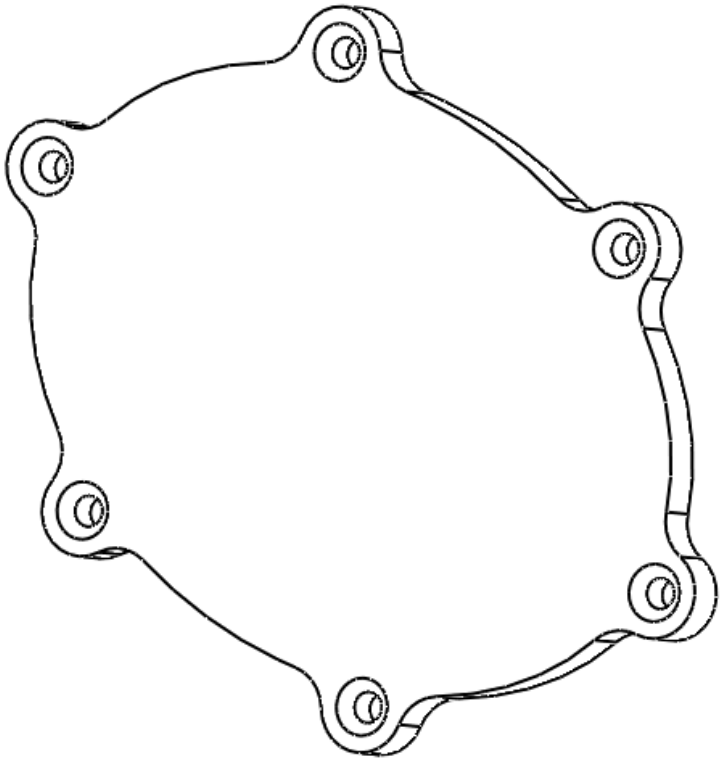

Figure 98: Camera lens.

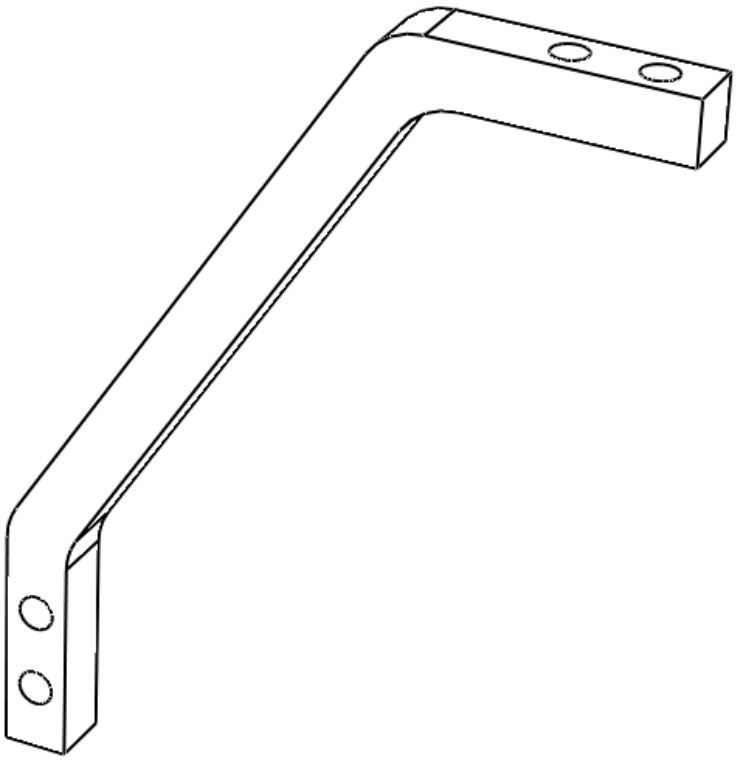

Figure 99: Mount to downward facing camera.

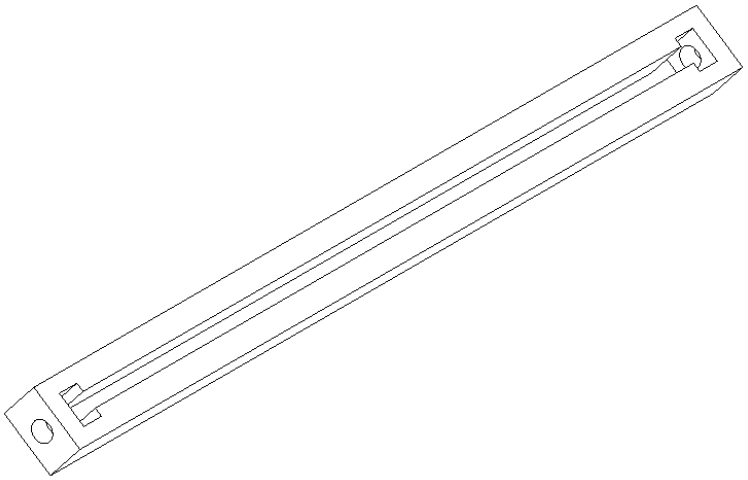

Figure 100: Mounting rod for electronic rack system.

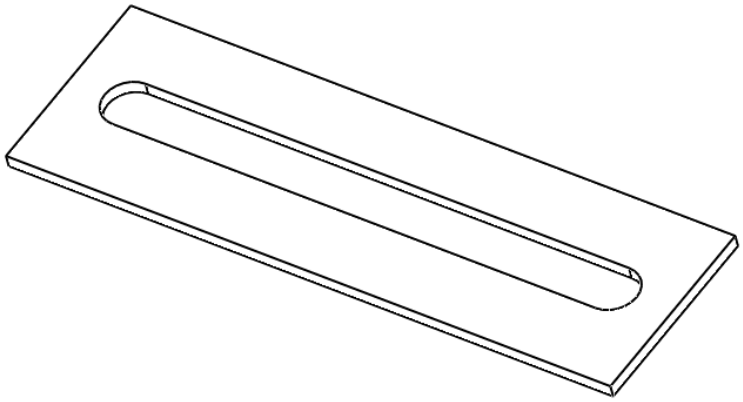

Figure 101: Part for mounting electronics to the rods.

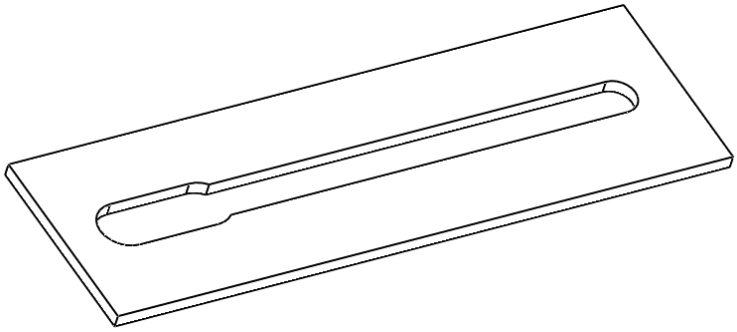

Figure 102: Part for mounting electronics to the rods.

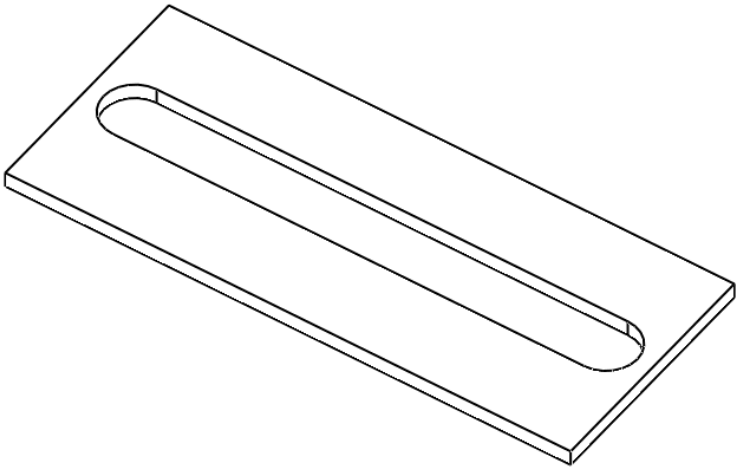

Figure 103: Part for mounting electronics to the rods.

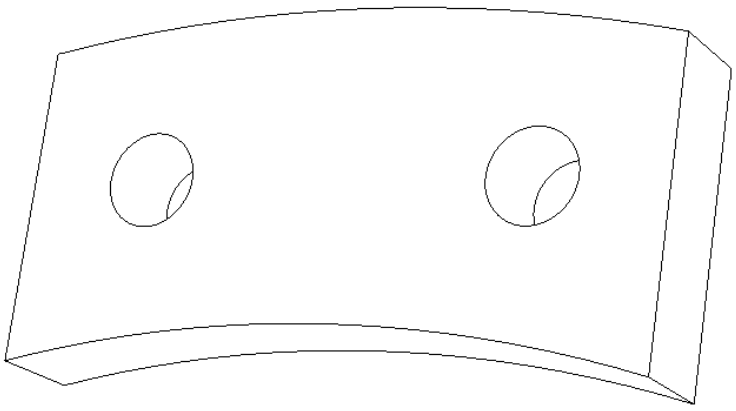

Figure 104: Guide for electronic rack system.

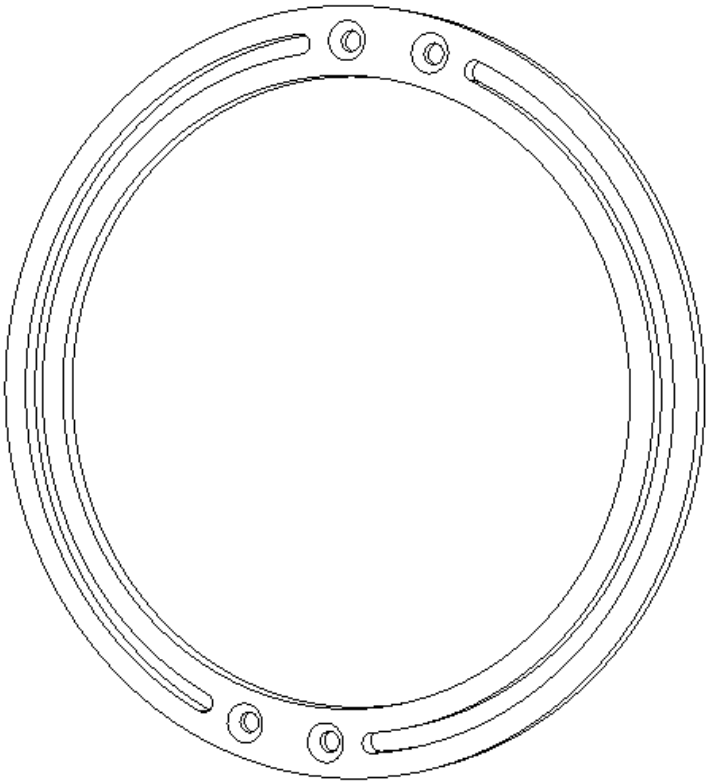

Figure 105: Inner plate for electronic rack system.

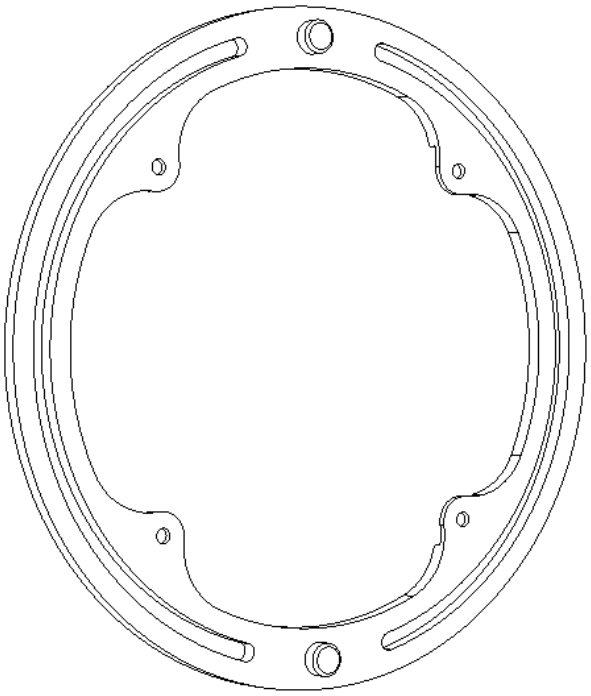

Figure 106: Outer plate for electronic rack system.

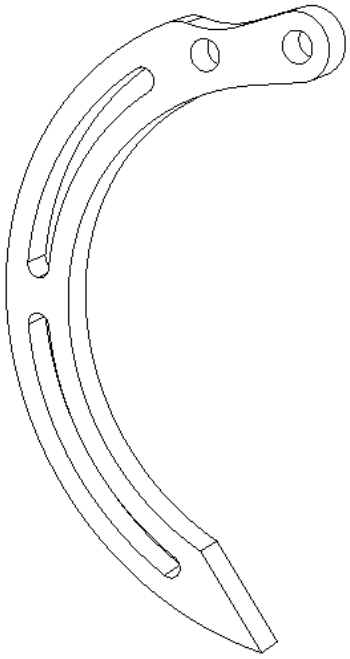

Figure 107: Claw to the gripper.

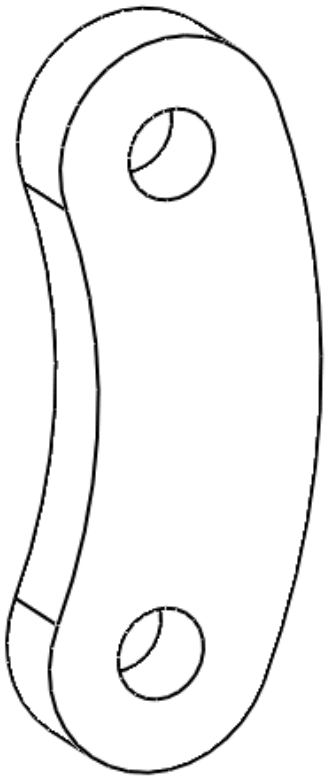

Figure 108: Link for driving claws to the gripper.

