KEA GOES DEEP

Copenhagen School of Design
& Technology

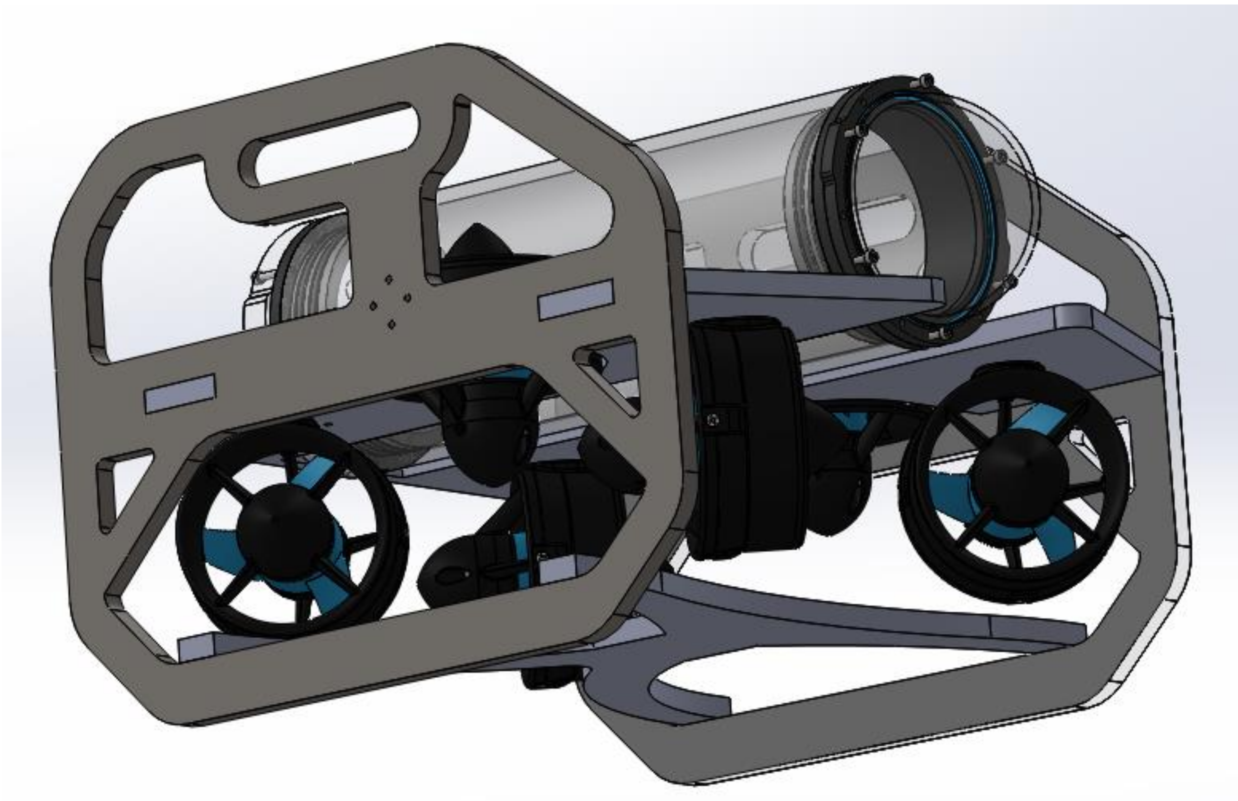

Table of content

Abstract

KEA Goes Deep is a student association that was created to build a prototype of a ROV able to reach the depth of our local oceans. The MATE competition had been a dream for our school for four years and this was the first year that we had the right skill sets to accomplish getting accepted to MATE. We are a group of students studying various subjects ranging from production technology and electronics to computer science. Our team is truly multicultural since we have members from all around our globe counting USA (San Francisco), Denmark, Slovakia, Norway, Iceland, Poland and even from Romania. When working on the ROV we gathered new experiences and knowledge from different fields of our studies. We started with a PVC tube version to explore to possibilities of building a functioning ROV. From that experience we were able to build our final ROV "Robbie", that we are bringing to the competition. Now the frame is improved, gripper is rotating and the design is more rugged than ever. To check the functionality and waterproofness of electronics enclosure we tested ROV in professional military diving tank. In the course of building the ROV we helped to develop smaller prototypes for "ROV Maker Competition" that is an organization that offers an intensive, creative and exciting learning program to students from technical high schools of Denmark. Joining the MATE competition was our dream from beginning of the project and we are sincerely proud of being able to represent KEA in USA.

