Robotics project course CDT508/DVA425

PROJECT NAIAD

January 14, 2015

Weronica Kovala
weronica.kovala@hotmail.com
http://www.naiad.se

Hardware
Anette Hilmersson
Lennie Carlén Eriksson
Ande Hana
Patrik Pärlefjord
Martina Öhlund

Software
Jonatan Scharff Willners
Hampus Carlsson
Joakim Gustafsson
Omar Jamal
Nahro Nadir
Wenkai Wang

Supervisor:
Mikael Ekström

Mälardalen University
Västerås, Sweden

Abstract

Naiad is an Autonomous Underwater Vehicle, created by students from Mälardalen University in Västerås, Sweden. The aim for this robot is to, at first, enter a competition in San Diego, USA called RoboSub during the summer of 2015. The long term goal is to clear the Baltic Sea from toxic waist. To do all of this a set of sensors, actuator to complete missions given to it in an autonomous fashion.

This report describes the different parts of the construction and development of Naiad. The report covers all parts of the system, mechanics, electronics and software.

Table of Contents

1 Introduction

This is a report of project Naiad. The name Naiad comes from the Greek mythology where the Naiads were water nymphs that controlled different bodies of fresh water[43]. Somewhat like the Greek Naiads this Autonomous Underwater Vehicle (AUV) will work towards a cleaner and safer underwater environment, both on its own and together with other Naiads as a group.

The long term goal for Naiad is for it to screen, map and maybe even clean the bottom of the Baltic sea from toxic and in other ways hazardous waist. The seas has been seen as a dumping ground for all sorts of things for decades. 23 000 barrels, containing 9 tons of mercury, is buried in the seabed outside of Sundsvall, Sweden [18].

The short term goal is to enter RoboSub 2015. To do this the AUV has to be able to do easy missions based on vision and sensors, using different tools such as torpedoes, markers and a manipulator of some sort. The project members has worked in different groups aiming towards one common goal and the results are a functioning autonomous robot that can complete simple missions.

2 Aim

This project was conducted during the project courses CDT508 (Robotics - project course) and DVA425 (Project in advanced embedded systems). The purpose was to create a fully functioning robot that could compete in the 2015 RoboSub competition in San Diego, USA. To do this a number of things had to be implemented such as a vision system, a set of sensors and actuators as well as some kind of mission plan and control system.

2.1 Objectives

A set of requirements was set by the client. The requirements consisted of three parts, functional requirements, missions for RoboSub 2014 that should be implemented and nonfunctional requirements. The functional requirements were physical things that should be done, the nonfunctional requirements were for example that the software should be layered, with hardware support as the first (lowest) layer and that the documentation was to be written in LaTeX. To test the system and prepare for entering RoboSub 2015 the requirement to perform some of the missions from RoboSub 2014 was added. Since the project was already started, some of the requirements was already fulfilled.

1. Functional requirements
   - 1.1 The robot should be built using a mono-hull.
   - 1.2 The hull should be easy to open and close.
   - 1.3 The cabling should be very efficient and easy to manage.
   - 1.4 A communication cable for remote operation should use either Ethernet or an optical fibre.
   - 1.5 The robot should be able to operate using external power (preferable through an umbilical including optical fibre/Ethernet).
   - 1.6 The umbilical ends in a box that connects to the robot.
   - 1.7 There should be a Kill Switch.
   - 1.8 There should be a Mission Switch.
   - 1.9 There should be indicators like LEDs and/or LCD-display.
   - 1.10 Actuators
     - 1.10.1 Every actuator has its own micro-controller.
     - 1.10.2 All actuators should operate over the CAN-bus.
     - 1.10.3 The motors should preferable be BLDC, but ordinary DC-motors can be used.
     - 1.10.4 Make new own BLDC-driver. One Controller two motors.
     - 1.10.5 There should be two droppable markers.
     - 1.10.6 There should be two unpowered torpedoes.
     - 1.10.7 There should be a simple manipulator that can grip an object and release the object.
     - 1.10.8 A LED-light for cameras.
   - 1.11 Sensors
     - 1.11.1 All low rate sensors should operate over the CAN-bus.
     - 1.11.2 High speed sensors should operate over Ethernet (GIMME-2).
     - 1.11.3 Every sensor has its own compute power.
     - 1.11.4 There should be a pressure sensor.
     - 1.11.5 There should be a sensor for water temperature.
     - 1.11.6 There should be a sensor for salinity in the water.
     - 1.11.7 There should be a speed logger (measures the speed in two directions.

1.11.8 There should be an 9-DOF IMU.

1.11.9 There should be an independent gyroscope for the yaw angle.

1.11.10 There should be an active two dimensional sonar.

1.11.11 There should be a passive two dimensional broadband sonar (20kHz - 40kHz).

1.11.12 There might be a system for finding direction and distance of cooperative robots based on ultrasonic sensors.

1.11.13 There should be two stereo camera systems or one system that fast and accurate can tilt (Two GIMME-2).

1.11.14 The fibre optical gyro shall be installed.

1.12 Communication

1.12.1 The communication system between nodes should use the CAN-bus.

1.12.2 The project should study Space Plug and Play Avionics (SPA) system and implement part of it.

1.12.3 There should be a communication mechanism for robot-robot communication (in case of cooperative robots).

1.13 Software

1.13.1 The firmware (Software for the micro controllers) should be downloaded via the CAN-bus.

1.13.2 The firmware should be handled by a configuration manager.

1.13.3 Compare the latest Vasa-code and Naiad-code, extract the best parts. Document the arguments.

1.14 There should be a simulator for the mechanical and electronic parts.

1.15 There should be a simulator for the software system.

1.15.1 The simulator should be able to handle more than one instance of the robot.

1.15.2 The simulator should handle inter robot communication.

1.15.3 The GUI of the simulator should be used to monitor the actual robot in operation.

2. Missions for Robosub 2014 that shall be implemented.

2.1 Pinger: Go and bring things on the bottom and move.

2.2 Gate: Go through gate with visual servoing.

2.3 Bouy: Identify and hit.

2.4 Bin: Identify and drop marker.

2.5 Torpedo: Find opening and fire torpedo.

3. Nonfunctional requirements

3.1 The programming language should be Ada and using the Ravenscar profile.

3.2 The software should be layered, with hardware support as the first (lowest) layer.

3.3 The software should be based on principles for component based software engineering.

3.4 The documentation should be written using Latex.

3.5 The preferred computer in the system is the GIMME-2.

3.6 The batteries should be easy to change.

3.7 Make a complete test plan.

3 Background

The Naiad project was started by 18 MSc students in the fall of 2013. Their aim was to enter RoboSub 2014 and to create an AUV which could help locate, map and possibly even remove some of the toxic and hazardous waist.

Since this project was already started, a lot of things were already done. When looking at the first list of requirements given from the client, a set of items could be crossed of immediately.

1. Functional requirements
   - 1.1 The robot should be built using a mono-hull.
   - 1.2 The hull should be easy to open and close.
   - 1.3 There should be indicators like LEDs and/or LCD-display.
   - 1.4 Actuators
     - 1.4.1 Every actuator has its own micro-controller.
     - 1.4.2 All actuators should operate over the CAN-bus.
     - 1.4.3 The motors should preferable be BLDC, but ordinary DC-motors can be used.
   - 1.5 Sensors
     - 1.5.1 All low rate sensors should operate over the CAN-bus.
     - 1.5.2 High speed sensors should operate over Ethernet (GIMME-2).
     - 1.5.3 Every sensor has its own compute power.
     - 1.5.4 There should be an 9-DOF IMU.
     - 1.5.5 There should be two stereo camera systems or one system that fast and accurate can tilt (Two GIMME-2).
   - 1.6 Communication
     - 1.6.1 The communication system between nodes should use the CAN-bus.
   - 1.7 Software
   - 1.8 There should be a simulator for the mechanical and electronic parts.
2. Nonfunctional requirements
   - 2.1 The programming language should be Ada and using the Ravenscar profile.
   - 2.2 The preferred computer in the system is the GIMME-2.
   - 2.3 The batteries should be easy to change.

This meant that the team had some things to take in to account when the project started. Since the mono-hull was already made, this meant that the groups had to adapt. Other part of the things that were already done had to be continued. For example, the fact that every sensor was to have its own compute power, the electronics group had to make sure that this was possible.

4 Method

This semester 12 MSc students has been working with the Naiad project, as supposed to the 18 MSc students working with it the previous year. The new group was divided into smaller teams by discussing every members background and interests. The decision fell on five groups, three groups working with software and two with hardware. For the software groups, one was working with the vision system, another with the graphical user interface and the third with the Naiad main control software. The hardware group was divided into two smaller groups where one focused its work on electronics and the other on mechanics. The two software groups was managed by a software manager and the hardware groups by a hardware manager. The hierarchy is shown in fig. 1.

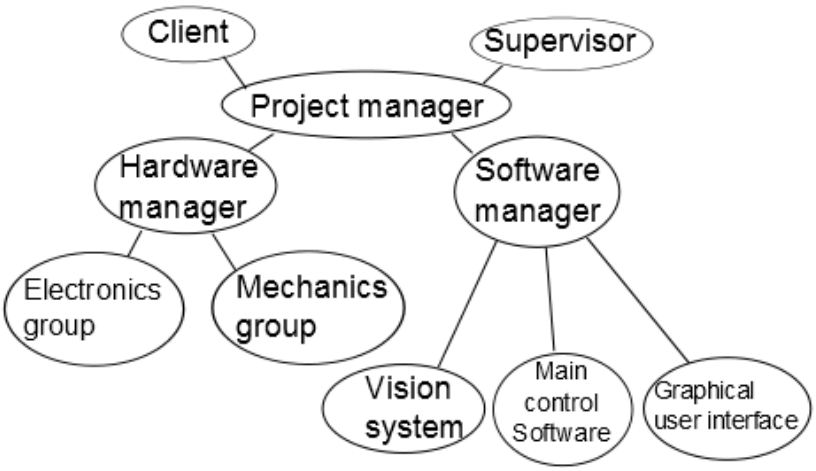


Figure 1: The hierarchy of the project.

The project was managed in a scrum like method where different tasks were handed out using a web-based system called Trello[17]. Here the team members could easily access and see which tasks where available. The tasks where prioritized using a five grade scale, where a low number means a high priority.

1. Crucial, needed for operating in water.
2. Important, needed for maintaining control in water.
3. Necessary, needed for entering RoboSub.
4. Good to have, needed to win RoboSub.
5. Not required, needed for other missions than RoboSub.

All of the requests from the client was prioritized together with the whole group. Files and documents was shared using a cloud service called Dropbox[2].

The project was managed in a way where the team members were trusted to solve the problems in their own way. The project manager worked as a bridge between the groups trying to motivate the team members towards the same goal. The administrative work, as well as most of the sponsorship work, was conducted by the project manager. The group managers did the administrative work of the groups, which included weekly reports to the project manager on the progress in the group as well as prioritizing the work.

Since the project was already started when this group took over, some time was put into getting familiar with the previous work. Some of the previous work had to be redone to better suit the purpose and current goals of Naiad. Almost all of the constructed hardware was kept in its previous state, however changes were made both to software and to some of the hardware designs that were yet to be implemented.

5 The competition

The initial goal for Naiad, after its completion, is to compete in RoboSub 2015. Since the missions are similar from one year to another the tactics and analysis is based on the rules from the rules and missions document of RoboSub 2014.[38]

5.1 Overview

On the competition website it is stated that the goals for the competition is to provide students with the opportunity to experience system engineering, accomplish realistic missions with AUVs and furthermore to build on the relationship between the young engineers and the developers and producers of different autonomous vehicle technologies.

There are two main parts to how the team and robot is judged. From their submitted journal paper, website and movie the team is scored with a subjective score from the judges. The team is scored with a performance measure from their execution of the missions. The sum of these scores result in the final score.

5.2 Subjective judging

The rules state that each vehicle will be subject to a static judging during the competition. The judges evaluate the vehicle's technical merit, safety as well as its craftsmanship. In the subjective and static judgings the following is measured.

- Utility of website
- Technical merit
- Written style
- Technical accomplishments
- Craftsmanship
- Team uniform
- Team Video
- Discretionary static points

All of these can give a total of 400 points.

Weight, size of the AUV and size of the markers and torpedoes will also be measured in the static judging.

5.3 Missions

The following missions are described in the rules of RoboSub 2014 [38].

The team has 15 minutes to perform all missions, finishing before will give bonus points, T minutes left will give $T * 100$ in bonus points.

The first mission is always to pass through the validation gate, if this is done, the AUV is free to continue with the rest of the missions. 100 points is rewarded for passing through the gate, another 150 points is rewarded for maintaining a fixed heading through the validation gate.

5.3.1 Path

The path consists of segments leading the way to the different missions. The mission is to always follow the path to the next mission. The path segments are 0.15 m wide by 1.2 m long sections that are covered in orange duck tape. 100 points per followed segment will be awarded.

5.3.2 Control Panel

In this mission there are three buoys, two of which will be self-illuminated with high power RGB LEDs (RGB-buoy). These RGB-buoys will cycle through each of these three colours every 5 seconds. The aim is to bump these to get a desired colour. The desired colour will be presented at the start of the competition. The first bump at a buoy will stop it from toggling between the different colours, every subsequent bump after that will toggle the buoy through the colors. This mission will award a maximum of 1500 points. 400 points for bumping at least one of the buoys, either one of the RGB-buoys or the solid coloured bouy. For setting one of the RGB-buoys, 700 points will be rewarded. For setting both the RGB-buoys to the desired colour 1500 points will be rewarded.

5.3.3 Maneuvering

The maneuvering mission goal is to pass a PVC-construction. The AUV should either pass over it or around it. The construction for this mission can be found in figure 2. There are two different ways

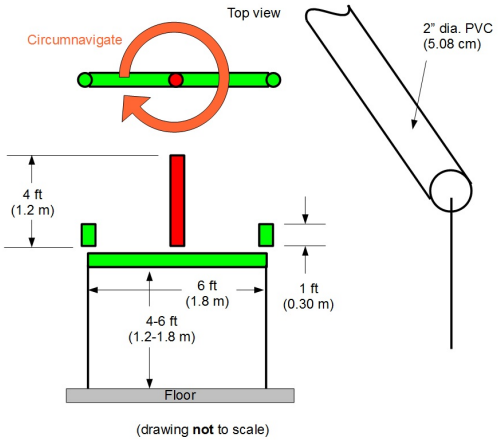


Figure 2: Design of the construction for the maneuvering mission

to score in this mission. One where the AUV should pass over the horisontal section, between the red riser in the center and the green risers on the outer end of the horisontal section. This gives 600 points if more than half of the AUV is passing below the highest point of the center riser. If the AUV passes higher than that, it is awarded with 400 points. The other way is to circumnavigate the center riser. The method and orientation for the circumnavigation is optional for the teams. The AUV may also extend further out than the Green risers. To score using this method, the AUV should do a full 360°rotation around the center riser. If the AUV is able to do so with more than half the vehicle above the top end of the center riser, 1000 points are awarded. If it completes the circumnavigation with at least half the vehicle under the top of the center riser, 1400 points is awarded.

5.3.4 Landing Site

The Landing Site consists of four black boxes placed in a square with white borders. The boxes are 30 cm wide and 60 cm long and each box has a 15 cm wide white border. The different bins will contain different silhouettes, as shown in fig. 3. Only one silhouette from each column will be in the four bins. One of the silhouettes in the boxes will be designated as the primary target and another will be designated as the secondary target. Teams are awarded the most points for dropping one marker in the primary target and one in the secondary, 1400 points per marker. Half the points, 700 points per marker, is awarded for dropping markers in any bin.

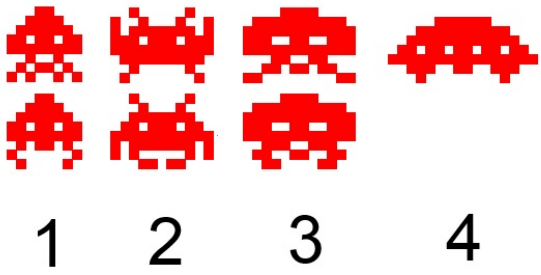


Figure 3: The possible silhouettes for the Landing Site mission

5.3.5 Brunch

The Brunch mission is to fire the two torpedoes into the right holes of a green, square board. The board will have a picture of a spaceship. Above the spaceship there will be four 12.7 cm circular cutouts with 17.8 cm black borders. Below the spaceship two bigger holes can be found, 25.4 cm cutout circles with 30.5 cm black borders around them. During each run, two of the smaller circles and one of the larger will be covered, leaving 3 open holes. The goal is to fire the torpedoes through each of the smaller cutouts, however points will be rewarded for firing through any hole. Getting a torpedo through the big hole gives 700 points per torpedo, firing a torpedo through one of the smaller holes gives 1000 per torpedo. To receive full marks, both the smaller targets has to be hit individually, this award the team 2500 points.

5.3.6 Reroute Power

The Reroute Power mission consists of a 91 cm yellow square with eight blue circles distributed evenly in a square 15 cm in from the border of the square. Four red power pins will be arranged on the eight blue circles so that 4 circles has power pins and 4 are empty. The goal is to remove one pin, place it in an unoccupied blue circle. This will be rewarded with 1000 points per power pin. Just removing one power pin will result in 300 points per power pin. Placing the power pin back in it's previous position provides 700 points per power pin. Maximum score for this mission is 4000 points.

5.3.7 Recovery Area

There are two separate parts to this mission, where both needs to be done consecutively for the team to receive all points.

The constructions for this task are two octagons constructed from PVC pipes, three moon rocks in the colour grey, three cheese structures in the colour green and a sample box. The octagon will be 2.7 m at its widest. The moon rock will be a 15.2 cm cube and the cheese will be a 10.4 cm high. The Collection Site is a 1.2 m square that is 7.6 cm high. There are also two pingers, located directly under the structure holding the Sample boxes, the octagons is floating on the surface directly above this structure. This is called the Recovery area, from which there is a path segment pointing directly at the Collection Site. At the start of each run one of the two pingers will be turned on and the moon rocks and cheeses will be positioned at the Collection Site. The team captain can choose to switch the active pinger, this should then be done when the object has been collected but before the AUV has surfaced.

Points are distributed in three different categories for this mission. For surfacing, the AUV should surface with no portion of itself outside the octagon. Surfacing in the wrong octagon is rewarded 500 points whilst surfacing in the right octagon rewards the team with 2000 points. To receive full marks for recovery, the AUV has to capture the object so that it is constrained in at least three degrees of

freedom when the AUV surfaces. The maximum reward for recovery is 1200 points. Dropping the object should be done from the surface, ability to do so is rewarded with 500 points. For hitting the Sample box with said object, teams are rewarded another 1000 points per object. Ability to put more than one similar object in the same sample box doubles the points, as well as the ability to put objects in both sample boxes.

5.4 Tactics and analysis

The missions were categorised into three different categories; solved, potential and not possible. The idea is that the missions listed as solveable are the ones where only software development is needed at this stage, potential is the ones where we have all the hardware but it is not finalised. The missions listed as not possible are the ones where there is not even a clear solution for the hardware required for this mission. Since there is no design for the manipulator at this moment, the missions requiring this are listed as not possible.

Solveable

- Gate - the gate rewards points for passing through, and additional points for maintaining a fixed heading through it. To make sure to maintain a fixed heading the vision system should be used.
- Control Panel - the control panel can give the team a maximum of 1500 points. For this the requirement is the vision system.
- Maneuvering - The only requirement for this is the vision system.
- Path - The only requirement for the path is the vision system.

Potential

- Landing site - To succeed with this mission both vision system and a system for releasing the markers need to be working.
- Surface - To do this the AUV must be able to find the pinger using the hydrophones.
- Brunch - To do this the vision system and a system for firing torpedoes is needed.

Not possible

- Reroute power - Rerouting power requires both the vision system and the manipulator (gripper).
- Collecting objects - Collecting objects requires both the vision system, the manipulator and the hydrophones to be working.

According to this a rough guesstimate would be that Naiad could receive 2250 points, doing the missions that are solved at the moment. If the missions listed as potential were to be solved before the competition, an additional 3400 points.

6 Mechanics

The main hull was already designed and constructed by the students from the previous year. This meant that a lot of the design decision had already been made. For example the over all shape of the robot and the thruster configuration was decided last year. The tool plate on the bottom of the robot as well as the wings was not constructed. These needed to be designed in detail and manufactured. The features of the tool plate, such as torpedoes, markers and a gripper, also needed to be designed. Several solutions for these have been studied.

6.1 Background

Since the robot is to be submerged, some kind of water protection was required for the electronics. The solution in Naiad is a watertight chamber for the electronics. Also the motors need to be mounted on something. The previous team had constructed the main hull and with that decided how this chamber as well as the thruster configuration were to look.

6.1.1 Main hull

At the beginning of this project, the main hull was already milled and all six motors were mounted. There were also compressing spring latches mounted on the hull for the purpose of attaching the lid of the main hull.

One side of the hull has two differently shaped countersinks which are intended for holding a kill switch and a mission switch respectively. The lid has a window for viewing an LCD display which may show status messages to the user when the robot is operating. The window consists of a clear Polymethylmethacrylate (PMMA) plastic which has an O-ring between itself and the lid to secure a waterproof system.

There is also an O-ring track around the top edge of the hull to further waterproof the system with a custom-made O-ring. Furthermore, there are two camera housings built into the hull, one facing forward and the other facing downward. These camera housings have, like the window on the lid, plates of transparent PMMA so that the surroundings can be observed by the cameras from inside of the hull.

At the bottom of the hull there are four tapped holes for the purpose of being able to have e.g. an Ethernet cable through the hull for manual control of Naiad or for connecting peripherals from outside of the hull.

6.1.2 Thruster configuration

To allow Naiad to move underwater in all three directions x, y and z and to control all three orientations, roll pitch and yaw, six thrusters were mounted. This is the minimal number of thrusters to the six degrees of freedom. The configuration of the thrusters was made by the previous group and it was their belief that this configuration to be more agile and responsive than the configuration Vasa had [12]. However the thrusters were never tested in the water by the previous group.

The previous group, using the Solidwork model, created a matrix of the relative effect of each thruster on each component, this can be seen in figure 5. The matrix were used to convert the thrusters value from the optimal thruster to the real thruster configuration. More information about thruster configuration can be read in Naiad AUV motion control system [37].

6.1.3 Toolplate

The previous group had also designed a prototype of how the design of the tool plate and front tool plate could look like. This design however was not finish to be manufactured. The design suggested also where the pneumatic box and the gripper could be placed, see figure 7 below. There where also

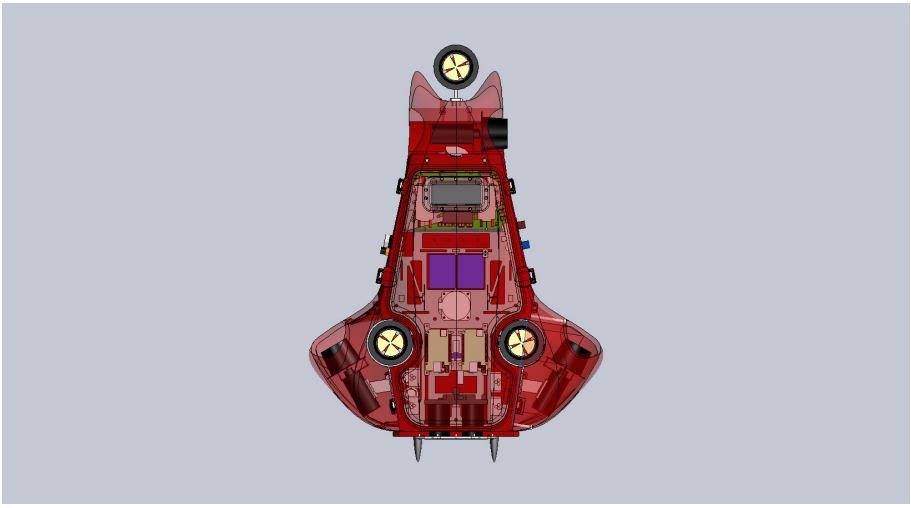


Figure 4: The six thrusters which allows NAIAD AUV to get the six degrees of freedom.

Thr	x	y	z	x^o	y^o	z^o
1	−0.866025	0.5	0	−0.0205	−0.35507	0.248963
2	0	−1	0	0.035	0	0.338
3	0.866025	0.5	0	−0.0205	0.035507	0.292265
4	0	0	1	0.111	−0.095	0
5	0	0	1	−0.025	0.476	0
6	0	0	1	−0.161	−0.095	0

Figure 5: The configuration Matrix. Figure 6 below explains the index.

Thruster	Position
1	Horizontal front right
2	Horizontal back
3	Horizontal front left
4	Vertical front left
5	Vertical front right
6	Vertical back

Figure 6: Thruster indices

a prototype of how the gripper and the pneumatic box could look like, but neither designs were not ready to be manufactured. Both the gripper and the pneumatic box needed to be redesigned.

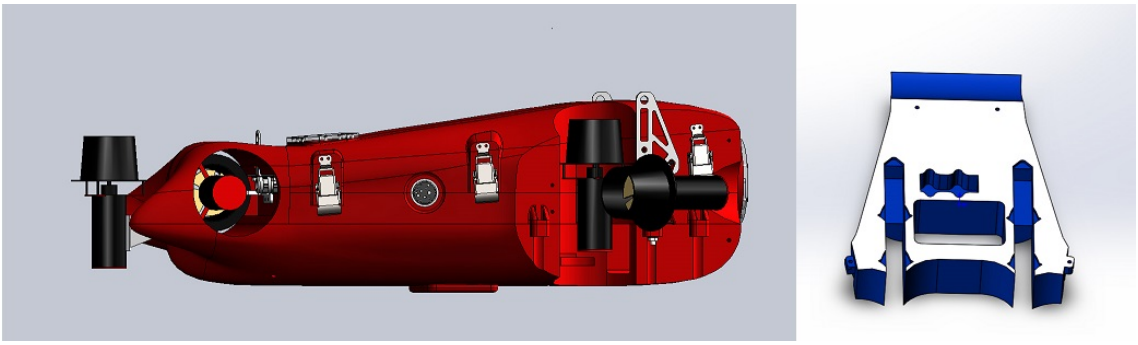


Figure 7: A and B shows the old design of tool plate.

6.2 Equipment

When participating in RoboSub, there is some equipment needed to be present on the AUV (Autonomous Underwater Vehicle) in order to fulfill all the goals of the competition. Naiad will therefore be equipped with two markers, two torpedoes and a pair of grippers. Additionally, there will also be two speed loggers equipped, one mounted on the tool plate and the other on the lid. This section will discuss the design and implementation of each of these tools and how they can be further developed.

6.2.1 Speed logger

Since Naiad has trouble knowing its position in water when there is nothing to navigate after, a speed logger was built. The speed is calculated from a sensor that measures how fast the turbine is rotating in the pipe. The speed logger has seven main parts and three screws. The base plate is formed so that water will easily flow into the pipe where the turbine is mounted. the pipe is built out of a plexi tube and this is the only part except for the screws that is not created with a 3d printer. The pipe exists so that the flow of water is only measured in one direction, else water flowing from the sides might disrupt the speed values in a certain direction. The turbine is built so that it will be as light as possible for it to be able to spin easily and another goal is that no water flow should get passed it without making the turbine spin. The turbine is held in place by the use of an axle connected to a wheel bearing that is mounted in a centerpiece. The centerpiece has three legs so it will be stable and at the same time not disrupt to much of the water flow. it is held in place in the pipe by the use of 3 screws. finally the centerpiece has a cap on the other side from the turbine, this is to make the water flow better and to protect the wheel bearing. The assembly of the speed logger can be seen in fig. 8.

6.2.2 Markers

Naiad should be equipped with two droppable markers. This requirement is for the RoboSub competition. The previous design of the marker assembly consists of two PMMA pipes, one short and one longer, each connected to a top and a bottom plate. The short tube is connected to the top plate on which there are screw holes for attachment to the tool plate. The bottom plate acts as a stabilizing bottom and holder for the other pipe. The design can be seen in Figure 9.

According to the old design they are to be operated by a pneumatic system. However, it was decided that using pneumatics for the markers would be over complicated for the sole purpose of moving a releaser a few millimeters back and forth. Thus, it was considered if it was possible to replace the pneumatics by using solenoids to release the markers instead. It was considered a good idea and the system was therefore replaced. This meant that the design of the marker holder needed to be adjusted as the existing design of the marker holder would be too large and therefore not fit in the tool plate. A new system with solenoids has not yet been tested as it is still in the planning phase.

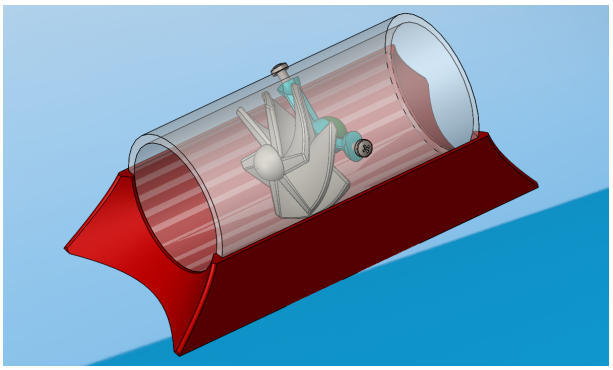


Figure 8: Assembly of the speed logger

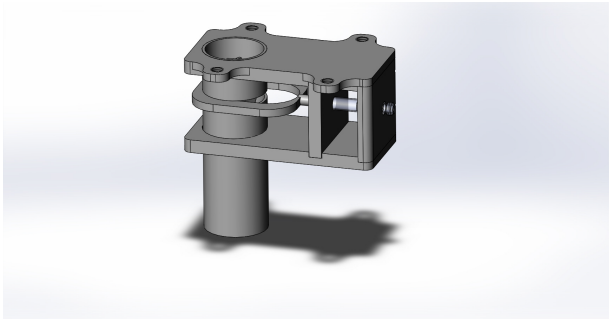


Figure 9: The previous version of the marker assembly.

With this new system for dropping the markers, the markers also had to be modified to fit in. A loop was added to the end of the marker so that it could be hanged onto the solenoid pin. Additionally, the shape of the marker had to be changed as the previous marker consisted of a sphere with fins and this design made no room for adding the loop to the marker. See Figure 10 for the old and the new design respectively.

The markers now work in such a way that they are activated by a solenoid retracting a pin which the marker hangs onto. The marker will then fall down and out of the pipe. The markers should be heavy in the bottom so that they do not float up to the surface or stay still after release. There were a couple of different tests made to see if the marker was heavy enough to sink relatively straight down, with different methods of making it heavy. As this was made late in the project, a solution for making the markers heavy had to include using items that were easily accessible in order to test them in time. The first method was to add some short screws to the bottom of the markers while they were 3D-printed. However, this created air bubbles inside the marker and also the screws did not create a uniform and balanced mass. This lead to that the markers drifted heavily when being dropped into water. Another idea was to make the marker hollow in the bottom and print the marker in two parts, where each part had a hollow part in the shape of half of a sphere. Then, plaster would be added to each part to fill up the holes. They would then be glued together and a layer of glue was added to the whole bottom part of the marker in order to protect the plaster from water. There were two prototypes made of this solution, one marker with fins and another without fins. When tested in water, the marker without fins worked much better. However, It did not move ideally as it still drifted some. To read more about the placement of the markers with their holders, see Section 6.3.3.

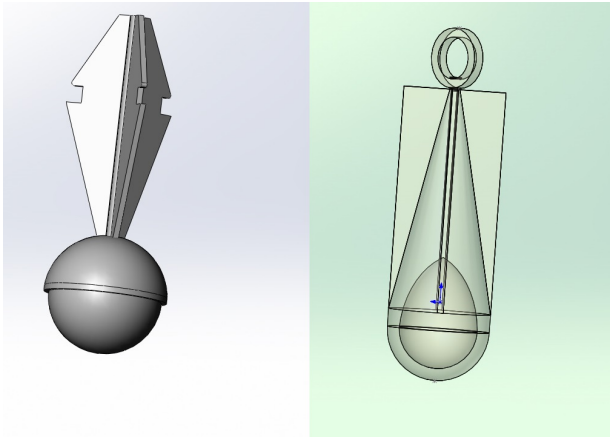
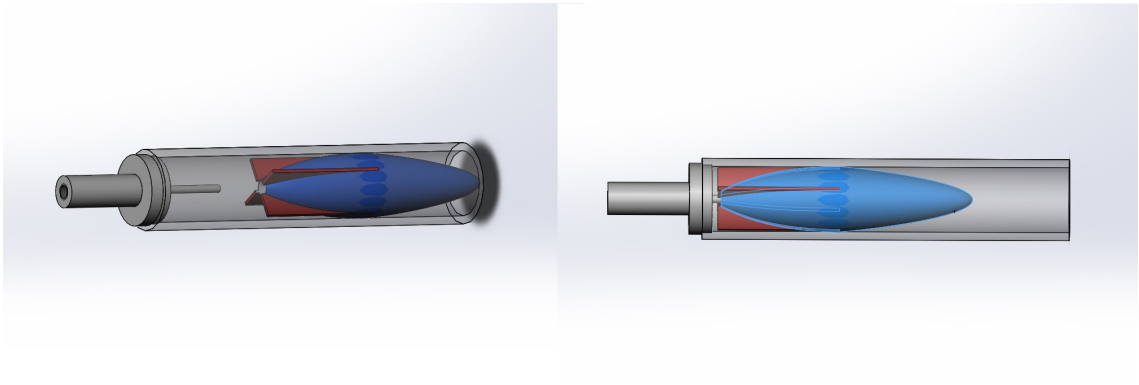


Figure 10: To the left is the first version of the marker and to the right is the second version of the marker.

6.2.3 Torpedeos

The basic idea of the torpedo launcher is that a small tube is attached to a plexitube and the torpedo is mounted inside the plexitube by a small hole behind the torpedo where the small tube comes in. The torpedo is then attached and stable inside the launcher waiting to fire. By a pneumatic system a pressure goes through the small tube to the torpedo which pushes the torpedo away, see the figure 11 below. However, since markers and new gripper will not use the pneumatic system, it is decided to also use solenoid for the torpedoes. The design of a solenoid torpedo have not yet been designed.



k,

Figure 11: The torpedo launcher.

6.3 Tool plate

In the old design the battery box which sticks out from the main hull is visible, see the fig 7. The new design has a bigger tool plate that encloses the battery box, see fig 12 below. There are also room for two markers, gripper, pneumatic box, temperature and pressure sensor.

The redesigning of tool plate was dependent on the gripper and pneumatic design. Since it was decided to make a new gripper than the suggested one and this new gripper didn't have a design, it was difficult to redesign and manufacture the tool plate.

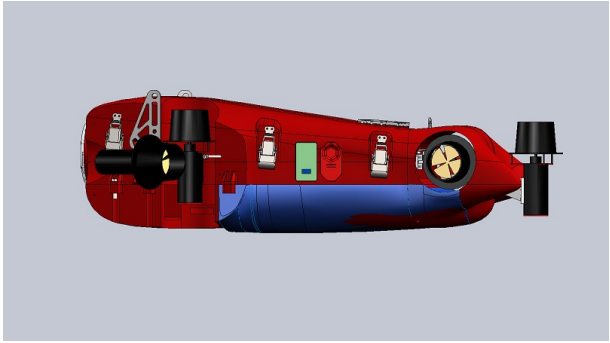


Figure 12: The new toolplate make battery box invisible

The solution was to make a modular design which are independent of the equipment's design. The new design is a plate with many holes which can be attached to the main hull by screws. The tool plate have then been divided into several parts each equipment have it's own part. Each part will then be attach to the tool plate by screwing it too the tool plate holes.

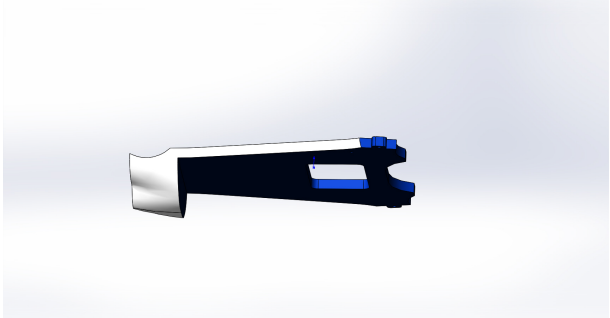


Figure 13: The new design of a modular tool plate

6.3.1 Pneumatics

In the beginning it was decided to use pneumatic system to operate the markers, the torpedoes and the gripper. Since the new gripper is considered to not use the pneumatic system to operate anymore, it's was suggested that the marker and the torpedo could use solenoids instead, this is much less complicated. Therefor for future work it is recommended to look into if the torpedoes could work with solenoid instead. The one thing that could cause a problem using solenoid to operate torpedoes is the space in the front tool plate.

6.3.2 Sensors

Naiad will also be equipped with one pressure sensor and one temperature sensor. Both sensors will be placed in the back of the tool plate. Where the sensors will be attach to are not designed yet. Since Naiad uses a modular tool plate, a box will be designed were both sensors will be attached. The box will then be screwed to the tool plate..

6.3.3 Markers

The markers should preferably be placed near a camera so the system can detect the target for the marker. There are two camera systems, one facing forward and one facing downwards. As the markers are supposed to fall downwards, this leaves one camera system to place the markers near. They can be placed in front of or behind the bottom camera housing. The markers are placed between the front tool plate and the bottom camera housing. This space is limited, but as the back of the tool plate is crowded this placement seemed like a good choice.

6.4 Front tool plate

Different prototypes were designed to select one final improved design of the front tool plate. The idea was to make a simple design which would meet all the requirements and at the same time possible and easy to mill. The requirement was to make room for four headlight LEDs, two torpedoes and one sonar. The design should be modifiable and at the same time easy to mill.

Naiad needed to be submersible and the first front tool plate was redesigned and manufactured to make Naiad submersible for the tests. Since this part was temporary and only needed for the tests, it had to be manufactured with a low cost. Therefore, this part was divided into eight parts to make it possible to use free waste materials. See fig 14 below for the first versiom of the front tool plate.

All parts where glued together and painted to make it waterproof. The design is simple, there are eight empty spaces inside the front tool plate which can be used to balance out the weight of Naiad and make it submersible.