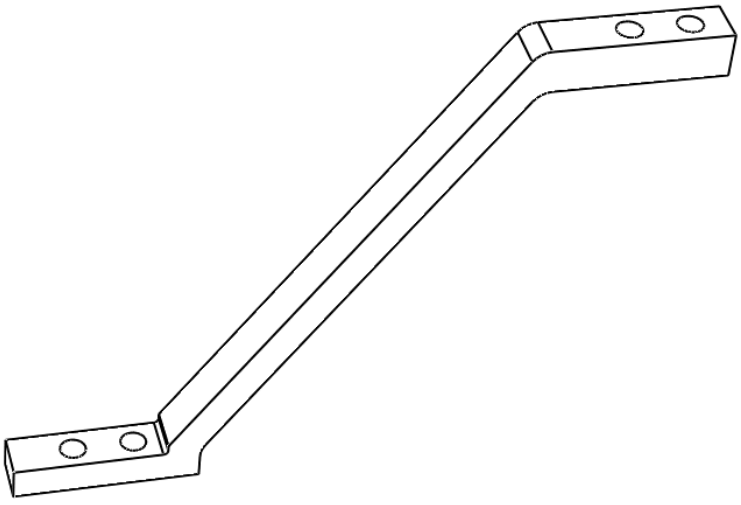

Figure 109: Gripper mount.

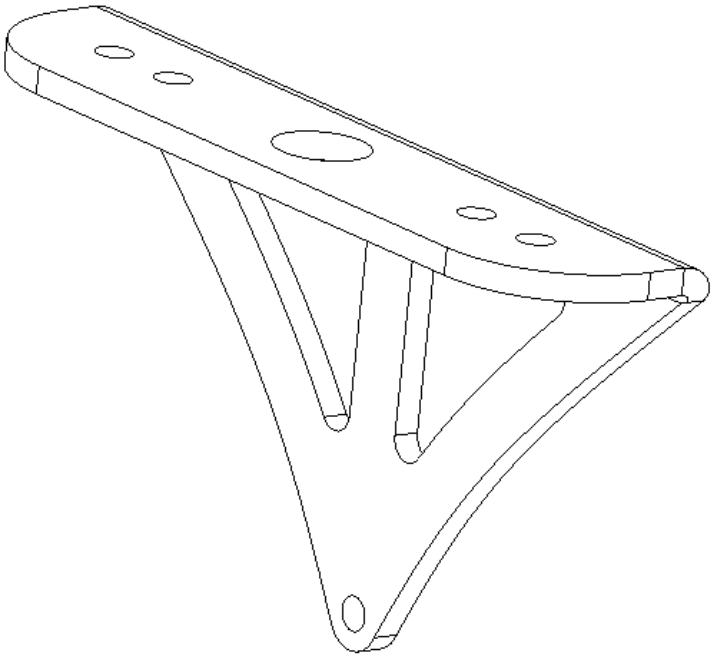

Figure 110: Gripper support.

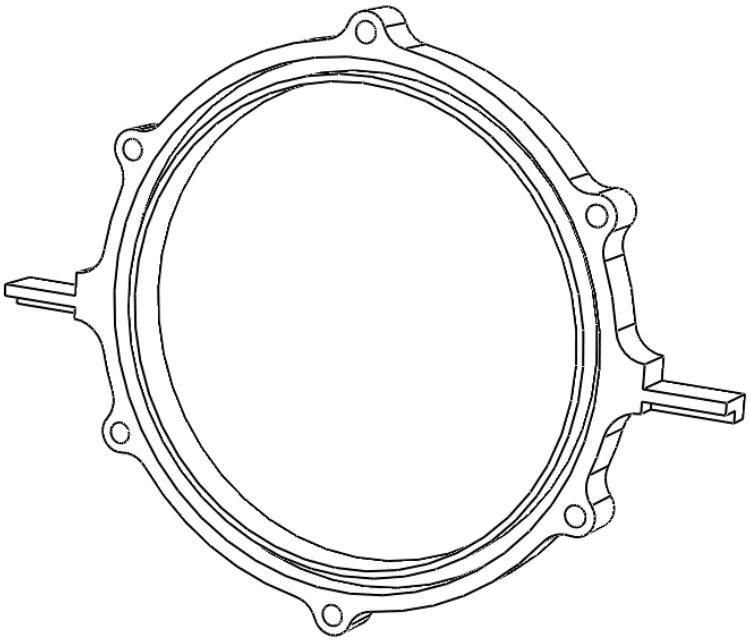

Figure 111: Inner flange.

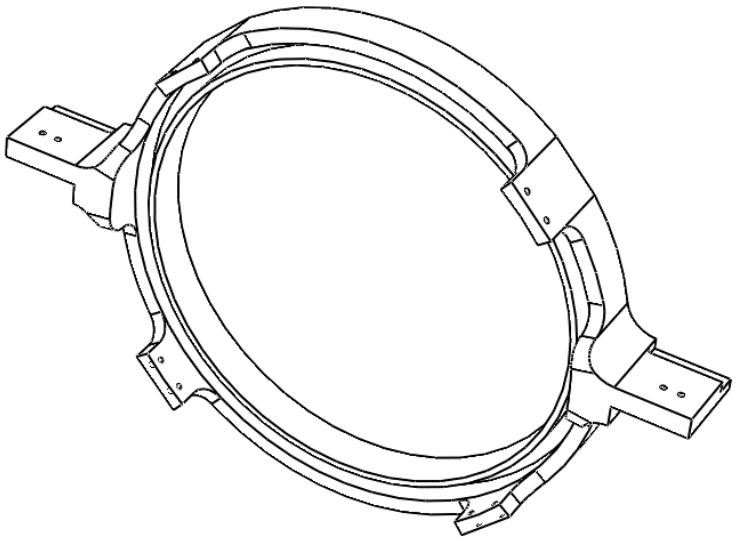

Figure 112: Outer flange.

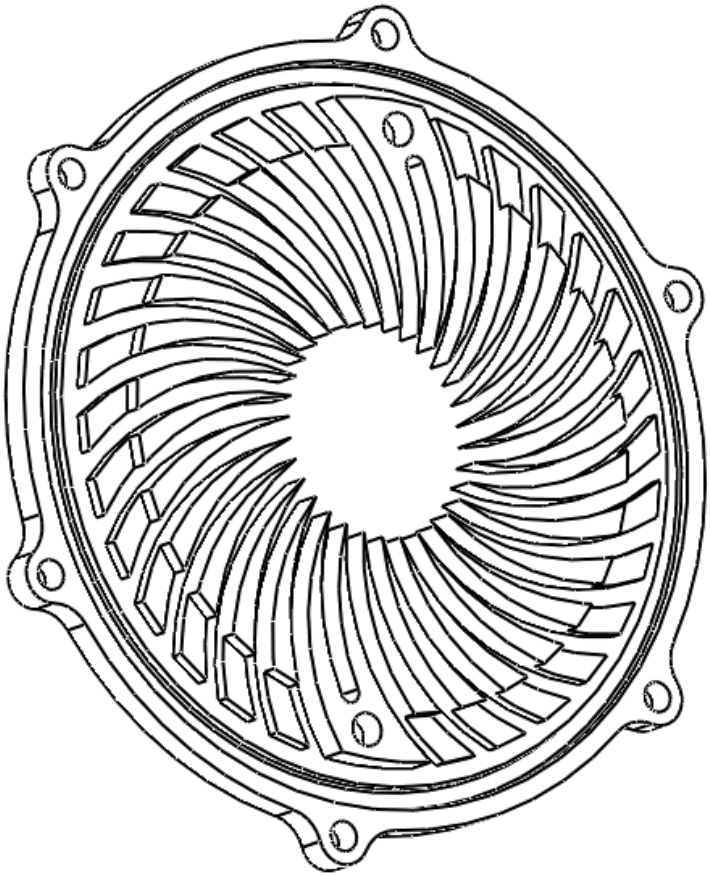

Figure 113: Inner lid.

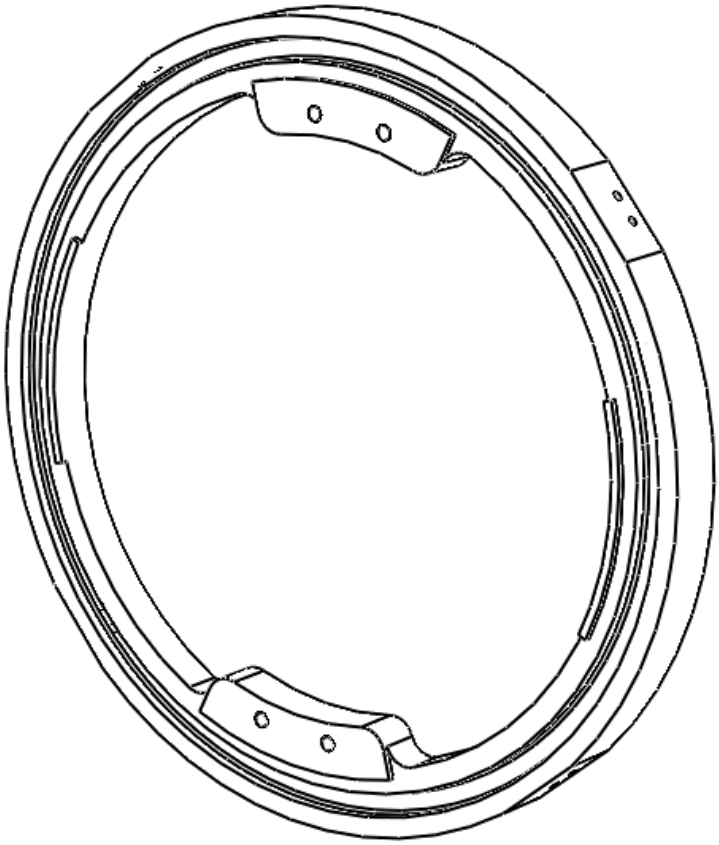

Figure 114: Outer lid.

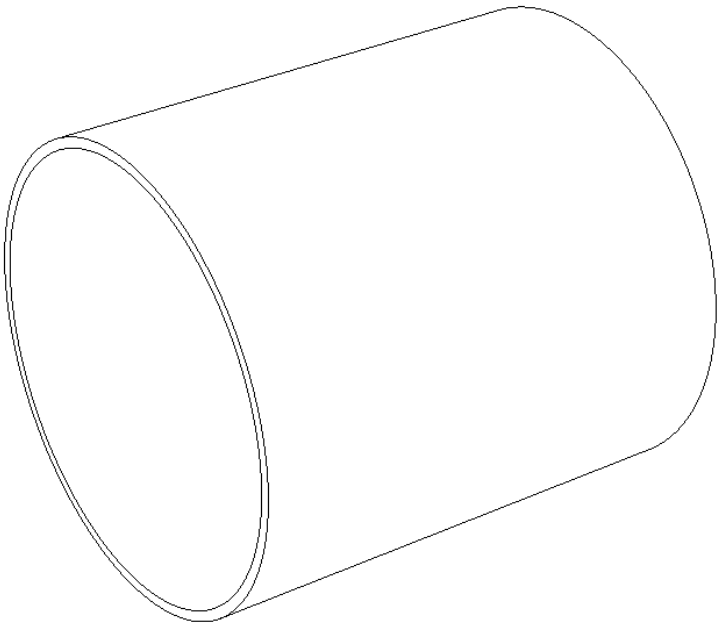

Figure 115: Plastic tube.

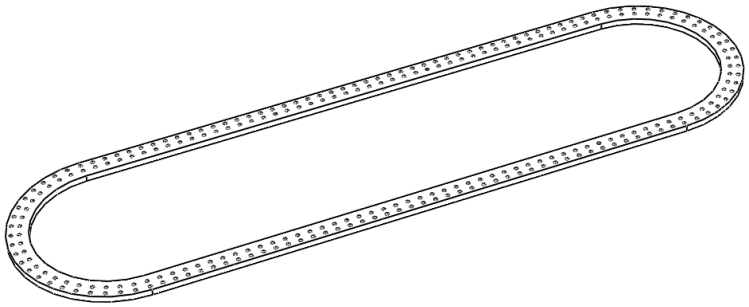

Figure 116: The frame.

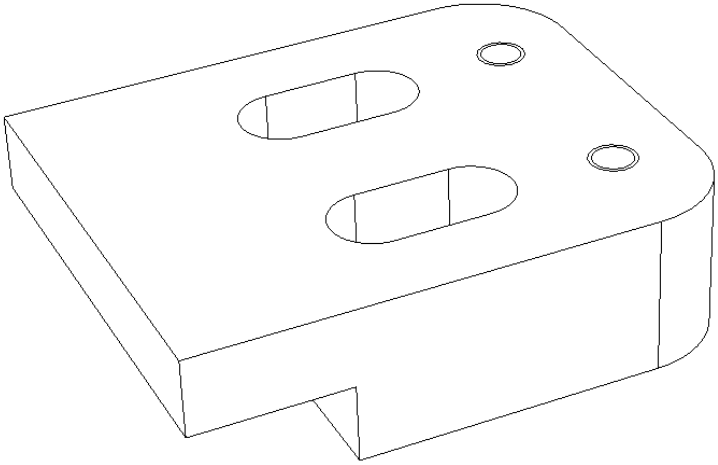

Figure 117: Tubes mount attached to the frame.

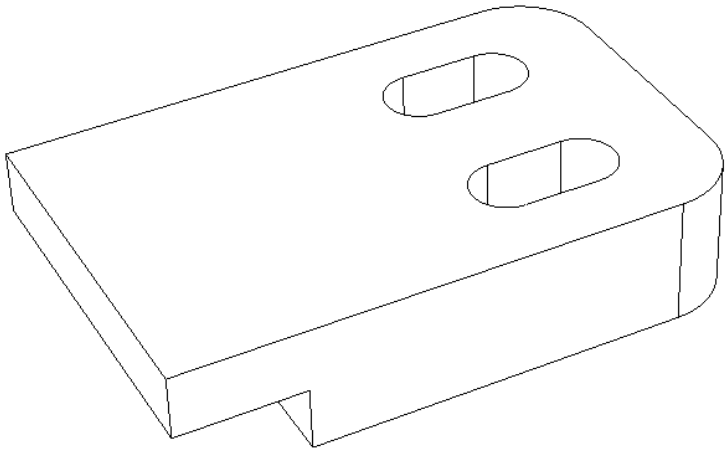

Figure 118: Tubes mount attached to the frame.