Design

Our team mentor Christopher Nelsen and our technically design academy are supporters of the maker mentality and we support the maker spirit and open source building principles. With that being said we accepted the challenge of making this ROV project with the mindset that we were not going to reinvent the wheel. Our outlook on building is, yes others have already done what we are trying to accomplish but, it hasn't been done by us. We exercise the use rapid prototyping and reverse engineering at every step of our design process. We were inspired by the Blue Robotics and we add one to build and take apart.

Tether

We used a rope and zip ties in our attempt to make the tether float because we calculated that the water displacement of polypropylene rope would be enough to make the electric cables float. We found out that Polypropylene was a material that was easy to find and had a relative density that was low and would float in water. We used a 12mm polypropylene rope on our tether and that helped the buoyancy but did make it completely buoyant. If we want this principal to work we would need a 20mm rope to displace enough water to make our heavy cable buoyant in water. In the future we would want to use a thick hollow braid polypropylene rope and pull the cables through the middle.

Figure 1Blue Polypropylene rope

Manipulator

The manipulator tool mounted on the Robbie was specifically designed to tackle the challenges of the 2017 MATE ROV competition. Throughout the design process our gripper team went through several design iterations and manufacturing methods, before ending up with our current design.

Manufacturing method

Originally we wanted to 3D-print all our manipulator parts in ABS plastic in order to enable rapid prototyping and quick conversion from CAD-file to physical product. However, due to issues with getting consistent quality prints and varying printer availability we decided to move away from 3D printing and over to laser-cutting the parts in 5mm acrylic. This enabled even quicker iterations and more consistent results, but came at the cost of a full re-design of the manipulator because of the limitations of subtractive machining compared to additive manufacturing.

Process

Our design goal was to make the manipulator as simple as possible, without sacrificing on functionality. We started out researching the tasks we needed the manipulator to perform, and created a Product Design Specification (PDS) that we used as a guideline throughout the process. Moving from there we researched mechanisms we could use to satisfy the functional goals we had set. The principle behind the gripping mechanism has stayed more or less unchanged throughout our process, with the only major changes being the re-situation of the servo driving the gripping motion.