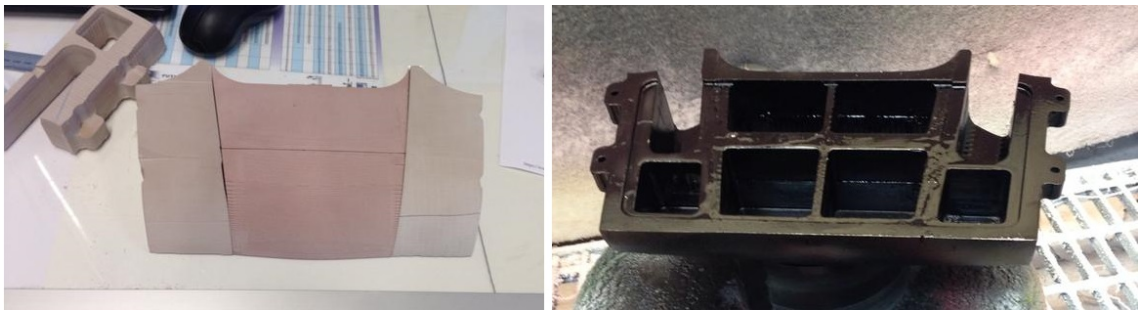


Figure 14: The first version of front tool plate which made Naiad submersible for the tests.

The second version of front tool plate was manufactured by milling as one big part and it's made by plastic. The plastic has a good heat resistance, the impact strength is high, it is high resistance to abrasion and it has 1.25 g/cc density which is much heavier than water [1].

Naiad needed headlights for both cameras, the one facing forward and the one facing to the bottom. In the beginning it was planned and designed so that the headlight for the front camera should be placed in the main hull. However, the LEDs used for headlights easily became very hot and therefore needed a cooling system. This was difficult to provide inside the main hull where all the electronic are placed, so the headlights were moved to be placed in the front tool plate instead which was the only place for headlight which could provide them a cooling system and was close to both of the vision systems at the same time.

Two headhlight boxes will be placed on each side of front tool plate, see the fig 16 above, and one box with two LEDs inside will be attached to the bottom of front tool plate, fig 19 below.

There are two cables coming out from the main hull that have to go through the front tool plate and out to the thrusters. This is why there is an empty space on each side on the back of the front tool plate which gives a free passage to the cables, see fig 17 below.

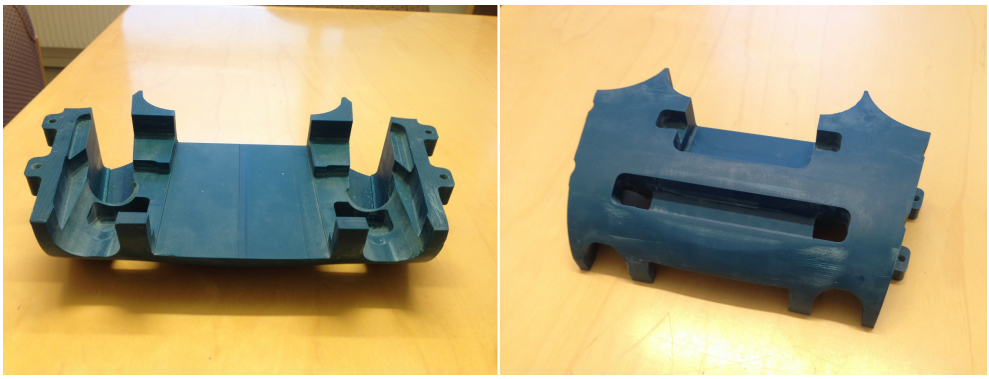


Figure 15: The second manufactured front tool plate.

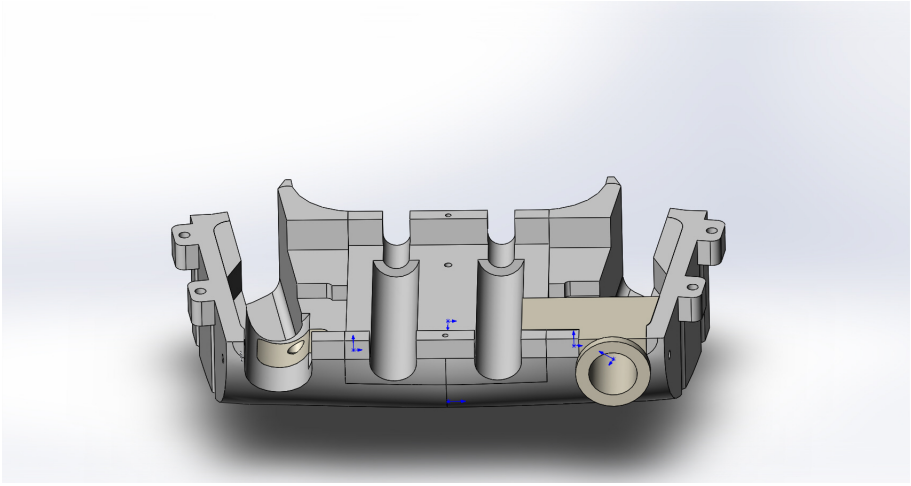


Figure 16: Front tool plate with right highlightbox

The problem with the design was that it was difficult to make room for all the equipment, to fulfill all the equipment requirements and to make it easy at the same time to mill.

Since a sonar will be used to scan the front and the bottom it had to be placed in the front of the front tool plate. To make it possible to scan both front and bottom the sonar needed to be placed with a 25 degree angel facing down.

The requirement for the sonar made it impossible to mill the front tool plate, the 25 degree angel was impossible to mill. The solution was to make a sonar box and by screws attach it to the front tool plate. The box would also make the sonar changeable in case the sonar would not be needed or needed an upgrade. More information about the sonar box read 6.4.2 below.

To make it possible to screw the sonar box to the front tool plate two plugs were designed, milled and glued to the front tool plate, fig 18 below. The plugs were made since they had a 25 degree, like the sonar box, which was impossible to mill together with the front tool plate as one whole part. One of the plugs is designed to give path to the cables which comes out from the sonar box. The cable from the sonar box will come out through the empty space behind the front headlights and go in through the tool plate into the main hull. All the cables can go the same way into the main hull. However, depending on which system is used to fire torpedoes a different path can be used. For more information about how the torpedo launcher works read 6.2.3 below. There is an empty space around

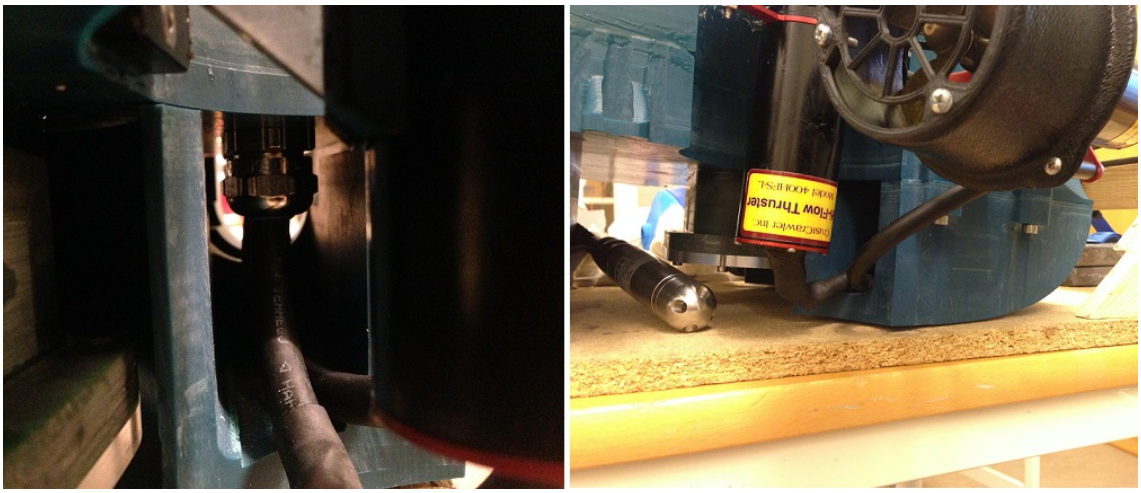


Figure 17: Front tool plate with right highlightbox

the cameras which makes it possible for wires to go through into the tool plate.

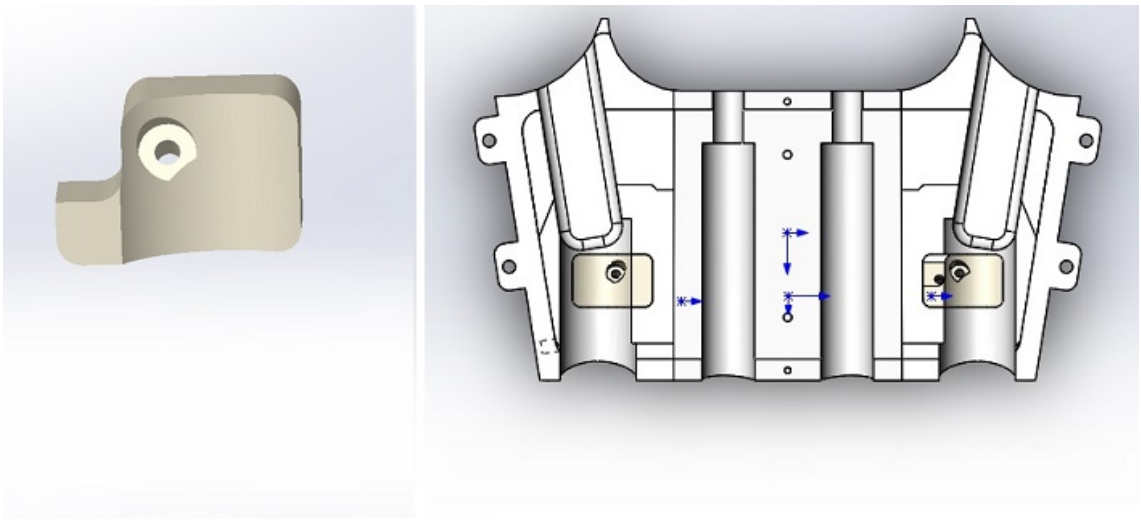


Figure 18: The first picture shows the plug and the second shows where the plugs are glued

One of the Robosub requirements is to have a torpedo launcher which should fire at a target in the competition. Naiad will have two torpedoes placed in the middle of the front tool plate close to the front camera system, more information about how the torpedoes works read 6.2.3 above.

6.4.1 Headlights

It was a challenge to make a good design of the headlight boxes. The design needed to fulfill the requirements and at the same time be easy to manufacture. The basic idea of the headlight boxes is to make it easy to change the LEDs in the future, but the most important was to provide it with a cooling system. The design is simple, a proxiglass is attached by screws to the front of the box which are waterproofed by an O-ring (rubber) inbetween. The glass is then easily removable which makes

LEDs changeable.

The headlight box could be manufactured in two different ways. If it is made by plastic then the bottom, which the LEDs will be attach to, should be made by aluminum. The other idea is to make the whole box made by aluminum, which will cost more but is much more effective of reducing the heat.

Each of the front boxes was divided into two parts, milled separately and glued together to one whole part.

The cooling system for the bottom box is simply that the whole box is made by aluminum and the empty space between the box and the front tool plate will provide a water flow around the box. The cooling system for the front boxes are much more simple, it is based on the same idea that the whole boxes are made by aluminum and there is an empty space behind which provides a water flow.

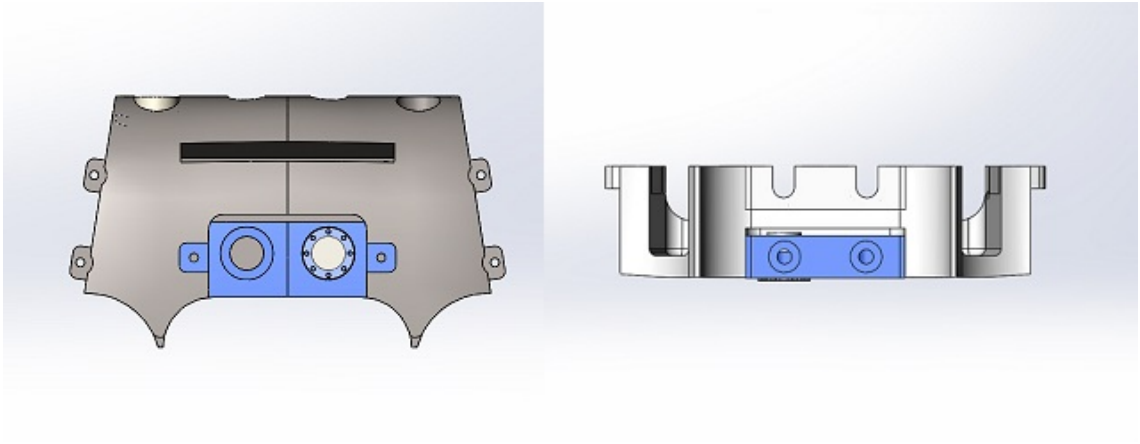


Figure 19: Bottom highlight box

6.4.2 Sonar

Many different prototypes were made to make it possible to mill. The final solution was a box with a 90 degree angel in one side and a 115 degree in the other the side. On each side of the boxes there is one hole for the screws and one side has one more hole for the sonar cable. The box was milled and sent to Deep Vision to attach the sonar to it and to make it waterproof.

Figure 20: The sonar box, picture one shows sonar attached to the box.

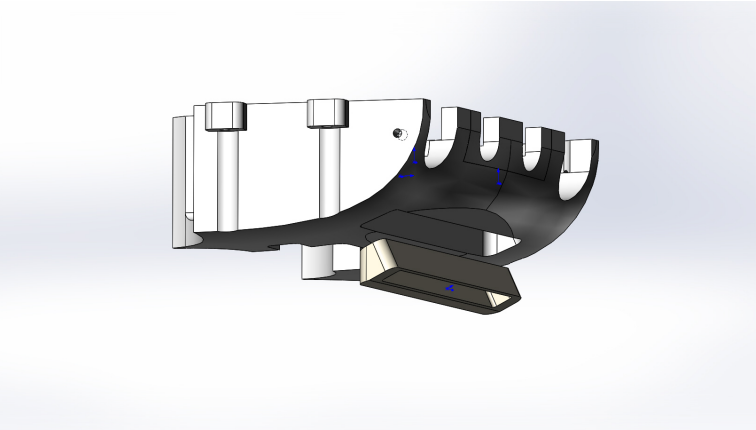


Figure 21: Shows where the sonar box are attached.

6.4.3 Torpedoes

Since torpedoes was only required for the Robosub competition and may not be needed in the future use, the design of both the front tool plate and torpedo launcher should be modifiable. Therefore a torpedo launcher box was designed and a part of it was manufactured. The torpedo launcher will be attached to the box and the box will be attached by screws to the front tool plate.

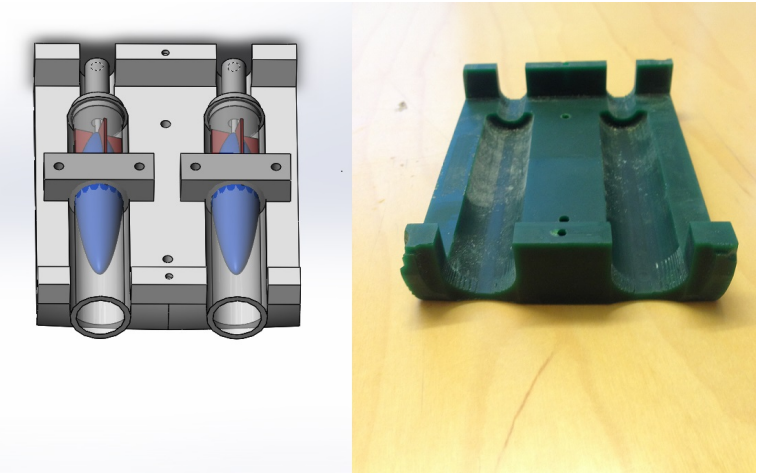


Figure 22: The torpedo launcher box.

The torpedo box is also made by the free waste material and the box is a prototype for using the pneumatic system torpedo launcher, see figure 22.

6.5 Wings

Naiad has a pair of wings which are located on the sides of the front. These serve the purpose of hiding the thrusters and to ensure that Naiad moves nicely in the water. According to the thruster specifications, the thrusters need to be re-greased after each use. This means that the wings also need to be removed from the hull after every use, as they cover the screws that needs to be removed for

re-greasing. With this in mind, the wings were redesigned with the intent of making them as easy as possible to remove while not harming the hull. Many possible solutions were discussed and almost every idea included splitting the wings into a top and a bottom part, but had different methods of how to secure the wings to the hull.

One design involved using magnets for attaching the wings to the hull and also using a magnet to attach the two wing parts together. This would make it very easy to remove and reattach the wings. Also, if using the right magnets, they would be able to keep the wings firmly to the hull and not risk them falling off. Furthermore, there would be no need to drill holes in the hull for attaching threads, reducing the risk of breaking the hull in the form of cracks in the plastic or the point force from the weight of the wings on a few small areas of the hull.

Another solution was to attach the wing parts with screws onto the hull by the use of threads and having a screw or magnet for attaching the two parts together. The bottom part would then be secured to the hull and only the top part would be removed when re-greasing.

The latter solution was also the final solution, where the two parts would be connected to eachother with magnets to make it easier to remove and attach the top part.

6.5.1 Hydrophones

Naiad will use four hydrophones. Two of them will be placed on each wing, one in the right wing and the other one on the left, and the other two will be placed in the back of the tool plate.

The hydrophone will be screwed to a box, see figure 23 below, and then the box will be screwed to the bottom part of the wing. The idea is to make the hydrophones easy to remove and easy to mount.

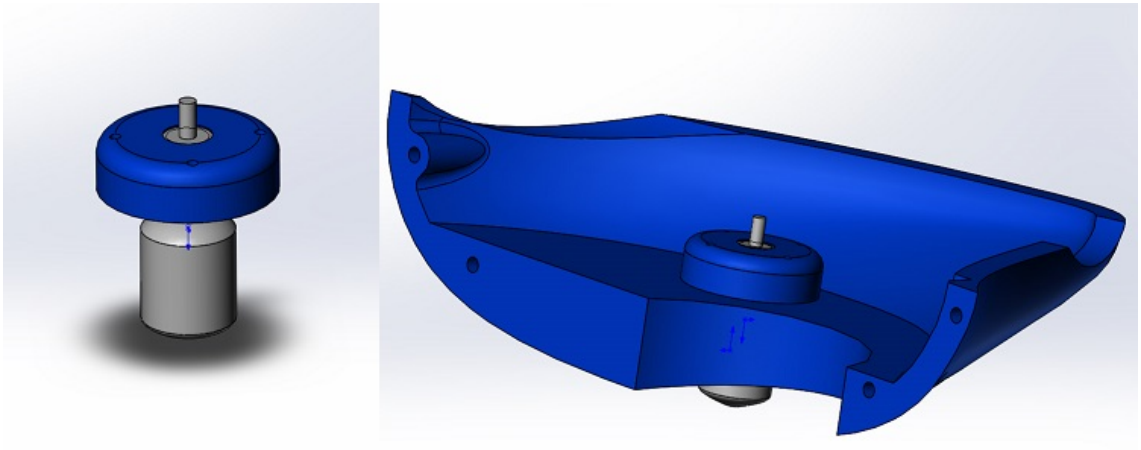


Figure 23: The first picture shows the hydrophone attached to the box. The second picture shows the box attached to the bottom part of the wing.

6.5.2 Coloured status LEDs

It was decided that LEDs should be added to the wings of Naiad for the purpose of showing different status of what is going on by using colour codes. The final design consists of a box which fits a PMMA plastic tubes with LEDs in them. This box is integrated into the wings and the top part has the same shape as the wing. Figure 24 shows the LED box itself and Figure 25 shows the box integrated into the wing.

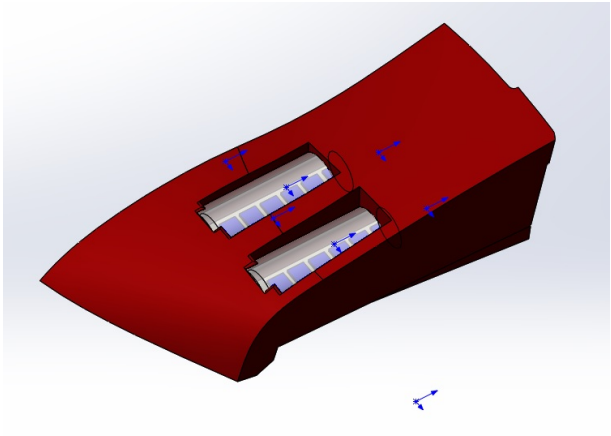


Figure 24: The LED box separated from the wing.

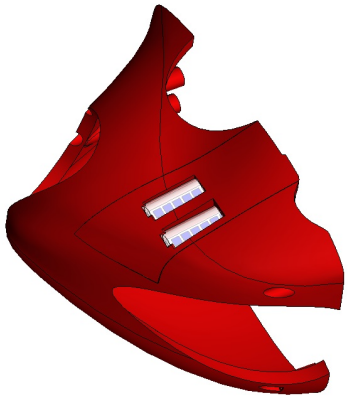


Figure 25: The LED box merged with the wing.

6.6 Safety

In this section the safety aspect of the mechanical part in this project will be discussed. As safety is an important factor in any situation, especially when constructing an underwater vehicle, a safe system has been adapted in order to protect those who operate together with Naiad. Safety can always be improved, but in this project it has been taken as far as time has allowed and the safety which was already present when taking over the project has been further improved.

6.6.1 Thruster safety nets

The thrusters that are used for Naiad do originally not have safety nets covering the openings. This results in an increased risk of injury, e.g. by accidentally grabbing hold of the thrusters and placing one's fingers inside of the opening when manually maneuvering the underwater vehicle. It was therefore decided that safety nets should be designed and 3D-printed for mounting on the front and rear openings of the thrusters.

Several designs were 3D-printed and tested, with varying thickness and width of the net structure. The design which finally was used can be seen in Figure 26. However, after a couple of underwater

tests it was discovered that some of the safety nets had broken and seemed fragile so the choice of material was questioned. The safety nets are printed in Acrylonitrile butadiene styrene (ABS) plastic, which becomes more fragile when exposed to UV-light, but there should be no major harm to the material by exposing it to water. Then, it was thought that the breakage could have been due to the chlorine in the water. This was tested by adding chlorine to a pot of water and letting a safety net lay in the chlorine mixed water for about a week. There were no major changes to the piece. With this in mind, the safety nets most likely broke because they were too weak in the construction or they were pressed onto when being handled.

The design of the safety nets was changed after this discovery and the new design, which can be seen in Figure 27, includes a locking mechanism which is intended to hold the thruster cap in place much better and remove the need of having a nut on the screws to secure the safety nets to the thrusters. The first prototype of the nets was slightly too tight, resulting in that the net stuck to the thruster without the need of any screws at all. It was decided that this method worked well and was therefore adapted for the time being.

This time the safety nets were printed in polylactide (PLA) plastic instead of ABS. PLA is supposed to be more fragile to water, but a piece of PLA was tested in the chlorine water at the same time as the ABS and, like the ABS, the PLA did not undergo any major changes.

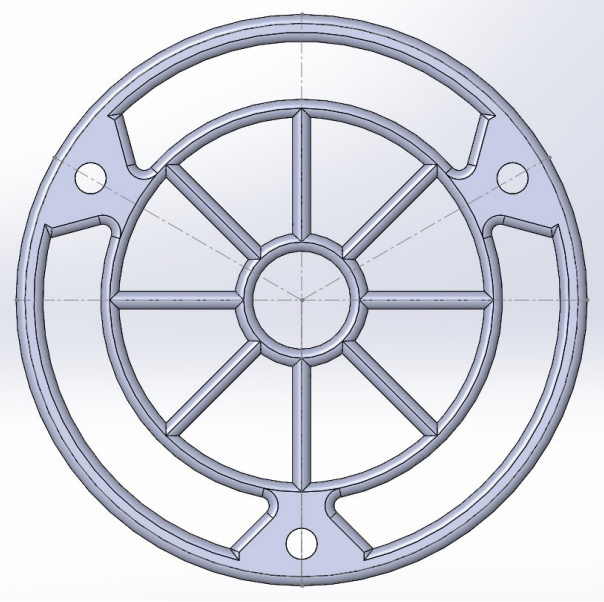


Figure 26: The safety net used during most of the project.

6.6.2 Kill switch and mission switch

The kill and mission switches add safety to the system. In case of emergency, the kill switch will shut down the motors when being removed from the hull. The mission switch makes sure that Naiad does not start a mission without the switch being activated. If this was not the case, it could lead to unexpected movement and as a result cause injuries on people around Naiad or damages to Naiad itself.

The kill -and mission switches were redesigned as the previous version turned out not to work with the placement of the magnets. Depending on the size of the magnets, they were either too weak to interact with the hall sensor or interfering with the hall sensor too much as the magnets for attaching the switches to the hull and the ones for reacting with the hall sensor were too close to each other.

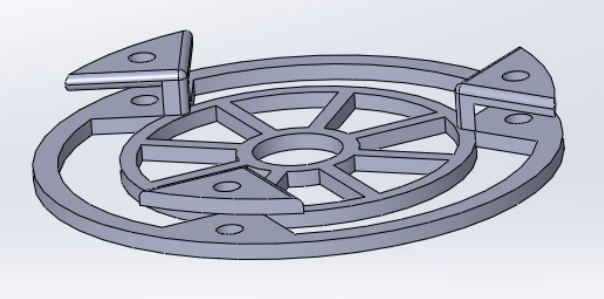


Figure 27: The latest safety net with locking device.

7 Electronics

To support the processing power of the robot as well as the sensors, surrounding electronics need to be in place. Some of these were created by the previous group and this project has continued on those platforms. Below are some of the cards that this group has continued on.

7.1 Background

When the project started this year a lot of the fundamental electronics had already been created. This meant that the overall electronic design was decided as well as thrusters and motor-drivers. The power supply was complete as well as the base of the Controller Area Network (CAN) bus. The Microcontroller unit (MCU) AT90CAN128 was decided as a CAN-bus control unit and stacks to mount and connect the cards was complete. There are a lot of redundancy in the system, for example, each CAN-card transforms from 24 to 5V. An explanation for this from the previous team was to increase robustness in the system. If a leakage occurs and the 5V on the power board and 24V is shortened the CAN-card will not burn. Each CAN-card can handle 12.6-48V making them less sensitive to spikes in the system than if one 5V source would go straight in to all of the MCUs.

7.1.1 Generic CAN

From the previous team, a generic CAN-card was created that has the version number 4.4. This card has nine analogue and eight digital IO-channels. As well as SPI, two UART channels and CAN-bus interface. The card is equipped with a AT90CAN128 MCU giving each card its own computing power. This makes it possible for the card to interpreted CAN-messages, make computation and give out an response. There was a requirement from last year that all of the sensors were to have its own computing power. This requirement was removed during this project. For the most part the card is fully functional. There are some bugs still though. For example the footprint for the 100nF electrolyte capacitor to the left of the switched DC-DC converter is backwards. This can be seen in fig 28

7.1.2 Thrusters

The previous team had decided to use the Crustcrawler 400HFS thrusters[40]. They are specially designed for AUV and ROV applications, and are, compared to their size, quite powerful. We had no reason to question this decision since they were already mounted and running. The motors were controlled by the Phoenix Ice2 HV 60 motor controller from Castle Creations [34]. The drivers, originally designed for RC air crafts, were bought with custom built firmware, designed to be able to instantly switch rotational direction. The previous team had not been able to get the special firmware settings to work.

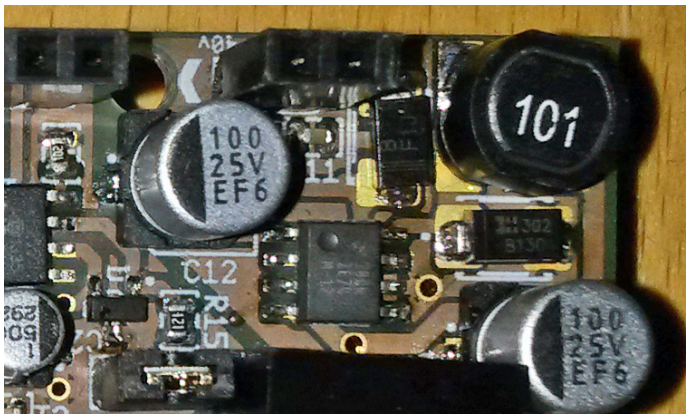


Figure 28: The capacitor marked C12 have an incorrect footprint, the footprint is backwards compared to the shoe on the capacitor.

7.1.3 Stack

To connect and support the CAN-cards a board that makes it possible to stack these on top of each other was created last year. These holders could hold three cards and the cards was soldered to the stacks. This created some problems with fitting add-on cards. Also if a card would break it was difficult to replace unless all three cards was replaced.

7.1.4 Power supply unit

A power supply unit (PSU) existed, which was created to supply the different parts of the system from a single source. This card have not been changed during this project. A CAN-card on the PSU can control the supply to different parts of the system. The only thing that can not be control entirely by the software on the CAN card is the supply to the motors. For the thrusters to be supplied the they need to be enabled by the card but the kill-switch also need to activated. The kill-switch consists of two physical pins than need to be connected in order to supply the motors. This switch can not be overwritten or controlled in software but are build in to the board. The kill-switch controls the power to the thrusters as well as the 12V output but *not* the rest of the system. The CAN-bus and 5V output can be started even if these pins are not connected. There are a few patches on the printed card from Würth electronics. These patches have been added to the schematics but no board design has been made. The schematics is backward constructed from the physical card since proper documentation was absent from the previous group. Even though it has been proofed several times there might still be errors it it. To read more on how the card works see Appendix E. The schematics can be found in Appendix A.

7.1.5 Inertial Navigation System board

When the previous group was working on Naiad the design of the Inertial Navigation System (INS) board was started. A design was proposed for the Fiber Optical Gyroscope (FOG) and was built but not tested. During the summer there was another electronics project where one person had as a mission to test the INS board and discovered that it did not work. The design was not good enough because the circuit was not complete and instead worked as an current generator. A new design was proposed during the summer project and a lot of tips for improving the card was proposed. The proposed board was not built so the design ideas could not be tested. During this years project the ideas was taken into account. Improvements was then made and a new card was built and have been undergoing testing.

7.2 Generic CAN

The generic CAN-cards from the previous group can handle several kinds of signals. This is a good base but to effectively use the cards features additional peripheral electronics are required. Therefore shields or add-on cards for these CAN-cards to add functionality to them has been created. These add-on cards are discussed in more detail in their separate sections. In the beginning it was sufficient to only do these add-on cards to the old CAN cards. When it was decided to save space by controlling several motor controllers on one card, it was discovered that the PWM signals on the old card was insufficient both in number and in resolution. It was therefore decided to make a new generation of the card. This time with six high quality PWM signals since the MCU could provide this but at the moment these outputs were not connected.

To speed up the process of creating this card, as much as possible was kept from version 4.4. At the time the card was redone, no schematics of the card was available but the board-design file was accessible. The required traces were simply re-drawn and the rest left as is. The user-LED, a LED that can be controlled in software, was occupying one of the high quality PWM signals from the MCU. It was therefore moved from MCU pin 6 to pin 14. After miscommunication in the group it was believed that the SPI interface were not used and was therefore removed in version 5.0.

It is believed that it is possible to keep the SPI in future generations. By moving the two PWM signals that are currently located where the SPI used to be in 4.4 to the other side of the 5V (where the M_PWM are located on version 4.4). This would require the motor extension board and the LED-control board to be modified as well. If this was done the same card should be able to be used on all locations of the system adding to the robustness of the system and require less spare parts to be brought to San Diego. At the moment both version 4.4 and version 5.0 are in the system.

7.3 Stacks

The stacks from the previous group were replaced with stacks that had connectors on them to increase flexibility. The connector used is a four pin bottom entry socket header from Würth electronics. The headers are placed on the back of the stack and the CAN-cards need to be inserted *though* the card and then the header. Other wise the connections wont line up properly. The break away headers on the CAN-card need to be angled and longer than standard. The pins need to go through both card and connector and show on the back on order to make a secure fit. These stacks have 4 levels. Not all are intended to be used at the same time. This is due to size and space. There is nothing electronic that limits that all connections are used at once. The levels are there as redundancy as well as making it more versatile. If extra stability is decried the two pins to next to the power supply are not connected on any level and can be used as extra support.

7.4 INS board

The INS board is powered and delivers navigational data to a CAN-card. It delivers the data that the Inertial Measuring Unit(IMU) measures through a USART interface to the CAN card. It also measure values from the Fiber Optical Gyroscope(FOG) and sends it from the INS board to the CAN-card through a SPI interface. The INS board also offer the FOG the safe and stable power levels it requires. The FOG is very sensitive to voltage levels, especially when it comes to the 5V, 12V & -12V lines, there are safety solutions implemented. As an example, to start the 5V three requirements needs to be fulfilled. A digital pin needs to be set high through software and the 12V & -12V lines need to be active. This is because the FOG might break if the 5V power is on without $\pm$12V being active. When these requirements are fulfilled the FOG is active and the INS board can start supplying data to the CAN-card.

7.5 LED controller

The LED controller board is made to control all the lights. Naiad will have two sets of light containing two regular and one ir-headlight. One pointing forward and one downward. It will also have two RGB LEDs on each wing to make it easier to understand decisions made by Naiad. The LED card has two power supplies. One from the PSU powering the LEDs and one from the CAN-card powering the logical circuit. The RGB LEDs on the wings is controlled by a CAN-card through a SPI interface. To activate an LED 16 bits needs to be sent. The headlights is controlled by a Pulse Width Modulated(PWM) signal. One PWM signal for the headlights facing forward and one for the headlights facing downwards. The duty cycle of the PWM signal determines the brightness of the headlights. The experimenal LED-controller board can be seen in 29.

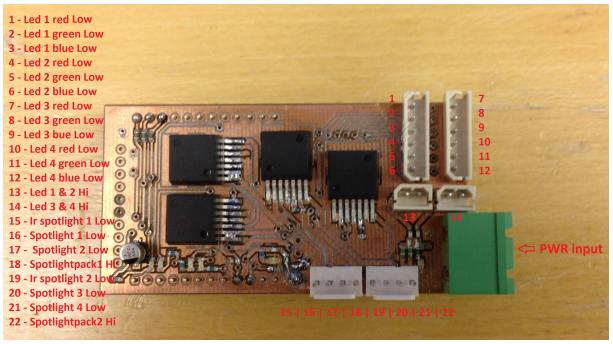


Figure 29: Picture of the experimental LED controller board

7.6 Hydrophone

One of the missions in the RoboSub competition is to locate the resurfacing point. This location is marked with an acoustic pinger, located at the bottom of the pool underneath the exit site. The pinger sends out a 4 millisecond ultrasonic beep every 2 seconds [38]. To determine the location of the pinger, four hydrophones are used to determine the direction the sound is coming from. By comparing the differences in time when the hydrophones hears the ping, a direction towards the source can be calculated. The reason for four hydrophones is that, to be able to locate a source in the 3 dimensions, observation of the signal will also be needed in 3 dimensions. With three hydrophones you would be able to get the the angle to the object in the plane, but not know how far off the plane the source is. This setup might have worked in the competition, with the pinger located on the bottom. Since the Naiad is built for a more general purpose, an exact direction in three dimensions is more usable in future applications.

The sound from the pinger needs to be drastically amplified in order for the MCU to handle the signal.The water is not a silent place, so the signal has to be filtered in order to pick up the desired frequency. This is done by the hydrophone shield card, which picks up the signals from the four hydrophones. The card has three main tasks:

- Phantom power
  The hydrophones gives a much more desirable signal if they are supplied with a constant voltage. This is a done by a voltage division and a big capacitor for stabilization. This power supply can be used for all four hydrophones.

- Filter and amplification
  The signal runs through a passive high pass filter with a tunable cut off frequency. It is able to filter out frequencies below 10-50 kHz. After the filtering the signal is amplified in two stages.

After an initial amplification the signal is split. One unaltered signal is buffered. Another signal is used as reference voltage for the last amplification stage. These signals are fed as input to the last amplification, where the difference between these two signals are amplified almost 200 000 times to get a clear input to the mcu .

- Digitalization and voltage restriction
  After the final amplification, the signal is smoothend to a digital high by a capacitor and picked up by one of the interrupt pins on the CAN-card.

No equipment for testing the circuit properly has been available. The pinger in the lab is limited to 20kHz and the circuit has worked at close range. No open water test has been conducted. The motors rattle and make a lot of loud noises. Since these sounds origins from a source very close to the hydrophones and the sounds are built from a wide range of frequencies, the sound is almost impossible to filter away, at least with the analog filter approach. Therefor, the only way to pick up the sound from the pinger is to turn the thrusters off and listen, whilst dead in the water.

A digital filter was ruled out due to lack of funds, time and hardware with high enough sampling frequency. The digital filter, with way sharper edges, may have been able to listen for the pings and completely remove any noise from the motors.

7.7 Speed-logger

The purpose and mechanical design of the speed logger can be found in section 6.2.1. The electronics parts of it is divided to an outer sensor and an inner analysing circuit.

7.7.1 Sensor

The sensor is based on two Allegro A1301 hall sensors. They translates differences in the magnetic field to analogue voltages. These are placed side by side with 5 mm spacing on a small PCB near the rotating turbine. The propeller has magnets mounted on its blades, and by reading the timing differences between the sensors, rotational direction and speed can be determined.

7.7.2 Board

The signal from from the sensors needs to be altered in order to work as an interrupt edge for the responsible CAN-card. The IC circuits on the sensor has a default voltage of half of the supply voltage. To get an accurate reference voltage, a third A1301 sensor is used inside the hull as a reference voltage. The difference between the inside and outside sensors is amplified in order to trigger the interrupt on the MCU.

7.8 Motor configurations

7.8.1 Motor extension board

The motor extensions board is in it essence an adapter. It provides the motor controller with 5V and GND from the CAN card as well as a PWM signal in a suitable connector. It has six three pin WR-WTB connectors from Würth electronics, one for each motor controller. These were chosen since they prevent the user from connecting the cable upside down and is also has a locking mechanism. The current version of the board is made to be used with the 5.0 CAN-card.

7.8.2 Motor controllers

The controllers was configured with the Castle Link V3.52.10 software. The reason the previous team had trouble with it was that since the firmware was altered from the supplier, some of the files for the setup program had to be replaced. The software is able to configure a number of thruster settings,

one of which is the reverse type. The ”crawler reverse” setting enables the motors to rapidly change rotational direction in order to make the AUV more agile.

7.9 Remote control

When the lid is closed and the Ethernet cable is not plugged in to Naiad there is no easy way to change settings. The remote controller is made to work on land. With the remote control one can navigate and change settings presented on the LCD display. The remote has 4 buttons and one slide switch. The four buttons is for navigating through the menu and the switch is for turning the remote on or off. Since there are no power saving function built in, the batteries in the remote controller will only last for 24 hours unless the switch is turned off after use. The remote uses a commercially bought transmitter and receiver circuit for sending data on the 433MHz FM band.