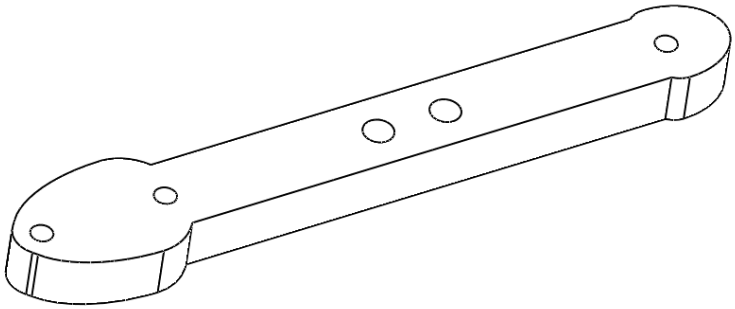

Figure 119: Thruster mount.

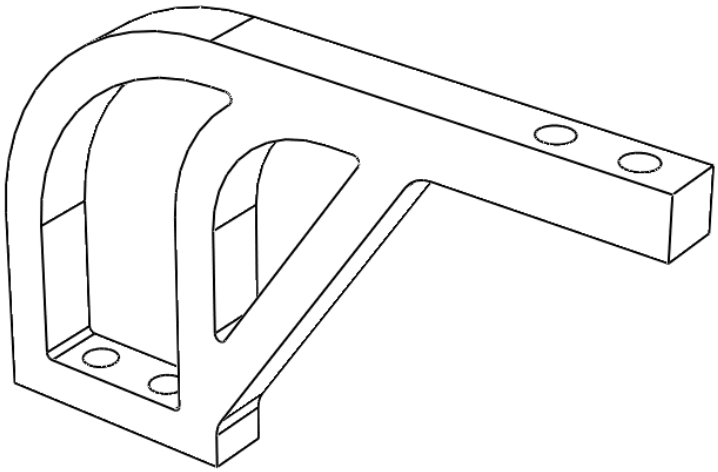

Figure 120: Mount that attaches thruster mount to the frame.

Figure 121: Thruster mount back.

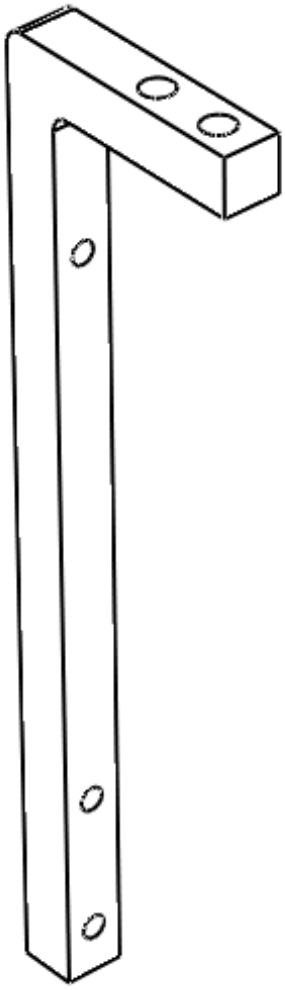

Figure 122: Thruster mount front.

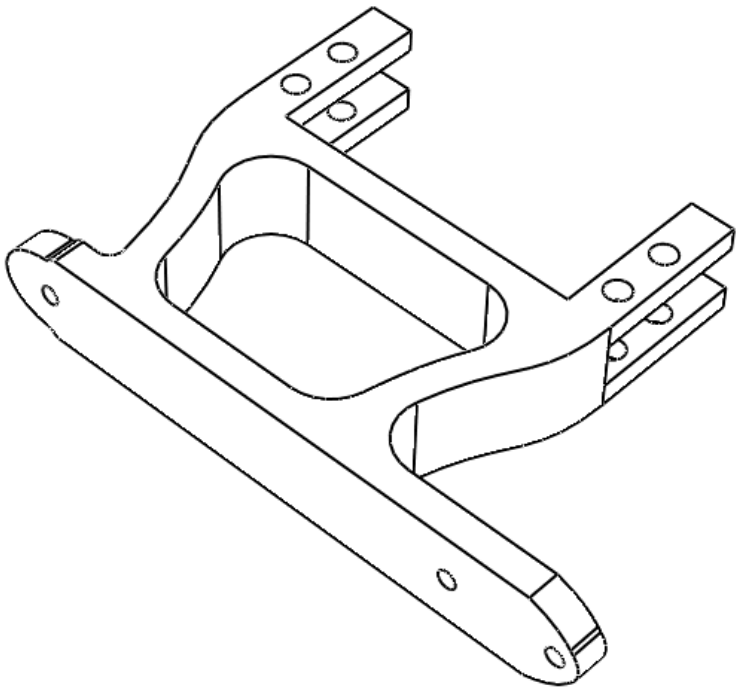

Figure 123: Thruster mount horizontal.

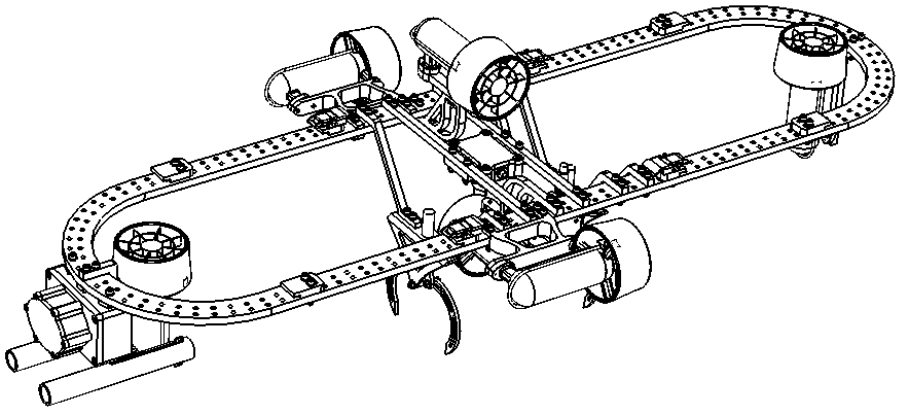

Figure 124: Complete robot without tubes.

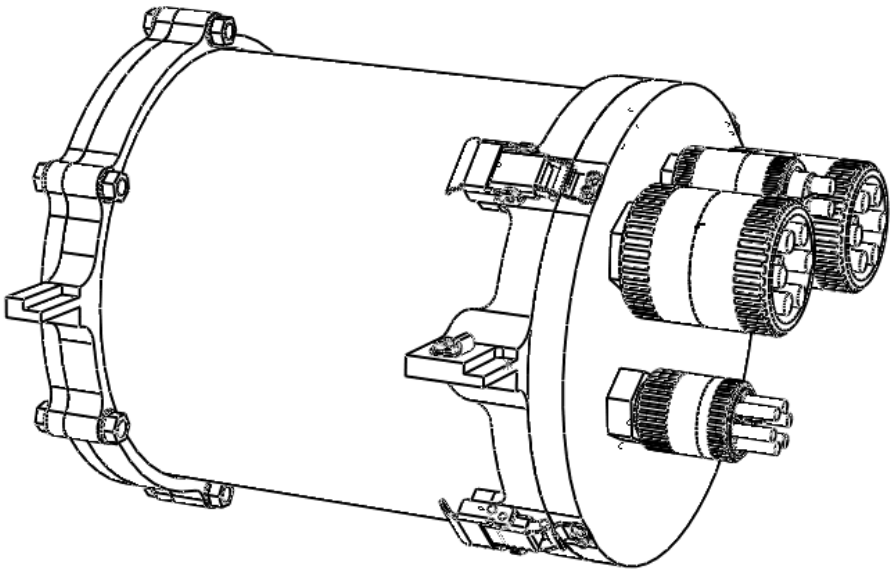

Figure 125: Backbone and miniITX.

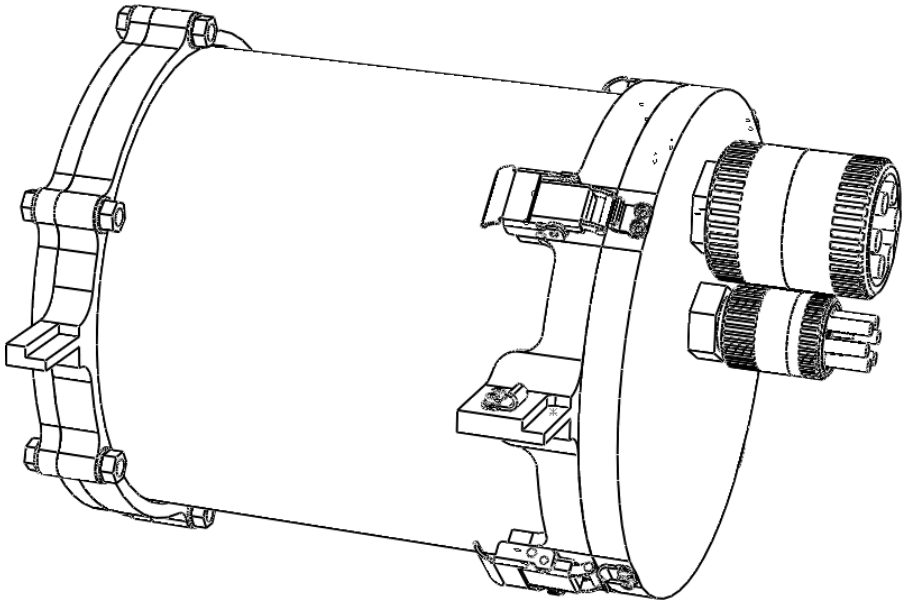

Figure 126: Power- and motor controller.

H User guides

H.1 Electrical

To set up the electrical system of the Vasa there will be a set up sequence. To simplify this procedure the upcoming part will list the set up in a numerical list. Some of these stages may already be completed but checking this list will verify if the system is ready.

1 Insert Router - card, I/O - card, and any other card in use in the sharkboard.

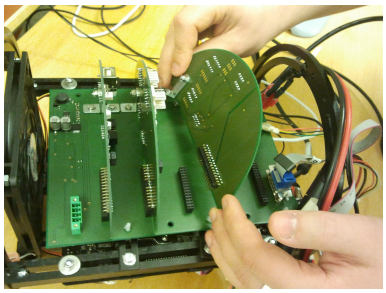

Figure 127: Top view of backbone header.

2 Connect wireing to all of these nodes, in the basic system this includes cables SIG 020:BB-PS, SIG 010:BB-IMU, USB010:ITX-BB.

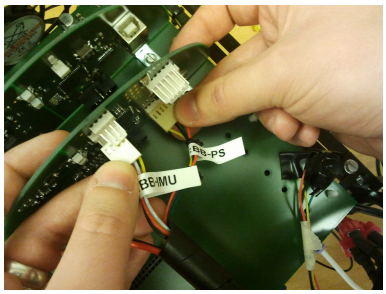

Figure 128: Top view of backbone header.

3 Connect power to the Backbone(PWR060:PB-BB).

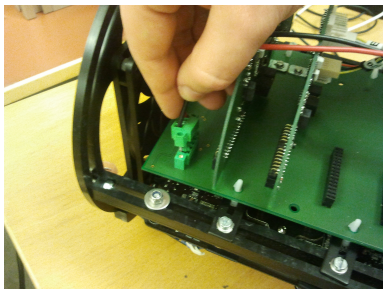

Figure 129: Top view of backbone header.

4 Connect Ribbon cable to backbone(CAN010).

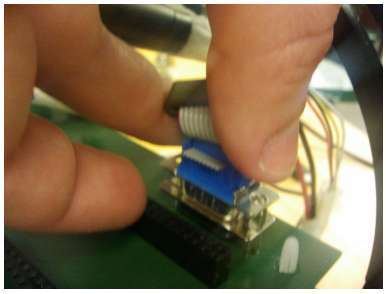

Figure 130: Top view of backbone header.

5 Connect Firewire cables(FIR 010:ITX-CAM, FIR 020:ITX-CAM).

6 Ensure proper Connections in the MINI-ITX including PSU device.

7 Connect Ribbon cables to Power board and Motor-controller(CAN020).

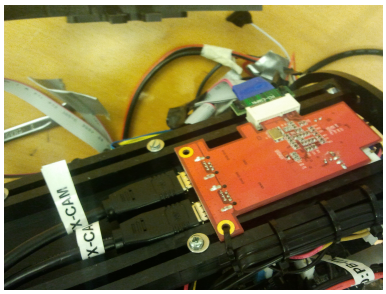

Figure 131: Top view of backbone header.

8 Connect Power distribution card power connector.

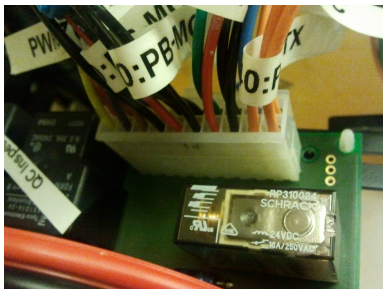

Figure 132: Top view of backbone header.

9 Connect power cables to motor-controller and h-bridges(PWR070:PB-MC - PWR150:PB-MC).

10 Connect motors to H-bridges note the numbering next to the H-bridges corresponds to the correct motor,verifiy this with programmers. Connector from motorcontroller to H-bridges (SIG 070:MC-MC - SIG 120:MC-MC) are specified for a sertant position and the cable order on the headers may vary from diffrent ports to the H-bridges. If modifications are done, make sure the order of colors acording to image is correct).

11 Connect battery to system. make sure nothing gets warm or smells. It is important to note that the system is intended to run underwater and shuld not run outside of water more than necisary, this includes standby modes. Nether should the robot never run in direct sunlight outside of water. Fan will start at the time of powering the system.

12 close hulls.

13 Install hulls in robot.

14 connect water sealed cables. Connect the power cable to the forward hull last, and make sure the robot is in an suspended state (standby mode) by deactivating the mission switch.