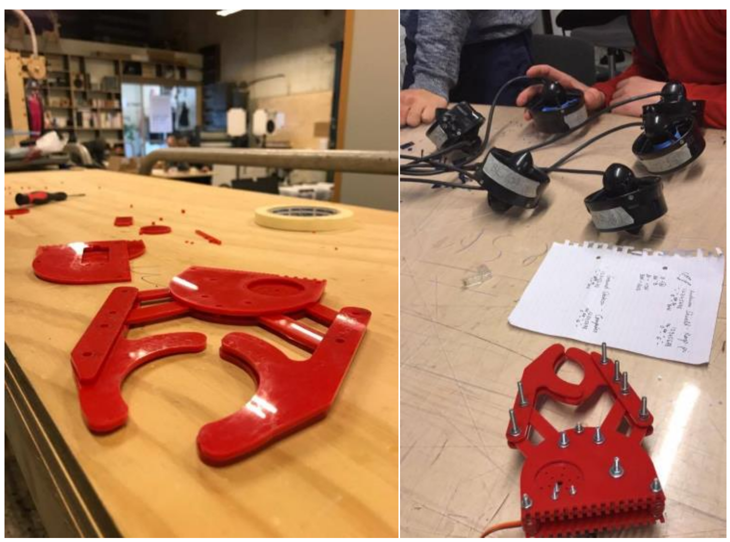

Figure 2 Different gripper design iterations

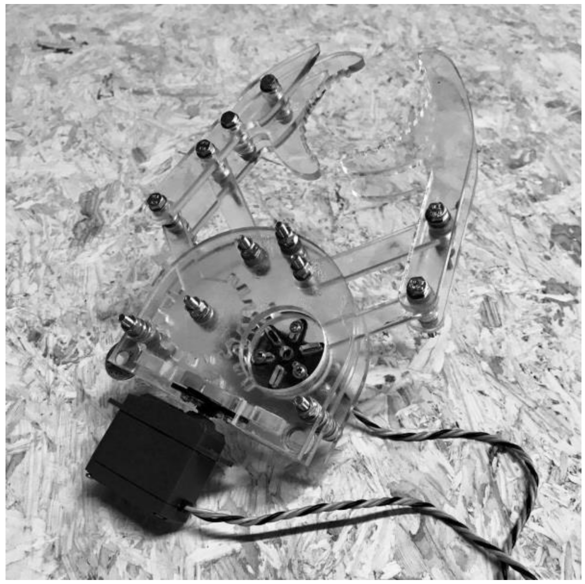

Figure 3 Different gripper design iterations

As shown in the pictures of our previous iterations, our first mechanism for opening/closing relied on having a servo directly connected to the driving cogs, on the part of the manipulator that would rotate. This proved to be impractical, as the wire running from the servo and to the electronics compartment would tangle on the rotating axis and be at risk of damaging the insulation.

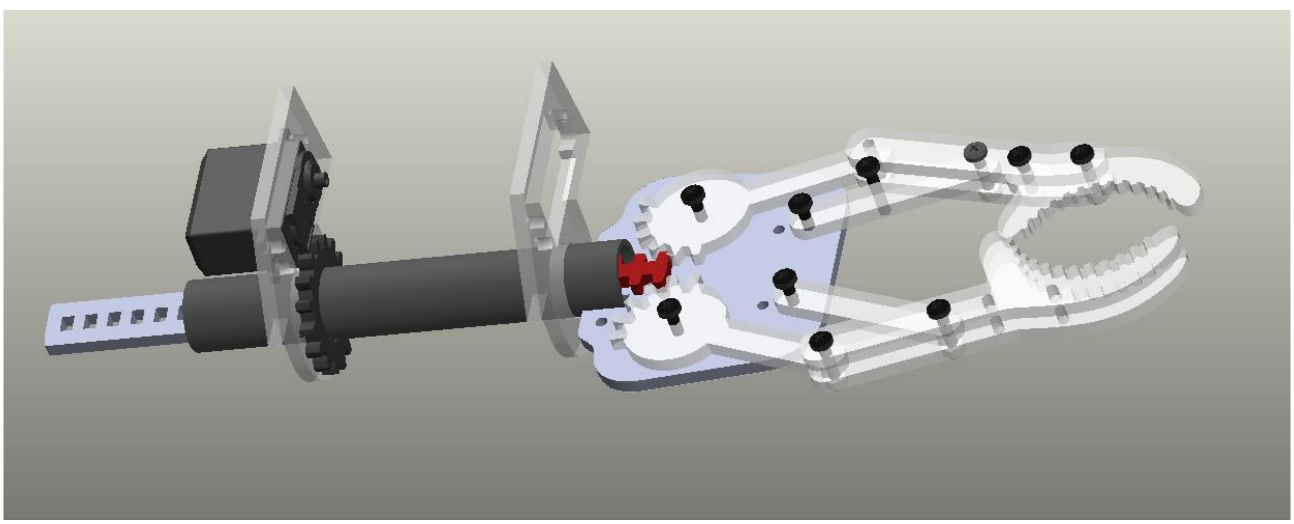

Figure 4 Gripper 3D rendering

As showed in the 3D model of our current design, we ended up with a 3-axis design with both servos placed on the ROV instead of on the rotating part of the manipulator like on our earlier iterations. This, while slightly complicating our design, dealt with the issue of the wires from the servo tangling on the rotational axis. This could also have been resolved by using a waterproof slip ring, which would keep our design slightly simpler, but in the end we decided to scratch that idea, because of difficulty sourcing the part.

Function description/mechanics.

The two mechanisms we use for our manipulator gives It the ability to rotate and to open/close. The rotational mechanism is driven by a servo modified for continuous rotation that runs a cog which drives a hollow driveshaft. This driveshaft is connected to the "gripping platform", but is freely rotating. Inside of the hollow driveshaft is the shaft of the opening/closing mechanism. The shaft is made up of three parts that all are independent of each other when it comes to rotation, which enables the servo that drives the mechanism to stay stationary while the gripper rotates.