7.10 Actuator board

The actuator board is a board made to control the solenoids for the markers and the torpedoes. The board can have one or two inputs and four outputs. One input can power either two or four actuators depending if the two pins located below the fuses named pwr_bridge is bridged. The power inputs can be at any level between 0V to 24V. The actuator board does not have any current regulators for the outputs, this is because each output will only be activated for a short time, when the markers get dropped or torpedoes shot. The solenoids that the outputs will power will also be submerged in water, so the heat will not be a problem. Each solenoids is controlled by a digital high/low signal that is sent from a CAN-card. The experimenal actuator board can be seen in fig. 30.

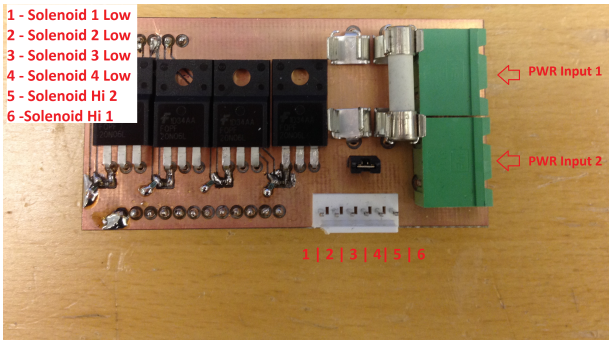


Figure 30: Picture of the experimental actuator board

8 Software

8.1 Introduction

In the beginning the team was split into three subgroups;
The vision group, two persons working full time on vision. Without vision, no missions above regular movement can be achieved, so getting this system to work was crucial for future development.
The Graphical User Interface (GUI) for mission control, one person working for half the project. This part was made so that a person not knowing much or anything at all about the software inside Naiad should be able to program the missions for the competitions or even real missions in the future.

Naiad main control software, three persons working full time and one person working half time. This group had the responsibility to get Naiad moving correctly, in water and simulations, all nodes on the CAN-bus and that all internal and external communication was working as supposed.

The software system is almost redone from scratch, except some firmware and vision from Vasa and old Naiad. The main reason to redo most of the work was to make the complete system more modular, in the end a node based on Transmission control Protocol (TCP) system which is communicating with JavaScript Object Notation (JSON) [25] was built. With this system done, the three subgroups (vision, mission interface and Naiad main control) would make it easier to implement and test their software without having to rebuild the whole system. This also lead to a more agile/scrum like development, due to small nodes which could be done and tested faster, without having to change much or anything at all in the existing system.

On the BeagleBone Black (BBB) [22] and CAN-bus all code running is written in Ada, the vision is written in C/C++ and external programs are written in Ada, C#, Matlab and python. For the ease of communicating between the same and different languages, the JSON standard was used over TCP. BBB uses Universal Asynchronous Receiver/Transmitter (UART) to communicating to a CAN-node connected to the CAN-bus that joins everything to one system.

Robustness and reliability are key components in a robot, so most code in Ada has been written taking Ravenscar [31] in consideration. Some fail detection is also implemented in the CAN-bus to be able to restart the system if needed. The BBB main routing node can also check what other nodes is connected.

To achieve an greater form of modularity, space plug and play avionics[32] have been started to be implemented, as for now when a new CAN-node connects, it will send its name to the power supply unit(PSU) CAN-card, this information will then be passed on to the BBB, this can also be done on request, so that the PSU CAN-card asks the system what cards is still connected, gather the information and send it to the BBB.

8.2 Background

A lot of resources for software existed at start, both from Naiad and Vasa(which had parts of it based on RALF II[30]). The firmware for AT90CAN128 [20] was done and working quite good. Some vision was able to be reused. Help from the previous Naiad team to get started quickly with both CAN-cards and BBB have been of significant help.

8.3 Communication

All of the node to node communication is based on either CAN or Transmission Control Protocol (TCP) where the former is communication between CAN-cards and the later is between nodes that are either on a BBB, a GIMME-2 card or other nodes that are connected via Ethernet. A node on the CAN-bus is still able to communicate with a node on a BBB by passing through a translation node.

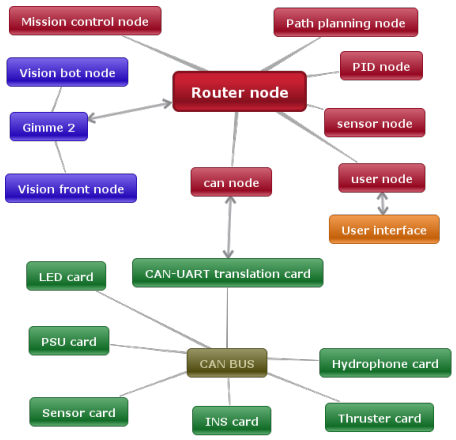


Figure 31: Complete flowchart of software

8.3.1 Node based TCP communication

All of the TCP communication is routed by a central server node that will relay messages to the desired destination node which is specified in the message itself. Having a central node where all the messages pass through adds a lot of overhead but it also adds a lot of flexibility and simplicity for a user. It allows each client to only require a connection to the server while still being able to communicate with all of the connected nodes. Furthermore, this makes it easy to add more nodes to the system with less effort as the other nodes does not have to establish a new connection, they only have to send a new type of message. The center node can also easier track which nodes are alive and connected rather then track every single cross connection between clients.

8.3.2 SPI

The SPI on Naiad is using the standard master/slave configuration, where CAN-cards are acting as masters and the attached peripherals are the slaves. There are two peripherals that are using SPI in Naiad, an LCD display and an ADC that is connected to a fiber optic gyroscope[3]. The highest bit rate possible for SPI on the CAN-cards are 8Mbit per sec as the clock rate of the CAN-card oscillator is 16MHz. The high bit rate is needed as it allows faster data sampling from the fiber optic gyroscope for a better precision when using the data for integration.

8.3.3 UART

Universal asynchronous receiver/transmitter (UART) is the communication link between CAN and TCP where a CAN-card and a BBB is connected. This link have the lowest communication bit rate which in turn sets the limit of the combined bit rate to and from all of the sensor cards. While the BBB support CAN, it does not support the high voltage that is on the bus which is why UART is used instead. Both the inertial measurement unit (IMU) and the side scan sonar is using UART to send the raw sensor data. The IMU is connected to a CAN-card and the side scan sonar is connected directly to a BBB due to the high amount of raw data it produces. The bit rate for all UART communication is 115200 bits per second.

8.3.4 CAN

The Controller Area Network (CAN) protocol is fully hardware supported on the microcontroller AT90CAN128, supporting the standard CAN controller 2.0A & 2.0B. Advantages of using CAN is the ease of adding additional nodes and the fact that it is a broadcast protocol. This allows a very modular system where new nodes can enter the bus without the need of establishing connections to already existing nodes. One drawback of CAN is the limited data bandwidth. To reduce the load on the CAN-bus, many calculations are done directly on the CAN-cards, so the data sent is directly usable.

8.4 BeagleBone Black

8.4.1 Background

The BeagleBone Black (BBB) is a single board computer with a 1GHz single core ARM processor and 512 MB of RAM. It is the hardware that is most similar to a desktop environment in Naiad which makes it more convenient to place the more abstract software on. The software however can not be cross compiled as there is no proper cross compiler for Ada. This mean that all code has to be natively compiled on a ARM architecture, where the BBB itself is generally the most convenient. The BBB supports CAN but because the limitation of using 3.3V for logical input and output compared to 22-24V on the CAN-bus. It is instead connected to a CAN-card via UART which translates all of the CAN messages. The UART on a CAN-card is however using 5V logic, this is solved with a simple bi-directional level shifter converter. The operating system is a scaled down, prepackaged armhf[11] version of Ubuntu. It only requires around 400MB which makes it possible to only require the on board 2GB flash for operating system and software.

8.4.2 PID-controller

One Proportional Integration Derivative (PID) and five PD controllers was implemented in the Naiad robot to control the position and orientation to achieve desired state of the robot.

State estimation

The initial state of the robot sets the global space coordinate system that will act as a reference frame under the runtime. The robot in it self has its own reference coordinate system which is the local space of the robot that all the motor calculations will be calculated in.

Since it is hard to get a good estimation of the position by integrating the acceleration data the positional estimation was neglected and the IMU was only used to get the roll, pitch and yaw angles which were more accurate and their estimated values did not drift.

Error calculation

The PD/PID controller bases its control values on the difference (error) between the desired and the current state of the robot. This error is handled as a vector that contains the error of the positional and the orientational states of the robot, see fig. 32.

Since the natural floating of the robot gives a stable horizontal orientation the most important factor of the orientation is to get a proper planar motion. This is done by keeping track of the angle around the Z axis, the yaw angle.

By keeping the position and orientation data relative to the starting state of the robot, it is possible to calculate which motors should be activated and how much depending on the current state of the robot. This is done by transferring the desired state that is given in global space coordinates into coordinates of the robots local space.

When the difference between the desired state and the current state is determined the control values that represent the desired speed of the motors can be determined by the equations 1-13.

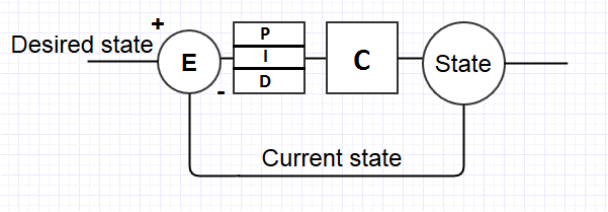


Figure 32: Simplified PID control loop

$$O_{error} = O_{desired} - O_{current} \tag{1}$$
$$XY_{error} = XY_{desired} - XY_{current} \tag{2}$$
$$Depth_{error} = Depth_{desired} - Depth_{current} \tag{3}$$
$$O_p = K_p * O_{error} \tag{4}$$
$$O_d = K_d * (O_{error} - O_{previous}) \tag{5}$$
$$XY_p = K_p * XY_{error} \tag{6}$$
$$XY_d = K_d * (XY_{error} - XY_{previous}) \tag{7}$$
$$Depth_p = K_p * Depth_{error} \tag{8}$$
$$Depth_i = Depth_i + K_i * Depth_{error} * dt \tag{9}$$
$$Depth_d = K_d * (Depth_{error} - Depth_{previous}) \tag{10}$$

Figure 33: Error equations

$$O_{output} = O_p + O_d \tag{11}$$
$$XY_{output} = XY_p + XY_d \tag{12}$$
$$Depth_{output} = Depth_p + Depth_i + Depth_d \tag{13}$$

Figure 34: Control values

The control values are added and multiplied with the corresponding motion component of each motor which gives a motor value between 0 and 255 where a value below 128 means that the motor is going backwards and a value higher than 128 means it is going forward. The calculated motor speeds is then sent to the CAN node that sends the values to corresponding motor controller.

8.4.3 Sensor fusion

The sensor fusion is aimed to calculate the attitude which include yaw, pitch, roll and integrate the accelerate value of x, y and z twice to get the origin position. Since the value of the pitch, roll and accelerometer z value from the IMU are good enough, so about the sensor fusion part now is mainly focus on the accelerometer x and accelerometer y. The accelerometer X value and accelerometer Y

value that comes from the IMU have a lot of noise. This makes it is quite hard for the NAIAD to go straight and do other tasks. The integration of the accelerometer value is the velocity value, and the integration of the velocity value is the distance. As can be seen after integration twice, the small noise from the accelerometer can become very big. This will lead Naiad to not be able to get the correct distance value. Even though there is a Kalman filter in the IMU device, but the accelerometer value x and y still have a lot of noise, that is quite bad. Now we let the value go through the Kalman filter that we build again.

Sensor fusion calculation by using Kalman Filter

First here is some basic knowledge about Kalman filter, it is a very powerful tool when trying to controlling a noisy system[41]."It will try to estimate the state of the system, based on the current and previous states, this tend to be more precise than the measurements alone."[44] Kalman Filter is discrete, it rely on measured samples taken between repeated but constant periods of time. This means that the result is approximated good enough, but there is something that can't be known between the samples[36] and [39]. The soul of the Kalman filter is the five equations which is shown below:

$$X_{k|k-1} = A * X_{k-1|k-1} + B * u_k \tag{14}$$

$$P_{k|k-1} = A * P_{k-1|k-1} * A^T + Q \tag{15}$$

The equation(1),(2) above are Time update which means they are the prediction value for the next round calculate.

$$K_g = P_{k|k-1} * H^T / (H * P_{k|k-1} * H^T + R) \tag{16}$$

$$X_{k|k} = X_{k|k-1} + K_g * (Z_k - H * X_{k|k-1}) \tag{17}$$

$$P_{k|k} = (I - Kg_k * H) * P_{k|k-1} \tag{18}$$

The equation(3)(4)(5) above are measurement update. These updated value will correct the value that is wanted[42]. About the parameters of A,B,P,H,Q,R,K, more details will be found in[45]. For now the Kalman filter which depends on Naiad's can be build from a mathematics model system. The parameters is set manually like P = 1000.0, Q = 0.01 and R = 0.1. When Naiad is kept on the table, the accelerometer x and y value suppose to be 0, but the real value is not 0, so value is set as the error value. The average error is calculated and then removed. When integrating the accelerometer value to get the velocity, the velocity value still not 0. The average error is recalculated and removed again. After that the velocity is quite small. Now if the absolute value of velocity is in the range 0 to 0.5E-03, then the velocity is set to 0, otherwise remove 0.5E-03 from the value. When the velocity value is integrated the distance is good enough to be used.

8.4.4 Mission control

Since each mission is more or less unique, a routine for each is needed. Most of the missions gets feedback from sensors and then update the relative coordinates for the PID-controller or if an action should be needed, for example fire torpedo. It's heavily depending on the data from vision in form of a distance vector to the object or objects that's interesting for completing the mission.

8.4.5 Path planning

Path planning in water differs from other mediums due to the special characteristics of water. The more static environment and the ability to move freely in three dimensions reduces the complexity of navigate without obstacles interfere with the path. For these reasons a simple path planner was implemented for the main purpose to aid the PID-controller by splitting movement to smaller parts which ensures that the proportional part of the controller doesn't exceeds the maximum values for the motors, which would make control of the unit close to impossible.

8.4.6 Side scan sonar

The side scan sonar [23] sends out a sound then measuring the effect received, over 60 degrees split in to 1000 points with 1 byte resolution for up to 25 meters distance. This can in the future be a huge asset when it comes to aiding the vision system or to gain some more knowledge about an objects.

8.4.7 Can node translator

The CAN node is a simple node that is translating messages between Ethernet and CAN. All of the received Ethernet messages with the correct structures will be translated into CAN messages which is sent to the CAN-bus via UART to a CAN-card. All of the expected incoming CAN messages must be predefined to know where the message should be sent. The CAN node does also convert data that have been split because of the limited eight single bytes structure of the data in CAN messages. Due to the nature of UART the node uses busy waiting to read the data. To reduce the CPU usage, the node goes to sleep for a moment while the UART buffer is empty. This means that a message sent from a CAN-card might be slightly delayed in a BBB's buffer.

8.4.8 Border Gateway

The TCP communication is based on a server – client structure. The server will act as a central node and be a gateway to all of the other nodes. This makes it simple to decide where messages should be sent as it can just be specified in the messages itself, not requiring a client to be directly connected with the node it wants to talk to. A drawback is that all the messages has to be sent twice, effectively doubling the number of messages on the network. This also means that each client only needs one open connection which is to the server. All but the initial communication is based on JavaScript Object Notation (JSON). So all messages contains variables with variable names expressed as a string. This allows a simple and compressed way to send data structure over TCP between programs written in different languages. A drawback is that variable names in the structure have to be predefined in some languages which makes it less dynamic.

To initiate a full connection to the server, a client has to first connect to a predefined port that is the same for all clients. When a client is connected on that port, the server expects a message containing a string of three characters. The string is important as the server will use that as the reference name for that connection and pass all messages that targets that name to that connection. After the initial string is received, it will first check if that name already have connected in which case it will do nothing but drop the current connection. If it is a new connection, the server will connect to an internal thread and pass the name on. The server has a predefined number of threads that will be ready to handle connections. When a thread receives a name, it will open a server connection and wait on a port that is related to the name itself. The port number is calculated by using the each letters as a number in base 26. It makes the possible port range go from 0 (aaa) to 17576 (zzz), depending on the name. When the client connects to the new connection, the initial set-up connection is done. A thread that is now handling a connection will sit and wait for an incoming message of the size 256 bytes. This is done to simplify and speed up the handling of messages as the alternative is to either read every single byte to check for termination sign or to always start a message by sending a single integer that will tell the size of the following message. In the message itself, there should be a JSON variable called "target" containing a three character long string with the name. This will tell the server where it should send the message. If the variable does not exist or if the target name have not connected yet, the message will be thrown away.

After an initial connection is established at least once, there is no requirement to reinitialize the connection if the connection would drop, as the server connection will reset on that port, but it will also not do any harm. This also means that a specific thread will never change it's port after it has been set. Because the server is running on a Linux operating system, it is configured to use the keepalive setting on the TCP connection which will probe to see if the connection have been physically disconnected. The keepalive timer is depended on a setting in the operating system, the default time

before checking is 7200 seconds. The current timer is set to detect a broken physical connection after around 3 seconds. Other connection issues can trigger a detection of a client drop much faster. It does not have any timeout settings enabled, so it will never drop the client by itself. The maximum number of clients is currently 20. The server will never be busy waiting as when it waits for a connection or when it is waiting for a message, it will sleep until it is awaken by an interrupt from the operating system due to activity on the Ethernet. This will only happen if the buffer is filled, which is if it receives 256 bytes. The server does also support the ability to mirror or/and disable traffic to nodes. All that is required to set the server node as the message target, set a variable name to 'ethid' 2 or 3 respectively and send boolean with a variable names of the target node where the traffic should be manipulated. Note that on some operating systems, opening a port requires higher user privileges, for example by using sudo on a Linux operating system.

9 GIMME-2

The GIMME-2 board system is equipped with two 10 Megapixel image sensors. The GIMME-2 has a zynq processor which has an ARM Cortex A9 dual-core processor and FPGA in the same chip. The board has 8 GB of RAM . There is a slot for external memory storage. The operating system installed on the GIMME-2 board is Peta Linux. All instructions for building the operating system on GIMME-2 can be found in appendix M.

The system retrieves missions from the decision center and sends back information according to the given mission. Fig. 35 explains how the system works.

9.1 Operating system

The operating system used on the GIMME-2 board is Peta Linux 2014.3 customized by Xilinx for Zynq processor which is based on the vanilla kernels. There are two ways the user can interact with the GIMME-2 board either through a serial port interface (USB cable) or through SSH client(Ethernet cable). Creating a bootable Linux for zynq on GIMME-2 board is challenging due to the fact that the files available on the Xilinx website are mostly for the Zynq evaluation board. The files that are necessary for booting Linux have to be built from scratch, the procedure and tools used for building these files are described in details in Xilinx wiki web site [19]. At first the First Stage Boot Loader (FSBL) starts and does the early system initialization then the universal bootloader starts which loads the kernel image and the wrapped ramdisk image.

9.2 Vision system software

The software in the vision system is a major issue especially since it is dealing with live captured images in underwater conditions, which have changing lighting conditions that will affect the image detection system.

A vision system has been made using OpenCV to deal with all the issues that the vision system can face.

In order to have a better calibration between the GIMME-2 cameras and the real live values a red light detection system has been made which was the start for building the vision system.

9.2.1 OpenCV for GIMME-2

OpenCV has been used in order to implement vision methods since it has a lot of functions that could be used to have a better detection and enhance the underwater image, the challenge was to build OpenCV for GIMME-2 board. Arm-linux-gnueabi cross compiler has been used for building OpenCV for GIMME-2. All instructions of building OpenCV for GIMME-2 can be found in appendix N.

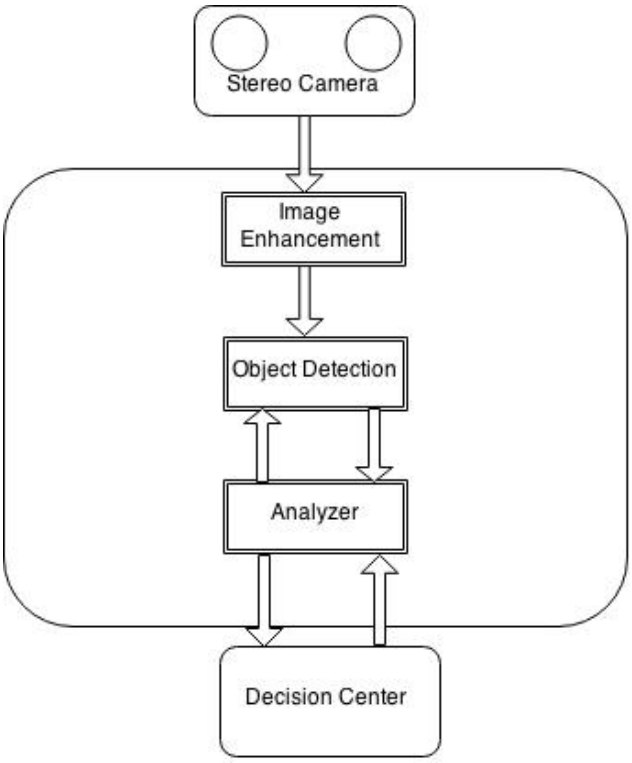


Figure 35:

- Stereo Cameras: continuously takes images underwater.
- Image Enhancement: Enhance incoming images from stereo camera.
- Analyzer: Analyze incoming mission(type of object to detect) and send back required parameters to Mission control center.
- Object Detection: Detect objects according to the Analyzer and send back feedback to the Analyzer.

9.2.2 Image Enhancement

Underwater images are not clear, when the depth increases, the amount of light on objects decreases and the light distribution becomes irregular [9]. So before further image processing, enhancement of incoming images from cameras has been done by converting the image to YUV(Y referred to as luminance or luma, while U and V is the chroma) color space and perform Contrast Limited Adaptive Histogram Equalization (CLAHE) on the Luminescence (Y) channel for improving image contrast [9]. Dilate and erode which are functions used in OpenCV are also used to enhance input images.

9.3 Red Light Detection

Detecting the red light is very important since that it is the base that all the detection in the vision system depends on especially for the real life calibration. Working with OpenCV, two red light detection systems has been made. The first depends on the red light channels inside the image which called the red boost method [13]. The second depends on taking the histogram of the red channel inside the picture and back project the most red part in the histogram which is called the histogram method [7].

The red boost method starts with reading the BGR image and split it into three channels red, green and blue. Then take away the green and blue and keep the red only. Once the green and blue colours is taken away use the multiply function used in OpenCV to boost up the red color only and save the final image in a matrix of type Mat.
After that, threshold the red boosted using threshold image function used in OpenCV and keep only the highest red and save the threstholded image. The threstholded image will be sent to the moment class used in OpenCV to get the x and y of the center of the detected red.

Histogram method works by finding the histogram of the image and then back project the highest red position in the source image. It starts with reading the BGR image and then split it into three planes. Using the split function in OpenCV a green, blue and red plane will be generated and sent into the calcHist function used in OpenCV with a specific histogram range depends on the hue range for the red colour.
Once the histogram calculation is done for the three planes a normalization is required to assure that the histogram is in the range for the BGR image. Once that is done the highest red can be detected by calling the calcBackProject function in OpenCV [8] which will threshold the image using the hue range and by that will detect the red light, and this image will be sent to the moment class to get the x and y of the center of the detected red.

9.4 Object Detection

For object detection, template matching and shape matching has been tested, but they are not good when the object is rotated or scaled; the irregular light distribution underwater would mean having extensive number of templates which would be time consuming so feature detection algorithms and classification algorithms has been used to detect the required objects and additional complex objects.

Once the object is detected by the stereo camera, the system will find the necessary parameters. It sends back the x coordinates of the object in both images and then the distance is determined by the difference between these two coordinates.

The distance is found by multiplying F (The focal length of the camera) with B (The distance between the cameras) and divide the answer by disparity (The difference between the x coordinates from left and right images) as seen in equation 19,and then the final answer is multiplied with a calibrated value to get the distance in real life[14] and [4].

$$Distance = (F * B)/disparity \tag{19}$$

To find the yaw of the detected object the y coordinates of the detected object is needed and the distance to the object and some gain value found from a lots of testing in real life conditions. First multiply the difference in x between the object and the center with the distance to the object and the

gain value as seen in equation 20, and then divide the answer by 100 to get the final value in meters as seen in equation 21.

$$posy = posy(xdifference) * Gain * distance \tag{20}$$

$$posy = posy/100 \tag{21}$$

Finally to find the yaw value of the object the mathematical function arctan is needed with parameters for (pos y) and distance as seen in equation 22.

$$Yaw = atan2(posy, distance) * (180/PI) \tag{22}$$

Finding the z position of the detected object can be done mathematically with the need of distance to the object and some gain value found from a lot of testing in real life condition and finally the y difference between the object and the center as seen in equation 23 and 24 .

$$posz = posz(ydifference) * Gain * distance \tag{23}$$

$$posz = posz/100. \tag{24}$$

9.4.1 SURF Algorithm

SURF(speeded up robust features) is a fast, and more accurate algorithm compared to others [16]. When using SURF four steps should be followed:

1. Key points selection from both images (object and scene).
2. Eextraction of descriptors from those key points.
3. Match descriptors by matcher.
4. Analyze matches [15].

For matching flann matcher and knn (k-nearest neighbor) matcher has been used, and knn give. Better results. As a final result of SURF it can be a very useful method for complex objects and it can detect some of required objects, but not all of them because they do not have too many features and edges.

The SURF algorithm APIs is not included within the OpenCV repository is intended for development of so called extra modules. All instructions of building extra modules with OpenCV for GIMME-2 can be found in N.

9.4.2 Classification

One of the best object detection methods is the Haar Cascade Classifiers. It is a machine learning based method, which uses a cascade trained by hundreds of sample pictures called positive and negative images [6].

The haar training is the most difficult and time consuming part of the haar cascade method. Once the negative and positive images is gathered the user can start making the sample images which combine the negative and the positive images in one file once the samples images is finished the final step left is to start the training and generate the Haar trained file.

The instructions for training your own Haar Cascade can be found in [5] and [10].

9.4.3 Gate Detection

For gate detection the enhanced image has been converted to HSV (Hue, Saturation and Value) and segmented for a specific color. Edge detection is used to separate desired edges, and then we find contours.

9.5 CAN-card

Two versions of the CAN-card is used, the one developed last year and a modified version of that one to be able to use six PWM signals on the same card.

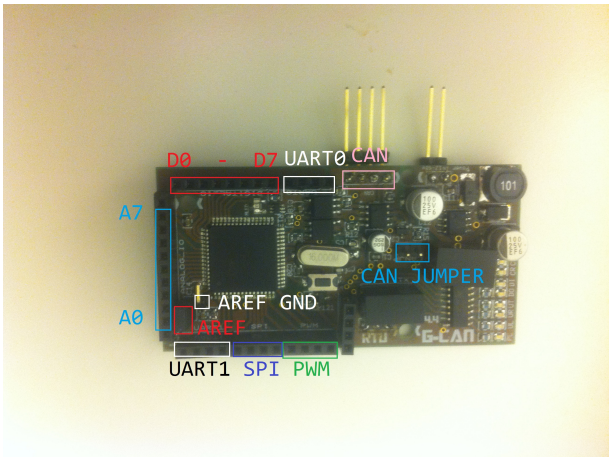


Figure 36: Generic CAN card version 4.4

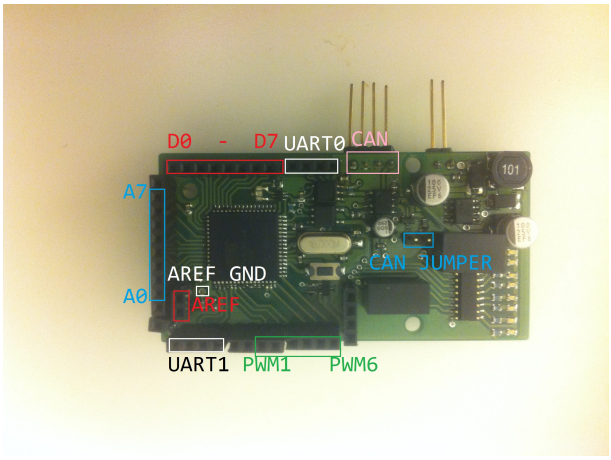


Figure 37: Generic CAN card version 5.0

To use the analog ports as analog inputs a jumper on AREF is required. For the CAN-bus to work it needs atleast one jumper over CAN Jumper pins in the pictures above, but more is not a problem, so preferably use one jumper on each card.

The CAN-cards got Atmels microprocessor AT90CAN128 [20] coupled with a 16MHz oscillator. It got hardware support for Serial Peripheral Interface (SPI), Controller Area Network (CAN) and Universal asynchronous receiver/transmitter (UART) protocol. The AT90CAN128 microprocessor is programmed with the JTAGICE MKII[26] or JTAGICE 3[27] from Atmel or equivalent together with the software Atmel studio 6. To program, attach the programmer to the analog pins.

9.5.1 Thruster node

From the previous years requirement one micro controller should be assigned to each motor. This requirement was removed this year which lead to the decision to let all motors become one node with a mutual micro controller, saving both energy, reducing heat and lessen the load on the CAN-bus. The PWM signal for the motors runs on 50 Hz and a duty cycle between 5 and 15%. The cards receives instructions from CAN messages where a byte of data represents the desired speed.

9.5.2 Sensor node

The sensor node handles all the analog sensors such as the pressure sensor [28], the temperature sensor and the salinity sensors. For the time being, only the pressure sensor has been implemented. It is able to give the depth of the unit down to three bars pressure which translates to roughly 20 meters. A temperature and at least one salinity sensor is planned to be implemented as well in the future to help the system estimate movement etc. First an sample from the pressure sensor should be taken to measure the value at the surface, the analog-digital-converter (ADC) has a resolution of 10 bits, therefore the resolution of the depth would be

$$20metres/2^(10) = 20000centimetres/1024 = cmdepth/analogvalue \approx 2cm/1analogvalue \quad (25)$$

So the real depth of Naiad would be

$$(currentvalue - initialvalue) * 2 = centimetresbelowsurface \quad (26)$$

9.5.3 Inertial navigation system node

The main purpose of the INS card is to get the current orientation of Naiad. This is done thanks to an IMU and a fiber optic gyroscope. The IMU is communicating with UART. The IMU can be configurated to give a variance of data. Currently the outputting orientation in X, Y and Z axis as well as the true body acceleration in X, Y and Z. The output data of the IMU is sent as a string of characters containing both header information and the data itself. The INS card removes all of the header information by extracting specific sections of the string and converting them to float values. This heavily reduces the data that has to be transmitted and it makes the values possible to use immediately. As each float is four bytes and a CAN message can has eight bytes of data, each message can hold two values from the IMU. A total of three messages is required to send all the data for each time the IMU generates an output. The current IMU output frequency is 20Hz. An issue with the IMU is that it can drift in yaw due to the nature of the implementation of the IMU hardware.

To compensate, a fiber optic gyroscope is added to track the yaw orientation. The fiber optic gyroscope is generating an output voltage correlated to the angular velocity around yaw. This voltage has to be measured very precisely which is why it is connected to an external low noise ADC. The ADC is sending the voltage value via SPI, this allows for a high bit rate which is needed to get a small integration interval. The ADC is acting as a slave in the SPI communication and it requires to first be selected through a slave select pin by pulling the voltage to a digital low. When the ADC is activated, it requires an external clock to drive the transmission. The clock should be starting with a rising edge when transmitting data. The initial procedure of the CAN-card is to send instruction to set the sample frequency of the ADC and also tell it to send data continuously so the CAN-card can avoid asking for data each time. As the ADC uses 24 bits to store the voltage value, each measurement has to be sent over three transmissions as the word size of the SPI is eight bits. The CAN-card will turn the slave off by turning slave select pin digital high.

9.5.4 Power supply unit node

This node controls how and when to start electronics and motors and also feedback to and from outside the robot in form of an LCD screen and a remote control. By controlling power to other nodes is has

the possibility to restart the whole system on command or if needed. The mission switch is connected to this node, by removing the switch it will send a command to the BBB allowing them to run the mission, if put back during a mission, the mission is paused until the switch is removed.

9.5.5 Translator node

Naiad mainly uses two different communication protocols to communicate between nodes, CAN and TCP. The link between them is UART as the BBB don't support the high voltage of the CAN-bus. The only task for this card is to read all the messages on the CAN-bus and pass them on to a BBB via UART and, vice versa, it translates UART messages to CAN messages. If this card sends too many UART messages over a short amount of time to a BBB, the BBB can crash.

9.5.6 LED node

When the robot is in the water, debugging and understanding the system by reading the LCD is hard, so by being able to control a total of 4 rows, 2 on each wing, of RGB LEDs, a better understanding is achieved and it will draw more attention to the robot. The card is also controlling power of the headlights in the front and bottom to support the vision system.

9.5.7 Hydrophone node

As the time difference between the signal generated from different hydrophones is small, the time stamping for each are based on four different interrupts to avoid busy waiting. When all four interrupts are triggered, a calculation can be made to get the source direction of the sound. This will send a message to update the system. The direction calculation is not implemented. When a interrupt is triggered, the interrupt on that pin is disabled. This is required because the CAN-card will get multiple pulses from the same source instance. The interrupt pin is enabled when either all four interrupts have been triggered or after a set time.

9.6 User Interface

9.6.1 Mission control interface design

The user interface is an integrated development environment, it is a graphical application written in C#, it allows the user to drag the mission from the missions panel and drop them to the painting panel, use straight line arrow or a curve line arrow to connect the missions. It also includes the PID control setting part and CAN message control part, all the output from this interface are JSON strings.

The flowchart that in fig 85 show the whole design of the user interface. First finish the drag-and-drop part, second make use arrow to connect with the node come true, finish the JSON string output through the TCP communication.

The drag and drop part mainly use the Microsoft Visual Studio tool, the drag part use the toolStrip bar which is a develop tool you can drag button from it easily. The drop part use the painting panel which you can drop some components on it. If there is a task or an arrow was dropped on the panel, then this component is added to the PaintItem List<>. The purpose of doing this is to make sure it is easy to manage all the items on the panel, then the painting panel will trigger the onPaintBackground and onPaint method, so the panel will refresh and repaint. The flowchart that in fig 39 is the design flowchart for the drag and drop event part. After dropping a task on the panel, the node should still be moveable. When click the mouse, the first thing need to do is to check which side of the mouse was pressed, if it was left button, then set the move value equal to true and get the start position of the node, calculate how far did the mouse move and set the new coordinate equal to the node, refresh the paint panel and set the move value equal to false. The flowchart that in fig 40 is the drag node move on the panel design.

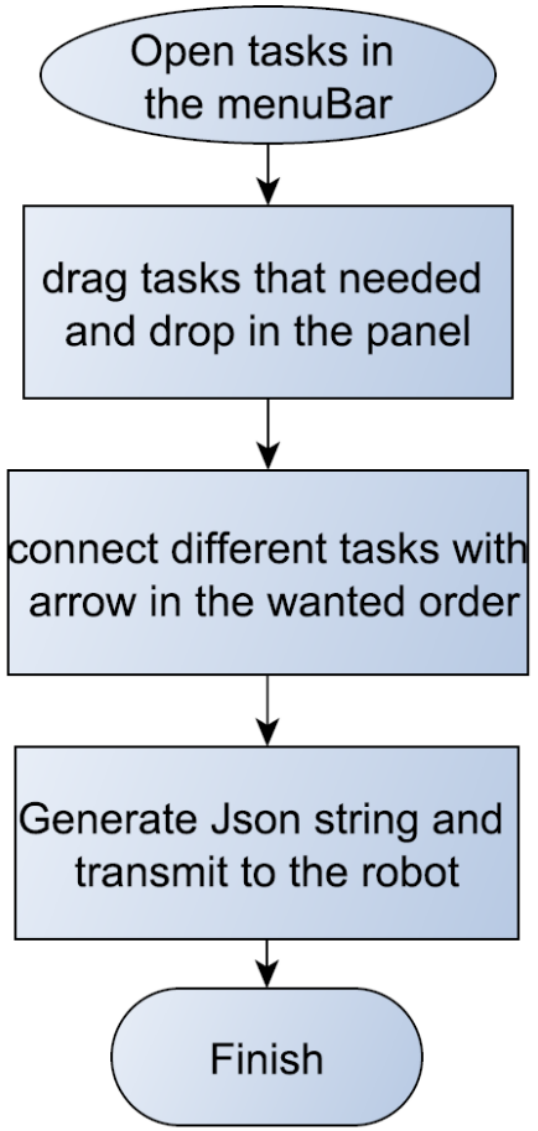


Figure 38: IDE over all design flowchart

create JSON and subJSON string design

Before trying to create a JSON string, there is one thing that needs to be done, sorting all the items in the PaintItem list<>, add nodes into the unitNameList<>, add straight lines into the straightLine-NameList<>. Add curve lines into curveLineNameList<>, and also make sure all the units in order depend on their coordinate by using Sort() method. Until now it is time to create the first JSON object, json1, set the checkMark value equal to false. The checkMark is used to check whether the straight line is the last straight line or not. The first thing that needs to be done is to check how many straight lines in the straightLineNameList<>, if there is no straight line, call CreateSubJsonString() function to create a subJSON string which is the callBack string and add this sub string to json1. If there are lines in the straightLineNameList<>, create another JSON object json2, find the last line in

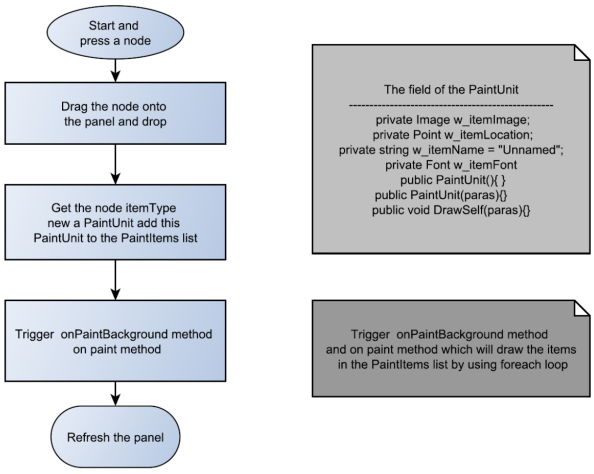


Figure 39: The drag and drop design flowchart of IDE

the list. Compare the coordinate with all the units in the unitNameList<>, if the unit's coordinate is equal to the line's startPoint, then add the unit name to json2 and also add json2 to json1,set the checkMark equal to true. Now compare all the unit's coordinate with the last line's endPoint and do the same thing as the startPoint, but set checkMark to false. Then use the foreach function to create json string for all the unit that left in the unitNameList<>. The flowchart that in fig 41 is the flowchart design for the create JSON string and subJSON string.