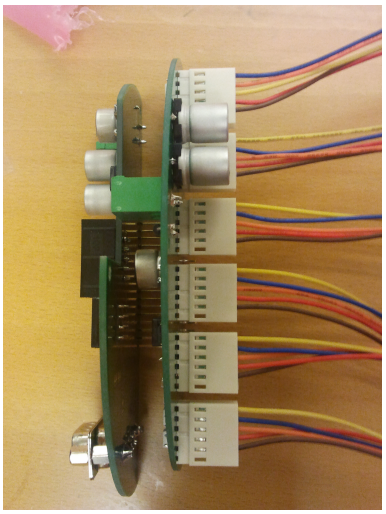

Figure 133: Top view of backbone header.

Figure 134: Top view of backbone header.

H.2 Camera tutorial

This section describes how the cameras are operated. The easiest way to do this is to use a program called Coriander. For manual image acquisition the libdc1394 functions must be used.

H.2.1 Coriander

Coriander is a program that offers control of your firewire camera. It supports dynamic adjustment of camera parameters and can set auto modes (auto exposure, auto gain etc.). Figure 135 and 136 shows two important tabs for adjustment and information in Coriander.

- Under **Camera Select** the connected cameras can be displayed.
- Under **Camera information** the specific GUID for each camera is displayed. This ID is used to identify the camera and is essential when dedicating a direction for a camera. Vasa uses two cameras with different directions; one directed forwards and one downwards. The GUID is used to distinguish the two cameras.

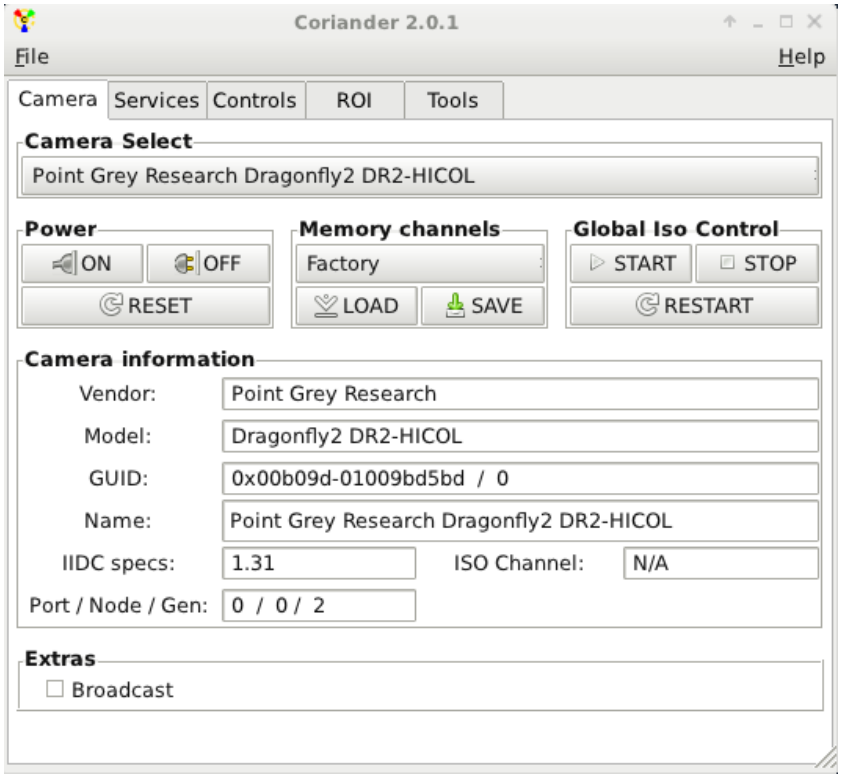

Figure 135: Coriander camera tab.

In this figure all the controls of the camera parameters are displayed. Parameters can be set to OFF and MANUAL. Some of the parameters can also be set to AUTO if it is supported by the connected camera.

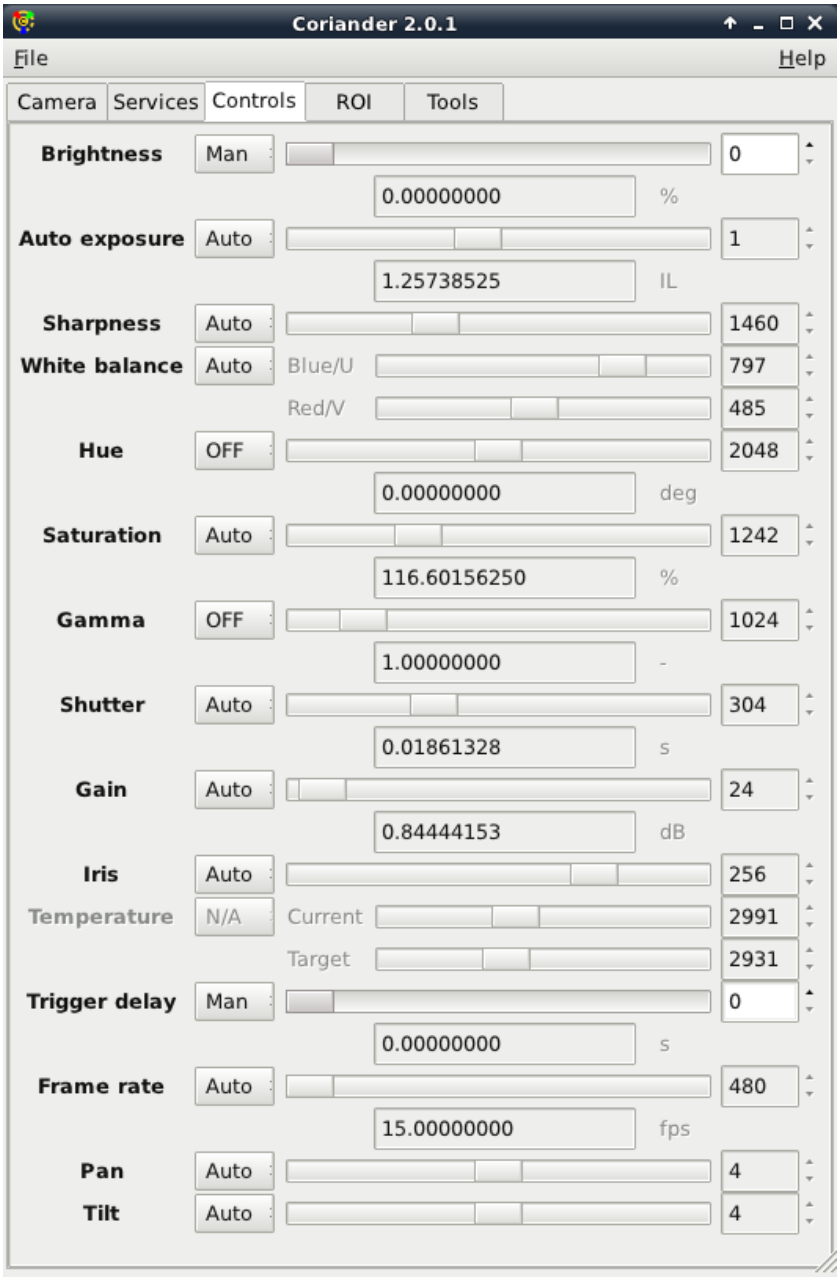

Figure 136: Coriander service tab.

Following pictures are example pictures taken with Dragonfly 2 and coriander. They show how well the camera and auto iris lens works despite different light conditions. In all three pictures the settings are the same. All possible settings are set to auto.
In figure 137 the illumination conditions are very high and represents a very bright day.

In figure 138 the illumination conditions have been changed. The illumination provided by the ceiling lights are gone and the only light that is provided comes from the side. The auto iris and the auto camera parameters adjusts them self to provide the best possible image.

Figure 137: Coriander shot with full room light.

In figure 139 the illumination conditions are very low. The room lights are providing no light at all. The only light that comes in to the room is light slipping through the cracks between the window and the blind. As displayed the colors have faded away but it is still possible to see the contours.

Figure 138: Coriander shot without ceiling lights.

H.2.2 Capture a frame with libdc1394

The code for capturing a frame with libdc1394 and convert it to OpenCV is shown below:

```
int Get_IPL_Frame(dc1394camera_t *camera,IplImage *cv_frame)
{
    dc1394video_frame_t *frame=NULL;

    dc1394error_t err;

    err = dc1394_capture_dequeue(camera, DC1394_CAPTURE_POLICY_WAIT, &frame);
    DC1394_ERR_CLN_RTN(err,cleanup_and_exit(camera),"Could not capture a frame\n");

    uyvy2bgr(frame->image,(unsigned char*)cv_frame->imageData,MAX_WIDTH*MAX_HEIGHT);
    dc1394_capture_enqueue (camera, frame);

    return(TRUE);
}
```

Listing 19: Image acquisition with libdc1394

H.2.3 Conversion to OpenCV

conversion from libdc1394 frame to an openCV frame.

```
void uyvy2bgr(const unsigned char *src, unsigned char *dest,
```

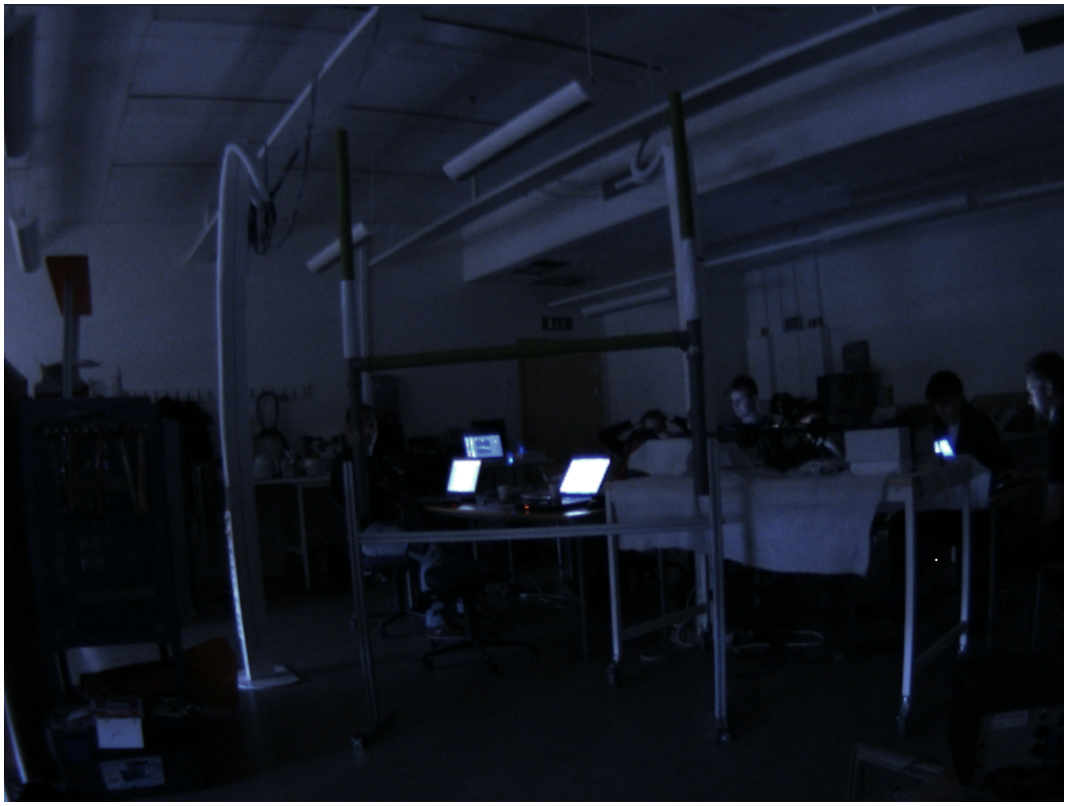

Figure 139: Coriander shot with no lights.

```
                unsigned long long int NumPixels)
{
        register int i = (NumPixels << 1) - 1;
        register int j = NumPixels + (NumPixels << 1) - 1;
        register int y0, y1, u, v;
        register int r, g, b;

        while (i > 0) {
                y1 = src[i--];
                v = src[i--] - 128;
                y0 = src[i--];
                u = src[i--] - 128;
                YUV2RGB(y1, u, v, r, g, b);
                dest[j--] = r;
                dest[j--] = g;
                dest[j--] = b;
                YUV2RGB(y0, u, v, r, g, b);
                dest[j--] = r;
                dest[j--] = g;
                dest[j--] = b;
        }
}
```

Listing 20: Conversion from libdc1394 frame to OpenCV frame

H.3 Mechanical Guide

This section will describe how to assemble the robot. See Appendix [J] for bill of materials. The ID numbers for all parts can be found in that table.

H.3.1 Step 1: Attach thruster to mount

Mount the thruster [ID 130] to the thruster mount [ID 48] with three M3 screws [ID 59], see figure [140].

Figure 140: Mount the thruster mount to the thruster.

H.3.2 Step 2: Assemble camera

Use four M3 screws [ID 26] and four spacers [ID 10] to mount the camera [ID 15 and ID 17] into the back of the camera housing [ID 27], see figure [141].

Mount one o-ring [ID 22] to the back of the camera housing. When the o-ring is attached, mount the front of the camera housing using four M5 screws [ID 30], see figure [142].

Next step is to attach the smaller o-ring [ID 23] to the front of the camera housing. When the o-ring is attached, use six M3 screws [ID 29] to attach the camera lens [ID 16] to the camera housing, see figure [143].

Now the front camera housing is complete, see figure [144].

H.3.3 Step 3: Attach to the frame

Mount the thruster assembled in step 1 and camera assembly to the frame using M5 screws [ID 5 and ID 9]. Mount the parts in the center of the radius of the frame, see figure [145] and [146].

H.3.4 Step 4: Attach thruster to mount

Attach the thruster [ID 130] to the mount [ID 54] with three M3 screws [ID 59], see figure [147]

Next, attach the thruster attached to the mount to the frame, see figure [148]. Use two M5 screws [ID 5].

Now, the assembly so far looks like this, see figure [149].