Motors

After a lot of research and debate we decided to go with servos for both the gripping and rotating function of the manipulator. Initially we wanted to use a waterproofed stepper motor for the rotational mechanism, as we wouldn't have to modify it for continuous rotation. However, due to availability and ease of implementation, we decided to use the servo type we already had available for this, and modify it for continuous rotation. This was accomplished by gluing the potentiometer in a neutral position and removing the bracket which would normally rotate the potentiometer.

One of the biggest challenges we had with our manipulator design was waterproofing our servos. The servos we use are HobbyKing HK15328D O-ring sealed waterproof servos. Sadly the advertised waterproof rating was, in lack of a better term – misleading. After water-damaging two servos we decided to cover the electronics compartment in epoxy and fill the gear train with marine grease in order to have another barrier for water to enter, in addition to the factory O-rings. This turned out to be a good learning experience, as we still had trouble with our servos malfunctioning after additional waterproofing. After some troubleshooting we realized that adding the marine grease seemed to give the gear train too much resistance, which again led to the servos burning out.

In the end, we managed to find a less viscous marine grease that solved the problem, and this combined with potting the servo-casing in epoxy turned out to be a very simple, yet functional way of waterproofing them. We confirmed that our waterproofing was adequate for the depths we would operate at by running them for over an hour at a depth of 6m.

Future improvements

We are pretty pleased with our end result, but there is definitely still room for improvement. In our current design we use stainless steel screws and locking nuts as fasteners. This, while quick and easy, leaves us vulnerable to over-tightening of the screws, which could lead to increased friction and strain on the servos, or even cracking of the acrylic. This could be solved by using flanged threaded inserts of the correct length instead of nuts, but in order to get the correct dimensions we would need to have them produced to specifications.

Our design would also benefit from increased modularity for the "fingers" of the gripper in order to have a wider range of uses, as it currently requires a screwdriver and a wrench to remove the 4 screws that hold the fingers.

Frame

Our first prototype was made out of 40mm plastic tube, using ABS 3d-printed connections.