PID control design

There are three parts on the PID control panel, the position control part, the orientation control part and the TCP communication part. The user can set the value for P,I,D from position x, position y and position z in the position control. From the orientation part the user can set the PID values for yaw, pitch and roll. When the values are set, the user can type enter the IP address and port number and then send the value to BBB, also as we can see there are a lot of values there, it is quite a waste of time to set all the values every time, in that case there are save and import buttons, when you get a group of perfect values, if you still want to use them later, you just need to press the save button and then when you need them next time, you can use the import function, which makes the user's work much easier.

CAN message setting design

In this panel, user can set the ID of the CAN message and also the value of the message, after this the programme can figure out the length of the message and display number on the screen. When the user type into the IP-address and port number and press the send button, the message will be send to Naiad.

There are some other function and feature that I didn't mention here, you will find them on the appendix part.

9.6.2 Simulation

Modes

The simulation node can be set in two different modes.

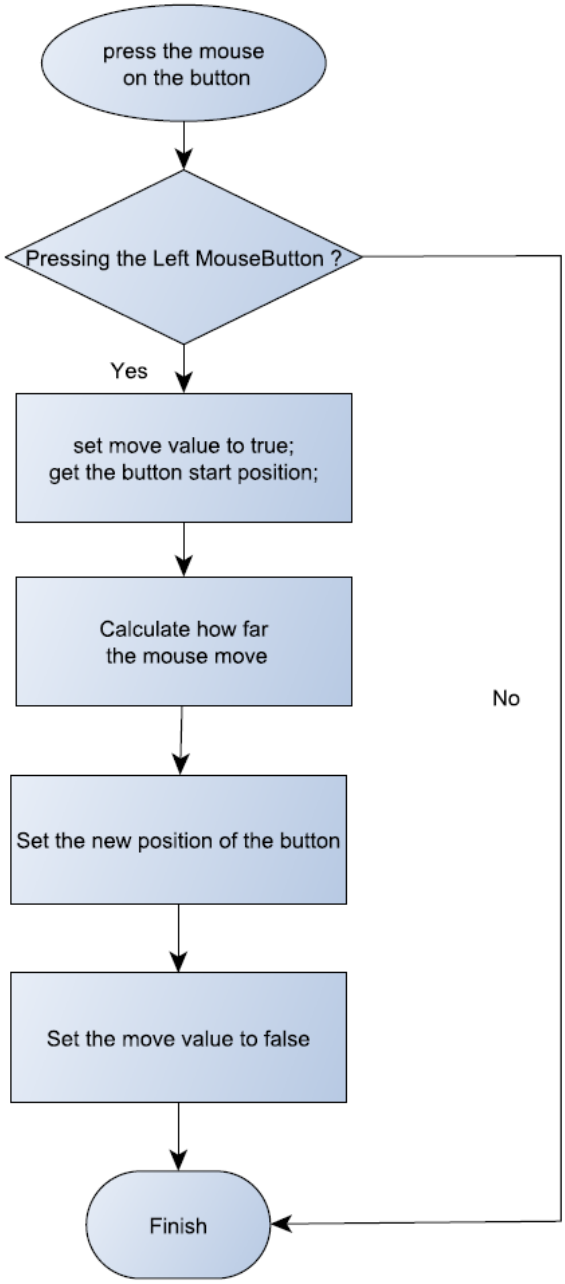


Figure 40: Drag node move on the panel design of IDE

- Simulation mode
- IMU mode

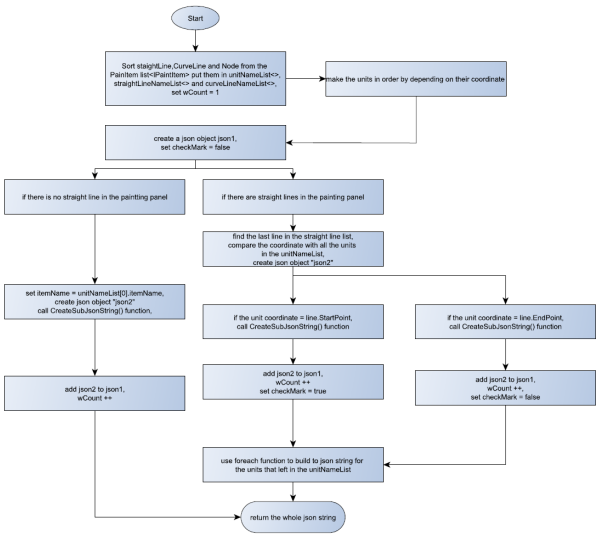


Figure 41: Create json string design of IDE

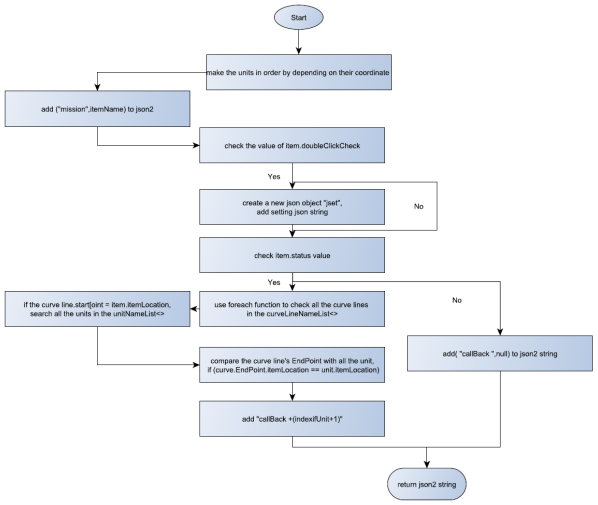


Figure 42: Create sub json string design of IDE

Simulation mode

In simulation mode a virtual simulated representation of the Naiad robot is run with simulated sensors and actuators. The simulation allows you to be connected directly to the PID controller through TCP sockets, sending equal data as the real sensor node would have done if the robot was running a real task and receiving the real motor values that the PID controller would send to the real robot. These motor values are turned into estimated forces that is shown as dynamic lines coming from the different motor positions to give a good visualization of the forces that is acting on the robot.

The calculations done in the simulation is based on the physical entities of the robot in order to get as accurate estimations of the forces that are acting on the robot as possible.

By converting the motor values from the PID controller into estimated forces its possible to use

these forces in the Newtonian equations to get an estimation of the translation of the robot and motion of inertia equations to estimate the orientation of the robot.

Based on Newtons second law of motion the translational and orientational acceleration can be determined by the equations 43 and 44.

$$F = m * a \tag{27}$$

Figure 43: Base equation to get the linear motion

$$\tau = I\alpha \tag{28}$$

Figure 44: Base equation to get the rotational motion

$$\begin{pmatrix} 0.866025 & 0.5 & 0.0 & 0.0 & 0.0 & 0.28 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.22 \\ 0.866025 & -0.5 & 0.0 & 0.0 & 0.0 & -0.28 \\ 0.0 & 0.0 & 1.0 & 0.355 & 0.230 & 0.0 \\ 0.0 & 0.0 & 1.0 & -0.355 & 0.230 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & -0.355 & 0.0 \end{pmatrix} \tag{29}$$

Figure 45: Thruster configuration matrix

Combined with the Newtonian equations a thruster configuration matrix was used that is based on the distances and angles of each motor relative to Naiads center of gravity (COG) that makes the motion calculations less computationally heavy and easier to edit. Each row in the thruster configuration matrix represent the corresponding motors motion components. The first three columns represent the X, Y and Z translational component and the last three columns represents the Roll, Pitch and Yaw rotational components.

To make an accurate estimation of the time varying local coordinate frame that the robot would be in after the forces have acted on the body a rotation matrix is determined by the equations 30-32[35].

Where S is a skew-symmetric matrix that contains the angular velocities that is integrated from the angular accelerations determined by the Newtonian equations.

IMU mode

In IMU mode the simulator can be set into a passive mode that shows a virtual visual representation of the estimated position and orientation that the system in real time has estimated. No calculations is needed in this mode more than rotating the virtual robot in the right order.

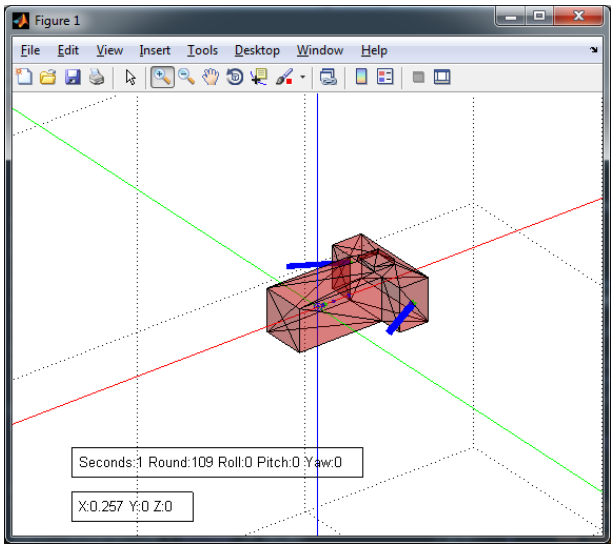


Figure 46: Visual representation of Naiad and forces while running in simulation mode

$$R^{'}(t) = S(\omega)R(t) \tag{30}$$

and,

$$R(t + \delta_t) \approx \delta_t R^{'} + R(t) \tag{31}$$

which can be combined to,

$$R(t + \delta_t) \approx \delta_t S(\omega)R(t) + R(t) \tag{32}$$

10 Results

The result of this project is a robot that is able to operate, autonomously, under water. The robot has a vision system and a sensor to determine how deep in the water it is. It is also able to execute some basic missions, such as move to this position.

The list of requirements as it was given from the client is found in 2.1. During the time of the project, some items were taken from the list of requirements for different reasons, these are marked with red in the list below.

1. Functional requirements
   - 1.1 Done: The robot should be built using a mono-hull.
   - 1.2 Done: The hull should be easy to open and close.
   - 1.3 Done: The cabling should be very efficient and easy to manage.
   - 1.4 Done: A communication cable for remote operation should use either Ethernet or an optical fibre.
   - 1.5 The robot should be able to operate using external power (preferable through an umbilical including optical fibre/ethernet).
   - 1.6 The umbilical ends in a box that connects to the robot.

1.7 Done: There should be a Kill Switch.

1.8 Done: There should be a Mission Switch.

1.9 Done: There should be indicators like LEDs and/or LCD-display.

1.10 Actuators

1.10.1 Every actuator has its own micro-controller.

1.10.2 Done: All actuators should operate over the CAN-bus.

1.10.3 Done: The motors should preferable be BLDC, but ordinary DC-motors can be used.

1.10.4 Done: Make new own BLDC-driver. One Controller two motors. Remark; this was not done, however the old drivers are now working properly

1.10.5 There should be two droppable markers.

1.10.6 There should be two unpowered torpedoes.

1.10.7 There should be a simple manipulator that can grip an object and release the object.

1.10.8 An LED-light for cameras.

1.11 Sensors

1.11.1 Done: All low rate sensors should operate over the CAN-bus.

1.11.2 Done: High speed sensors should operate over ethernet (GIMME-2).

1.11.3 Every sensor has its own compute power.

1.11.4 Done: There should be a pressure sensor.

1.11.5 There should be a sensor for water temperature.

1.11.6 There should be a sensor for salinity in the water.

1.11.7 There should be a speed logger (measures the speed in two directions.

1.11.8 Done: There should be an 9-DOF IMU.

1.11.9 There should be an independent gyroscope for the yaw angle.

1.11.10 There should be an active two dimensional sonar.

1.11.11 There should be a passive two dimensional broadband sonar (20kHz - 40kHz).

1.11.12 There might be a system for finding direction and distance of cooperative robots based on ultrasonic sensors.

1.11.13 Done: There should be two stereo camera systems or one system that fast and accurate can tilt (Two GIMME-2).

1.11.14 The fibre optical gyro shall be installed.

1.12 Communication

1.12.1 Done: The communication system between nodes should use the CAN-bus.

1.12.2 Done: The project should study Space Plug and Play Avionics (SPA) system and implement part of it.

1.12.3 There should be a communication mechanism for robot-robot communication (in case of cooperative robots).

1.13 Software

1.13.1 The firmware (Software for the micro controllers) should be downloaded via the CAN-bus.

1.13.2 The firmware should be handled by a configuration manager.

1.13.3 Done: Compare the latest Vasa-code and Naiad-code, extract the best parts. Document the arguments.

1.14 Done: There should be a simulator for the mechanical and electronic parts.

1.15 Done: There should be a simulator for the software system.

1.15.1 The simulator should be able to handle more than one instance of the robot.

1.15.2 The simulator should handle inter robot communication.

1.15.3 Done: The GUI of the simulator should be used to monitor the actual robot in operation.

2. Missions for Robosub 2014 that shall be implemented.

2.1 Pinger: Go and bring things on the bottom and move.

2.2 Gate: Go through gate with visual servoing.

2.3 Done: Bouy: Identify and hit.

2.4 Bin: Identify and drop marker.

2.5 Torpedo: Find opening and fire torpedo.

3. Nonfunctional requirements

3.1 The programming language should be Ada and using the Ravenscar profile.

3.2 Done: The software should be layered, with hardware support as the first (lowest) layer.

3.3 Done: The software should be based on principles for component based software engineering.

3.4 Done: The documentation should be written using Latex.

3.5 Done: The preferred computer in the system is the GIMME-2.

3.6 Done: The batteries should be easy to change.

3.7 Done: Make a complete test plan.

Most of the items listed as not done are at least started, many of them are almost finished but had to be left that way due to lack of time and/or funds. The different items marked in red were removed for different reasons. The requirement that every sensor should have its own compute power was removed, since that would require an extensive amount of extra cards inside Naiad. This takes up additional space and also creates heat inside the hull. All requirements that were related to inter robot communication were removed in agreement with the client since creating more than one instance of Naiad would not be possible during the time of the project. Furthermore, the requirement for writing all code in Ada was removed since it would be easier to do the vision programming in C instead. However, Ada has been used for all on-robot communication except on the GIMME-2 boards. Ravenscar was also taken in consideration when this was possible.

10.1 Mechanics results

The previous group had developed and constructed the main hull. Since the main hull was complete it was mostly peripherals that needed to be created this year. For example the wings. The wings has been 3D-printed with attachments, such as LEDs to show state in the program as well as mountings hydrophones.

Also the entire tool plate needed to be redesigned. The tool plate was divided in to a front and back tool plate and are designed to be changeable in case other functionality was desired. The front tool plate has been constructed, it contains most of the equipment for the RoboSub competition. For example torpedoes and markers. The side scan sonar is also mounted in the front tool-plate. The back is at the moment an empty support structure where tools can be attached. The simple manipulator has not been finalised since the project ran out of time.

10.2 Electronics results

The electronics group has continued the work from the previous year and added more functionality to the pre-existing cards. This added functionality have for the most part been by add-on cards to the exiting cards. A redesign of the CAN-card has also been done in order to get more out of the MCU itself. This new card can not at present time completely replace the cards developed last year but it is the belief of the group that this can be done with another redesign.

Some of the functionalities that has been added is hydrophones to be able to hear a pinger used in the RoboSub competition. A Speed-logger to help determine position in the water. The previous group worked on a card to support the FOG to help determine yaw position that this group seems finalised but more testing is required. RGB-LEDs have been added to the wings to communicate with the surface. A remote control to navigate though menus on the display when Naiad is on the surface has been constructed as well as a card to control the solenoids in the tool plate, these have also not been tested however. Also a side-scan-sonar has been built in to in to the system.

10.3 Software results

Most of the software and protocols running in Naiad have been tested firmly. By observation over time or by trying to break them. If an essential program failed, it has been prioritized until working as desired. By focusing on robustness, a good base for future work has been built. A lot of work has been put in to get a node based system running. With that working much time can be saved for testing new sub-programs since it is easy to add a new node. It also simplifies diagnostics to see what node is alive to take appropriate actions if a node goes down. This also helps for testing the robustness of that program. The PID-controller and path-planner have been tested both in simulation and in real life. The tests have been successful but some tuning is sting required to get optimal performance. The simulator has been tested with orientation and translation with correct results. The graphical user interface is yet to be implemented for sending missions to the robot. Tests for sending simulated missions and messages has been done successfully. Implementation in Naiad for receiving missions and messages from the GUI is not implemented. The vision system has been tested with acceptable result for the purpose, a problem have been the lenses narrow visual angle causing it to easy loose sight of objects. Also the frequency of the program needs to be increased to work efficient enough. Most of the CAN-nodes works without any problems. The most common problems seems to be caused by SPI or UART, the time needed to firmly fix the firmware for these protocols has not been prioritized due to time issues and that the impact would not be essential.

11 Discussion

The documentation of the work previously done in this project was not that well done, some parts were missing and some parts had faults in them. This made it harder for the team to move forward in their efforts to finish the project. Since the project was already started, the team members agreed that the best way was to continue with what already started. Restarting would take to much time. However as the project progressed, it was clear that some parts were to undocumented, others did not reach the right standard and had to be dismissed for that reason. This is to be seen as one of the main reasons as to why the project could not be finished on time. Not all parts had to be redone, some was harder to change, such as the hull which would have taken a lot of money and a lot of time to redo. Some parts were also fully functioning, which made it unnecessary to change them.

Since the system is currently specified for some of the RoboSub 2014 missions, this will have to be changed before the competition. However, there are only minor changes to the missions which means this will be an easy task.

Working in a larger group with shared documents have caused problems. Twice almost all documents have been accidentally erased only being able to be retrieved thanks to a computer not updated the folder. Working as a group there is also a possibility that more then one person is working in the same document at the same time. If saved by one this will overwrite the others. This have happened way more than once. For these reasons a better system then Dropbox should be used for sharing files.

The previous Naiad team had one person who was working full time on marketing and getting sponsors for the project. Since there was only 12 persons in this project group, it was decided that there was to many other things to do. This responsibility was supposed to be shared between all members of the group. However in the end, it was only the project manager working on this. As the project progressed it was apparent that the project would have benefited from having one person working full time on this since dedicated efforts are needed to get companies to move forward with sponsoring requests.

11.1 Mechanics discussion

The mechanics group is the group that have had to mostly adapt to what was given. This has both been a blessing and a curse. Some decisions on design was already complete meaning some workaround was necessary. At the same time a as it has been good to have a base to start from. Because of time and cost in manufacturing all of the constructed pieces from last year needed to be kept. For the most part the designers from last year hat thought abut how to attach the missing pieces. One place where this was not taken in to account was the wings. There are no way to attach the wings into the top of naiad at the beginning. Also the Motors needs to be graced aver every-time they have been in water. Meaning access to them is necessary. This complicated the design of the wings.

The tool-plate is milled just like the rest of the hull. Milling however eats allot of time. It is hard to get time on the milling machine in Eskilstuna. Every time someone else have been on the machine it needs to be recalibrated which also ate a lot of time. To reduce the amount of people wanting time on the machine as well as the complexity on some parts drove the mechanics group to learn the biggest milling machine available in Eskilstuna. This machine has a steeper learning curve than the other machines. Despite the extra time to learn it it feel in the end like it was the right choice.

The wings and some other smaller parts of the parts produced this year was printed in the schools new 3D-printer. This have not always been without frustration. It is not a high quality printer and sometimes several prints needed to be made before a satisfactory print was done. The printer material also did not seam to go with the chlorine in the pool we were testing the robot in. It is believed that a layer of paint will fix this.

Another frustration for the mechanics group is the vague description or waiting in response on how things should be mounted by other groups. This is something that is always a problem in all group assignment but because we only had one shot on most of the mechanic pieces it was extra important to get it all right the first time.

11.2 Electronics discussion

The electronics groups progress went really smoothly in the beginning. The cards from last year was already produced and add-ons seamed easy. However as further in in to the project we came the more it was important to not only be able to use the cards from last year but also understand them. Most of the documents regarding electronics from last year was sub-standard or missing completely therefore work on recreating this documentation have been done as well, taking up surprisingly large amount of time.

The electronics group have produced allot but as always when constructing hardware it takes longer time than expected and there had been an hope to reach further that we have done. For example have time to order final version cards from Würth and ideally even have time to produce these cards. This have not been the case. All cards have been prototyped and tested but further works needs to be done on most cards.

11.3 Software discussion

The software group has worked well together reaching most of the primary goals, achieving a reliable base for future work.

Documentation is just as important as the work you document on in projects of this magnitude. The documentation from the first Naiad team was hard to follow, making so that a lot (almost all) of the work had to be redone. This year the software team has focused on following a standard that should be easy to use and using at most 3-5 layers of code in libraries to avoid having to spend much time understanding where values and functions descend from.

By combining the previous firmware from Naiad and from Vasa, a well working firmware has been achieved in short time, there is still some issues, but overall it works good.

From the start the GIMME-2 boards was supposed to replace BBB but due to limitations of the operative system running on the GIMME-2 boards this would not be an effective alternative.

A lot of problems hard to explain have occurred in the CAN-cards. Whether it is Ada or the firmware that causes it, the persons working with have tried to figure out but without any real success. This are not logical issues, but rather for example crashing when leaving a procedure, variables changing values without reasons etc.

References

[1] Advanced materials renshape bm 5173.

[2] Dropbox. http://dropbox.com. [Cloud service; accessed 7-January-2015].

[3] Fiber optic gyro 8088000-112 b from saab.

[4] Find distance. http://warse.org/pdfs/2013/icctesp02.pdf/. [Online; accessed 26-October-2014].

[5] Haar cascade training. http://opencvuser.blogspot.be/2011/08/creating-haar-cascade-classifier-aka.html/. [Online; accessed 02-December-2014].

[6] Haar classification. http://docs.opencv.org/modules/objdetect/-doc/cascade_classification.html/. [Online; accessed 01-December-2014].

[7] Histogram and backprojection. http://www.robindavid.fr/opencv-tutorial/chapter4-histogram-and-backprojection.html/. [Online; accessed 16-November-2014].

[8] Histogram and backprojection in opencv. http://docs.opencv.org/doc/tutorials/imgproc-/histograms/back_projection/back_projection.html/. [Online; accessed 18-November-2014].

[9] Image enhancement. http://auv.dce.edu/software.htm. [Online; accessed 15-November-2014].

[10] Opencv haar training. http://www.tectute.com/2011/06/opencv-haartraining.html/. [Online; accessed 04-December-2014].

[11] Prepackaged ubuntu for armhf.

[12] Project vasa system overview.

[13] Red boost method. https://marcosnietoblog.wordpress.com-/2011/11/23/simple-highlighting-rgb-colors-with-opencv/. [Online; accessed 08-November-2014].

[14] Stereo camera. http://ieeexplore.ieee.org/stamp/-stamp.jsp?arnumber=6271270/. [Online; accessed 02-November-2014].

[15] Surf. http://docs.opencv.org/trunk/-doc/py_tutorials/py_feature2d/-py_surf_intro/py_surf_intro.html/. [Online; accessed 29-November-2014].

[16] Surf speed. http://computer-vision-talks.com/articles/2011-01-04-comparison-of-the-opencv-feature-detection-algorithms//. [Online; accessed 02-December-2014].

[17] Trello. http://trello.com. [Web system; accessed 7-January-2015].

[18] Tusentals tunnor med kvicksilver kan finnas i Östersjön. [Online; accessed 7-January-2015, published 25-December-2014].

[19] ug873. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug873-zynq-ctt.pdf/. [Online; accessed 04-October-2014].

[20] At90can128. http://www.atmel.com/Images/doc7679.pdf, 2015. [Online; accessed 5-January-2015].

[21] Atmel studio 6. http://www.atmel.com/microsite/atmel_studio6/, 2015. [Online; accessed 8-January-2015].

[22] Beaglebone black. http://beagleboard.org/BLACK, 2015. [Online; accessed 5-January-2015].

[23] Deep vision: side scan sonar. deepvision.se/products/side-scan-sonars/, 2015. [Online; accessed 5-January-2015].

[24] Gnat gps. http://libre.adacore.com/download/configurations#, 2015. [Online; accessed 5-January-2015].

[25] Json. http://www.json.org/, 2015. [Online; accessed 5-January-2015].

[26] Jtagice mkii. http://www.atmel.com/tools/AVRJTAGICEMKII.aspx, 2015. [Online; accessed 5-January-2015].

[27] Jtagice3. http://www.atmel.com/tools/JTAGICE3.aspx, 2015. [Online; accessed 5-January-2015].

[28] Pressure sensor. http://www.sensorsone.com/wp-content/uploads/2013/10/IMCL.pdf, 2015. [Online; accessed 7-January-2015].

[29] Putty. http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, 2015. [Online; accessed 13-January-2015].

[30] Ralf ii. http://www.idt.mdh.se/rc/page_id1/ralf-ii, 2015. [Online; accessed 5-January-2015].

[31] Ravenscar profile. http://docs.adacore.com/sparkdocs-docs/Examiner_Ravenscar.htm, 2015. [Online; accessed 5-January-2015].

[32] Space plug and play. http://www.kirtland.af.mil/shared/media/document/AFD-111103-031.pdf, 2015. [Online; accessed 5-January-2015].

[33] Winscp. http://winscp.net/eng/index.php, 2015. [Online; accessed 13-January-2015].

[34] Castlecreations. Phoenix ice2 hv. http://www.castlecreations.com/products/phoenix-ice2-hv.html, 2015. [Online; accessed 5-January-2015].

[35] Peter Corke. Peter corke - robotics, vision and control, 2011.

[36] Greg Czerniak. When can kalman filters help? http://greg.czerniak.info/guides/kalman1/, 2014. [Online; accessed 10-November-2014].

[37] ebm09003@student.mdh.se Emil Bergström.

[38] AUVSI Foundation. Official Rules and Mission AUVSI & ONR's 17th Annual RoboSub competition. http://auv.mcgillrobotics.com/wp-content/uploads/2014/05/RoboSub_Mission_Final_2014.pdf, 2014. [Online; accessed 21-November-2014].

[39] Johnson A sumadu Hisham Alrawashdeh. The kalman filter performance for dynamic change in system parameters.

[40] Crustcrawler Inc. 400hfs-l hi-flow thruster. http://www.crustcrawler.com/products/urov2/docs/HiFlow_400HFS-L_Thruster_Data_Sheet.pdf, 2015. [Online; accessed 5-January-2015].

[41] Unknown. Kalman filter applications.subject mi63:kalman filter tank filling.

[42] Unknown. Kalman filter for dummies. http://bilgin.esme.org/BitsBytes/KalmanFilterforDummies.aspx, 2014. [Online; accessed 20-November-2014].

[43] Unknown. Naiad. http://en.wikipedia.org/wiki/Naiad, 2014. [Online; accessed 4-December-2014].

[44] Unknown. A practical approach to kalman filter and how to implement it. http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/, 2014. [Online; accessed 14-November-2014].

[45] Unknown. What are all those matrices for the kalman filter? part i: x, f, p, h, r, u. http://forums.udacity.com/questions/1010153/what-are-all-those-matrices-for-the-kalman-filter-part-i-x-f-p-h-r-u, 2014. [Online; accessed 20-November-2014].