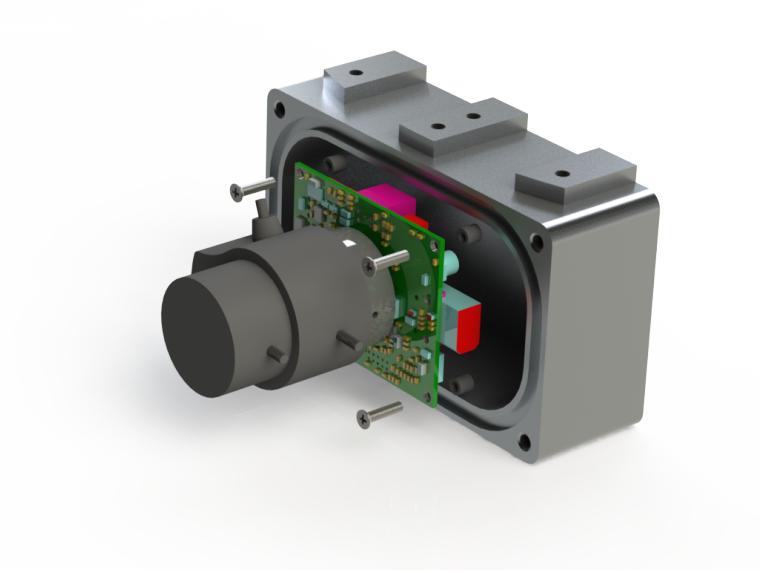

Figure 141: Camera housing assembly.

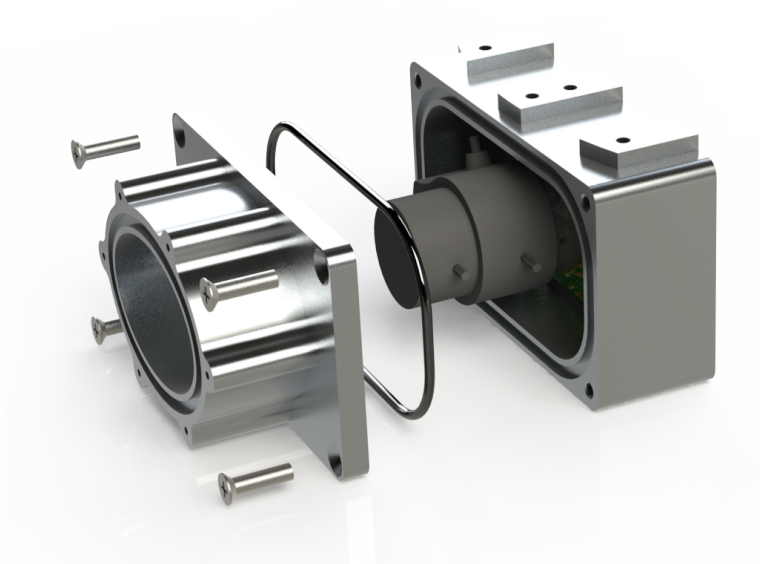

Figure 142: Camera housing assembly.

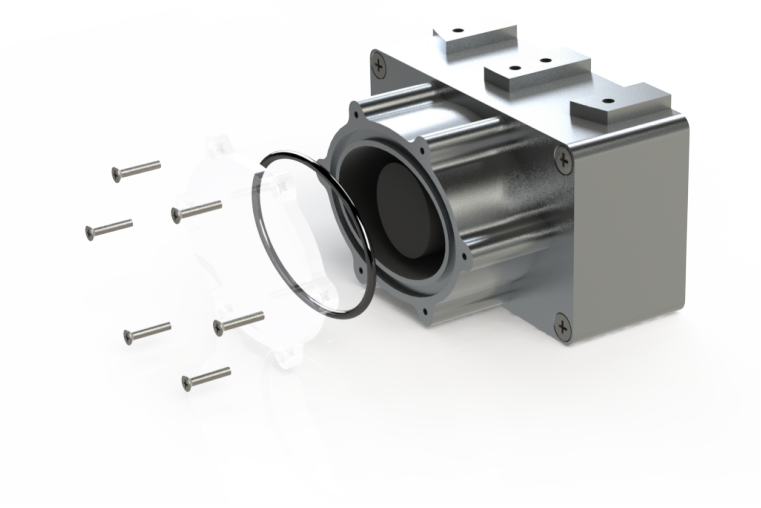

Figure 143: Camera housing assembly.

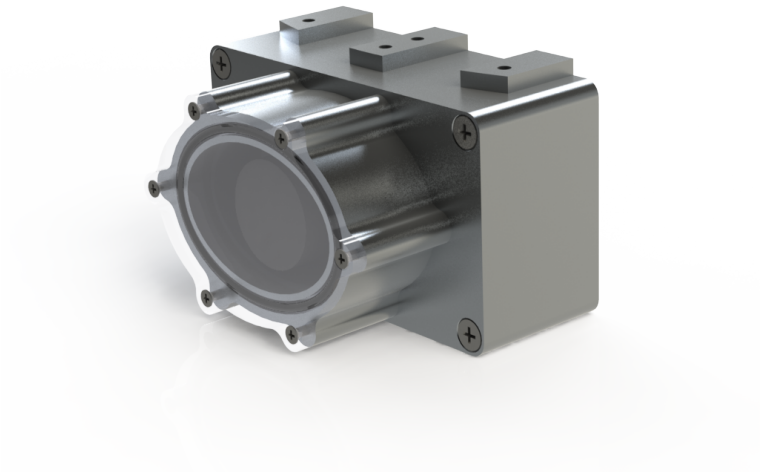

Figure 144: Camera housing assembly.

Figure 145: Mount the camera and thruster to the frame.

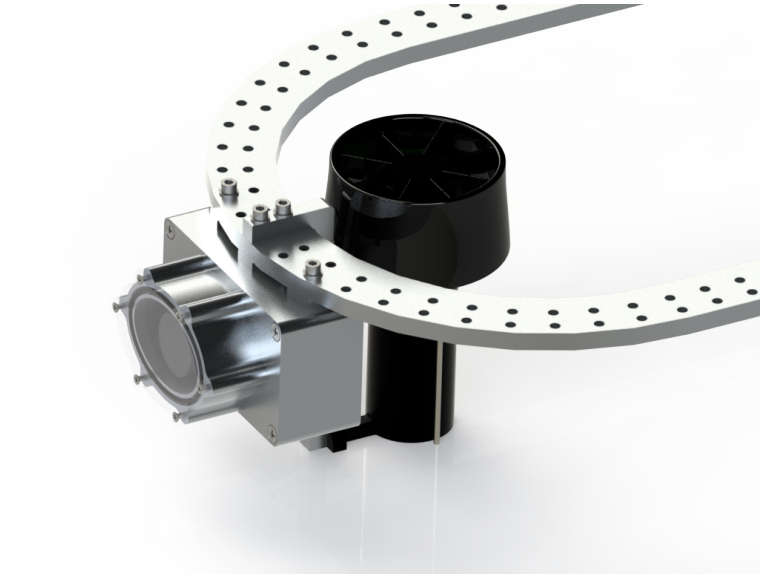

Figure 146: Camera and thruster are attached to the frame.

Figure 147: Mount the thruster to the mount.

Figure 148: Mount the thruster to the frame.

Figure 149: The robot assembly.

H.3.5 Step 5: Attach tube mounts to frame

This step is to attach mounts [ID 52] holding the tubes in the right position. This is mounted with sixteen M5 screws [ID 5], see figure [150]. The mounts are attached in the fourth hole, counted from were the holes are straight (and not in a radius pattern). This procedure is repeated for the other side as well.

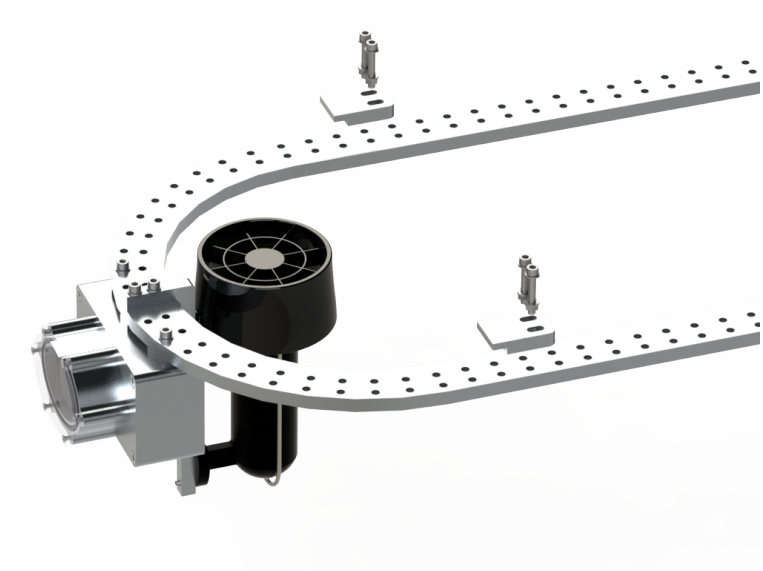

Figure 150: Mounts holding the tubes.

Mount the smaller mounts [ID 51] to the frame as the picture shows, see figure [151]. The mounts is placed closest to the middle of the robot, ten empty holes from the last mounts. Eight M5 screws [ID 61] are used in this assembly. This procedure is repeated for the other side as well.

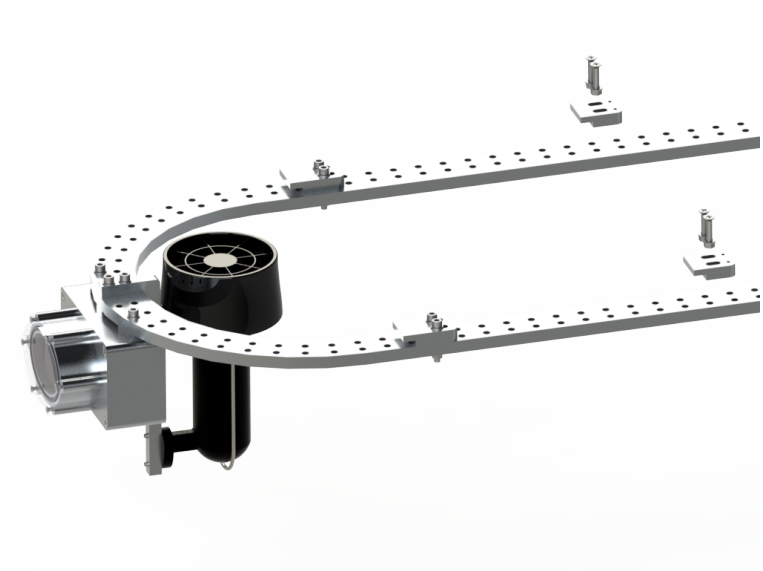

Figure 151: Mounts holding the tubes.

Now, the robot should look like this, see figure [152].

Figure 152: Frame assembly so far.

H.3.6 Step 6: Assemble the IMU-box

The next step is to assemble the IMU-box. The IMU sits inside the IMU-box [ID 131], see figure [153]. The IMU [ID 36] is attached to the box with four M3 screws [ID 42].

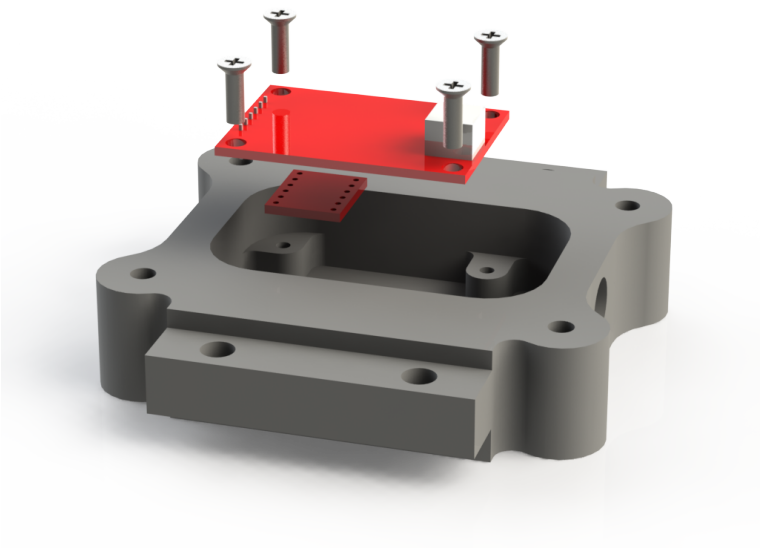

Figure 153: IMU assembly.

This step is to make the box waterproof, see figure [154]. Insert the o-ring [ID 41] to the lid [ID 132]. After

that, attach the lid to the box by using four M5 screws [ID 30].

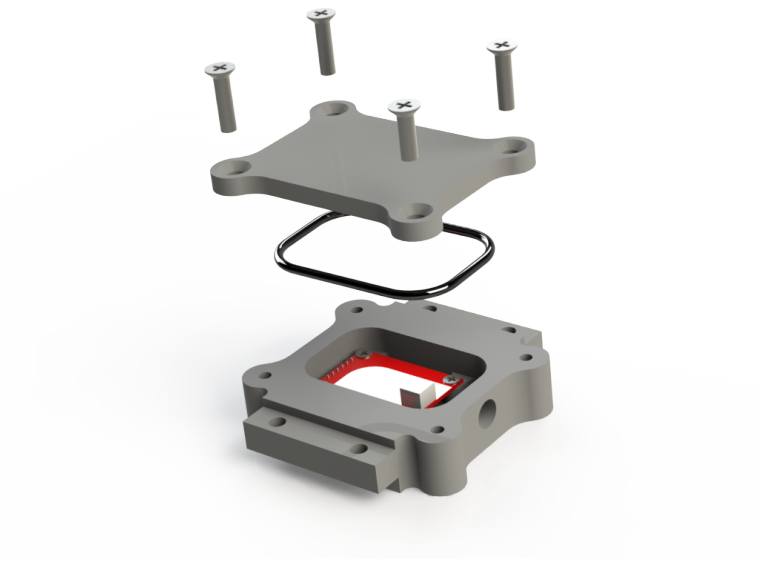

Figure 154: IMU assembly.

Now it is time to attach the IMU-box to the mounts [ID 50] that will attach the IMU assembly to the frame, see figure [155]. The short side of the mount should face the side were cables is inserted into the IMU-box. The mounts are attached with four M5 screws [ID 5].