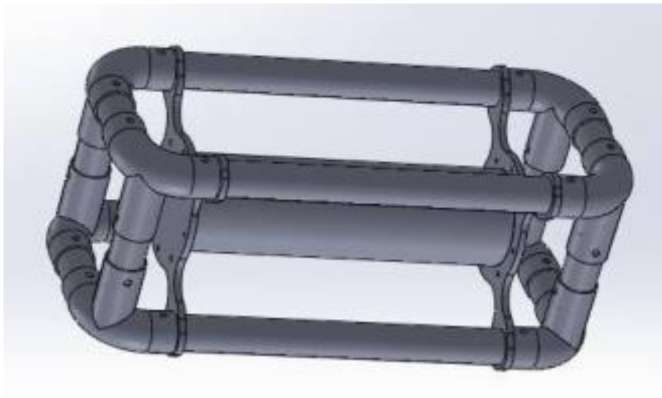

Figure 5 First frame design

Figure 6 First frame rendering

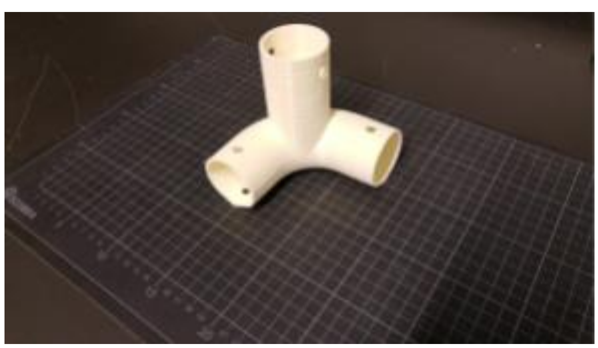

Figure 7 Frame contention

Figure 8 Electronic tube

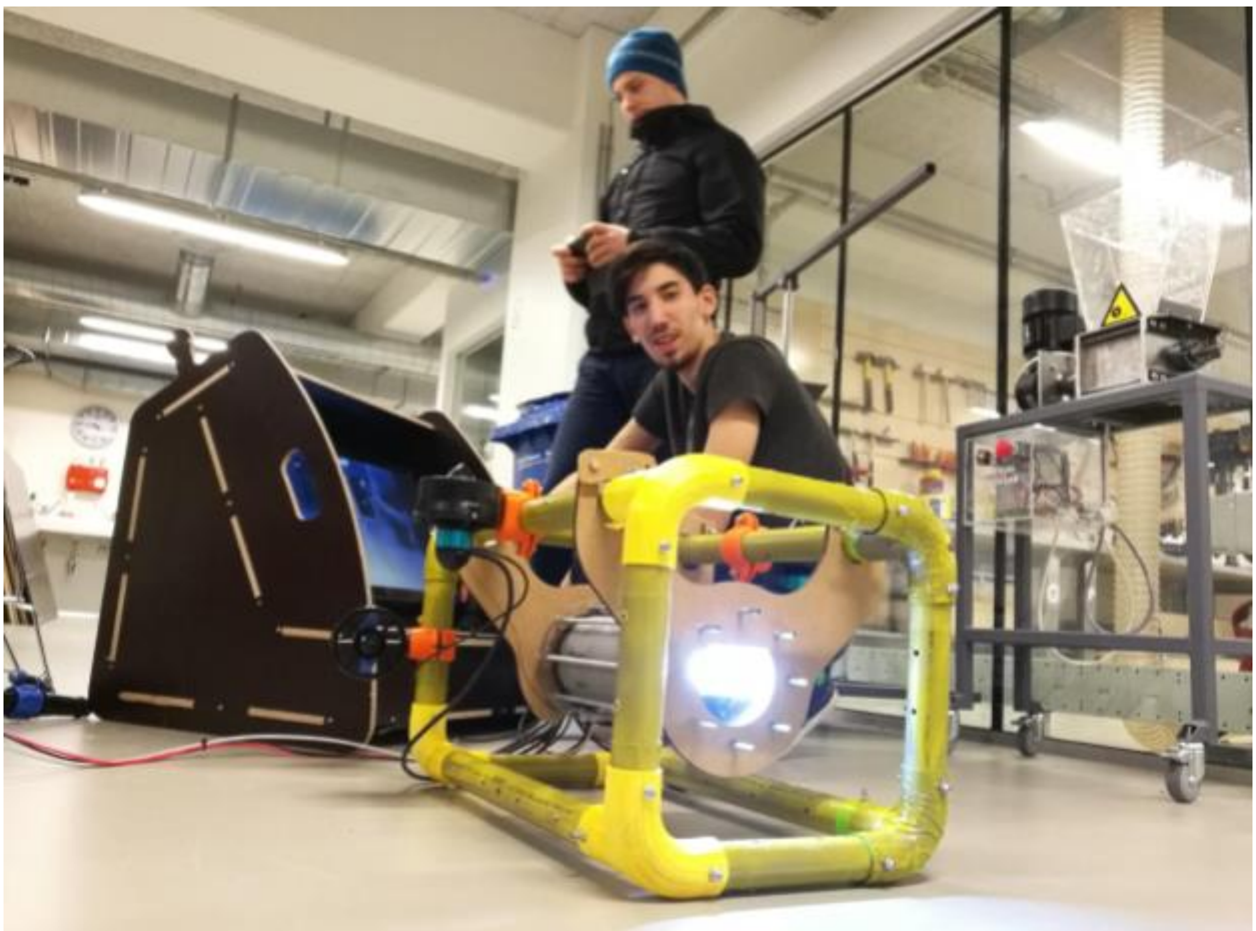

Figure 9 ROV with ground station

Final Design

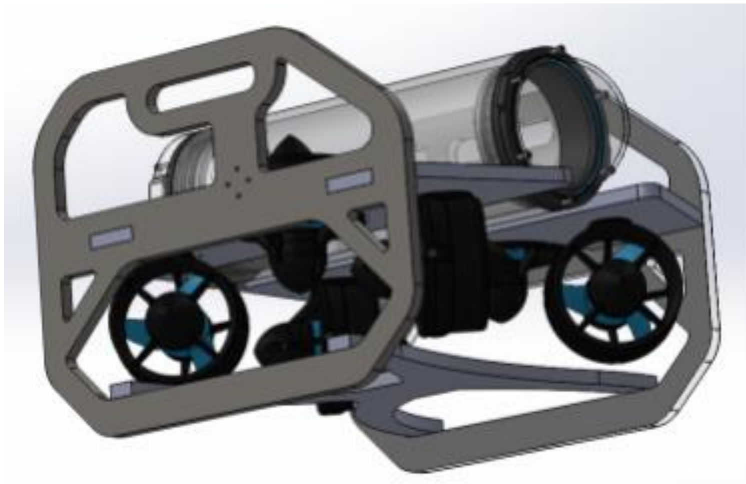

Figure 10 Final ROV design rendering

Our first step when designing the shape, size and frame design of our ROV was deciding how many thrusters we wanted to use, and how we were going to set them up. We decided on 6 thrusters, not so much for the power, as for the accuracy, and the configuration shown in the following image.

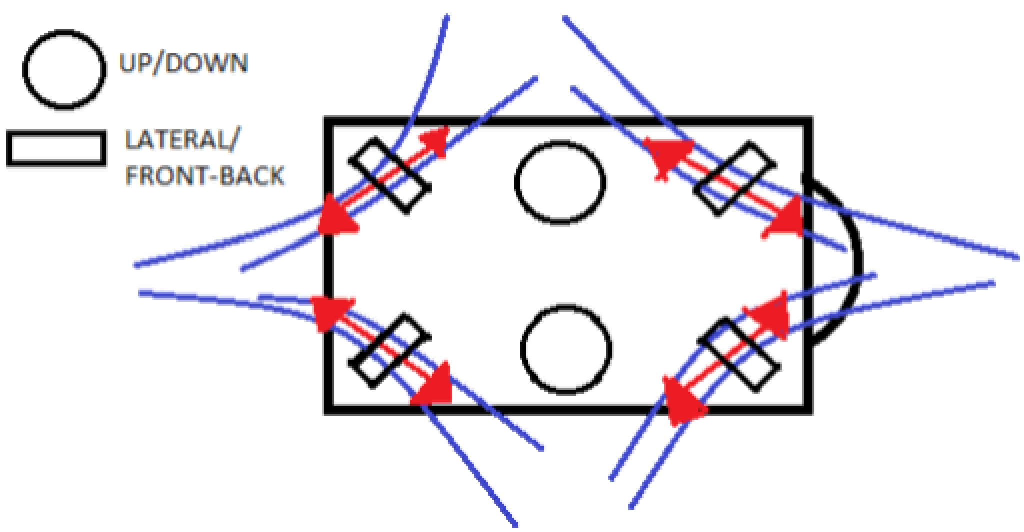

Figure 11 Thrusters configuration

We went with this thruster setup, because after thoroughly researching the subject, we decided this configuration was best suited to our vehicle and the tasks we were going to accomplish with it. Furthermore this also provides more accurate movement, as all thrusters (excluding the up-down ones) work together and contribute to any lateral/forward-backward/strafing movement.

Our next step was to design the actual frame. After a lot of thoughts, and even more sketching, our goal was to use the least amount of material and connections to hold all of our parts in place, and still maintaining a strong, lightweight and rigid frame.