A Electronic schematics

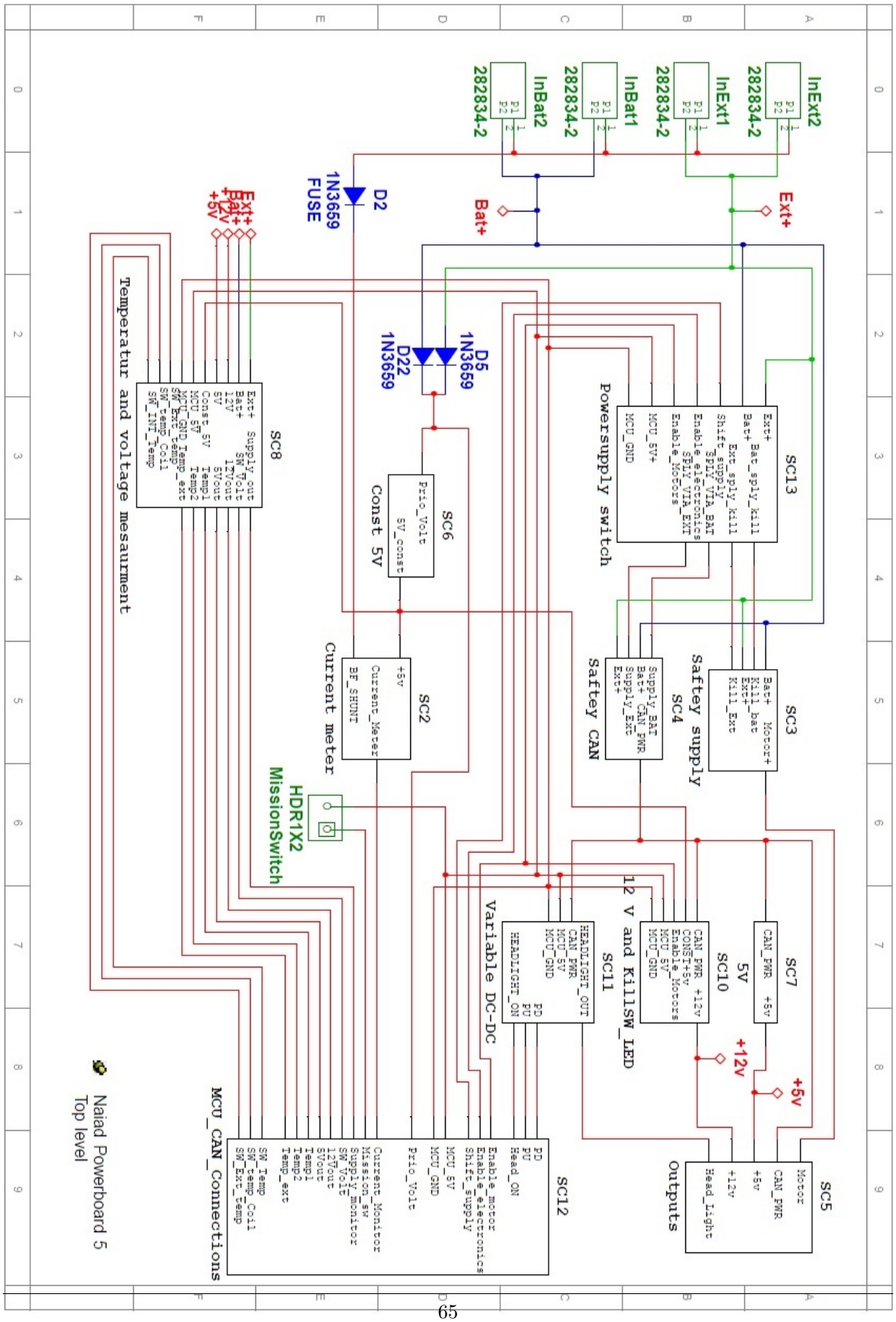


Figure 47: Topview of Powerboard

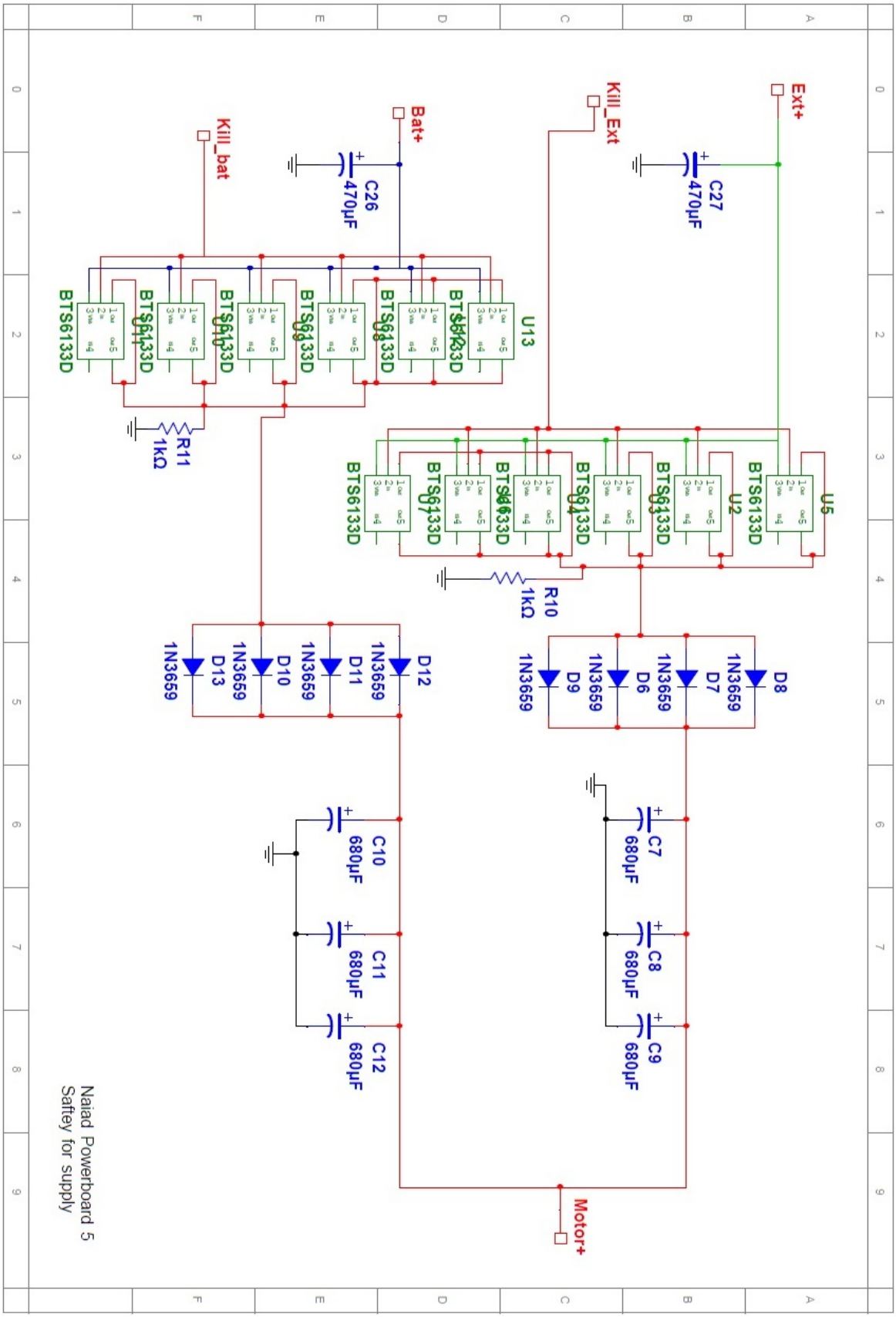


Figure 48: Saftey for supply on Powerboard

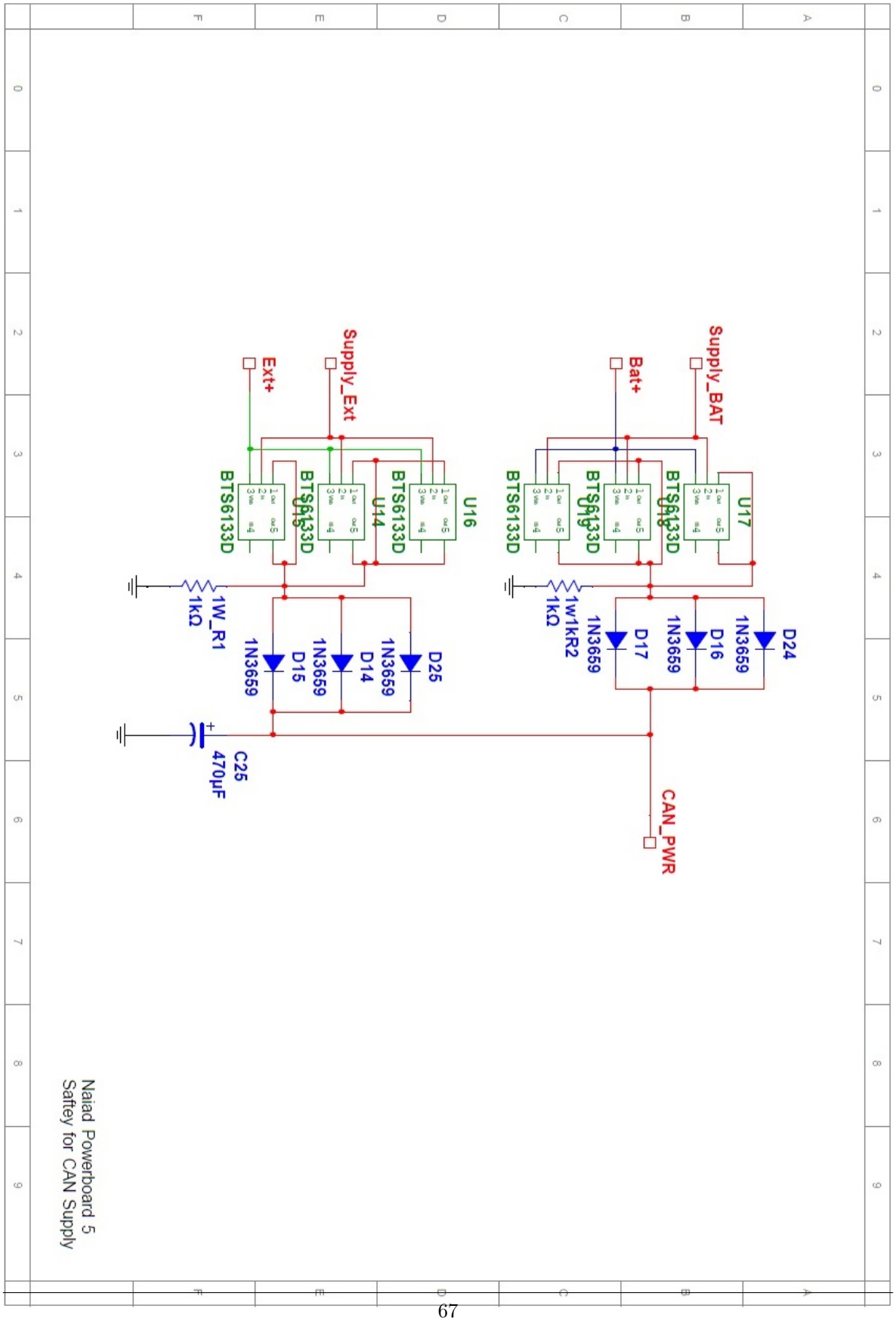


Figure 49: Saftey for CAN bus for Powerboard

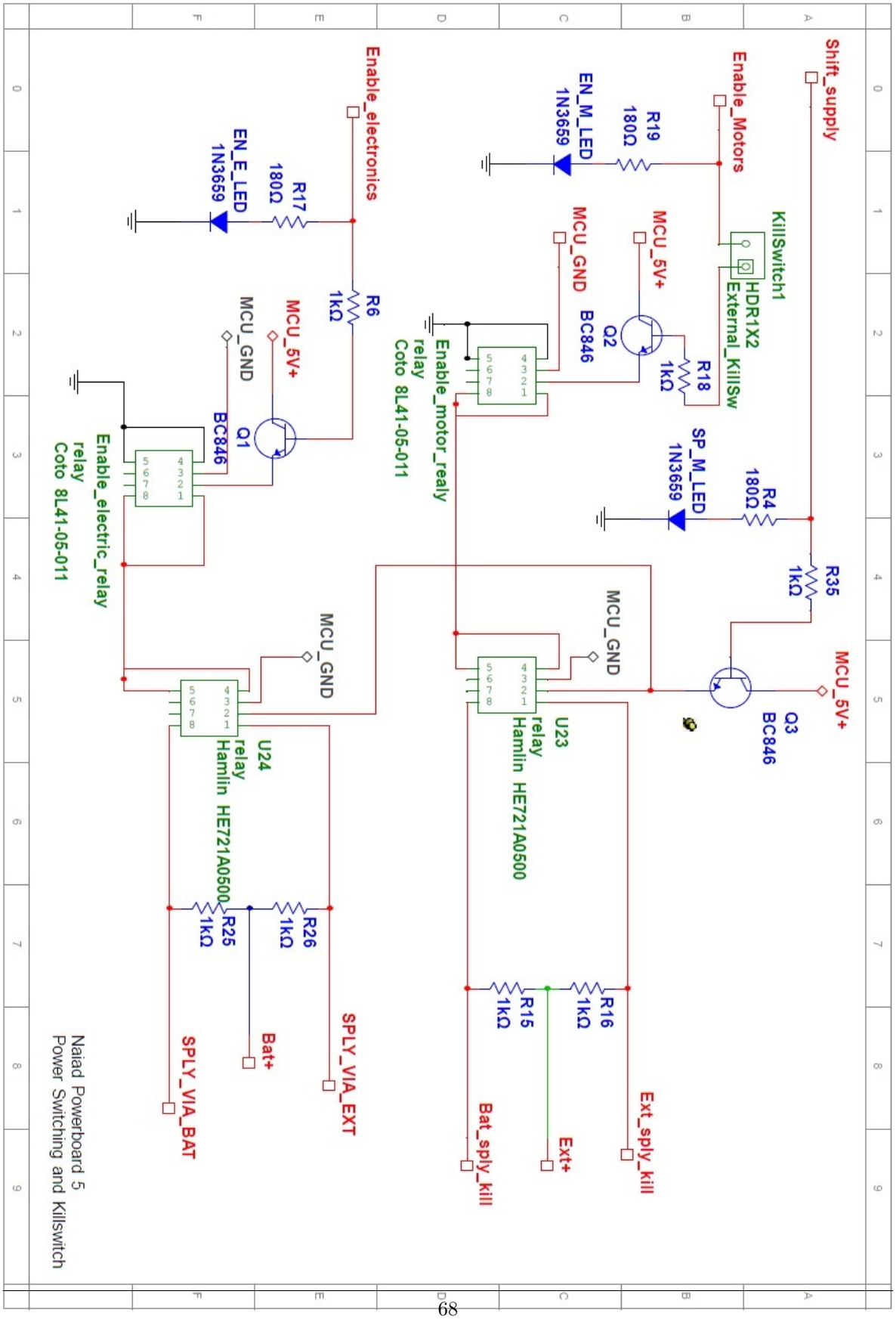


Figure 50: Control what supply to use and kill switch Powerboard

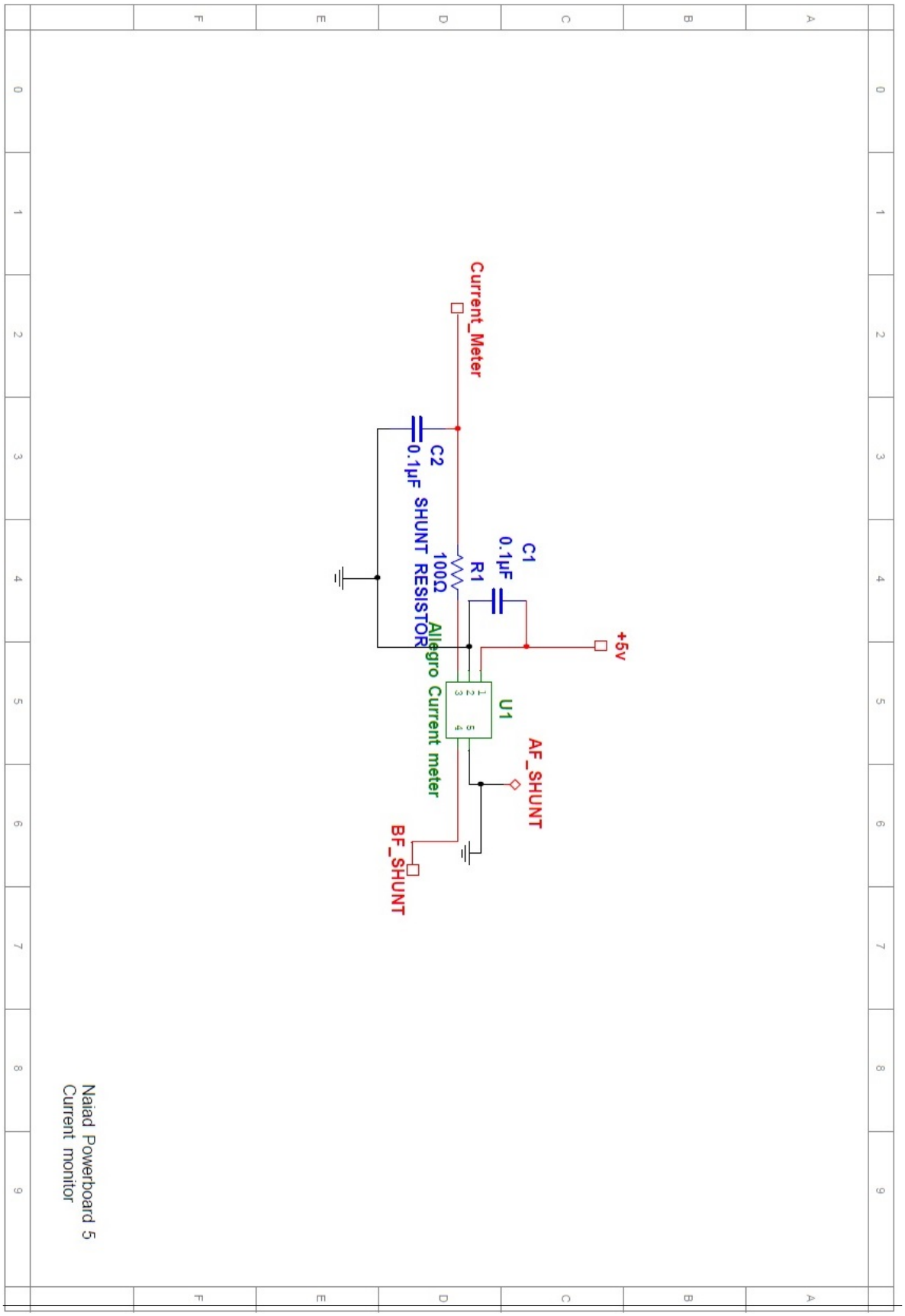


Figure 51: Current sensor to detect how much power the system use

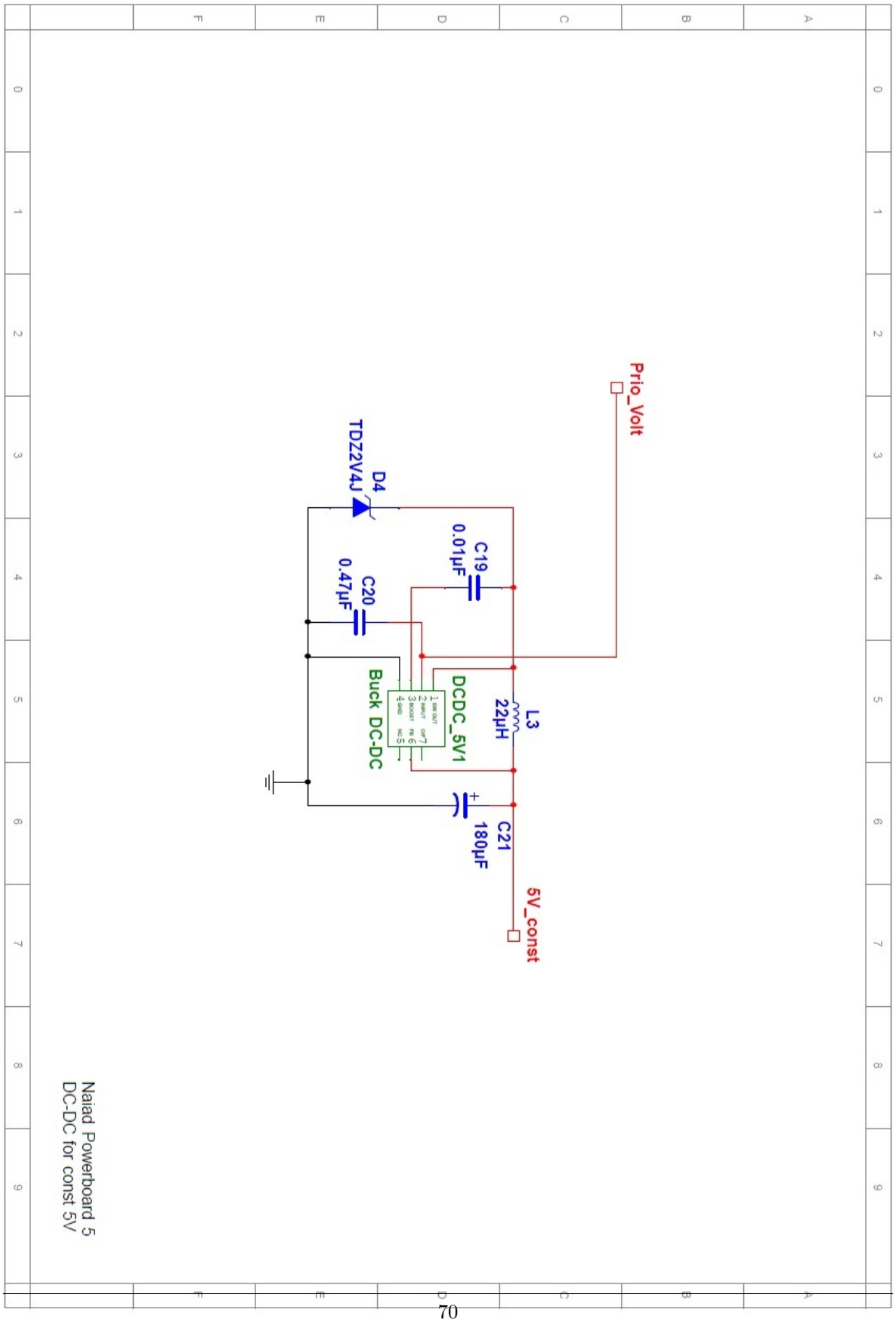


Figure 52: DC-DC for the constant 5V trace on Powerboard

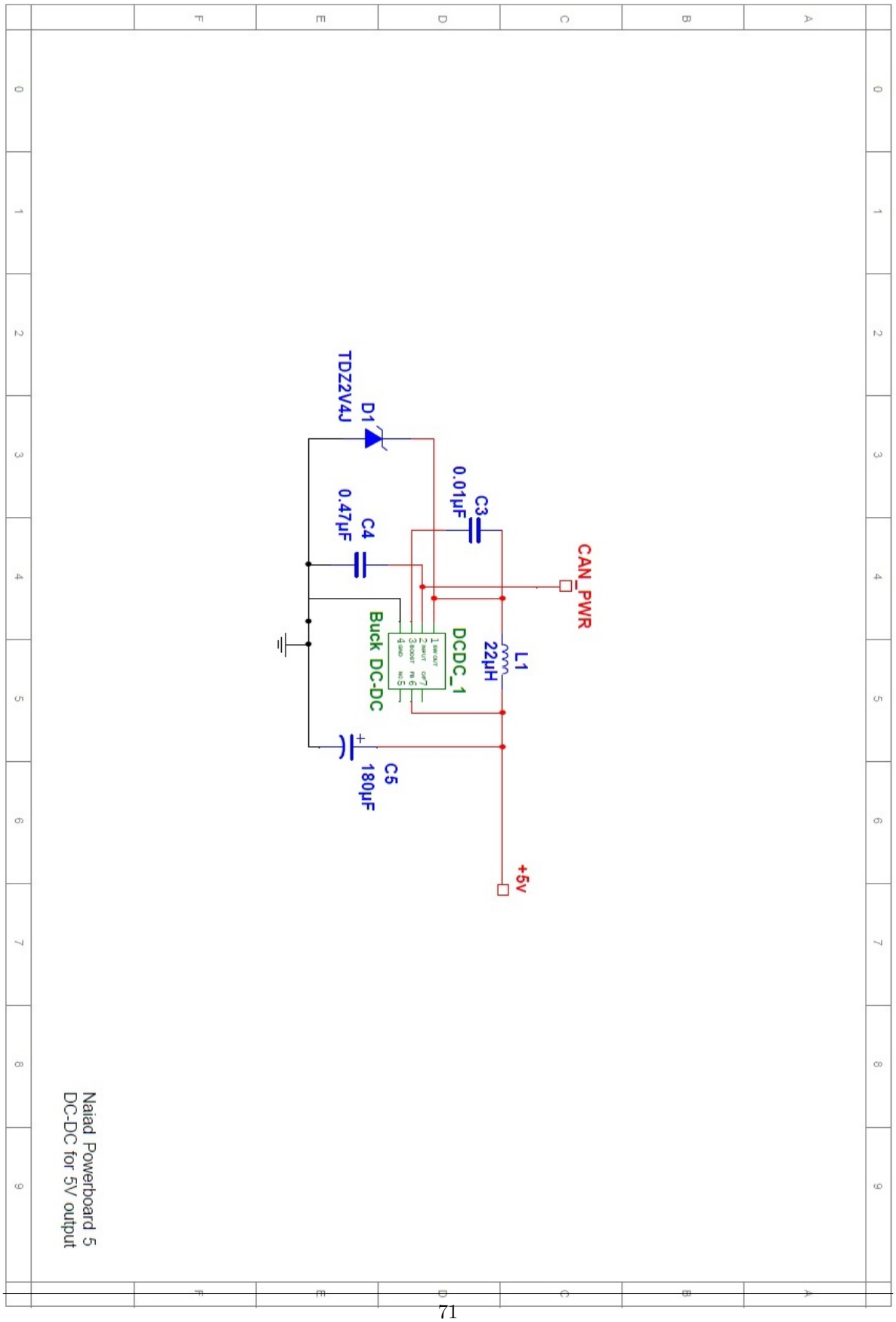


Figure 53: DC-DC for the 5V output on Powerboard

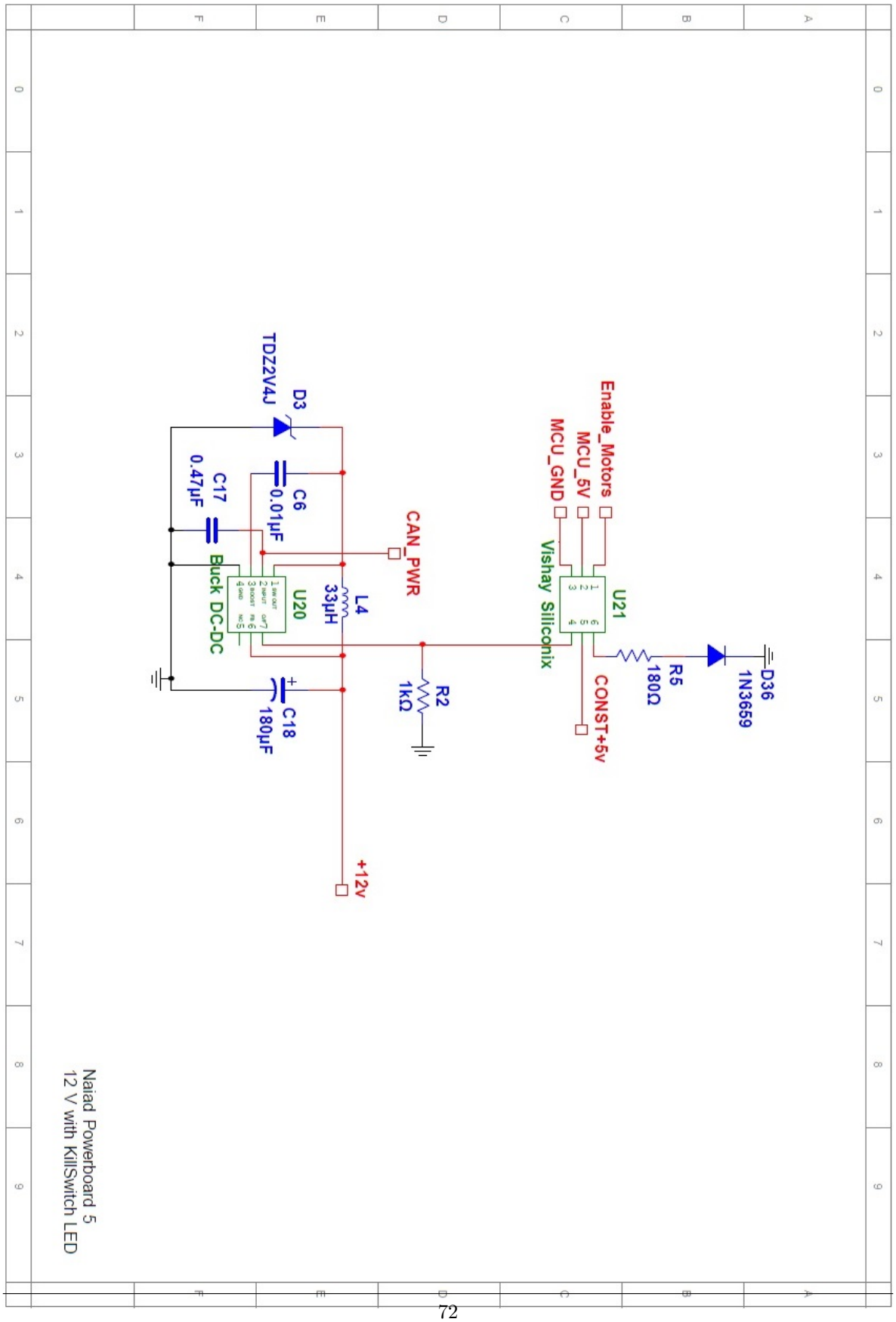


Figure 54: DC-DC for the 12V output on Powerboard

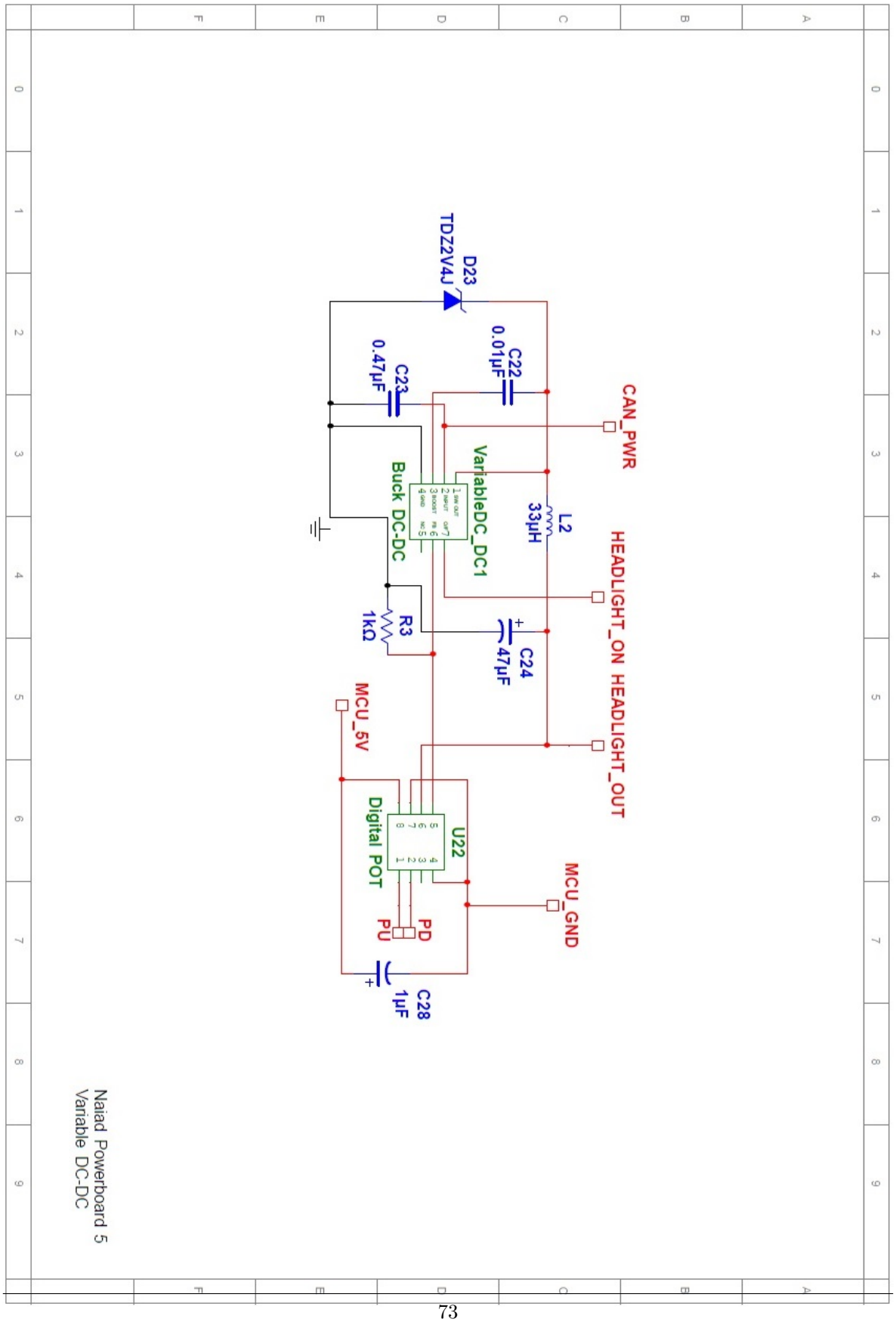


Figure 55: DC-DC for the variable output on Powerboard

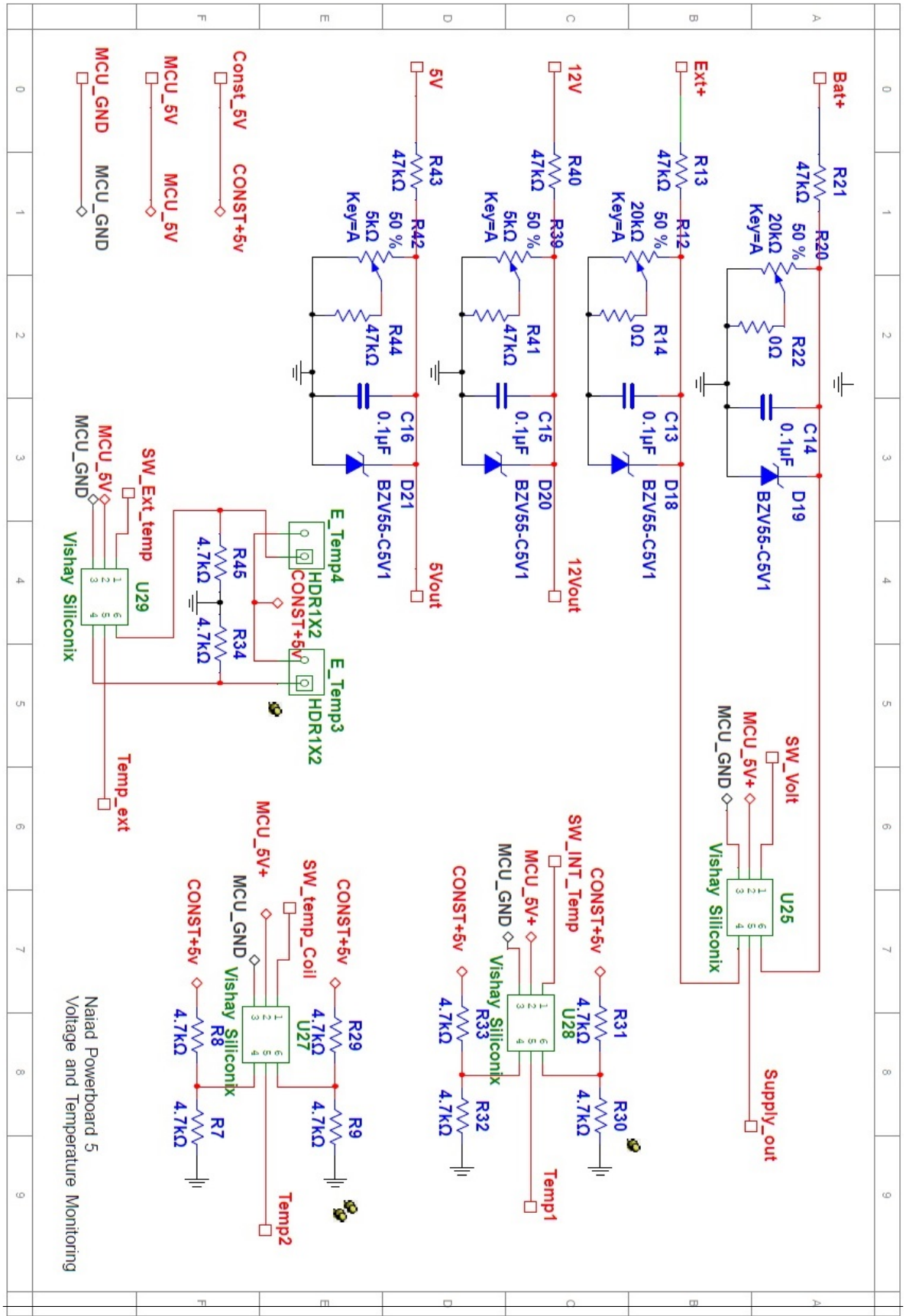


Figure 56: Voltage and temperature monitor for Powerboard

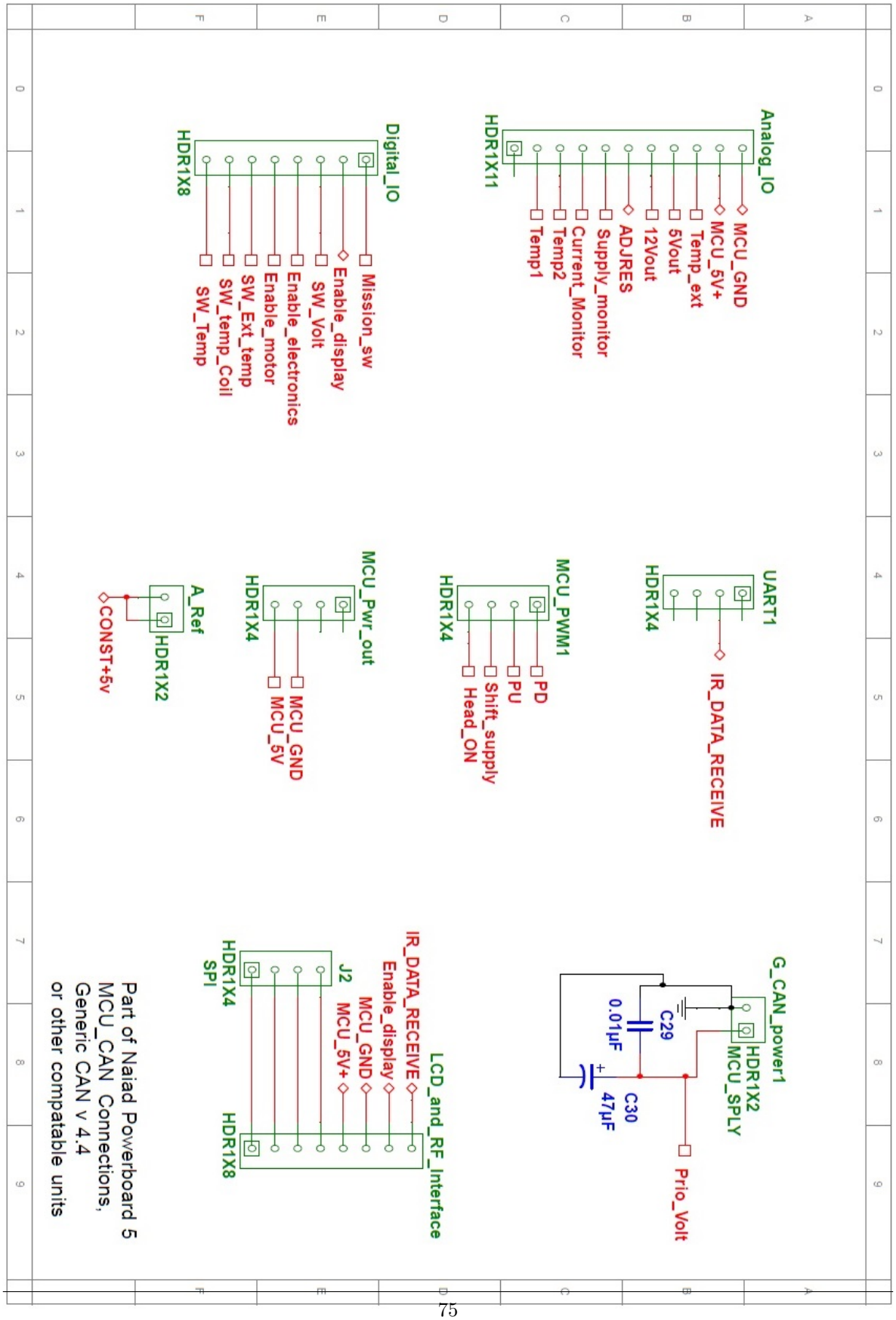


Figure 57: Connection with MCU CAN card on Powerboard

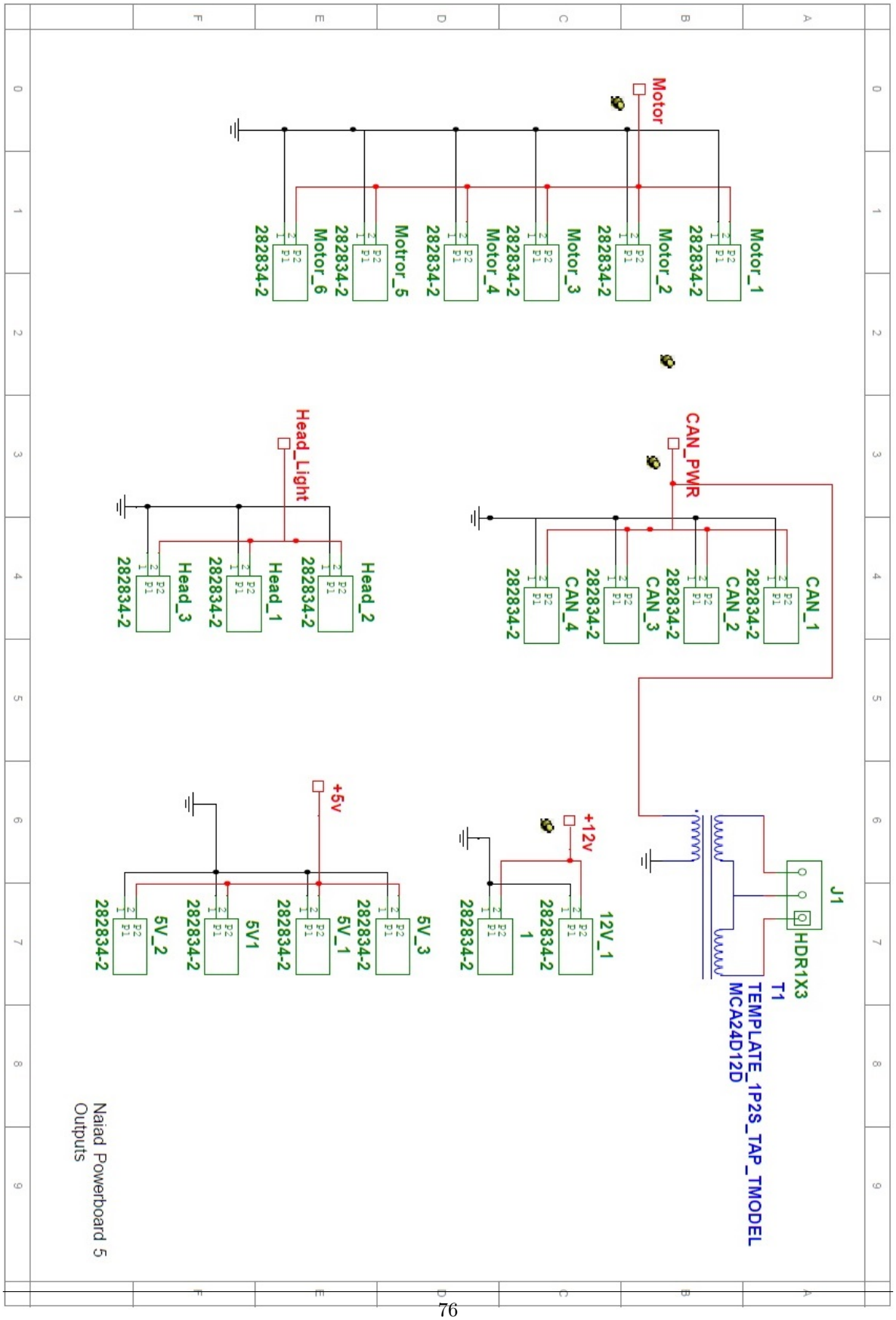


Figure 58: Outputs on Powerboard

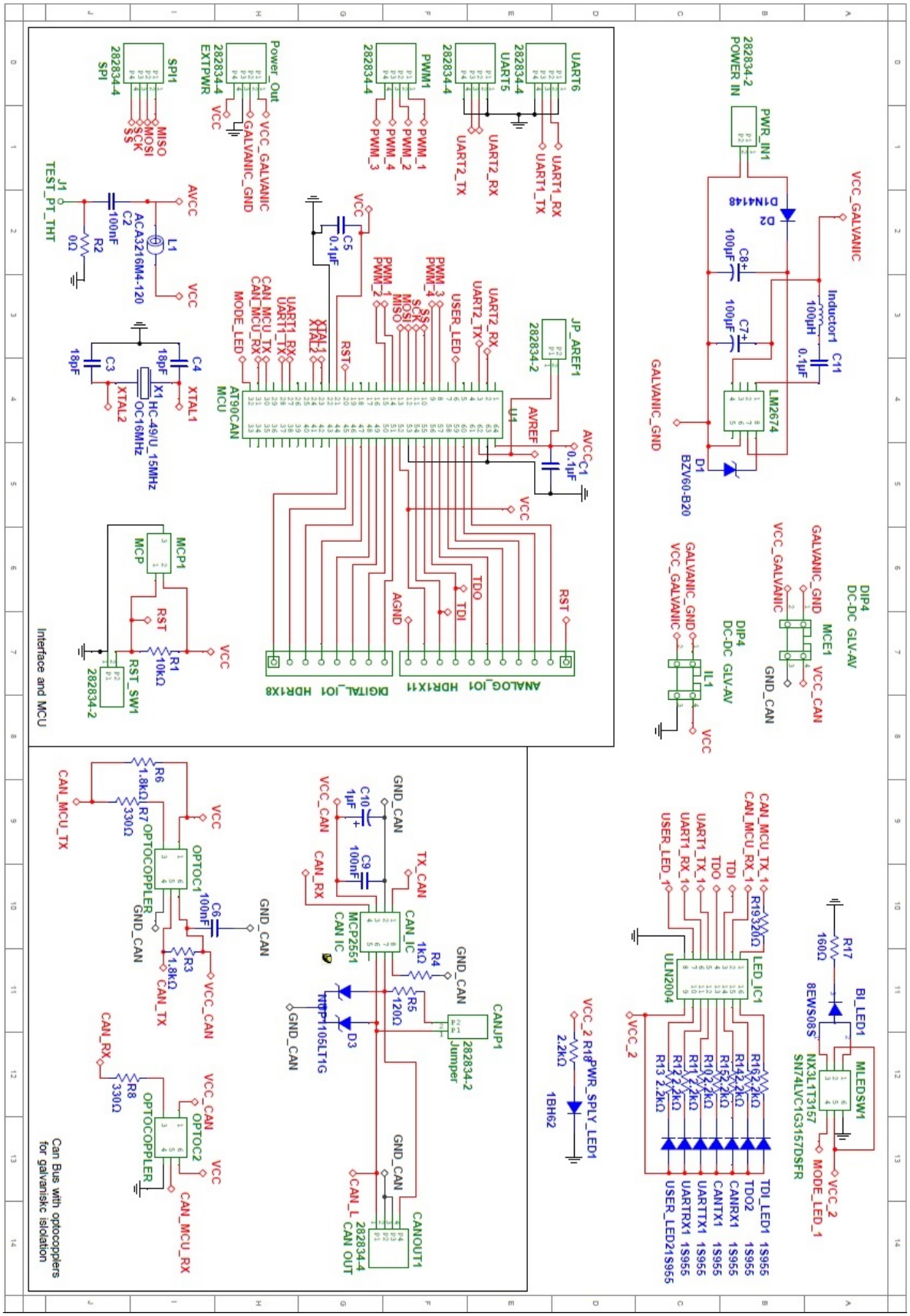


Figure 59: Generic CAN-card 4.4

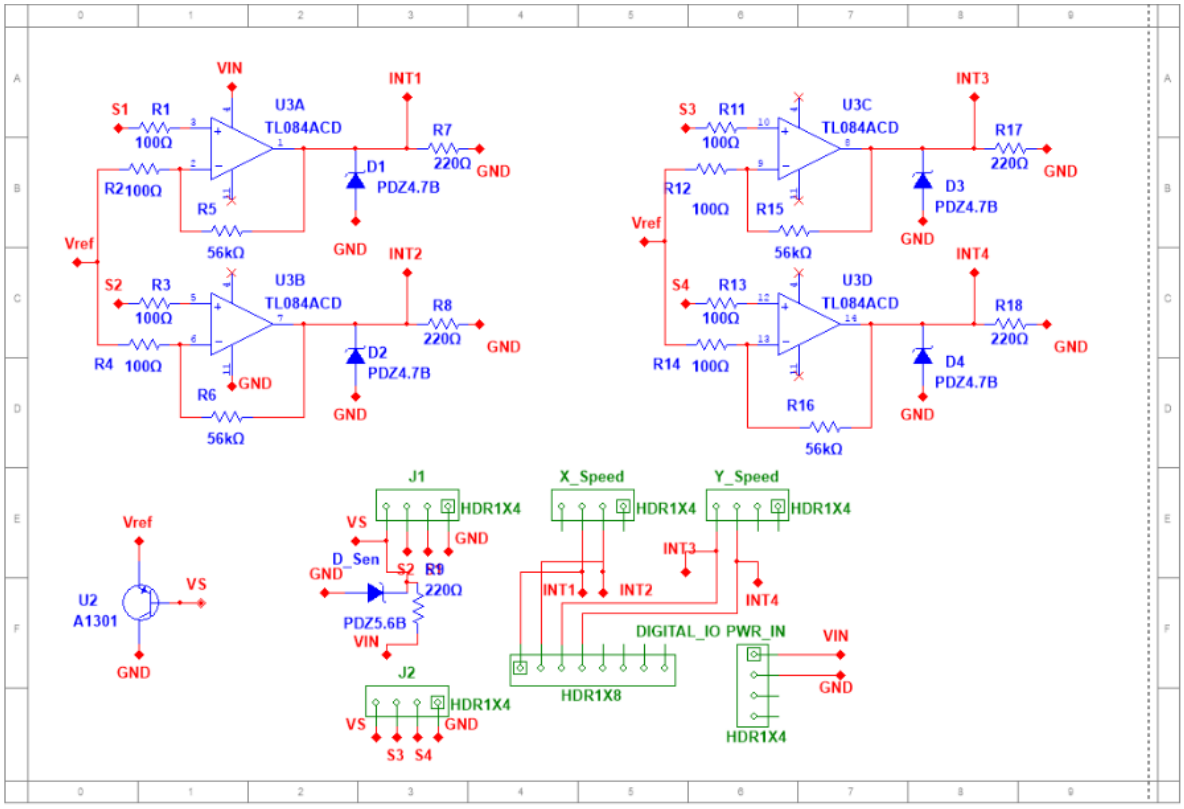


Figure 60: Speedlogger sensor card

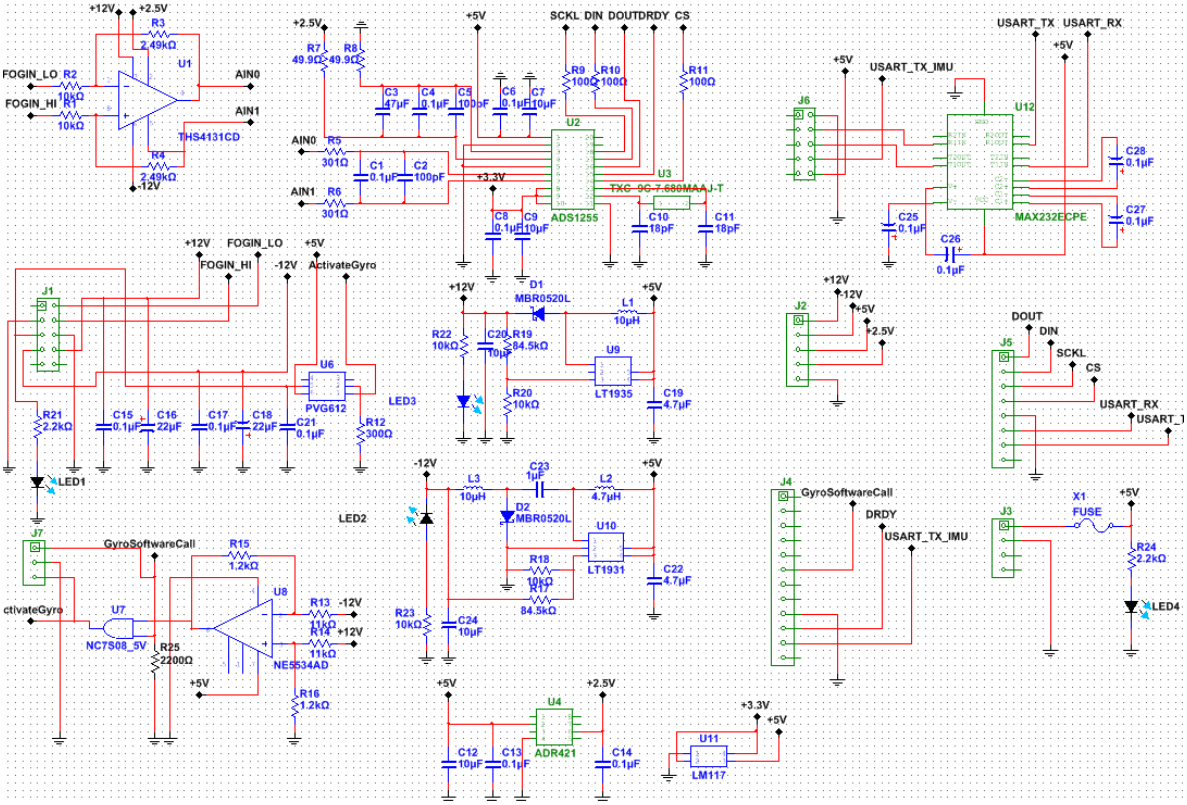


Figure 61: Speedlogger sensor card

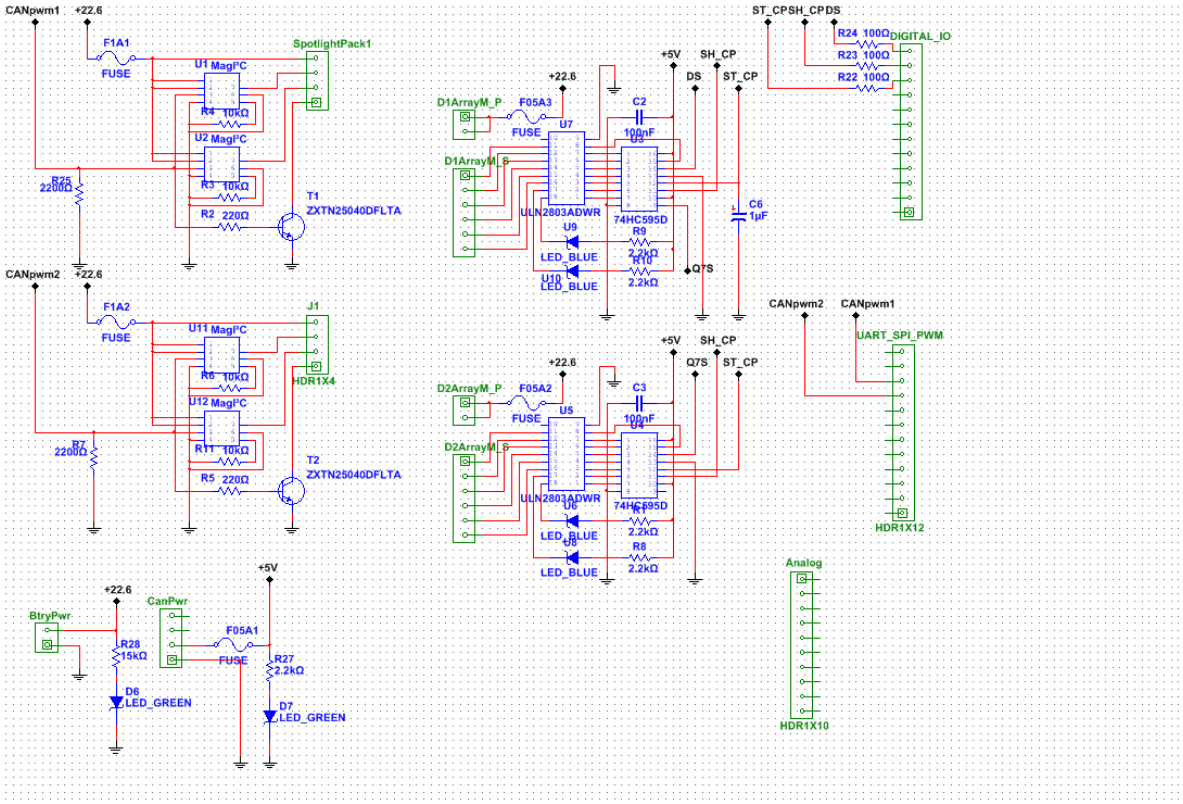


Figure 62: LEDcontrol-board

B System test

Introduction

The goal for the tests described in this test plan is to get the Naiad robot ready for competing in RoboSub 2015 in San Diego, USA. In order to easier find the problem if one should occur, some tests to find the problem has been constructed. These tests can be looked at as sort of a flow chart of what to do if something is not working as it is supposed to.

B.1 The vision system is not working properly

Purpouse of the test case

The purpouse of this test case is to find the problem when the GIMME-2 board is not working properly.

Description

There are a number of different problems that can occur when using the GIMME-2 in Naiad. This test is designed so that finding the problem should be easier.

Resources

Electronics of Naiad including at least one GIMME-2 board with lenses on the camera sensors and a multi-meter.

Preconditions

Before starting this test the electronics should be powered up and the system should try taking and analysing pictures from the GIMME-2, however for this test to be relevant there has to be a problem somewhere in this process.

Post conditions

The goal for this test is that the GIMME-2-system should be working properly.

Flow of events

First thing to check is that the GIMME-2 is powered correctly. Check this using a multi-meter, there should be 12 V where the GIMME-2 is powered see fig. 63. If there is no power to the GIMME-2 card, make sure to check the connections back to the batteries from the GIMME-2. Make sure to check the cables and the power board for the GIMME-2. Important to remember when checking this power board is that the card is inverted so that the plane is Vcc and that the trace is ground. The input to this board should 24 V and the output 12 V.

If all of these seem to be correct, the problem is probably with the GIMME-2 board, this is not easy to fix, replace the card. If the card seems to be working but the vision system still is not, the problem might be in the software for the vision system.

Check point	Measured
Measure power input in GIMME-2	
Check cables	
GIMME-2 power board	

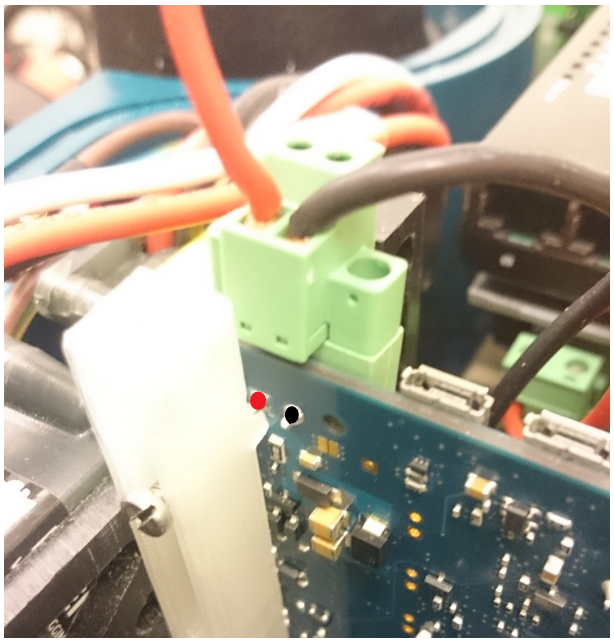


Figure 63: Measure on the black and red dot, there should be 12 V between them.

Inclusion/Exclusion points

This test counts as a pass if the problem can be found and the GIMME-2 board starts working. All other results will count as fails.

Special requirements

There are no parameters for this category.

B.2 Markers are not releasing as they are supposed to

Purpouse of the test case

The markers might not release as they are supposed to, the purpouse of this test is to find out why this problem occurs.

Description

This test is designed to find the reason to why the markers are not releasing properly.

Resources

Complete system including electronics and mechanics.

Preconditions

For this test case to be relevant the markers should not be releasing as they are supposed to.

Post conditions

The markers release when supposed to.

Flow of events

The first thing to check is that the markers are free to release. The next thing to check is to make sure that the solenoid is connected properly. Make sure to also check that the actuator shield is connected properly to its CAN card and that the CAN card is connected properly to the stack. After doing all of these things the next thing is to make a complete unit test of the actuator shield, and if that does not work a unit test on the CAN card should be done.

Check point	Measured
Marker can release properly	
Solenoid connection	
Actuator shield connection	
CAN connection	
Actuator shield unit test	
CAN card unit test	

Inclusion/Exclusion points

There are no exclusion points to this test.

Special requirements

There are no special requirements for this test.

B.3 A sensor is giving out a faulty value

Purpouse of the test case

There are a number of sensors on the Naiad system, the purpouse of this test is to identify the problem when one of them is giving out a faulty value.

Description

It might happen that one of the sensors give out a fault value, which could make the AUV act inappropriately. This test is designed to ensure how to find the problem if that were to happen.

Resources

Electronics of Naiad including the affected sensor(s).

Preconditions

A sensor in the system is giving out a faulty value when reading it in software.

Post conditions

The desired post condition is that the sensor works properly.

Flow of events

The thing to start with is checking that the sensor is connected properly, that the sensor shield it is attached to is mounted on its CAN-card properly and that the CAN-card is properly connected to its stack. If all of this is properly connected and the sensor is still giving out a faulty value, turn to the respective unit test of that sensor or the card it is mounted to. If the problem is still not resolved, make sure to do a complete unit test on the CAN card the sensor is connected to. The unit test for the CAN cards can be found in appendix D.

Check point	Measured
Sensor connection	
Shield connection	
CAN connection	
Sensor shield unit test	
CAN card unit test	

Inclusion/Exclusion points

This test does not test the sensors, it is just testing the connections and the mountings.

Special requirements

No special requirements for this test.

B.4 System does not start

Purpouse of the test case

The purpouse of this test is to find out what is wrong if the system will not start when you power it up.

Description

This test was created to ease problems with a system that does not start as it is supposed to.

Resources

Complete Naiad with electronics, batteries and hull, a multimeter,

Preconditions

The robot is not starting when the batteries are plugged in.

Post conditions

For passing this test, the system should start after this test.

Flow of events

Start by checking the connection from the batteries to the power cords. It they are connected continue with checking the power level on the batteries. This can be done using a multimeter. The batteries should measure above 22 V, if using a battery monitor all cells should be indicating green. If the voltage is running low, try charging the batteries and then start up the system. If the charger will not charge a battery that could mean that a cell is damaged, discard the battery.

The next step is to check the Power Supply Unit (PSU), the first thing to look for is the fuse on the PSU. This is marked in fig. 64 If the fuse is broken, make sure to replace it with a new one. Test if

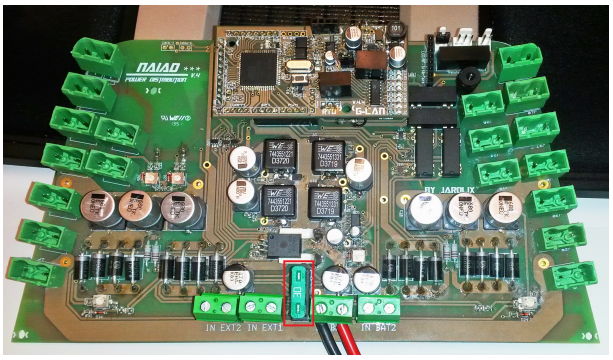


Figure 64: The red square indicates where the fuse can be found on the PSU

the system will start again. If this did not work, make a complete unit test of the PSU. The complete unit test of the PSU can be found in appendix E.

Check point	Measured
Battery connection	
Power level batteries	
Fuse status	
PSU unit test	

Inclusion/Exclusion points

This test will be counted as a pass if the system starts, all other results will count as fails.

Special requirements

There are no special requirements for this test.

B.5 Is the hull water tight

Purpouse of the test case

The purpouse of this test is to make sure that the hull is waterproof and that it will not leak when put in water.

Description

Testing if the hull is watertight is a continuation from the unit test of the hull. This test can be found at C.2.

Resources

Required resources for this test is the hull of Naiad, some paper, petroleum jelly, a long rope, and weights (for example in a weight belt for diving). A pool to sink the hull in is also required.

Preconditions

The preconditions for this test is to fill the inside of the hull with toiletpaper to easier see if there is any intake of water. The hull should then be closed according to the specification found in C.2.

Post conditions

The post conditions are the same as the preconditions.

Flow of events

Enough weights to sink the hull should be attached to the hull, make sure that the weights can not fall of when the hull moves around. It should be sunk to the bottom of the water and then left there for at least five minutes. Then it could be lifted from the water and dried on the outside. The lid should then be opened and the inside examined to see that there is no water on the inside.

Check point	Measured
Closed properly	
Weights attatched	
Sunk	
Lifted	
Outside dried	
Inside dry	

Inclusion/Exclusion points

The depth to which the hull is waterproof is not tested in this test.

Special requirements

There are no special requirements for this test.

B.6 Motor not starting

This case is a common problem in which one or more motor(s) for some reason is not starting.

Purpouse of the test

The purpose of this test is to identify the problem with the motor and find out why the motor is not running.

Description

The test is designed to identify any problems with why a motor is not running.

Resources

One person is required to perform the test.

Preconditions

The motor is not running on command.

Post conditions

The desired post condition of this test is a motor that is running.

Flow of events

When the motor is not working that means one of two things. Either the motor is completely quiet or the motor gives out a 2Hz beep-signal.

If the motor(s) is completely quiet that would imply that there is no power to the motor. In that case all the cables to the power supply needs to be checked. These cables are thick, one is red and the other one is black, and will come from the motor controller and should be plugged in to the PSU. Also the cables to the motors need checking, that is three cables, they are red, white and black from the motor controller and should match the shrink tubing on the cables going out to the motors. When checking the cables - make sure that the conwtb-female has not started to be loose, in this case the contact needs to be switched to a new one.