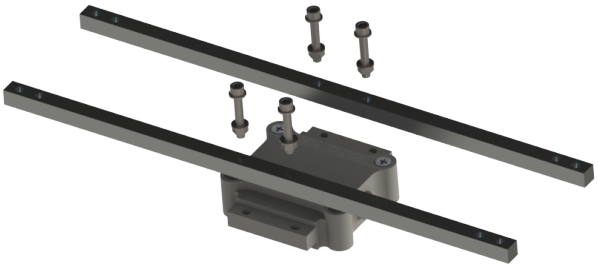

Figure 155: IMU assembly.

Then the assembly is ready to be mounted onto the frame of the robot. Use four M5 screws [ID 5] to attach the IMU assembly to the frame, see figure [156]. If you count from the tube mount closest to the middle of the robot, counted from the front, the IMU will be mounted with four free holes in between.

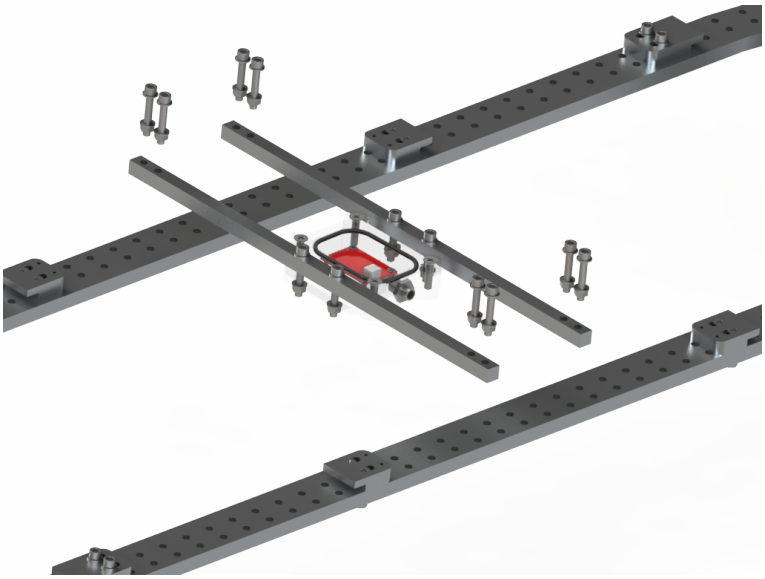

Figure 156: IMU assembly.

Now, the IMU isattached to the robot and looks like this, see figure [157].

Figure 157: Robot assembly so far.

H.3.7 Step 7: Assemble the middle camera

In this step, the middle camera will be installed. The first step is to assemble the camera housing, that can be seen in step 2. Then it is time for attach the mounts [ID 49], see figure [158].

Attach this assembly to the frame using four M5 screws [ID 5]. It should be two free holes between the

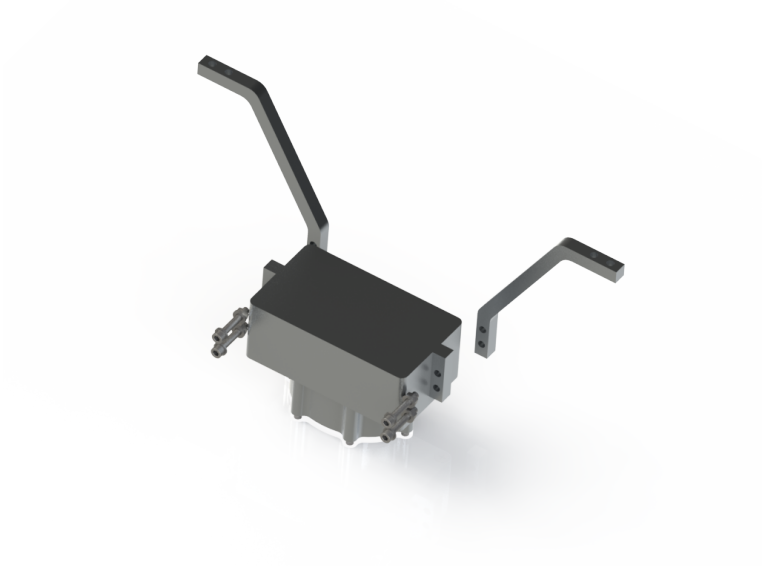

Figure 158: Camera down.

IMU mount and the camera mount, see figure [158].

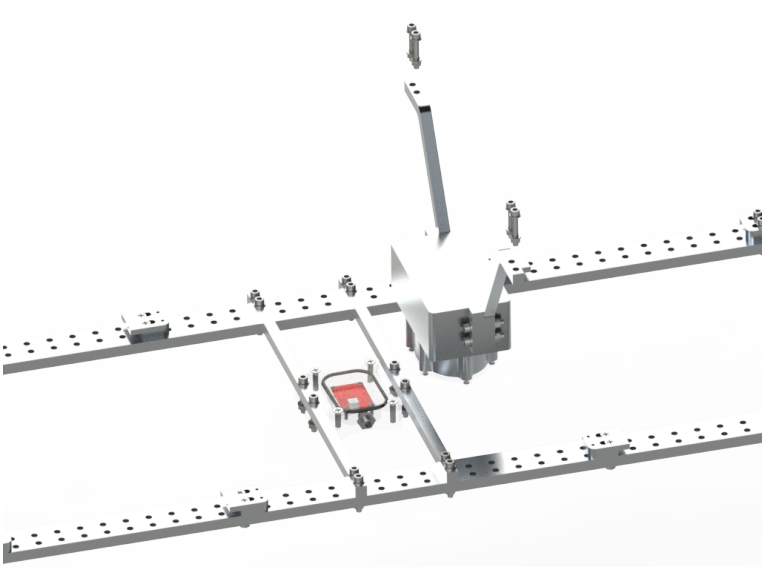

Figure 159: Camera down.

Now the robot looks like this, see figure [160].

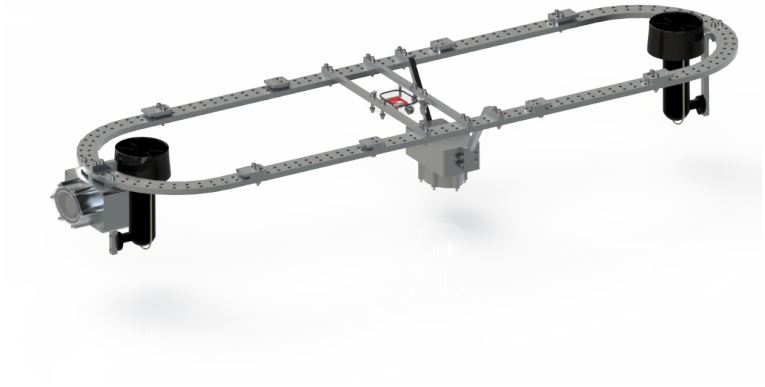

Figure 160: Robot assembly.

H.3.8 Step 8: Attach gripper mount

Now it is time for attaching the gripper mount [ID 57] to the frame, see figure [161]. Eight M5 screws [ID 5] are used. The mount closest to the front are two empty holes to the tube mount. The rear mount are one free hole from the tube mount.

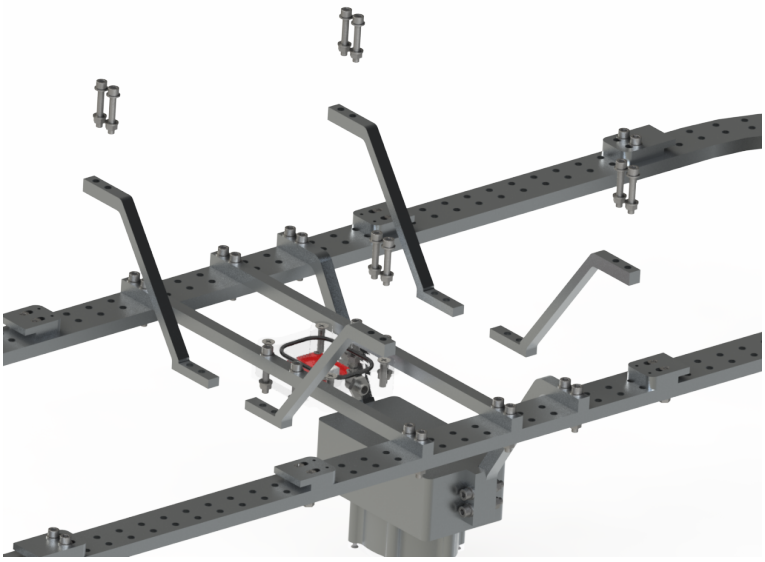

Figure 161: Attach the gripper mount.

With the gripper mount attached, the robot should look like this, see figure [162].

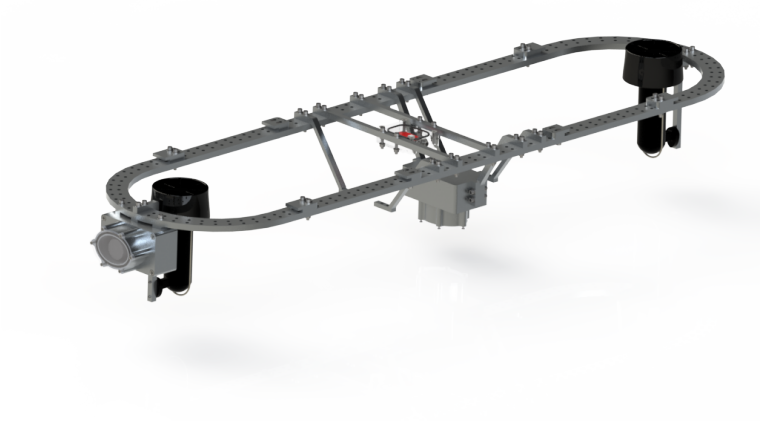

Figure 162: Frame assembly so far.

H.3.9 Step 9: Attach thruster to mount

During this step, the middle thrusters will be assembled, see figure [163]. Material used are one thruster [ID 130], three M3 screws [ID 59] and one motor mount [ID 56].

Figure 163: Attach the motor mount to the thruster.

Next step is to attach the frame mount [ID 55] to the assembly, see figure [164]. Two M5 screws [ID 62] are used.

Now it is time to attach this assembly to the frame, using two M5 screws [ID 5], see figure [165]. It should

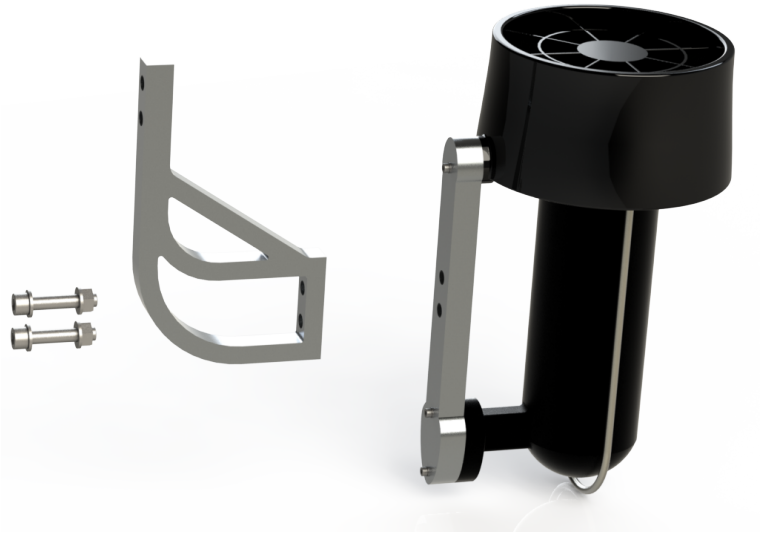

Figure 164: Assemble the middle thruster.

be one hole of air between the thruster mount and the front of the IMU mount.

Figure 165: Assemble the middle thruster to the frame.

Repeat the precious step to mount one more thruster on the opposite side if the robot, see figure [166]. The compete assembly so far should look like this, see figure [167].

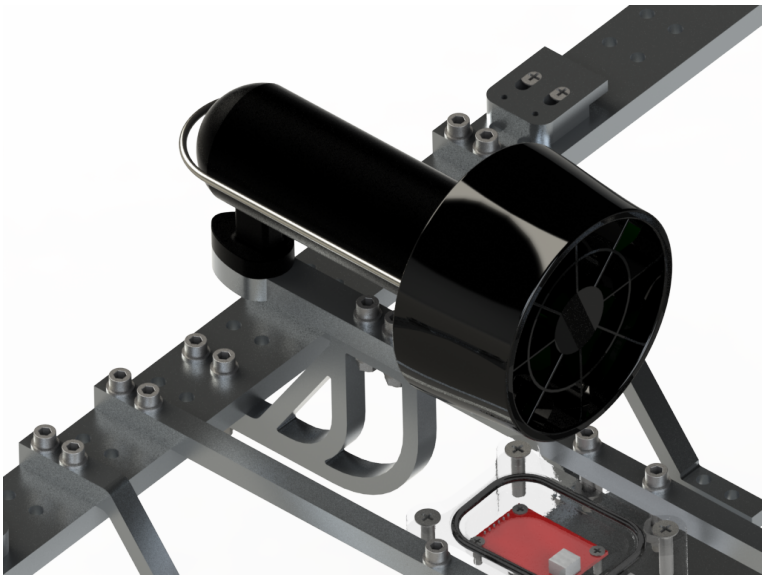

Figure 166: Assemble the second middle thruster to the frame.

Figure 167: Assembly so far.

H.3.10 Step 10: Attach excenter locks to the frame

In this step, we will prepare for mounting the tubes in the robot. First, four excenter lock [ID 133] must be mounted with eight M3 screws [ID 134], see figure [168] and figure [169].

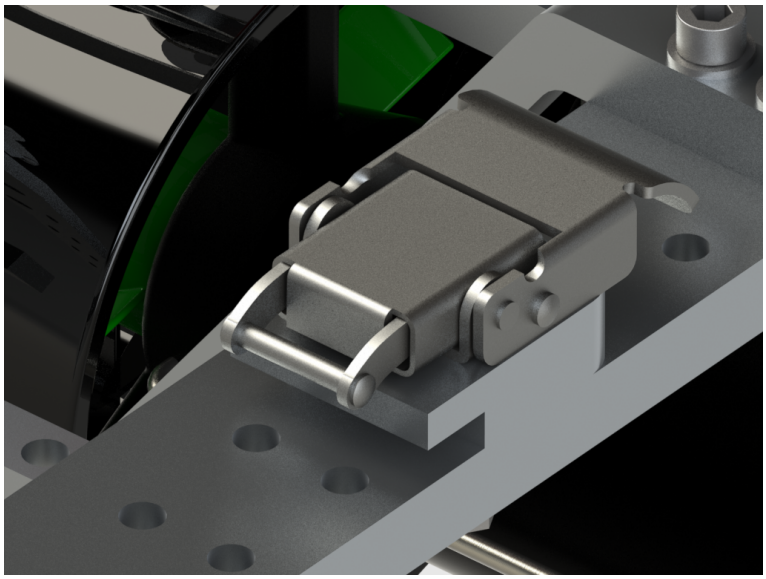

Figure 168: Attach the excenter lock to the frame.

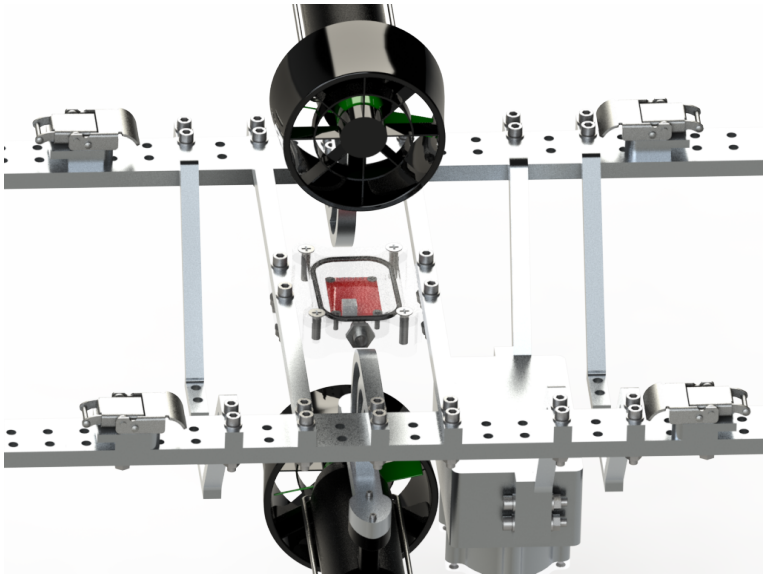

Figure 169: Four excenter locks are added to the frame.

H.3.11 Step 11: Assemble the tubes

Now, the tubes will be assembled. First step is to attach the electronic support system. The first thing to do is to attach the fan [ID 135] to the ring [ID 71], see figure [170].

The next step is to attach four rods [ID 73] to the ring, see figure [171]. Four M5 screws [ID 89] are used. Those mounting rods are adjustable and easy to move to the right position. The nuts are placed inside the rod, that will keep the nut in place, see figure [172].

The second ring [ID 72] is mounted to the rods, see figure [173]. This is made as before, using four M5

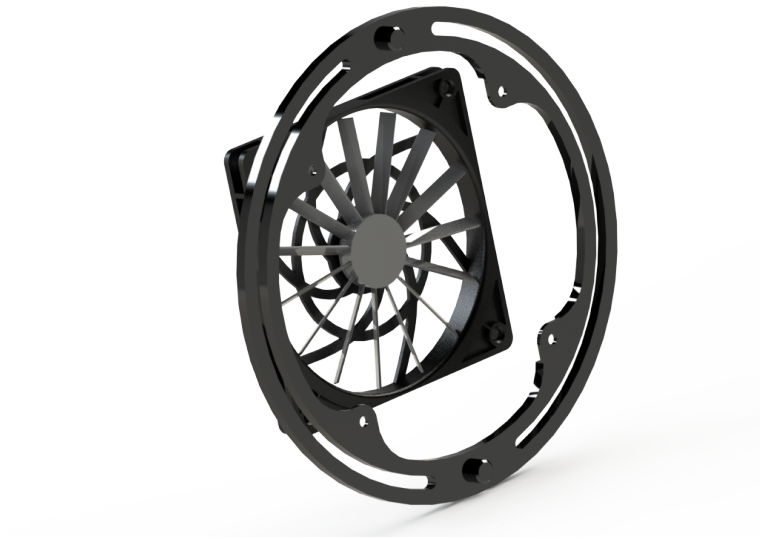

Figure 170: Attach fan to the ring.

Figure 171: Attach rods to the ring.

screws [ID 89].

The electronic support system should now looks like this, see figure [174].

In this step, the electronic support system is attached to the lid [ID 95], using two spacers [ID 81] and four M5 screws [ID 40], see figure [175].

Attach the mounts [ID 99] for excenter locks to the lid using M3 screws [ID 42], see figure [176].

Next step is to attach the mounts [ID 99] for excenter locks on the flanges, using four M3 screws [42], see figure [177].

Attach four excenter locks [ID 133] to the flanges, using eight M3 screws [42], see figure [178].

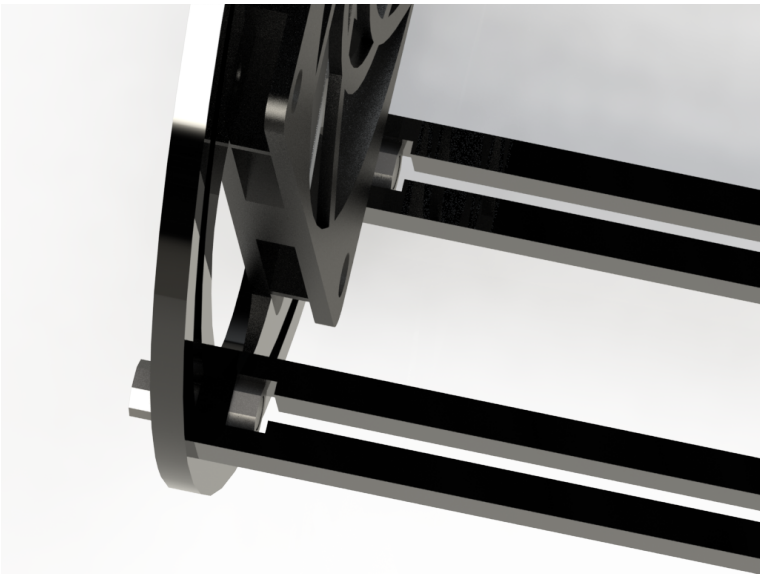

Figure 172: The figure shows the position of the nuts.

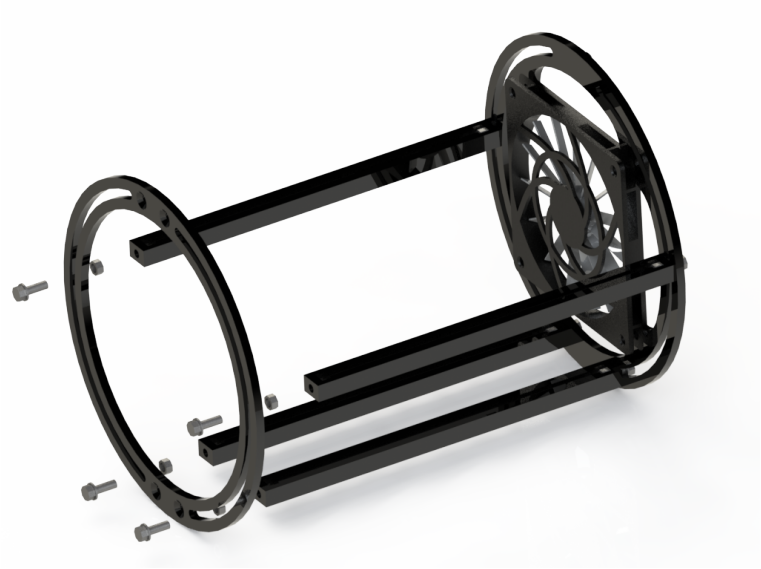

Figure 173: Assemble one more ring.

Now it is time for mounting the o-ring [ID 103] and lid [ID 93], see figure [179]. Place the o-ring in the o-ring slot on the lid. Six M8 screws [ID 102] are used to mount the lid.

Next task is to mount the electronic rack system to the tube, see figure [180]. Place the o-ring [103] in the o-ring slot on the lid. Close the tube by using the excenter locks.

Now the tube assembly is finished, see figure [181]. Repeat step 11 for the other tube, since the robot consist of two tube assembly's.

Mount the tubes to the frame, see figure [182].

Place the tube assembly on the frame of the robot, see figure [183]. Then slide the tube assembly into right

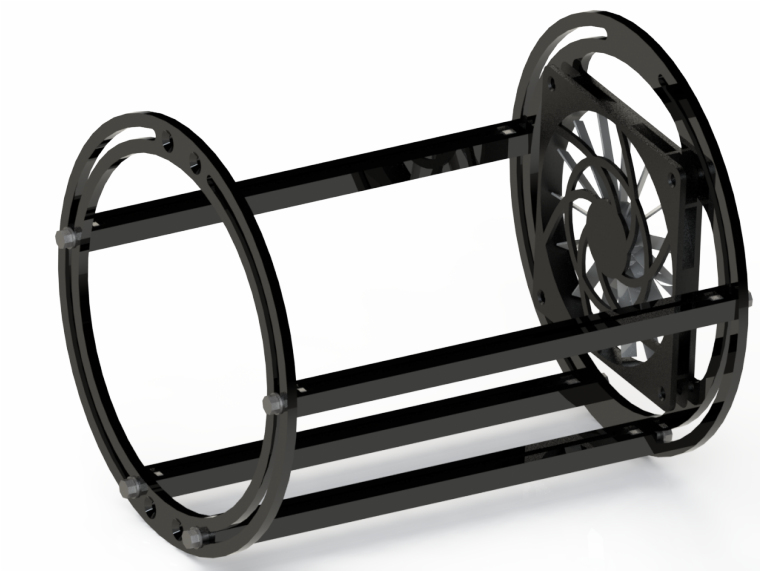

Figure 174: Complete electronic support system.

Figure 175: Attach the rack system to the lid.

position and lock it the excenter locks.

Now, the assembly of the robot is almost finished, see figure [184].

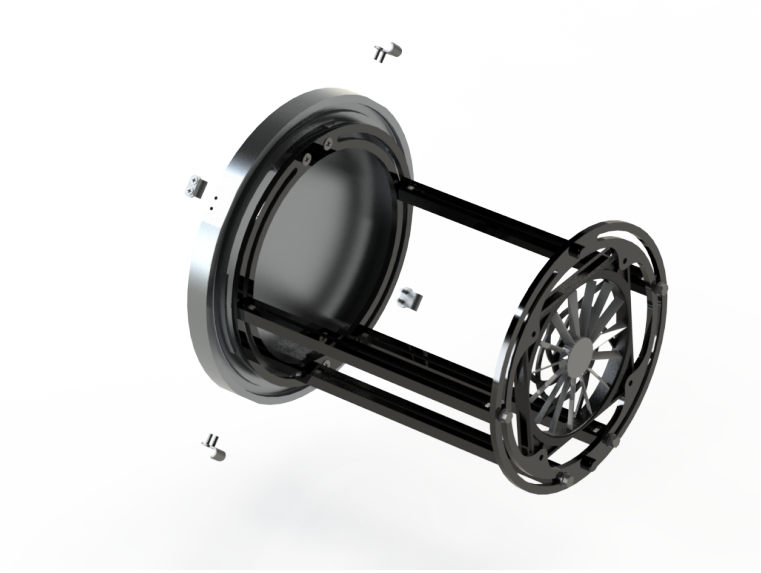

Figure 176: Attach the mounts to excenter locks to the lid.

Figure 177: Attach the mounts to excenter locks to the lid.

Figure 178: Attach the excenter locks to the lids.

Figure 179: Mount the o-ring and lid.

Figure 180: Mount the o-ring and electronic rack system.

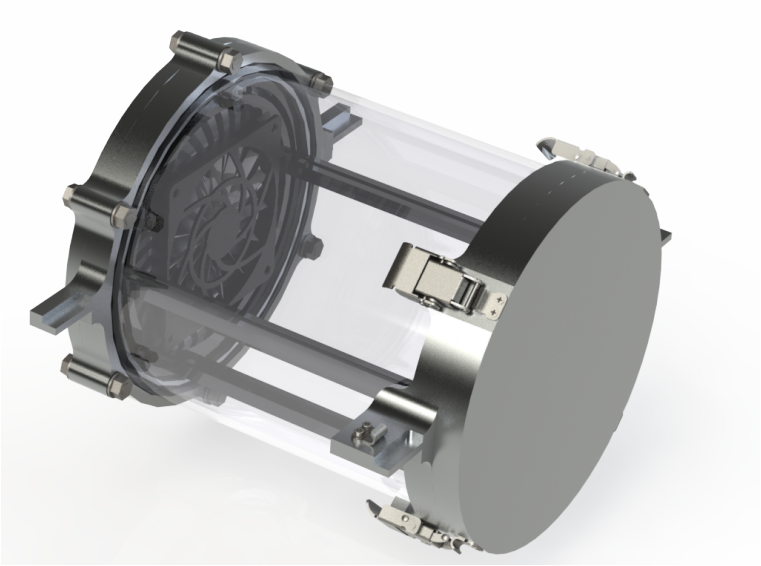

Figure 181: Tube assembly is finished.

Figure 182: Mount the tubes to the frame.

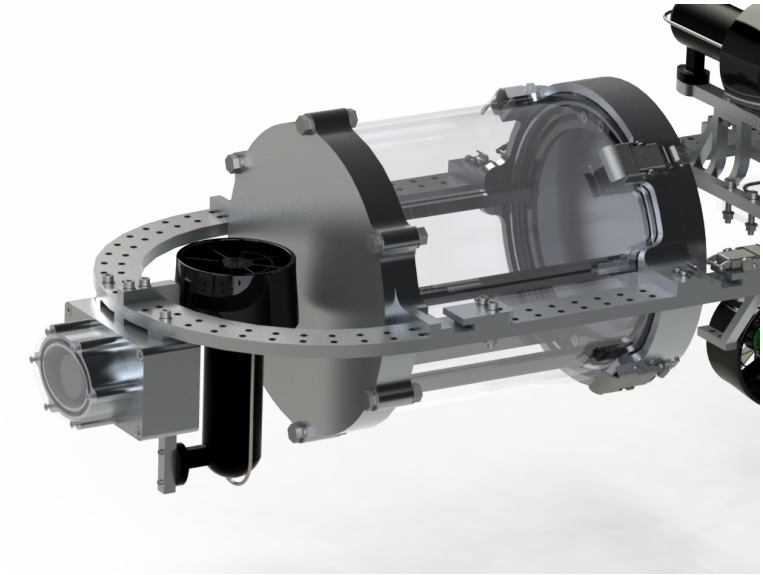

Figure 183: Slide the tube assembly into right position.