If the motor(s) are giving out a beeping signal at 2Hz, that would mean that there is no signal to the motor(s). In this case the signal cable need checking. This cable is orange/red/black and run from the motor controller to the motor extension board. Make sure that the motor extension board is properly connected to the CAN-card. Also make sure that the CAN-card is properly connected to the stack.

If all the cables and cards are plugged in as they are supposed and the system is still quiet this could mean that there is no power in the system, in this case, nothing should be working and all LEDs should be turned off. In this case, check that the batteries are plugged in and that they are charged properly. How to do this is better specified under B.4.

If there is power in the system even if the motor is silent and all the cables are connected correctly then this would mean that the PSU is not working properly. To test this card, use the complete unit test for the PSU found in appendix E.

Check point	Measured
2Hz beeps	
Power cables to motor controller	
Cables from motor controller	
conwtb-female	
Signal cable to motor controller	
Motor extension board	
CAN-card connection	
System power	
PSU unit test	

Inclusion/Exclusion points

This test does not test wether the motor is working or not. If the test is failed one could further test this.

Special requirements

There are no special requirements.

C Mechanic tests

Following is a series of tests that should be performed on the mechanical parts of the system.

C.1 Testing the o-ring

Purpouse of the test case

The purpouse of this test is to make sure that there are no dents in the o-ring, which would make it unsafe to use under water.

Description

For this test a person will feel through the o-ring to make sure that there are no dents in it.

Resources

O-ring, petroleum jelly

Preconditions

The preconditions are that the o-ring is dry and grease free.

Post conditions

After preforming this test the o-ring should be greased and for a pass it should have no dents. Any dents will count as a fail in this test.

Flow of events

For this test the person performing the test should have some petroleum jelly on their fingers and gently feel every piece of the o-ring to make sure that there are no dents in it. Make sure that the entire o-ring is properly greased.

Check point	Measured
Check for dents	

Inclusion/Exclusion points

The test will count as a pass if the o-ring is free of dents and can be completely greased. All other cases will count as a fail.

Special requirements

There are no special requirements for this test.

C.2 Closing of the hull

Purpouse of the test case

This test is supposed to make sure that the hull is water proof when closed.

Description

The test is done to make sure that the lid is closed properly to make sure there is no leakage when the AUV is put into water.

Resources

Required resources for this test is the hull of Naiad, some paper, petroleum jelly, a long rope, and weights (for example in a weight belt for diving).

Preconditions

There are no preconditions for this test.

Post conditions

After preforming this test the lid of the hull should be closed and the hull should be waterproof.

Flow of events

Firstly, the hull should be fitted with either the electronics, for an actual run with the robot, or with paper for testing the waterproofness of the hull.

Then the o-ring should be tested, this is done through feeling the o-ring all the way around using petroleum jelly to grease it thoroughly, for a closer description of this see C.1. Then it should be placed in the o-ring slot which before hand should be greased as well. Then the lid can be placed and closed to ensure waterproofing. When placing the lid, make sure that no cables are being pinched between the hull and the lid. Also make sure to place the lid from straight above so that the o-ring stays in place.

Check point	Measured
Test of o-ring	
Placement of o-ring	
Placement of lid	
Cable check	

Inclusion/Exclusion points

This test will not test which pressure the hull can sustain, it will just test that there will not be any immediate leakage.

Special requirements

There are no special requirements for this test.

D Unit test for CAN 4.4 used in Naiad

Unit test for the generic can card fitted in Naiad. **This test only test the basic electronics.** To fully test the card it is required to program it with a JTAG adapter and some kind of programming environment. The JTAG adapter is connected in to the analog-IO list on the left edge and are configured according to Figure 65. This document can be used also in order to understand the card and below there is a list over what the LEDs in the bottom right corner mean.

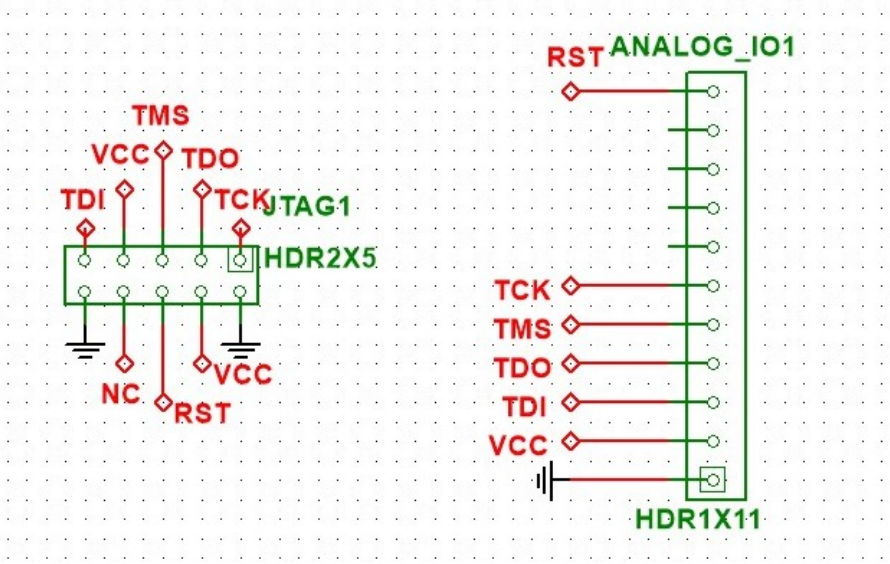


Figure 65: The JTAG adapted needed to program the card.

The card is divided in to three galvanically isolated fields. The fields are marked in Figure 66. One for the processor and peripheral electronics (marked red), one for the CAN bus (marked green) and one for power Input (marked yellow). The two yellow field are connected on the back.

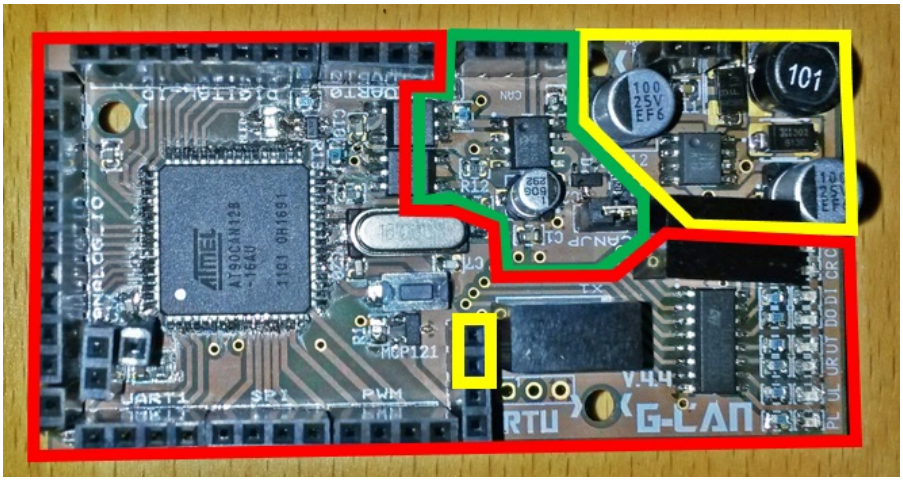


Figure 66: The CAN cards different fields marked, the fields are galvanic isolated

In the lower right corner there are a list of LED's the silk screen can be quite cryptic here is a

explanation.

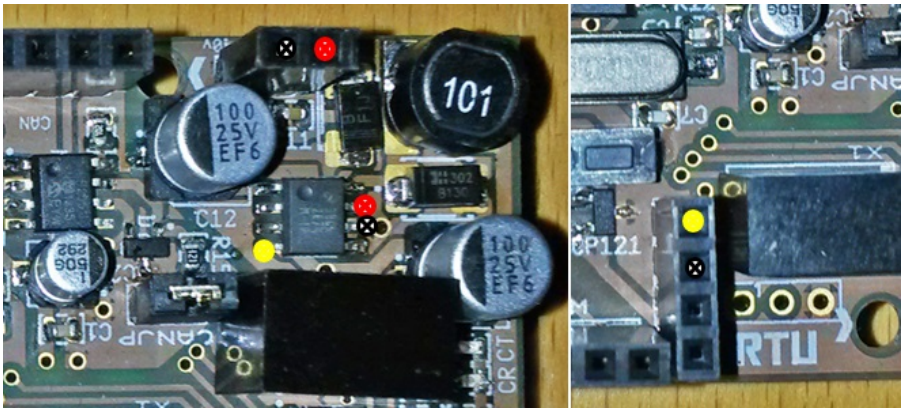
- CT = CAN Transceiver, light up when a CAN message in sent
- CR = CAN Receive, light up when a CAN message in received
- DI = TDI, has to do with the JTAG, lights up when the card is being programmed
- DO = TDO, has to do with the JTAG, lights up when the card is being programmed
- UR = UART Transceiver, light up when a UART message in sent
- UR = UART Receive, light up when a UART message in sent received
- UL = User LED, programmable LED that can be set by setting leg 6 on the processor. (In version 5 (also called motor CAN) this LED has been moved to pin 14)
- PL = Power LED should light up as soon as power is turned on

D.1 Required resources

- Card for testing
- Power supply, 12-48V
- Multi-meter

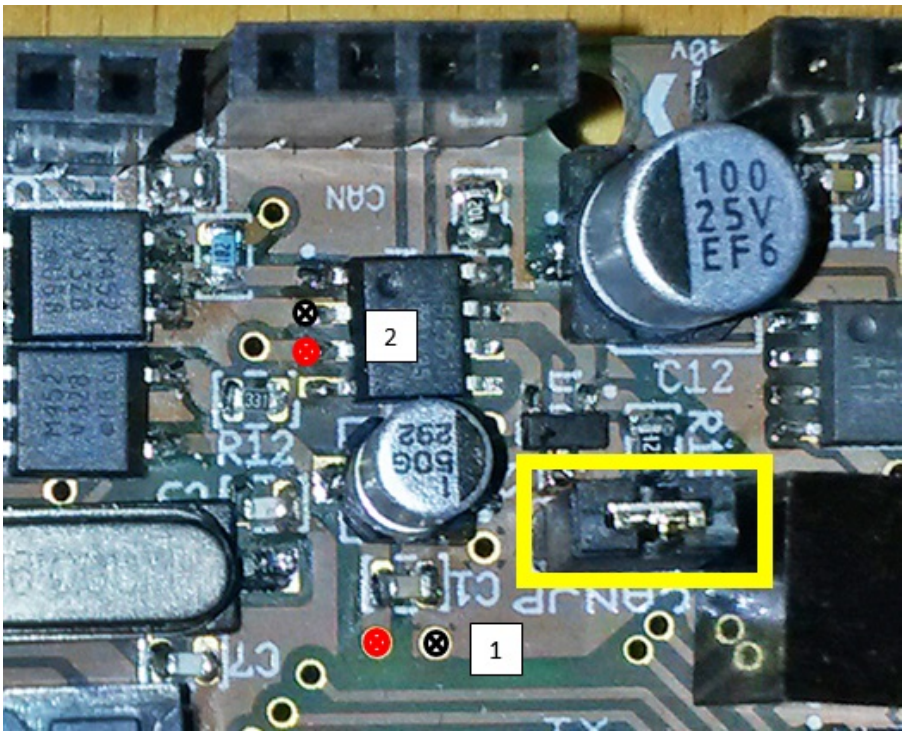
D.2 Input

The input consists of some protective circuitry and a switched regulator that takes the input and switches it down to 12 V. Connect the input voltage (12.6-48V) to the break way header on the top right corner. Ground is the left connector.



Point/Net	Pre-condition:	Desired	Measured	Commentary
GND		0V / 0Ω		Check that the GND on the card and supply are the same
Input	Supply with 12.6-48V	Input voltage 12.6-48V		As soon as the supply is connected the bottom LED in the bottom right corner should light up. If does not there is some errors on the card. Make sure the supply is correct. Then check the inputs to the DC-DC (the two middle legs on the right side)
Output	Supply voltage	12V		Measure between the same black as above and the yellow dot. If there is no output here it is most likely the DC-DC that's broken try to change that. Measure also in the breakaway header in the lower middle part of the card. If there is 12 V up in the corner and not here there is a problem with the traces on the back

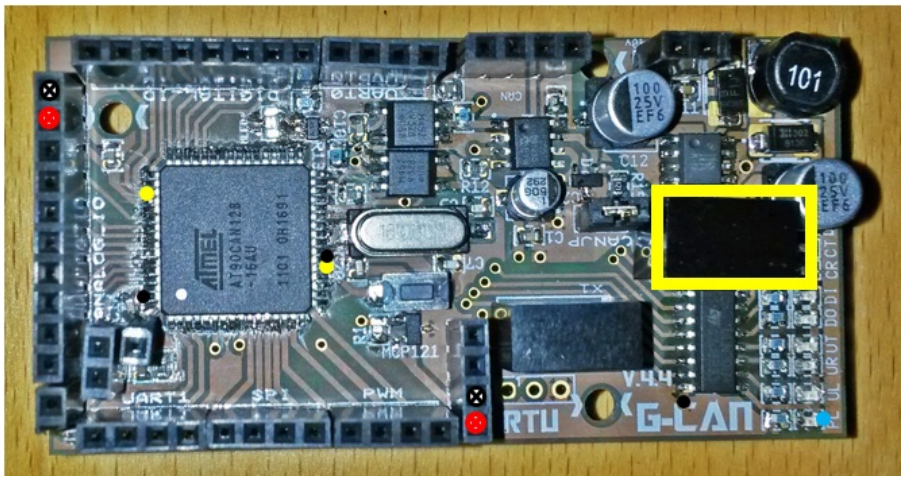
D.3 CAN power



Point/Net	Pre-condition:	Desired	Measured	Commentary
Can VCC	Supply voltage	5V		If there is no power between the dots (1) there is either the trace from the DC-DC or the DC-DC **below** (not in picture) that is broken. Check the traces and then try to replace the DC-DC
Supply for CAN	Supply voltage	5V		Check that the CAN circuit get its required input (2)
Jumper	-	-		It is not necessary to have jumper on all nodes but it is preferred. (marked yellow)

D.4 Processor with peripherals

The rest of the cards and the 5V output for shields are driven by the DC-DC marked in the picture in yellow.



Point/Net	Pre-condition:	Desired	Measured	Commentary
VCC	Supply voltage	5V		Measure both big red/back dots. If none are 5V measure leg 3 and 4 on the yellow marked DC-DC (3 is GND), replace if incorrect voltage here to. If one of the pair is 5V and not the other there is a broken trace somewhere.
Power LED	Supply voltage	LED on / 5V		The LED on the bottom in the bottom left corner should light up. If it doesn't check that the voltage between the blue dot and the closest small black dot is 5V. If so the LED or resistor is probably broken, try to change these. If not the trace between the LED and the DC-DC is probably broken.
Supply for MCU	Supply voltage	5V		Measure the small yellow and black dot located on the processor. (2^{nd} from the bottom and 3^{rd} from the top on the left and 5^{th} and 6^{th} from the bottom on the right)

E Unit test Naiad Power-board 4.4

This is a unit test for The Naiad power-board 4.4. This guide can also be used to understand the different parts of the power board. It covers most of the power board but does not go in to detail to check the temperature and voltage monitoring featured on the card.

E.1 Required resources

- Power board to test
- Power supply, 22-24V
- Multi-meter
- Working CAN-card with software that sets digital 3,4 high and control PWM 1, 2, 3 as well as take input on digital 7.

E.2 Input and prio-electronics

Plug in the card in at least one of the four supply terminal block located in the bottom of the card to a suitable source. 22-24V with minimum of 500 mA for testing is sufficient. **The ground is always the lead *closest* to the fuse!** You need to flip the cables if plugged on to Ext1 or Ext2 according to figure refinput.

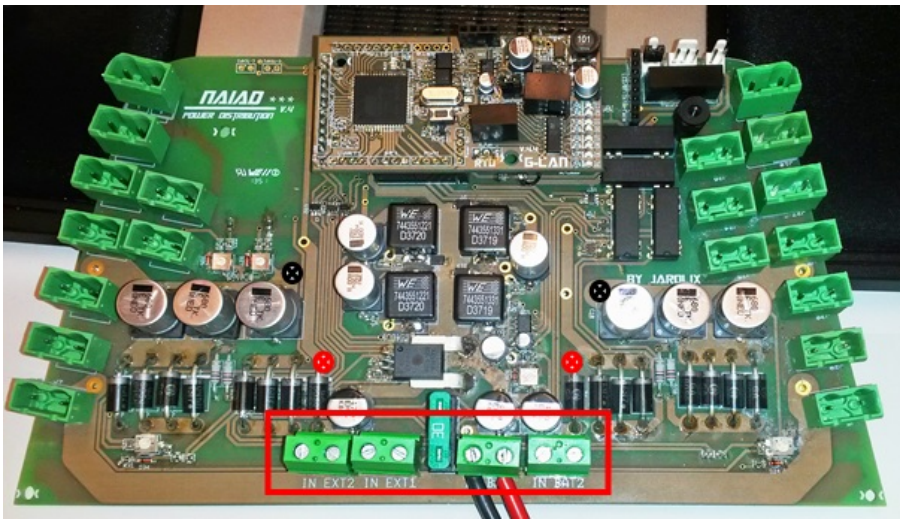
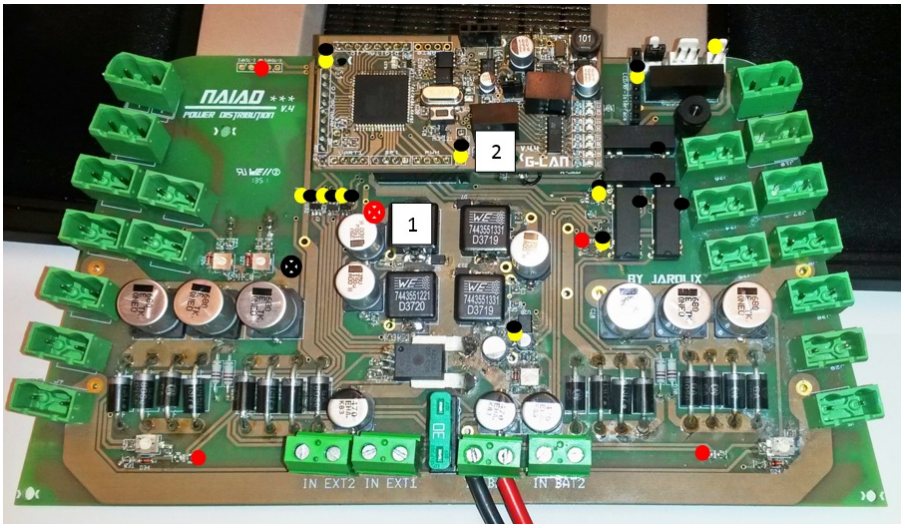


Figure 67: The caption you want with the picture

Point/Net	Precondition:	Desired	Measured	Commentary
GND	Supply with 22-24 V and 500 mA	0V / 0Ω		Make sure the black dots and the ground on the supply are connected. If not, discard the card.
Input	Supply with 22-24 V and 500 mA	Input voltage 22-24V		Make sure the That the voltage is correct and then that the wires are in the correct socket on the terminal block. If all these are correct check the Fuse.
Fuse	-	30 A Car fuse		Try another Fuse. If still no power is supplied to rest of card check the two points marked.
Prio-volt	Supply with 22-24 V and 500 mA	Input voltage 22-24V		If there still is no power on the dots there is either a problem with the terminal blocks or the diodes next to the dots. Change these and try again. If there is power here but the CAN-card is not lit, check the trace leading to the CAN-card power supply for damage.

E.3 Required peripherals on PSU



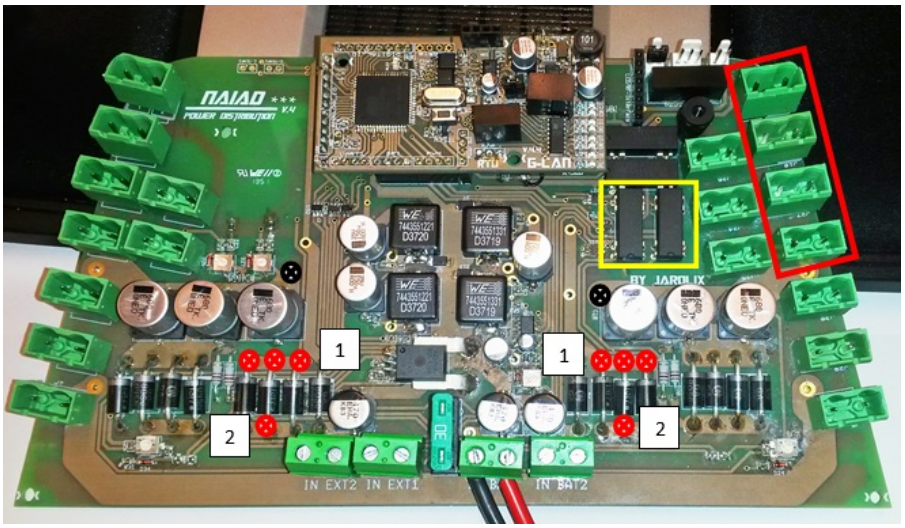
Point/Net	Precondition:	Desired	Measured	Commentary
Const 5V	Supply and Prio-volt works.	5V		This is used at several point on the card, but firstly check that the output is correct from the DC-DC regulator by measuring on point text to the box with a 1 in it.
Const 5V	Supply and Prio-volt works.	5V		Check the rest of the points smaller points, all should be 5V compared to the larger black it the above point but not one of these points a trace is broken.
MCU_VCC	Working CAN-card	5V		Firstly make sure that the CAN-card is working, then measure on the CAN-card between VCC out and GND, the small yellow and black next to the 2.
MCU_VCC	Working CAN-card	5V		Measure between all small black points and small yellow points, all should be 5V if the MCU VCC in 5V but not one of these points a trace is broken. (On the IC on the left it is the left most legs that should be measured, there is also another on the underside below the E-temp connectorś).

E.4 Kill switch / Mission switch

The mission switch sets digital pin 1 high.

Point/Net	Precondition:	Desired	Measured	Commentary
Mission switch	Working CAN-card	5V		Short the two pins on the mission switch. Then digital 1 should be 5V compared to MCU GND(the small red and black dot)
Kill switch	Supply with 22-24 V			When the pins are NOT shorted a red LED under the CAN-card should be lit. When the two connectors are connected the red light goes out. CAN-power is still on when the kill switch is disconnected but the motor power should be killed. Check these by measuring the CAN and motor output.

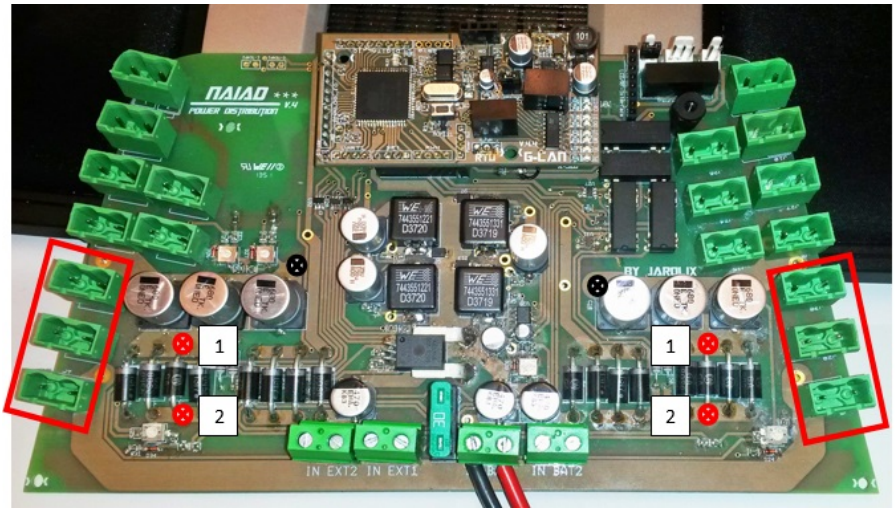
E.5 Can-Power



Point/Net	Precondition:	Desired	Measured	Commentary
Output	Software that enables Electronics	Input voltage 22-24V		Make sure it seems like the terminal blocks have contact with the trace. If they do try the points marked in the picture next to the 1
Can-power (dots 1)	Supply with 22-24 V and 500 mA	Input voltage 22-24V		If there is power here but not in the terminal block there is some kind of damage on the trace between this point and the output.
Can-power (dots 2)	-	Input voltage 22-24V		If there is power here but not at (1) there is something wrong with the Diode, change the diodes and try again. If not turn the card over and measure the switches.
Switch (marked yellow)		0Ω(or close)		The second leg need to be shorted to GND via a relay controlled by PWM 3. If it is not make sure the software is working and that the traces are undamaged. Try to ground the second leg or change the switches

E.6 Motor power

The power for the motors can be taken from the bottom three connectors on both sides of the card, these have to be activated in the software by setting digital 3 to high on the CAN-card.



Point/Net	Precondition:	Desired	Measured	Commentary
Kill switch				Make sure that the kill-switch's two pins are connected, the kill switch light should go out.
Output	Software that enables motors	Input voltage 22-24V		Make sure it seems like the terminal blocks have contact with the trace. If they do try the points marked in the picture next to the 1.
Motor Power (dots 1)	Supply with 22-24 V and 500 mA	Input voltage 22-24V		If there is power here but not in the terminal block there is some kind of damage on the trace between this point and the output.
Motor Power (dots 2)	Supply with 22-24 V and 500 mA	Input voltage 22-24V		If there is power here but not at (1) there is something wrong with the Diode, change the diodes and try again. If not turn the card over and measure the switches on the back

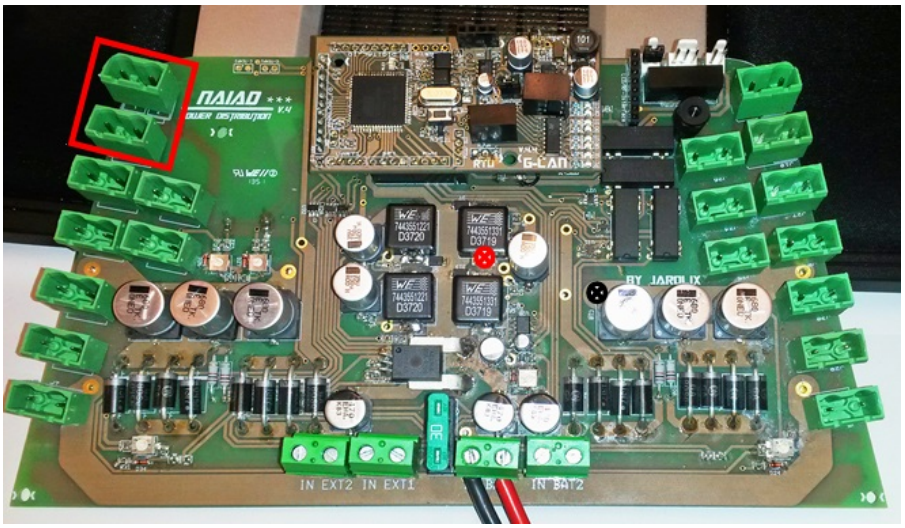
E.7 Headlight

These outputs are in the Ultiboard and Multisim file called Headlight but they are really a adjustable output voltage between 1.8 to 22V, all of the connectors have the same voltage however. They use the CAN-power which therefore need to work and be activated. To control these a digital potentiometer is changed by triggering pin PWM1 and PWM2. PWM1 decreases the voltage and the PWM2 increases one step for each pulse.

Point/Net	Precondition:	Desired	Measured	Commentary
Output	CAN-power working, Working MCU GND and MCU VCC	Input voltage 22-24V		Make sure it seems like the terminal blocks have contact with the trace. If they do try the point marked in the picture.
Headlight Power(dots)		Input voltage 22-24V		If there is power here but not in the terminal block there is some kind of damage on the trace between this point and the output. If not check DC-DC on the underside.
DC-DC		1:output voltage 2:22-24V 3:22-24V 4:0V 5:NaN 6:unkown 7:5V		If these are not correct check with the schematic for possible solution.

E.8 12 V

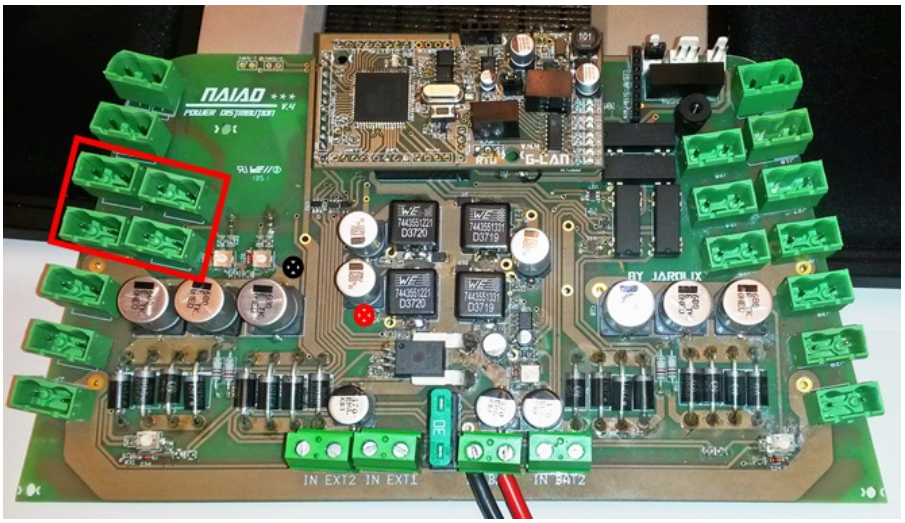
12 V output (however are more close to 11.6) are enabled together with motors and requires CAN-power to function.



Point/Net	Precondition:	Desired	Measured	Commentary
Output	Can-power working and kill-switch activated.	12V		Make sure it seems like the terminal blocks have contact with the trace. If they do try the point marked in the picture.
12V (dots)		Input voltage 22-24V		If there is power here but not in the terminal block there is some kind of damage on the trace between this point and the output. If not check DC-DC on the underside.
DC-DC		1:12V 2:22-24V 3:22-24V 4:0V 5:NaN 6:unkown 7:5V		If these are not correct check with the schematic for possible solution.

E.9 5 V

5V output are enabled by setting digital 4 high and requires CAN-power.



Point/Net	Precondition:	Desired	Measured	Commentary
Output	CAN-power working software that enables electronics (Digital pin 4 and PWM 3)	5V		Make sure it seems like the terminal blocks have contact with the trace. If they do try the point marked in the picture.
5V (dots)		Input voltage 22-24V		If there is power here but not in the terminal block there is some kind of damage on the trace between this point and the output. If not check DC-DC on the underside.
DC-DC		1:5V 2:22-24V 3:22-24V 4:0V 5:NaN 6:unkown 7:5V		If these are not correct check with the schematic for possible solution.

F Mechanics exploded views

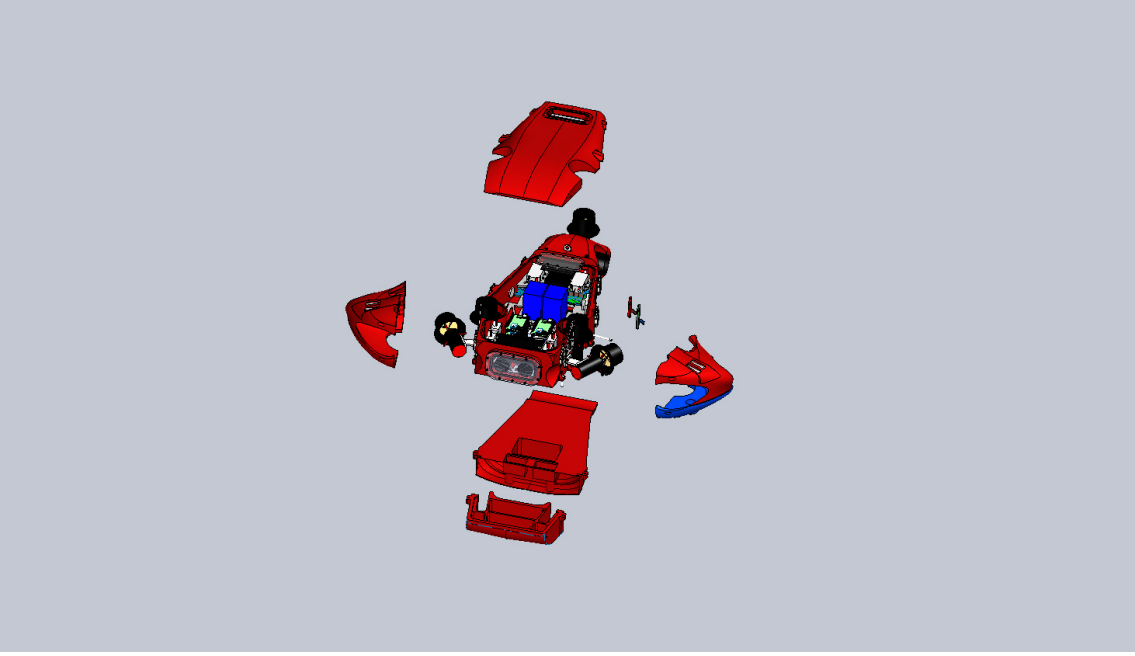


Figure 68: Exploded view of Naiad.

Figure 69: Exploded view of the left wing of Naiad.

G Thruster configuration

Figure 70: Thruster configuration matrix used in the PID controller and the Matlab simulation.

Motor	X	Y	Z	Roll	Pitch	Yaw
1	0.866025	0.5	0.0	0.0	0.0	0.28
2	0.0	1.0	0.0	0.0	0.0	0.22
3	0.866025	−0.5	0.0	0.0	0.0	−0.28
4	0.0	0.0	1.0	0.355	0.230	0.0
5	0.0	0.0	1.0	−0.355	0.230	0.0
6	0.0	0.0	1.0	0.0	−0.355	0.0

H Mechanical drawings

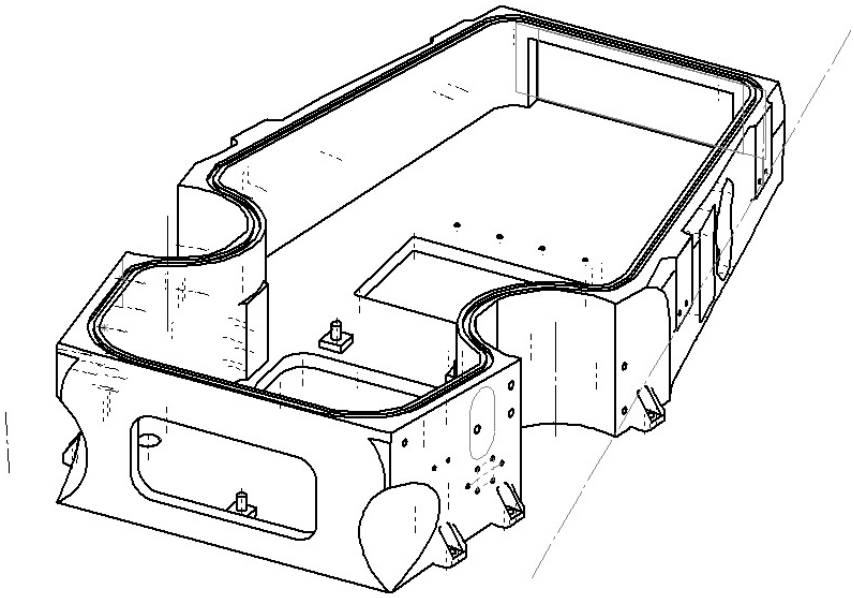


Figure 71: The safety net used during most of the project.

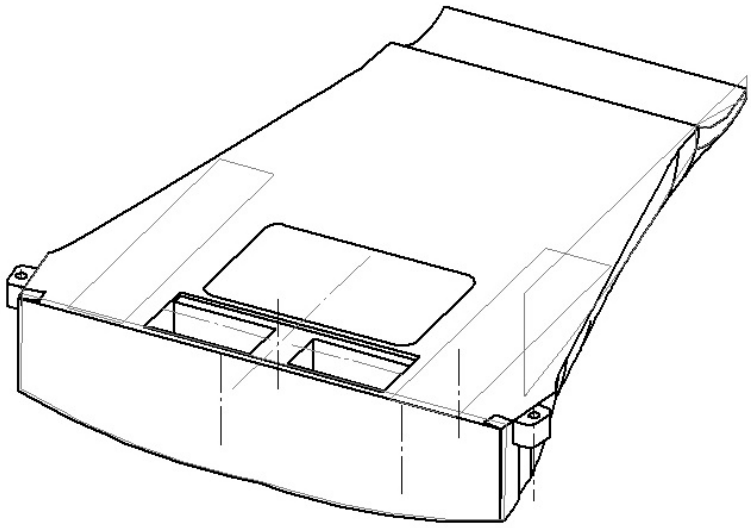


Figure 72: The tool plate. Here pictured with the old design of the marker holders integrated to the tool plate.

H.1 Introduction to Naiads software interface

This is the interface to understand the current communication of the system to help analyse and add new programs. If something is unclear the best way is to check the code for that program/node.
Reservation for small mistakes such as spelling of fields in JSON object etc. The field names can also be confirmed in the code.
All code should be commented and written with a standard such as:

Variables Starts with lower-case letter, other words with Upper-case as first letter. Each variable should start with what type it is (example unsigned_ 16 : u16VariableName). The name should be descriptive of its purpose.
If special type, such as records, the type name is x (example special type : xVariableName).

Functions/procedures Should start with upper-case letters in each word and each word separated with underline (example This_ Is_ A_ Procedure).

Libraries/files Should follow same standard as functions/procedures.

Comments All procedures and functions shall have comments on their purpose and as much as possible on why and how.

H.2 BeagleBone Black

For creating a new node you need to use the library TCP_ Client's function Set_ IP_ And_ Port at IP 192.168.1.1 with a 3 letter long name written in lower-case, which will be transferred to a unique port number. All communication intern this system will be with JSON objects with all fields in lower

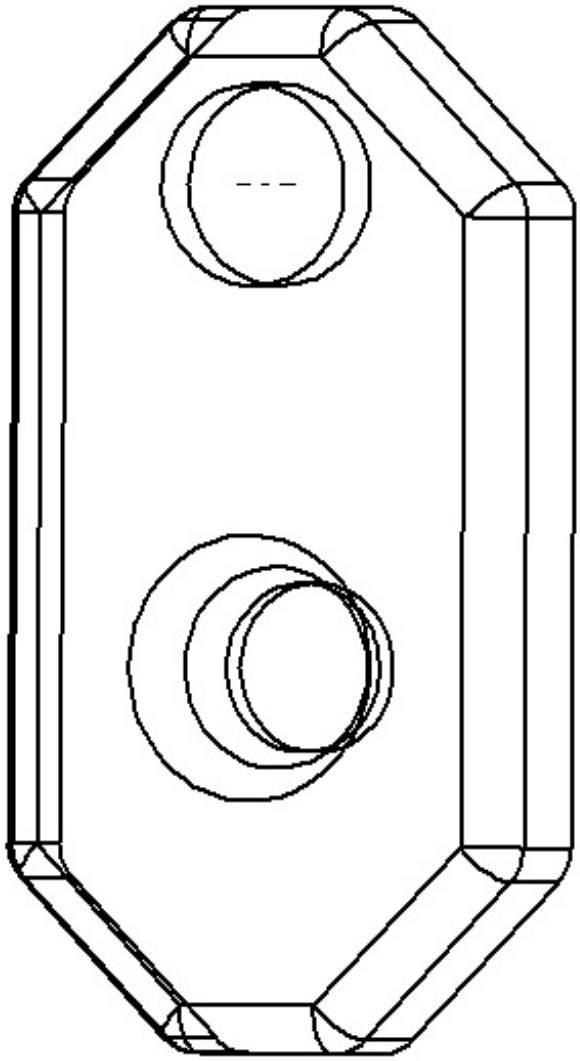


Figure 73: The emergency stop used for stopping the motors.

case letters. Following nodes is currently implemented and with the interface toward each of them, the first 3 letters is the target name.

If the library TCP_ Client is not used, the initial set-up and communication software has to be implemented again. As clients are only supposed to be connected to the server node, they only need one connection. To make a successful connection the client should first connect to port 854. This

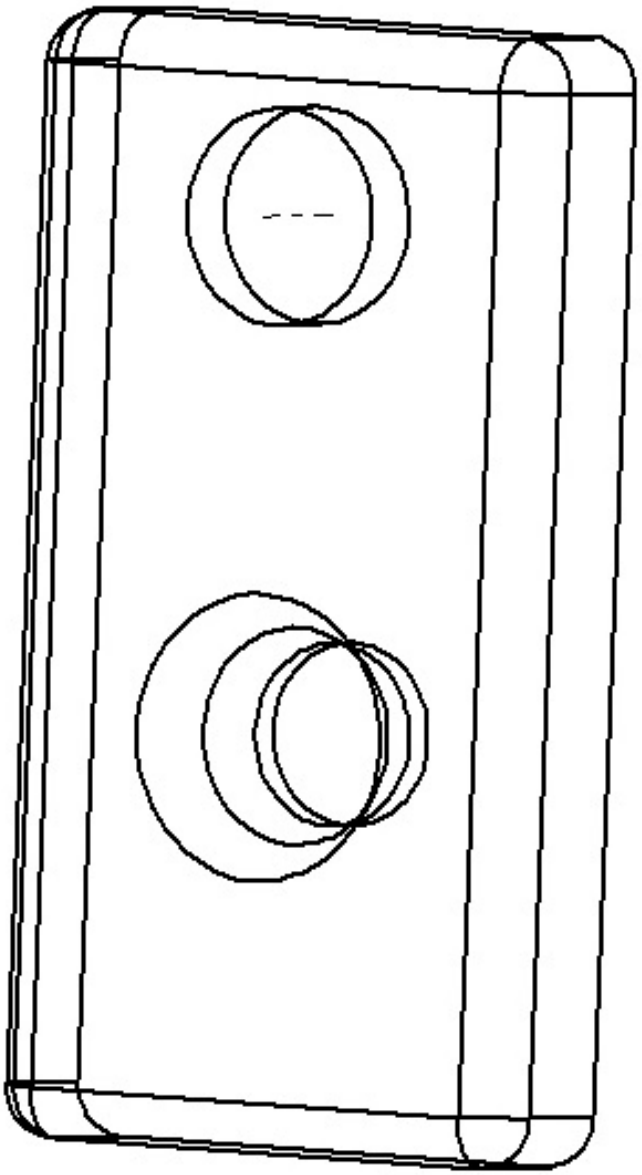


Figure 74: The mission switch used for starting missions.

connection is just to send a short message to the server, telling the name of the client. The name is a three character long string. It should be the same three characters as what other clients are using when

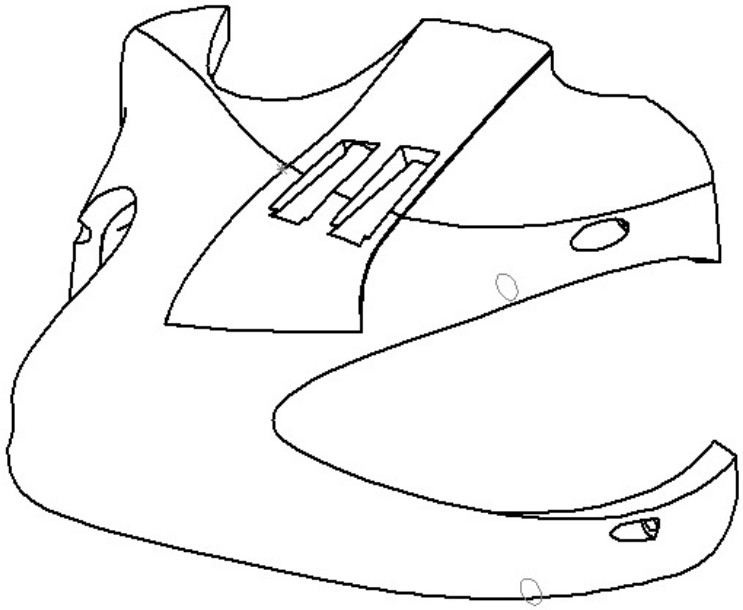


Figure 75: The right wing of Naiad.

they want to send a message to the this client. When the message is sent, the server will open a new server connection on a port that is based on the clients name. This is the connection where the client can now connect and where it can communicate with other clients. There will be a short delay before the server have successfully configured the new connection so if the client is trying to connect too soon after the initial message, it will fail and the client will have to retry. The port number is calculated by using the each letters as a number in base 26. It makes the possible port range go from 0 (aaa) to 17576 (zzz). A list of current name and the corresponding ports can see in "port.txt". All messages sent should always at least contain the two variables target and a sender. The variable target is a three character long string that should correspond to the name of the node which should receive the message and sender similar but it should instead be the name of the client that is sending the message. Another useful variable tag that is used are ethid, which tells the receiver what type of message it is. Another important thing the client has to consider is that all message should always be exactly 256 bytes. This is because the server will only react when its buffer is full, which is when it receives 256 bytes on a connection. The safest way to fill a message is by padding the message with spaces (ASCII value 32). When a client is waiting to receive a message, it should also expect a message size of 256 bytes. The server will never drop the connection intentionally. If the connection is broken, it will be because an issue with the server, like a crash or it can mean that the physical connection is broken.

All of the procedures above can be handled by a client library called "tcp_ client" that is written in ADA. This library will only be able to handle one connection and it is done by a separate thread that is running as soon as the library is included. As soon as the initialization procedure is called, it will connect properly and allow the client to send and receive messages from anywhere in the program.

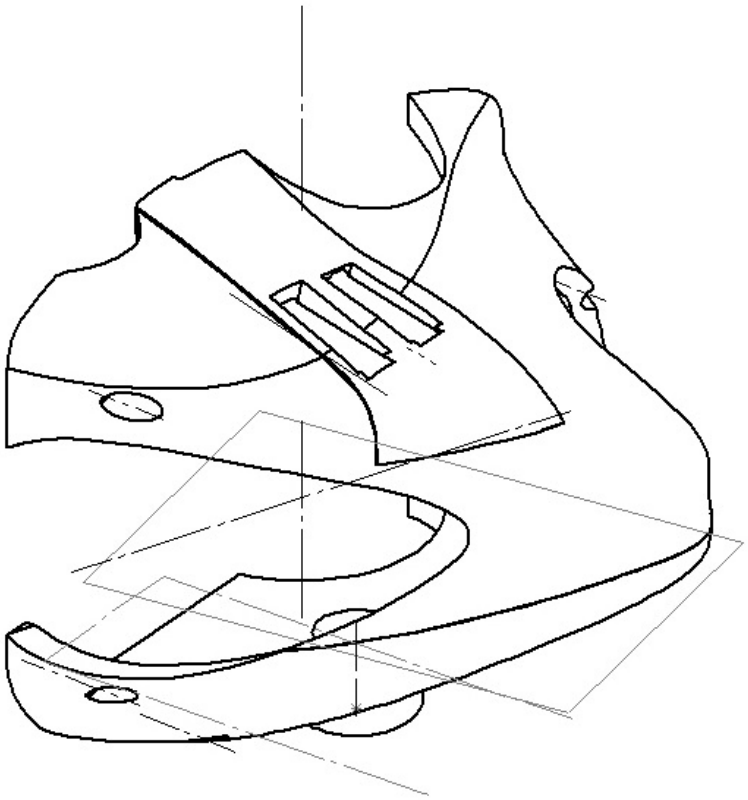


Figure 76: The left wing.

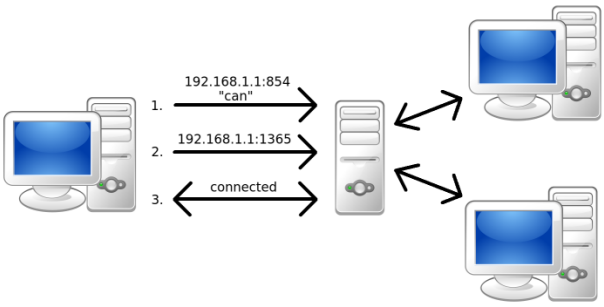


Figure 77: Initial TCP communication messages

This will also pad messages automatically and it will clean up received messages. The thread that handles the connection will also automatically try to reconnect to the server if the connection is lost.

H.2.1 bgw - boarder gate way

The main node routing all messages, all programs will connect to this one. User("usr") and simulation("sim") have the possibility to tell this node that it wants to listen to desired messages

"target" (string) name of the target with 3 letters (a-z).

"ehtid" (integer) which setting to change where 2 is to mirror data and 3 is to block data flow

"xxx" (boolean) true or false, xxx is the name of the node that the setting should effect.

H.2.2 usr - user

A computer connected to Naiad, may listen to messages to simulate in real-time, or send missions to the mission control node etc.

"target" (string) "usr"

"sender" (string) "pid".

H.2.3 sns - sensor

Handles all sensor data, sensor fusion and sending the data to the nodes that needs it. will get following message for update position and orientation

"target" (string) "sns"

"sender" (string) "can".

"accx" (float) accelerometer x.

"accy" (float) accelerometer y.

"accz" (float) accelerometer z.

"roll" (float) roll angle (in degrees).

"pitch" (float) pitch angle (in degrees).

"yaw" (float) z angle (in degrees).

H.2.4 pid - PID-controller

Calculates the needed values of the thruster to get to desired position and orientation. Can be absolute or relative coordinate/orientation. Message to set this is structured as following:

"target" (string) "pid".

"sender" (string) name of sender, most often "pth".

"ethid" (integer) 1 if absolute coordinate, 2 if relative.

"posx" (float) x coordinate (in meters).

"posy" (float) y coordinate (in meters).

"posz" (float) z coordinate (in meters).

"roll" (float) roll angle (in degrees).

"pitch" (float) pitch angle (in degrees).

”yaw” (float) z angle (in degrees).

Sends following after thruster values been calculated:

”target” (string) ”can”.

”sender” (string) ”pid”.

”ethid” (integer) 616 (can identifier for thruster values).

”len” (integer) 6.

”b1”.”b6” (integer(value 0 to 255)) value for thruster n.

H.2.5 pth - Path-planner

Gets positions and orientation from mission control and updates the PID-controllers desired values to get there. Updates the PID when new sensor data or new desired positions comes.

”target” name of the target with 3 letters (a-z).

”sender” name of the node sending with 3 letter (a-z).

H.2.6 msn - mission-control

Getting feedback from cameras and sensor to help complete missions by telling the path planner where to go.

”target” name of the target with 3 letters (a-z).

”sender” name of the node sending with 3 letter (a-z).

H.2.7 can - CAN-translator

Sends the CAN-messages to the correct program on BBB, and sends messages from BBB to the CAN-bus.
When a JSON message with the field target as ”can”, this node will receive that message and send it to the CAN-bus with following settings.

”target” (string) can

ethid (integer) can identifier (0-2047)

”len” (integer) number of bytes in message (1-8)

”b0”.”bn”.”b7” (integer) byte data 0-255, in index n

H.2.8 vsf - vision front

Will send distance, hight difference and yaw angle to the object of interest. The format of one of those messages is:

”target” (string) ”msn”.

”sender” (string) ”vsf”.

”yaw” (float) yaw value to object.

”x” (float) distance in horizontal to object.

”z” (float) distance in vertical to object.

H.3 CAN cards

To program the CAN-cards in Ada one will need to download and install GNAT Ada GPL 2012 for AVR microcontroller ELF format (windows).[24] to compile Ada code for the AT90CAN128. To download the code to the microcontroller use Atmel Studio[21]. The AT90CAN128 should have the following configurations:
BODLEVEL = DISABLED
TA0SEL = [ ]
OCDEN = [ ]
JTAGEN = [X]
SPIEN = [X]
WDTON = [ ]
EESAVE = [ ]
BOOTSZ = 4096W_ F000
BOOTRST = [ ]
CKDIV8 = [ ]
CKOUT = [ ]
SUT_ CKSEL = EXTXOSC_ 8MHZ_ XX_ 16KCK_ 0MS

EXTENDED = 0xFF (valid)
HIGH = 0x99 (valid)
LOW = 0xDF (valid)

The tool used to download the code to the microcontoller was JTAGICE3[27] and JTAGICE MKII[26]

To connect a new node simply connect a new node that communicate at a baud rate of 250000 bits/second. Following nodes are currently connected to the CAN-Bus and will have one section for outgoing message(if any) and for what messages they listen to(if any).

H.3.1 Ask for name of all nodes

If a message with Identifier 380 is sent, all nodes (except PSU) will return there name to PSU node. This can be used to check that all nodes are still alive.

H.3.2 Example message

ID Identifier number and if extended or not, for example (580, false), so far extended is never used.

Len Number of Bytes of data to be sent.

Data as array with position 1 to 8 values 0-255.

H.3.3 Thruster

If no messages received in last 1.0 seconds motors is set to neutral to avoid movement if something goes wrong in higher levels. Incoming messages:

616 6 bytes of data, data(1) speed of motor 1, .. , data(6) speed of motor 6.

623 1 byte, value 0-100, sets a software limit for the motor, value in percent, recommended to not go above 50

H.3.4 Sensor

Current for pressure/depth sensor, shield is able to connect to salinity and temperature also.
outgoing messages:

1593 2 bytes from pressure sensor, updated at 10 Hz.

H.3.5 INS

This card will send values from the inertial measurement unit and fiber optic gyro(when done), as 32 bits float values.
update frequency 20 Hz.
outgoing messages:

1594 byte 1-4 : Yaw, byte 5-8 : Pitch.

1595 byte 1-4 : Roll, byte 5-8 : Accelerometer Z.

1596 byte 1-4 : Accelerometer X, byte 5-8 : Accelerometer Y.

H.3.6 Power supply unit

The node activating all others electronics (BBB, CAN-bus etc). Following is done at startup:
Starting electronics, sets port A, bit 4 to true.
Starting motors, sets port A, bit 3 to true.
Incoming messages:

480 Restart system

481 Turn off system

482 Turn on system

486 allow auto self restart

487 don't allow auto self restart

489 Clear LCD

490-497 write text on LCD,

510 print Naiad on LCD

outgoing messages:

400 Stop mission (if mission switch gets attached)

404 Start mission (if mission switch gets removed)

1000 Name of a connected node byte 1-8 in ascii value filled out with spaces after name.

H.3.7 LED

Control the LED strips on the wings. incoming messages:

753 Turn of all lights

754 turn on port/starboard blinking

755 turn of port/starboard blinking

756 set left wing, if byte 1-6 /= 0 set led(1-6) = true

757 set right wing, if byte 1-6 /= 0 set led(1-6) = true

760 set pwm (data(1)) for front spotlight, value 0-255

761 set pwm (data(1)) for bot spotlight, value 0-255

H.3.8 Solenoid

Controlling up to 4 solenoids, by opening for 2.5 seconds then closing them. incoming messages:

988 activate port 0.

989 activate port 1.

990 activate port 2.

991 activate port 3.

H.3.9 Example node

I User guides

I.1 Electronics

To supply the system connect a suitable power source. The source should deliver 22-24V with atleast 0.5A to start the system. More amperage is needed to run the motors. Connect the supply to the connectors in the center of the bottom of the power board. The connectors are marked `IN EXT2`, `IN EXT1`, `IN BAT1` and `IN BAT2`. The source primarily used during the project has been a 6-cell Li-Po battery.

The BAT and EXT connectors share a common ground but are otherwise separated and the card will chose the voltage supply that have the highest potential. The EXT 1 and 2 and the BAT 1 and 2 are connected in parallel to each other respectively. To be able to start the whole system a MCU CAN card need to be mounted on the card with suitable software to control the system. In order to start the motor supply and the 12 V supply the kill-switch need to be activated as well.

The second significant part of the system is the CAN bus. The bus consists of several MCU CAN cards connected to each other. Most of these connections are aided with the help of stack cards in the robot. These consists of a card with two six pin TP cable connectors, one power-supply connector and eight four pin bottom entry break away socket headers. The TP connectors connect the different stacks as well as supply the stacks later in the chain with power to supply the cards. *One* of the stacks in the chain needs to be connected to one of the top three connectors on the right side of the power-board to the green connector on the bottom of the stack. The rest of the stacks are supplied via the TP cable.

On the top half of the card there are the eight four pin bottom entry break away socket headers. The socket headers is where the MCU CAN cards are to be connected. Make sure that the card goes *though* the card *first*. The breakaway header on the MCU CAN card need to be angled and longer than standard to make a secure fit. They need to go all the way though to the other side and show at least a couple of millimetres on the back.

The MCU CAN cards have several extension boards to add functionality to the system. What these do and how they work are discussed in its respected sections.

The brain of the robot is the beagle bone black (BBB). They are supplied by the power board and should be connected to the third and fourth connector on the left side from the top. This supply also need to be activated by the MCU CAN card mounted on the power board. The BBB is also connected to the CAN bus via small card. This card changes the logical level from 5V in the CAN cards to 3.3V that the BBB are using.

The final cards are the GIMME 2 cards. These should be connected to the GIMMIE 2 power board. This is then connected to the same supply as the CAN bus, one of the top three connectors on the right side of the power-board.

I.2 Mechanics

This section will describe how to prepare the mechanical parts of Naiad for underwater use and how to handle the robot afterwards. It also goes through how to connect cables through the main hull and how to modify the length of an O-ring.

I.2.1 Preparation for Underwater Usage

This section describes the steps to be done when preparing Naiad for use in water. Step 2-3 can be done either before or after the electronics have been placed into Naiad. However, to decrease the risk of accidental damage to the O-ring when adding the electronics, it is preferred to do it after all the electronics have been fitted.

1. Make sure the insides of the main hull and lid are dry.

2. Clean the O-ring track on the main hull with a wet towel or a piece of tissue and make sure the track is free from any dust or small hair.
3. Add a small amount of petroleum jelly along the whole track.
4. Clean the O-ring in the same manner as the O-ring track. It is important to check for any damage on the O-ring. Check for any cuts, dents or similar while cleaning it. Replace if it is damaged.
5. Add a layer of petroleum jelly to the O-ring and make sure no hairs or dust gets stuck on it.
6. Put the O-ring into the track. Make sure that the O-ring fits and is not too large. Preferably, the O-ring should be lying against the inner wall of the track. If it lies too much against the outer wall there is a high risk of it coming out of the track when, for example, putting on the lid. For more details about changing the size of the O-ring, see Section I.2.5.
7. Attach the lid to the hull. Make sure all compressing spring latches are firmly closed.
8. Make sure the kill-and mission switches are mounted on the side of the hull. When in the water, the kill switch is to be removed when there is an emergency and Naiad needs to stop. The mission switch is to be removed for starting missions.
9. Mount the front tool plate to the underside of the main hull in the very front to the pre-made holes with screws and nuts. Naiad is now ready to be put into water.

I.2.2 Post-usage

After Naiad has been into water, the following steps should be done (where step 3-4 can be done before or after removing the electronics):

1. Dry the outside of Naiad to eliminate the risk of accidentally dripping water into the hull when removing the lid.
2. Remove the lid and prevent damage to the O-ring by lifting the lid straight up.
3. Remove the O-ring and clean it from the petroleum jelly with a piece of tissue.
4. Clean the O-ring track free from the petroleum jelly.
5. On every thruster: Locate the screw on top of the thruster, just beneath the propeller. Unscrew it. See Figure 78 for the location of the screws.

   For the horizontal thruster in the very back of Naiad, first remove the two, horizontally placed, small screws on the part that covers the motor. Then unscrew the two screws that go vertically straight through from the top. Carefully remove the covering part. Unscrew the screw on the thruster. The placement of the screws can be seen in Figure 79.
6. Add grease in a large syringe.
7. Insert the syringe into the screw holes and inject grease until some grease comes out of the top of the thruster and any water has been pushed out. Repeat for all six motors.
8. Put all the screws back on the thrusters. For the covered thruster in the back, put the cover back on in reversed order from step 5.

I.2.3 Adaptions to the system

The following section describes optional configurations of Naiad.