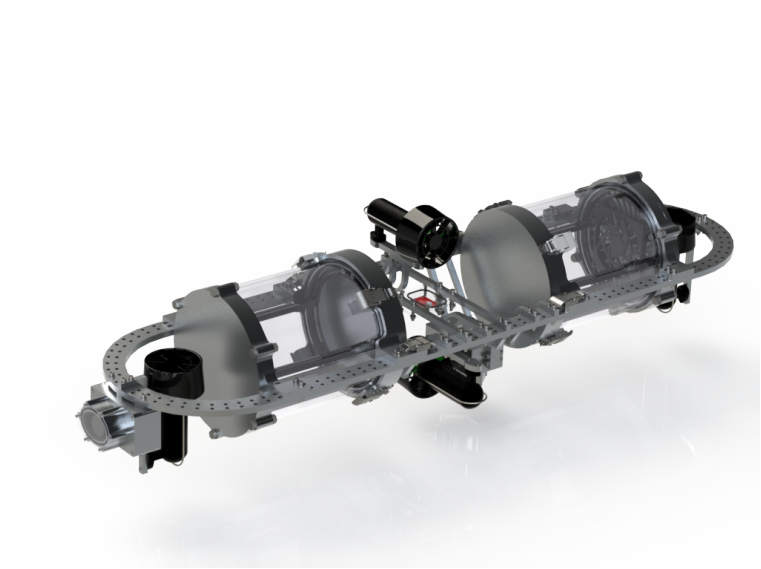

Figure 184: Slide the tube assembly into right position.

H.3.12 Step 12: Attach thruster to mount

The last step is to mount the two horizontal thrusters, see figure [185]. Use the mount [ID 53], thruster [ID 130] and three M3 screws [ID 97].

Figure 185: Attach the thruster mount to the thruster.

Attach this to the frame using four M5 screws [ID 5], see figure [186]. Mount the thrusters as close to the tube holder as possible.

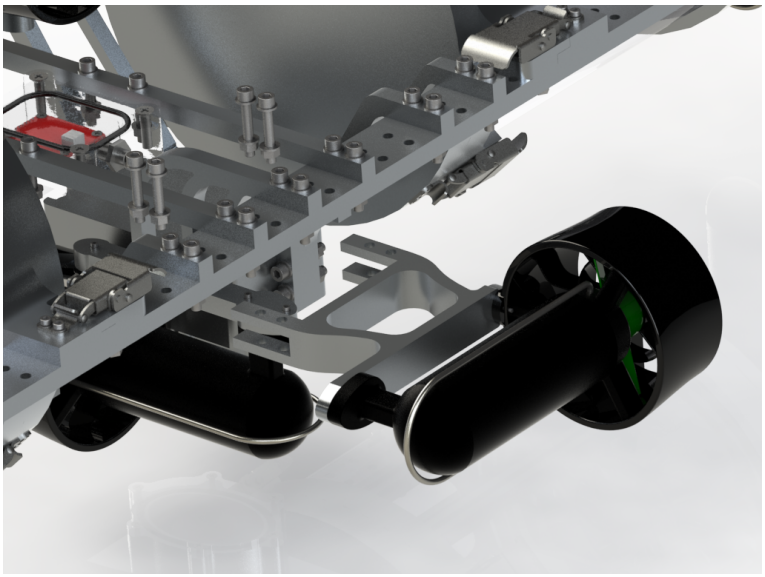

Figure 186: Attach the thruster to the frame.

Now the assemble of this thruster mount is finished, see figure [187].

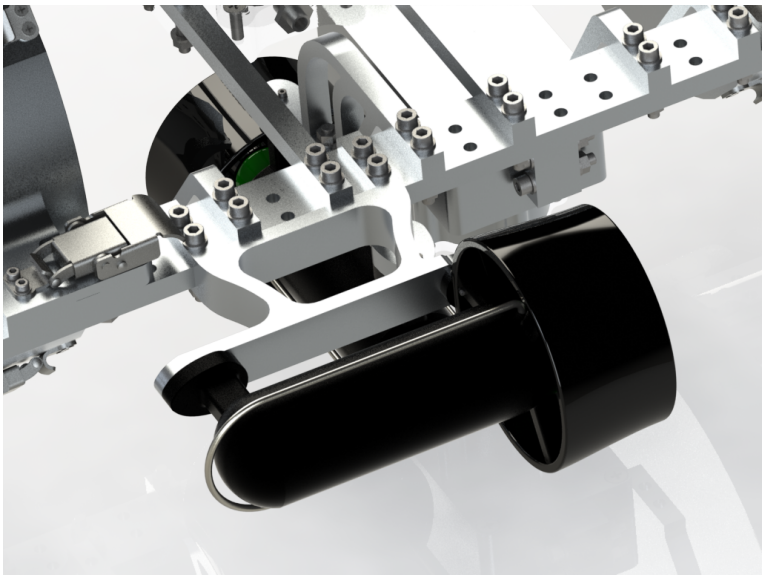

Figure 187: Attached to the frame.

Repeat this step for the other side, see figure [188].

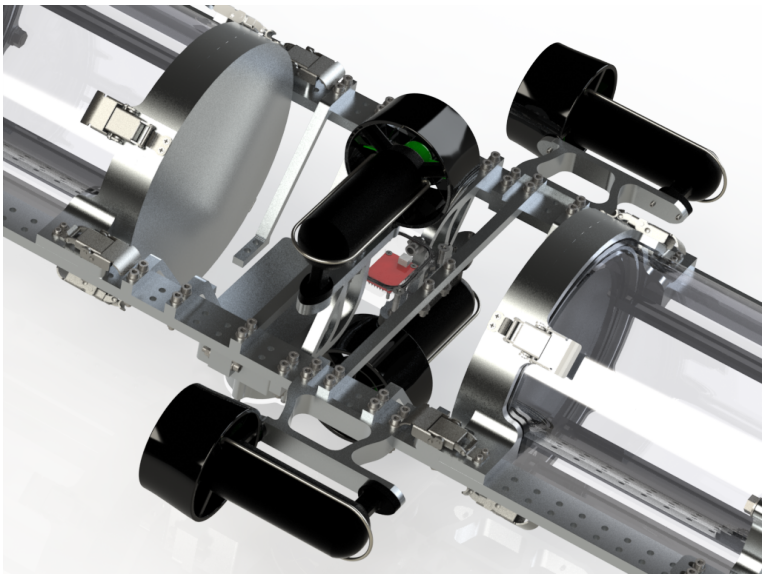

Figure 188: Two horizontal thrusters attached.

Now, the robot is finished, see figure [189].

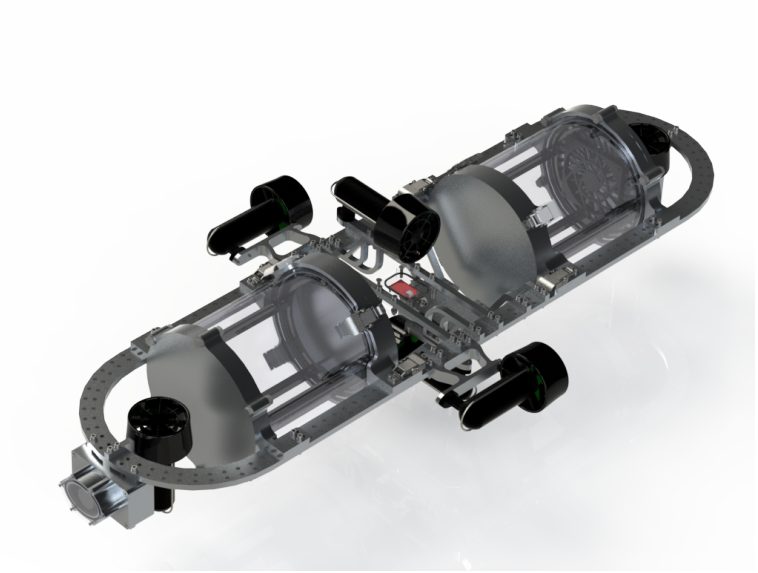

Figure 189: The robot is finished.

I Retailers and sponsors

Company	Contact person	Notes
ÅF	Tommy Klevin	ÅF sponsored the project with financial aids
Würth Electronics	Johan Lindquist	Electronic components & PCB's
Gullbergs Marina AB	Tommy Gullberg	Glued the tubes
Preciform AB	Per-Olof Blomkvist	Sponsored the project with anodizing the aluminum parts
PointGrey	Dorothy Milde	Cameras
Computar		Optics
SeaCon Global	Juan Carlos Braga	External connectors
Stainless Steel Welding HB	Katarina	Waterjet cutting

J Bill of materials

ITEM NO.	PART NUMBER	QTY.
1	Gripper_support	2
2	Claw_gripper	4
3	Gripper_link	4
4	Pole	2
5	Screw_M5_30mm	54
6	Nut_M5	78
7	Washer_M5	168
8	Screw_M5_16mm	4
9	Screw_M5_20mm	12
10	Distance	2
11	Main_body	6
12	Propeller_protection	6
13	Propeller	6
14	Protection	6
15	TG2Z3514FCS-2	2
16	Lense	2
17	Dragonfly2 Board	2
18	CS mount	2
19	Distantance	8
20	Body_back	1
21	Body_front	1
22	O-ring 97x3	2
23	O-ring 59,5x3	2
24	B18.6.7M - M3 x 0.5 x 16 Type I Cross Recessed FHMS --16N	6
25	B18.6.7M - M5 x 0.8 x 20 Type I Cross Recessed FHMS --20N	4
26	B18.6.7M - M3 x 0.5 x 13 Type I Cross Recessed FHMS --13N	8
27	Body_back2	1
28	Body_front2	1
29	B18.6.7M - M3 x 0.5 x 16 Type I Cross Recessed FHMS --16N	6
30	B18.6.7M - M5 x 0.8 x 20 Type I Cross Recessed FHMS --20N	7
31	Marker_pipe	2
32	Marker	2
33	Marker_pin	2
34	Marker_solenoid	2
35	Accelerometer_housing	1
36	Accelerometer	1
37	LLC	1
38	Accelerometer_housing	1
39	G□ngbussning	1
40	B18.6.7M - M5 x 0.8 x 16 Type I Cross Recessed FHMS --16N	1
41	O-ring accelerometer	1
42	B18.6.7M - M3 x 0.5 x 10 Type I Cross Recessed FHMS --10N	4
43	Torpedo_pipe	2
44	Torpedo	2
45	Base plate	2

ITEM NO.	PART NUMBER	QTY.
46	Marker_mount	2
47	Solenoid_mount	2
48	Thruster_moun_front_back_long	1
49	Camera_mount_middle	2
50	Accelerometer_mount	2
51	Frame to tube mount Long	4
52	Frame to tube mount Short	4
53	Thruster_mount_horizontal	2
54	Thruster_moun_front_back	1
55	Thruster_mount_center	2
56	Motormount	2
57	Gripper_mount	4
58	Frame	1
59	Screw_M3_15mm	18
60	Washer_M2	18
61	Flat_head_M5_30	8
62	Screw_M5_25mm	4
63	l4032-833-n-rf_pinne_3x3-5	24
64	el4032-833-n-rf_bygel	12
65	el4032-833-n-rf_platta	12
66	el4032-833-n-rf_grepp	12
67	el4032-833-n-rf_pinne_3x28-5	12
68	Bulkhead	2
69	Nut_and_washer	2
70	In-line connector	2
71	Circular plate outer	2
72	Circular plate inner	2
73	Mounting rod	8
74	PCB mounting part long	12
75	PCB mounting part short	2
76	Mini ITX	1
77	Backbone pcb	1
78	Slot cards	5
79	Fixure	10
80	SSD	1
81	Circular plate guide inner	4
82	Cooling fan body	2
83	Fan blades	2
84	plastic washer	10
85	Plastic washer - 2	2
86	Washer_M4	20
87	Screw_M4_20mm	14
88	M5-nut	16
89	M5x15-bolt	16
90	Nut_M4	16

ITEM NO.	PART NUMBER	QTY.
91	Screw_M4_25mm	2
92	Flange_inner	2
93	Lid_inner	2
94	Flange_outer	2
95	Lid_outer	2
96	PMMA tube	2
97	Screw_M3_10mm	24
98	Washer_M3	24
99	F?ste_excenterl?s	12
100	Washer_M8	24
101	Nut_M8	12
102	Screw_M8_40mm	12
103	O-ring 217x5,7	4
104	Seacon_female_8-16pin	1
105	Seacon_male_8-16pin	2
106	Locking sleeve female	1
107	Locking sleeve male	1
108	Seacon_female_4-24pin	3
109	Seacon_male_4-24pin	18
110	Seacon_female_locking_sleeve_4-24pin	3
111	Seacon_male_locking_sleeve_4-24pin	3
112	Seacon_female_2-12pin	2
113	Seacon_male_2-12pin	12
114	Seacon_female_locking_sleeve_2-12pin	2
115	Seacon_male_locking_sleeve_2-12pin	2
116	Pressure sensor	1
117	PCB mounting part long 4mm-3mm	4
118	floor	1
119	Power board	1
120	Motor controller floor	1
121	PCB1	1
122	PCB2	1
123	Fixure_short	4
124	H-Bridge	6
125	HX-SHCS 0.073-72x0.125x0.125-N	12
126	Narrow FW 0.073	12
127	HX-SHCS 0.138-40x0.188x0.188-N	8
128	Battery holder	2
129	Battery	2
130	Thruster_assembly	6
131	IMU_box	1
132	IMU_lid	1
133	Excenter_lock_assembly	12
134	M3x8	48
135	Fan assembly	2