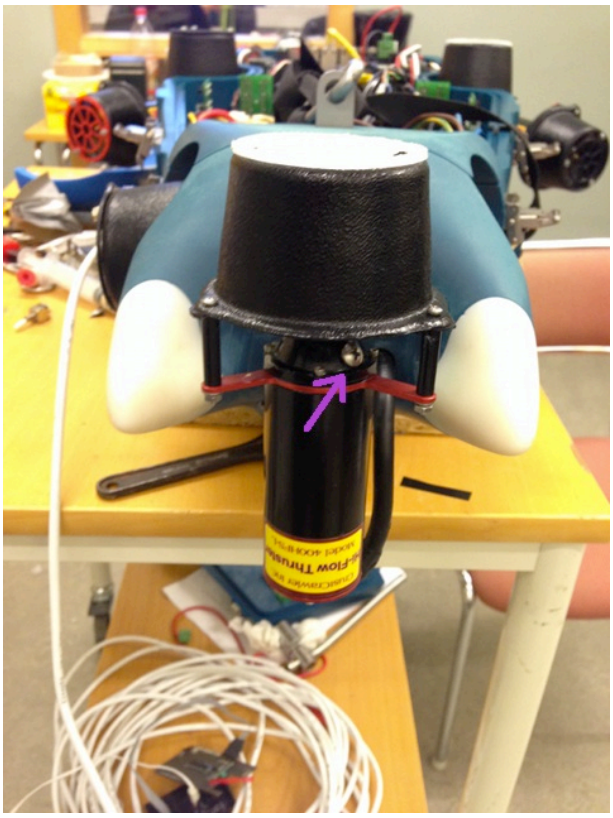


Figure 78: The location of the screw for grease refilling.

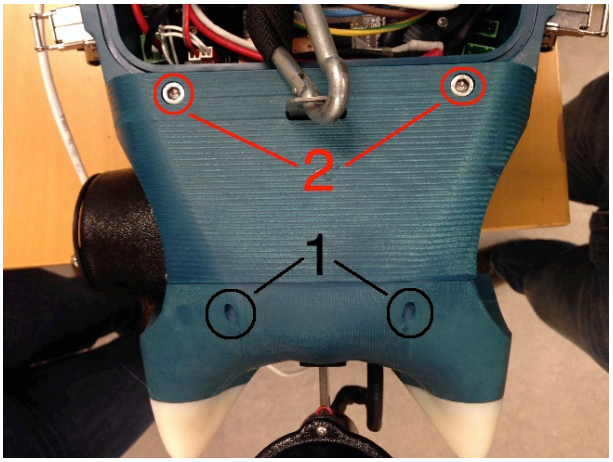


Figure 79: The two sets of screws for accessing the rear motor. Number 1 indicates the horizontal screws and Number 2 indicates the vertical screws.

I.2.4 Connecting Cables

Naiad can be equipped with equipment that may need to be connected to the electronics inside of the hull. This section describes how to manage this.

1. Locate the four plugs in the bottom of the main hull. Remove the plug of any of the four holes that is closest to the peripheral that needs to be connected to inside of the hull.
2. Insert the cable through a cable gland and then through the hole in the hull.
3. tighten the cable gland on the outside and put on a screw nut from the inside. Make sure not to tighten the cable gland too much as it could damage the hull.

I.2.5 Adjusting the length of the O-ring

If the O-ring is too large for the main hull, do the following steps in order to make it fit (if making a new O-ring, start from step 2):

1. Cut off the O-ring as perpendicular and straight as possible, to get nice and clean endings.
2. Put glue, preferably special O-ring glue, on both ends of the O-ring and then put the ends together.
3. Wait for the glue to dry and then check if the splicing is secured.

It is highly recommended to put Naiad with the new O-ring in water without electronics to check the quality of the O-ring repair.

J Interface user guide

J.1 How to design a mission

After the user run the user interface software, a window looks like below will appear, on left side of the screen is the mission chose and line chose bar, user can drag tasks from here and drop them onto the white section on the right and use the straight line and curve line connect all the missions together, after enter the IP address and the port number and pressed the TCP communication button,the server will receive a json string like the image below.

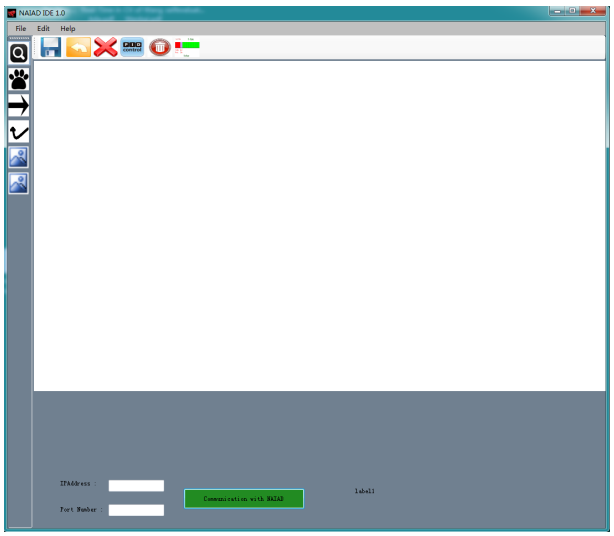


Figure 80: The over all view of the panel

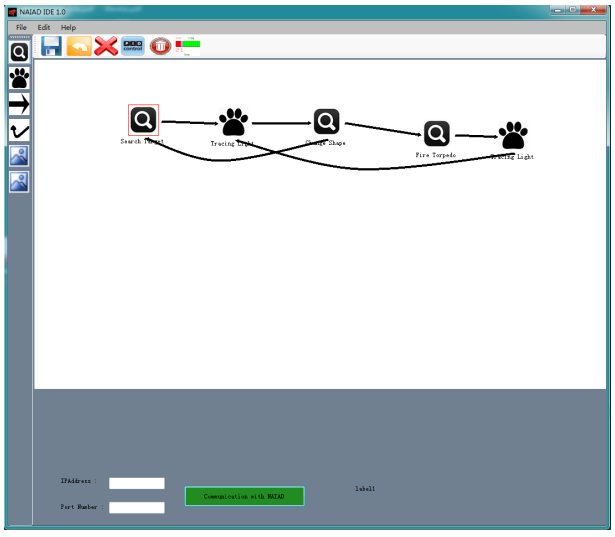


Figure 81: A example about how to use the interface

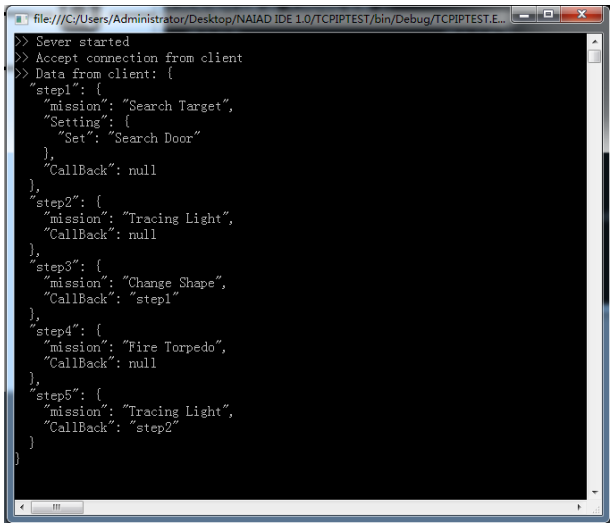


Figure 82: The output of the json string

J.2 PID control

When the user pressed the PID control part, an window like below will appear, the user can set the PID value on position X, position Y and position Z. After that user can set the PID value for the orientation yaw,pitch and roll. When finished all the setting, there are three buttons that user can chose:save button,import button and TCP communication button.

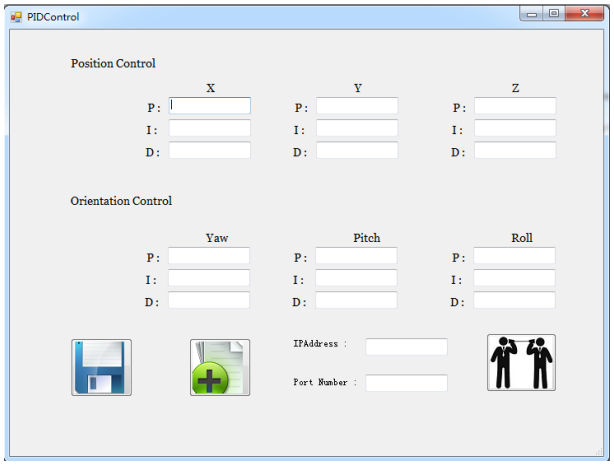


Figure 83: The PID control setting part

J.3 CAN message setting

After pressed the CAN message button, the user will see a window exactly the same as below, there user can set the ID of the CAN message and the value for different byte. After that user need enter the IP address and the port number, then all the information will be sent to Naiad.

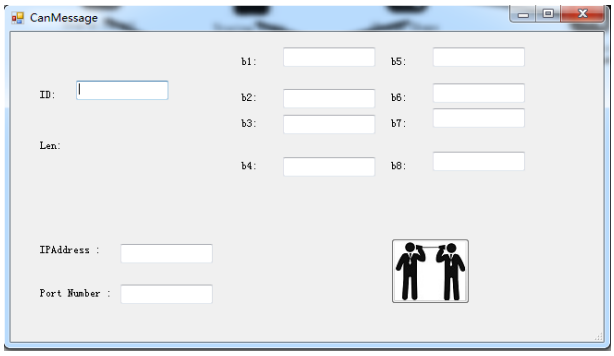


Figure 84: The CAN message setting part

J.4 About Naiad

When the user click the Help button,there are two options to chose,Welcome button and About button, after you click any one of them, you will see the image below, there you can find the basic things about Naiad and the website.

Figure 85: The About part

K Retailers and sponsors

Würth Electronics	Martin Danielsson	Electric components and PCB's
Robotdalen	Ingemar Reyier	Robotdalen sponsored the project with financial aids
PackSize	Fredrik Markström	PackSize sponsored the project with financial aids
FoU-rådet	Christer Petersson	FoU-rådet sponsored the project with financial aids
Deepvision	Fredrik Elmgren	Deepvision sponsored the project with a side scan sonar and associated its electronics
Kvaser	Lars-Berno Fredriksson	Kvaser sponsored with their own products

L Bill of materials

Material used, divided in to subsections for each card or part. Components that have been considered generic have not been given an order number. Only a value are given and the footprint in the order number column.

L.1 MCU CAN

Type	Quantity	Value	Retailer	Order number
Break-away-header, longer angled	1	2 pin		2.54mm
Break-away-header, longer angled	1	4 pin		2.54mm
CAN-Trans	1		Farnell	9758569
Capacitor	1	0.01μF		R0603
Capacitor	7	0.1μF		R0603
Capacitor	1	1μF	Farnell	2068682
DC-DC	1	12V-5V	Farnell	8728135
DC-DC	1	12V-5V	Farnell	2079696
DC-DC	1	40V-12V	Farnell	1469205
LED, red/green	1		Farnell	2146480
LED, switch	1		Farnell	2318469
Optocoupler	1		Farnell	1870803
Resistor	1	0Ω		R0603
Resistor	1	120Ω		R0603
Resistor	2	330Ω		R0603
Resistor	1	1kΩ		R0603
Resistor	2	1.8kΩ		R0603
Resistor	8	2.2kΩ		R0603
Resistor	1	10kΩ		R0603
Supervisor	1		Farnell	1851919
Transistor 7NPN	1		Farnell	1564361
X-tal	1	16.000MHz	Farnell	1640900
Stackable break-way-header	1	11 pin		2.54mm
Stackable break-way-header	1	8 pin		2.54mm
Stackable break-way-header	4	4 pin		2.54mm
Stackable break-way-header	1	2 pin		2.54mm
Jumper	1		Würth	609 002 115 121
LED, blue	8		Würth	150060BS75000
Diode, TVS	1		Farnell	1899525
Diode, Schottky	1		Farnell	1843713
Diode, Standard	1		Farnell	1612315
Inductor	1	100μH	Würth	744787101
Ferite	1		Würth	742792606

L.2 Naiad Power board

Type	Quantity	Value	Retailer	Order number
2.54mm Male Locking Header WR-WTB	4	2 pin	Würth	619 002 111 21
2.54mm Male Locking Header WR-WTB	1	3 pin	Würth	619 003 111 21
Analog Switch	6		Farnell	1333248
Break-away-header female	1	11 pin		2.54mm
Break-away-header female	1	8 pin		2.54mm
Break-away-header female	4	4 pin		2.54mm
Break-away-header female	3	2 pin		2.54mm
Break-away-header female	1	8 pin		2.54mm
Capacitor	3	150µF	Farnell	1850115
Capacitor	6	680µF	Farnell	2079238
Capacitor	3	470µF	Farnell	1850134
Capacitor	1	1µF	Farnell	2068682
Capacitor	2	47µF	Farnell	2326182
Capacitor	5	0.01µF		0603
Capacitor	7	0.1µF		0603
Capacitor	4	0.47µF		0603
Connector	19	2 pin 7mm	Würth	691 311 400 102
Current sensor	1		Farnell	1791390
DC-DC +-12	1	±12	Farnell	2079684
DC-DC 12V	1	12V	Farnell	2323493
DC-DC 5V	2	5V	Farnell	1469215
DC-DC Adj	1	Adj	Farnell	8207399
Diode, Schottky	16	45V 8A	Farnell	1431033
Diode, Schottky	4	40V 5A	Farnell	2114792
Diode, Zener	5	5.1V 500mA	Farnell	1097202
Fuse	1	30A	Farnell	9943900
Fusebox	1		Farnell	2292904
Inductor	2	33µH	Würth	7443551221
Inductor	2	22µH	Würth	7443551331
LED Green	2		Würth	151031VS04000
LED Red	1		Würth	151031SS04000
LED Yellow	1		Würth	151031YS05900
Potentiometer	2	5kΩ	Farnell	2328481
Potentiometer	3	20kΩ	Farnell	2328486
Potentiometer, digital	1	10kΩ	Farnell	1562065
Relay	2	Hamlin Relay	Farnell	1608141
Relay	2	Coto Realy	Farnell	1081673
Relay	3		Farnell	1574344
Resistor	8	1kΩ 1W	Farnell	1565353
Resistor	1	100Ω		0603
Resistor	5	1kΩ		0603
Resistor	6	4.7kΩ		0603
Resistor	10	47kΩ		0603
Resistor	3	180Ω 0.5W		
Smart power switch	18		Farnell	1440811
Termistor	4	4.7kΩ	Farnell	9528113

L.3 INS board

Type	Quantity	Value	Retailer	name/footprint
Differantial OP amp	1	pin	Farnell	THS4131CD
Analog to Digital Converter	1		Farnell	ADS1255
7.68MHz crystal oscillator	1		Farnell	9C-7.680MAAJ-T
2.5V reference	1		Farnell	ADR421ARZ
Optical relay	1		Farnell	PVG612
AND gate	1		Farnell	NC7S08M5
OP amplifier	1		Farnell	NE5534AD
DC-DC Converter -12V	1		Farnell	LT1935
DC-DC Converter +12V	1		Farnell	LT1931
DC converter 3,3V	1		Farnell	LM117
Rs-232 Transceiver	1		Farnell	MAX232ECPE
Fuse	1	1A	Farnell	R0603
Schotty diode	2		Farnell	MBR0520L
green diode	4		Farnell	0805
Inductor	1	0.1µF	Farnell	
Inductor	1	4.7µH	Farnell	
Ceramic capacitor	1	47uF	Farnell	1206
Electrolytic capacitor	2	22uF	Farnell	CAPAE430X540N
Ceramic capacitor	5	10uF	Farnell	1206
Ceramic capacitor	2	4,7uF	Farnell	1206
Ceramic capacitor	1	1uF	Farnell	1206
Ceramic capacitor	9	0,1uF	Farnell	1206
Electrolytic capacitor	4	0,1uF	Farnell	CAPAE430X540N
Ceramic capacitor	2	100pF	Farnell	1206
Ceramic capacitor	2	18pF	Farnell	1206
Resistor	6	10kΩ	Farnell	R0603
Resistor	2	2,49kΩ	Farnell	R0603
Resistor	2	301Ω	Farnell	R0603
Resistor	2	49,9Ω	Farnell	R0603
Resistor	3	100Ω	Farnell	R0603
Resistor	1	300Ω	Farnell	R0603
Resistor	2	11kΩ	Farnell	R0603
Resistor	2	1,2kΩ	Farnell	R0603
Resistor	2	84,5kΩ	Farnell	R0603
Resistor	3	2,2kΩ	Farnell	R0603

L.4 LED controller

Type	Quantity	Value	Retailer	name/footprint
LED-driver	4		Würth	MagI³C-LDHM
8-bit shift register	1		Farnell	74HC595D
Darlington array	1		Farnell	ULN2803ADWR
NPN bipolar transistor	1		Farnell	FZT651
Fuse	2	1A	Farnell	0603
Fuse	3	0,5A	Farnell	0603
Resistor	7	2,2kΩ	Farnell	0603
Resistor	2	220Ω	Farnell	0603
Resistor	4	10kΩ	Farnell	0603
Resistor	3	100Ω	Farnell	0603
Resistor	1	15kΩ	Farnell	0603
Ceramic capacitor	2	100nF	Farnell	0603
Electrolytic capacitor	1	1uF	Farnell	CAPAE430X540N
Green LED	2		Farnell	0805
Blue LED	4		Farnell	0805

L.5 Speed logger sensor

Type	Quantity	Value	Retailer	name/footprint
Linear Hall Effect Sensor IC	2		Farnell	A1301

L.6 Speed logger board

Type	Quantity	Value	Retailer	name/footprint
Linear Hall Effect Sensor IC	2		Farnell	A1301
Quad Operational Amplifiers	1		Farnell	TL084ACD
Resistor	8	100Ω	Farnell	0603
Resistor	5	220Ω	Farnell	0603
Resistor	4	56kΩ	Farnell	0603

L.7 Actuator board

Type	Quantity	Value	Retailer	name/footprint
Transistor N-channel	4		Würth	MagI³C-LDHM
Resistor	4	100kΩ	Farnell	0603
Resistor	4	2,7kΩ	Farnell	0603
Green LED	2		Farnell	0805

L.8 LED strip

Type	Quantity	Value	Retailer	name/footprint
RGB LED	6		Farnell	ASMT-YTB2-0BB02
Resistor 0,5W	2	100kΩ	Farnell	0803
Resistor 0,25W	1	200kΩ	Farnell	0603

L.9 Remote control

Type	Quantity	Value	Retailer	name/footprint
Encoder	1		Farnell	RF600E
Transistor N-channel	1		Farnell	BC848B
Resistor	1	2,2kΩ	Farnell	0603
Resistor	1	10kΩ	Farnell	0603

M Getting starting with GIMME-2 board

In order to interact with the GIMME-2 board the user will need some equipment for the initialization of the GIMME-2 board:
1.Micro-USB Male (Type B) to USB Male (Type A) connector or an Ethernet cable.
2.Power supply, the board should be powered with voltages from range +12V to +24V.
3.The operating system on the machine host is preferred to be Ubuntu 12.04.
4.A serial terminal like gtkterm is needed if the user use a USB cable.

Assuming that the GIMME-2 board already has an installed Linux on it the user can interact with the board either through the serial interface (usb cable) or through SSH interface. The user needs to make sure that the jumper on the GIMME-2 board is 0100 before powering on the board.

M.1 Using serial interface

Connect the micro-USB to USB connector to the port labelled USB0 on the board and connect the other end to the host PC. Set the voltage of the power supply to +12V and connect it to the GIMME-2 board. Now open the serial terminal and the booting steps can be seen there. Once the booting is over, the user gets a prompt on the screen.

M.2 Using Ethernet

Connect one end of the Ethernet cable to the middle port among the three on the GIMME-2 board and the other end to the host machine. Now open the bash terminal and execute ssh root@192.168.1.10. Type in (root) when password is prompted. The user gets a zynq> prompt on the screen.

Assuming that the user needs to do changes on the ramdisk image on the GIMME-2 board, the user will need to install the full package of Xilinx ISE Design Suite including the software development Kit(SDK) on the host Pc. Using SDK a new boot image can be created .if the user made some changes on the ramdisk image the user will need to create a new bootable image.

M.3 Install operating system on GIMME-2

For building the boot image from sources refer the Xilinx Wiki . The necessary files for creating a boot file for Linux on the GIMME-2 board is available on the Naiad's DropBox. If the root file system needs to be edited please follow the steps in section M.5 . For building the latest kernel version, follow the steps section M.8. The steps to make a bootable image for flash memory using the files in Naiad's DropBox are described in section M.9. Section M.10 describes how to program the flash memory through a JTAG interface.

M.4 Installing Platform USB Cable II drivers

For Windows users, no need to install the drivers as it comes with the package of Xilinx ISE Design Suite or Vivado.

For Linux users, first download and install Xilinx ISE Design Suite [1] or Vivado [2]
The installation process is guided using a GUI like in Windows.

Download the driver from [3]

Build the library by calling 'make'. If you are on a 64 bit system but want to build a 32 bit library, run 'make lib32' instead. Be sure to have the 32 bit versions of libusb-devel and libftdi-devel installed.

To use the device as an ordinary user, put the following line in a new file ”libusb-driver.rules” in /etc/udev/rules.d/

ACTION==”add”, SUBSYSTEMS==”usb”, ATTRSidVendor==”03fd”, MODE=”666” and restart udev.

If your cable does not have the ID 03fd:0008 in the output of lsusb, the initial firmware has not been loaded (loading it changes the product-ID from another value to 8). To load the firmware follow these steps:

Run ./setup_pcusb in this directory, this should set up everything correctly:
When XILINX is set correctly:
./setup_pcusb
When XILINX is not set, and ISE is installed in /opt/Xilinx/14.7:
./setup_pcusb /opt/Xilinx/14.7/ISE_DS/ISE

M.5 Modifying the root file system

Initrd and initramfs refer to two different methods for loading a temporary root file system into memory in the boot process of Linux kernel. For GIMME-2 board, we have a prebuilt initrd and initramfs image from Xilinx[4].

M.6 Modify an initial ramdisk of type initrd

Assuming that your image called (ramdisk.image.gz):
1.Unwrap the wrapped image.

```
$ dd if=./uramdisk.image.gz bs=64 skip=1 of=ramdisk.image.gz
```

2. Extract the initrd image from the gzip archive.

```
$ gunzip ramdisk.image.gz
```

3. Mount the initrd image.

```
$ chmod u+rwx ramdisk.image
$ mkdir tmp_mnt/
$ sudo mount -o loop ramdisk.image tmp_mnt/
```

4. Make changes in the mounted filesystem if required. Otherwise skip to step 5.

```
$ cd etc/init.d/
$ sudo gedit rcS
```

5. Go to folder location where tmp_mnt resides. Unmount the initrd image and compress the image.

```
$ sudo umount tmp_mnt/
$ gzip ramdisk.image
```

6. ramdisk.image.gz needs to be wrapped with a U-Boot header in order for U-Boot boot with it:

```
$ mkimage -A arm -T ramdisk -C gzip -d ramdisk.image.gz uramdisk.image.gz.
```

M.7 Modify an initial ramdisk of type initramfs

Assuming that your image called (ramdisk.image.gz):
1. Create a temporary folder

```
$ sudo mkdir /tmp/rootfs
```

2. Unwrap the wrapped image.

```
$ sudo dd if=uramdisk.image.gz of=ramdisk.image.gz bs=64 skip=1
```

3. Extract the initramfs image from the gzip archive and mount it.

```
$ sudo gunzip -c ramdisk.image.gz | sudo sh -c 'cd /tmp/rootfs/ && cpio -i'
```

4. Go to /tmp/rootfs/ and make the changes in the file systems if required.
5. Unmount the initramfs image and compress the image.

```
$ sudo sh -c 'cd /tmp/rootfs/ && sudo find .
    | sudo cpio -H newc -o' | gzip -9 > ramdisk.image.gz
```

6. ramdisk.image.gz needs to be wrapped with a U-Boot header in order for U-Boot boot with it:

```
$ sudo mkimage -A arm -T ramdisk -C gzip -d ramdisk.image.gz uramdisk.image.gz
```

7. Delete the temporary folder and old ramdisk image:

```
$ sudo rm ramdisk.image.gz
$ sudo rm -rf /tmp/rootfs
```

M.8 Build Kernel

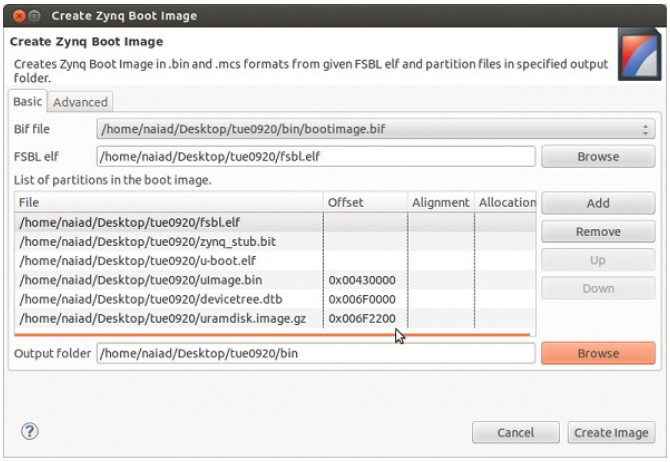
```
$ make ARCH=arm xilinx_zynq_defconfig
$ make ARCH=arm menuconfig
```

To produce the kernel image:

```
$ make ARCH=arm UIMAGE_LOADADDR=0x8000 uImage
$ CROSS\_COMPILE=/opt/Xilinx/14.6/ISE_DS/ -EDK/gnu/arm/lin/bin/arm-xilinx-linux-gnueabi
```

M.9 Make a Linux Bootable Image for QSPI Flash

1. In SDK, select Xilinx Tools > Create Zynq Boot Image. The Create Zynq Boot Image wizard opens.
2. Add the necessary files as shown in the figure and give the offsets(offsets can be changed depending on the GIMME-2 board):



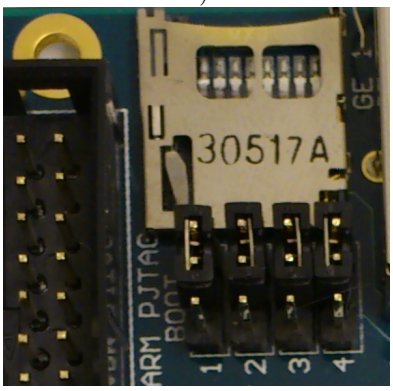
3.Click Create Image.

The Create Zynq Boot Image window creates following files in the specified output folder:
bootimage.bif
u-boot.bin

M.10 Programming the QSPI flash through JTAG

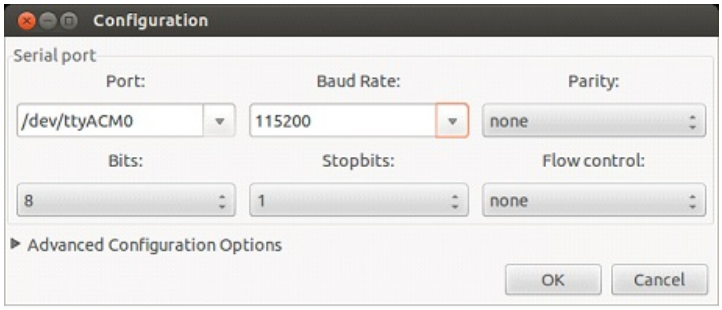
Requirements:
XILINX Platform USB Cable II (JTAG interface)
micro-USB cable OR Ethernet cable
Xilinx SDK

1.Set boot jumpers for JTAG-boot - 0000 - and power up the board (the boot jumpers are on the GIMME-2 board)



2. Program the FPGA - gimme_top.bit.

3. Start terminal (Prefered to used GtkTerm) and set the following configuration:



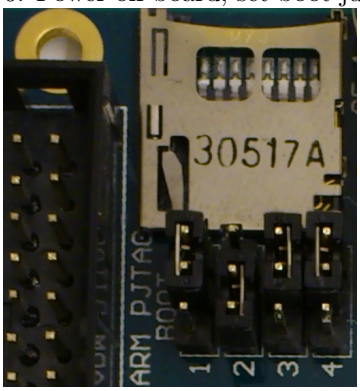
4. In SDK, start console (Xilinx Tools -> XMD Console) . Execute these commands:

```
$ connect arm hw
$ source <path>/ps7_init.tcl
$ ps7_init
$ ps7_post_config
$ dow <path>/u-boot.elf
$ dow -data <path>/u-boot.bin 0x08000000
$ con
```

5. As soon as you enter “con” in XMD console, switch to serial terminal.
6. In terminal, stop autoboot when it shows - ”Hit any key to stop autoboot: 3” and enter these commands:

```
$ sf probe 0 0 0
$ sf erase 0 0x01000000
$ sf write 0x08000000 0 0xFFFFFF
```

6. Power off board, set boot jumpers for QSPI-boot - 0100 - Power on board



7. Start terminal or connect on ssh 192.168.1.10 u: root, p: root

References

[1] http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html,date,2014-09-15

[2] http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools.html,date,2014-09-25

[3] http://git.zerfleddert.de/cgi-bin/gitweb.cgi/usb-driver?a=snapshot;h=HEAD;sf=tgz,date,2014-09-10

[4] http://www.wiki.xilinx.com/Zynq+Releases,date,2014-09-15

N Building / Cross Compiling OpenCV for GIMME-2

N.1 Install necessary packages

```
$ sudo apt-get install build-essential
$ sudo apt-get install cmake
```

N.2 Install cross compiler

```
$ sudo apt-get install gcc-arm-linux-gnueabi
$ sudo apt-get install g++-arm-linux-gnueabi
```

N.3 Create opencv directory

```
$ mkdir opencv_build
$ cd opencv_build
```

N.4 Download OpenCV

```
 $ git clone https://github.com/Itseez/opencv.git
```

To fix the version problem modify the compiler version in the platform file .

```
$ ./opencv/platforms/linux/arm-gnueabi.toolchain.cmake
```

Changed the GCC_COMPILER_VERSION variable's value from 4.6 to 4.7 to match installed compiler.
Disable Opencl options by editing CmakeLists.txt file, change ON to OFF in these three lines:
OCV_OPTION(WITH_OPENCL "Include OpenCL Runtime support" ON IF (NOT IOS))
OCV_OPTION(WITH_OPENCLAMDFFT "Include AMD OpenCL FFT library support" ON IF (NOT ANDROID AND NOT IOS))
OCV_OPTION(WITH_OPENCLAMDBLAS "Include AMD OpenCL BLAS library support" ON IF (NOT ANDROID AND NOT IOS))
Now create a sub-directory called build and cd into it:

```
$ mkdir build
$ cd build
```

N.5 Configure the build

```
$ cmake -DSOFTFP=TRUE -DCMAKE_TOOLCHAIN_FILE=
   ../opencv/platforms/linux/arm-gnueabi.toolchain.cmake ../opencv
```

Now run 'make' and 'make install':

```
$ make
$ make install
```

Now the user have all the include and lib files that the user need to build an OpenCV app for GIMME-2.
Copy opencv_libs and arm-linux-gnueabi_libs to GIMME-2 board.

N.6 Cross compile the program

```
$ export PKG_CONFIG_PATH=../opencv/build/install/lib/pkgconfig
$ arm-linux-gnueabi-g++  program.cpp `pkg-config --cflags --libs opencv`
$ arm-linux-gnueabi-gcc  program.c `pkg-config --cflags --libs opencv`
```

N.7 Run the program on GIMME-2

```
$ export LD_LIBRARY_PATH=/mnt/opencv_lib:
   /sdcard/arm-linux-gnueabi_lib pkg-config --cflags --libs opencv
$ ./program
```

N.8 Build extra modules for GIMME-2

The user can build OpenCV, so it will include the modules from this repository. Here is the CMake command:

```
$ cd opencv_build_directory
$ cmake -DOPENCV_EXTRA_MODULES_PATH=opencv_contrib/modules opencv_source_directory
$ make
```

If the user don't want all of the modules, use CMake's BUILD_opencv_* options.

```
$ cmake -DOPENCV_EXTRA_MODULES_PATH=opencv_contrib/modules -DBUILD_opencv_legacy=
    OFF opencv_source_directory
```

N.9 BeagleBone Black User guide

The most common way to interact with the BeagleBone Black (BBB) is by connecting to a terminal on it via a host computer. The software needed to do it differ depending on the operating system of the host.

N.9.1 Linux host

The Linux terminal can connect to a terminal in a BBB via Ethernet via the ssh command. SSH is included in most Linux distributions by default.

```
$ ssh ubuntu@192.168.1.1
```

On this line, ubuntu is the user name and 192.168.1.1 is the IP of the BBB. A prompt will ask for a password, the password is ubuntu.

N.9.2 Windows host

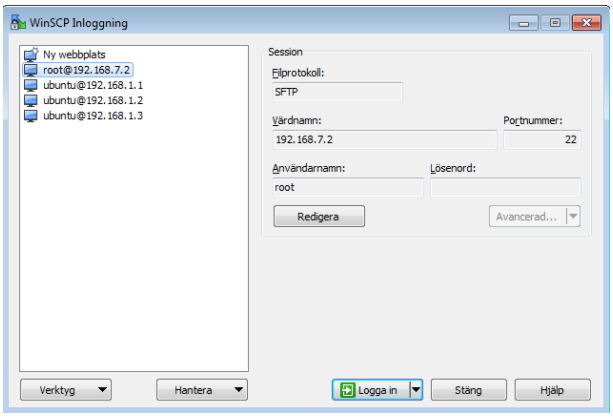


Figure 86: WinSCP

WinSCP[33], Secure Copy (SCP), is an application for managing files between Windows and BBB easier. One can move files freely between the both systems without terminal commands. If connected through Ethernet over the switch use hostname (”värdnamn” in the picture) 192.168.1.n where n is dependent on which of the BBB one wants to access. If connected through USB use hostname 192.168.7.2. With drivers, this can also share connection to internet for installation of applications etc in BBB. Username (”Användarnamn” in the picture) is either root or ubuntu. If ubuntu then password is ubuntu.

PuTTY [29] is an application to Secure Shell (SSH) in to BBB from Windows. This gives the possibility to remote control the BBB. By doing this one can compile and run applications.

N.9.3 a Linux environment

When connected, the user has to navigate in the Linux environment on BBB. This is not effected by the operating system of the host. Navigation in Linux is mostly done with two commands:

```
$ ls
```

This command ls show all the folders in the current directory.

```
$ cd Software/
```

The cd command is used to enter a folder.

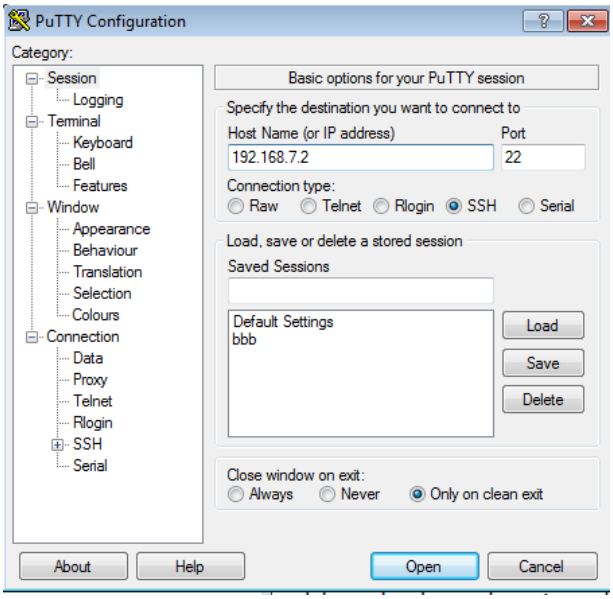


Figure 87: PuTTY

```
$ cd ..
```

The folder name .. will move up one level. Note that Linux is case sensitive regarding names.

All of the software is located in the folder /home/ubuntu/BBB/software/. All the binary files should be located in a folder called 'bin' within the folder Software. To transfer files or folders between a computer and a BBB, use the following command:

```
$ scp -r pid/ ubuntu@192.168.1.1:BBB/software/
```

The flag -r tells the program that the source 'pid/' is a folder while ubuntu@192.168.1.1 is the user on the target computer, in this case a BBB. The colon is the separator where the path target path begin. The default folder will be /home/ubuntu. By adding BBB/software/, the folder will end up at /home/ubuntu/BBB/software/pid with all its content.
If gcc is installed, a project can be compiled with gnatmake.

```
$ gnatmake -P mainproject.gpr
```

Or to only compile a single file:

```
$ gnatmake main.adb
```

To make the process easier, a makefile can be used. Here is an example with three commands. Note that the formatting is important.

```
this:
  gnatmake -P mainproject.gpr
all:
  gnat clean -r -P mainproject.gpr
  gnatmake -P mainproject.gpr
clean:
  gnat clean -r -P mainproject.gpr
```

The command make will run the first command in the makefile, 'this'. With other words, the project mainproject.gpr will be compiled. To run another command, add the name after make.

```
$ make all
```

This will clean the project and rebuild it from scratch. Multiple makefiles can be combined with a shell script so all of them can be compiled by just running the shell script.

```
$ make all -C software/myprojectfolder/
```

This command will run a makefile located in the folder software/myprojectfolder/, multiple lines can be added to compile mulitple projects. To run an executable file, use ./ before the name.

```
$ ./main
```

To exit a program, press ctrl + c. To check which processes are running use ps aux.

```
$ ps aux
```

There are currently a script file with some predefined functions. The file system.sh is a shell script located at /home/ubuntu/BBB/. It can make some tasks easier. It got 5 commands, kill, run, build, restart and move. The command kill together with a name will end that process.

```
$ ./system.sh kill pid
```

This will end the PID controller. To kill all the running nodes, leave leave the name empty. All of the commands except build can be used this way. With other words, the run command will look like this.

```
$ ./system.sh run pid
```

The run command will start the node as a background process but it will still output all the messages to the terminal. To hide the output, redirect the standard output with a pipe to null.

```
$ ./system.sh run pid > /dev/null
```

The build command will compile all projects. If the parameter all is added, all projects will be cleaned first. The build command does also move all the executable files to a specific binary folder. This folder is used by run and restart. The restart command simply kills a specified process and tries to run it again. Finally, the command move will only copy the executable file(s) to the binary folder.

N.9.4 BeagleBone Black configurations

To setup a new BeagleBone Black card (BBB) some configurations has to be done.

First, to install the Bone Debian operating system on the BBB simply download the latest release of Bone Debian operating system for the BBB at http://beagleboard.org/latest-images and mount it to a microSD card using any mounting software. One can either choose the eMMC version which installs the operating system onto the internal flash memory on the BBB or the version which runs directly from the microSD card which does not require any flashing of the internal memory.

Once the operating system has been installed onto the BBB some further configurations has to be done. To configure the Ethernet network communication on the BBB run the following command where the X should be a number between 2 and 254 that makes the BBB unique on the network.

```
$ ifconfig 192.168.1.X netmask 255.255.255.0
```

UART communication is allowed by echoing the desired UART ports to the BBB cape that is installed with the operating system.

```
$ echo BB-UART1 > /sys/devices/bone_capemgr.*/slots
$ echo BB-UART4 > /sys/devices/bone_capemgr.*/slots
```

Some of the commands could preferably be put into a boot up script since it sometimes stores the new configurations temporary.