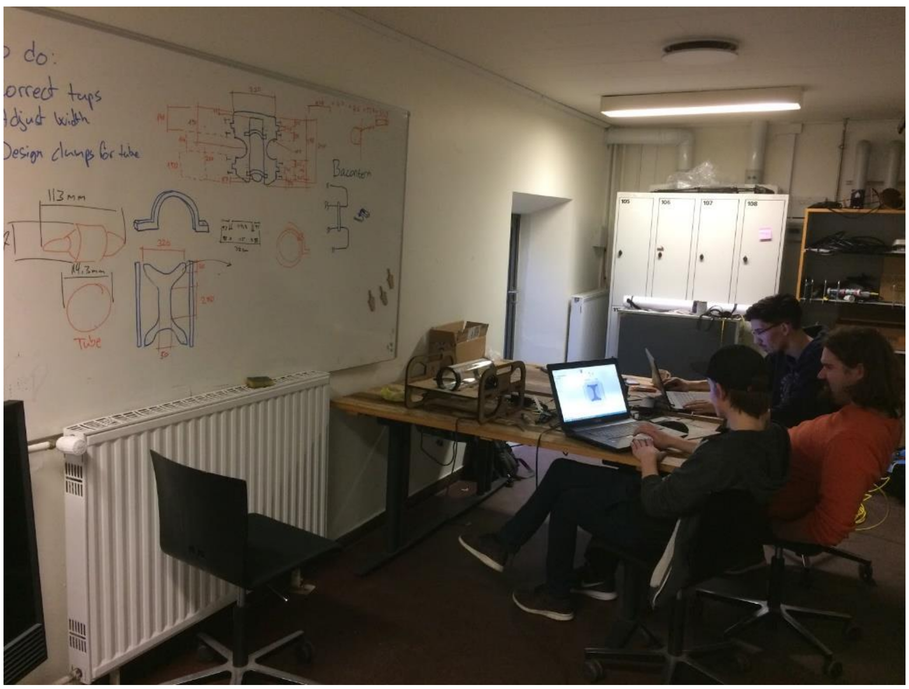

Figure 12 Frame team

Final parts

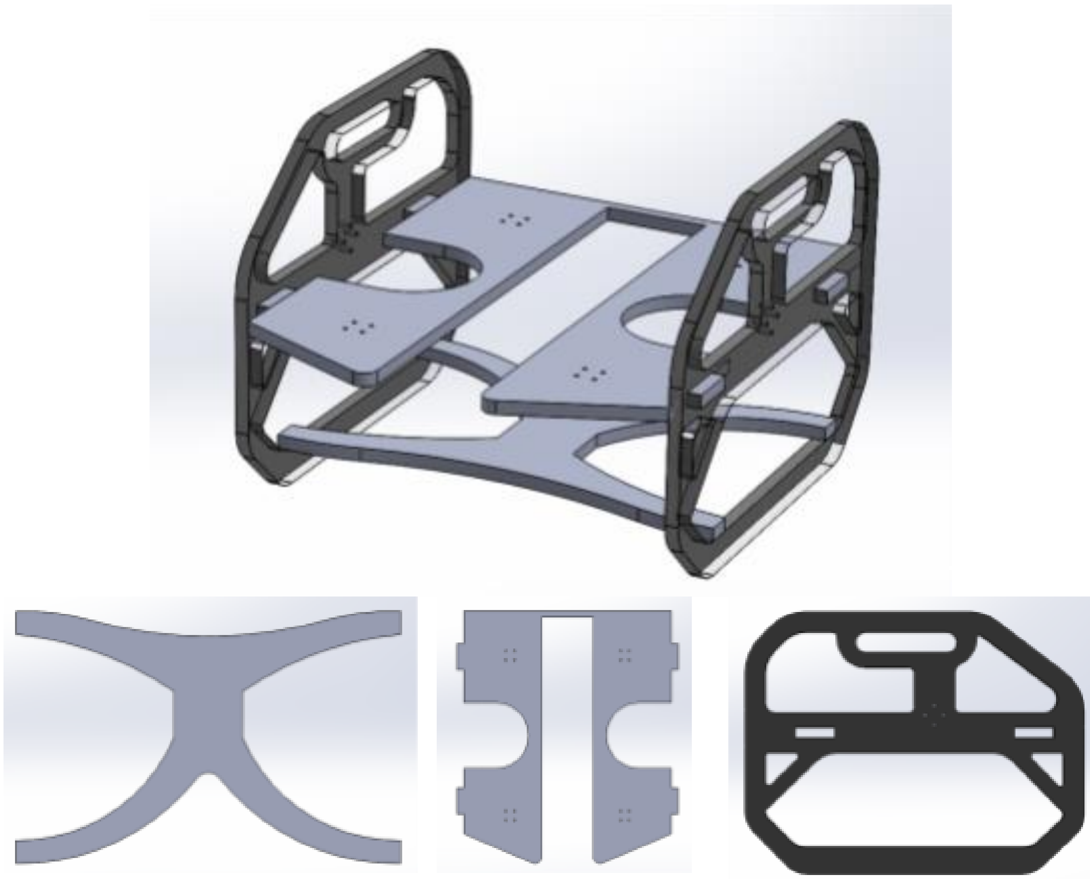

Figure 13 Frame

We made a flow simulation in Solidworks, then laser cut a prototype frame out of MDF, to have a more tangible feel of the size of the frame and after several iterations with minor adjustments, we decided it was ready to go.

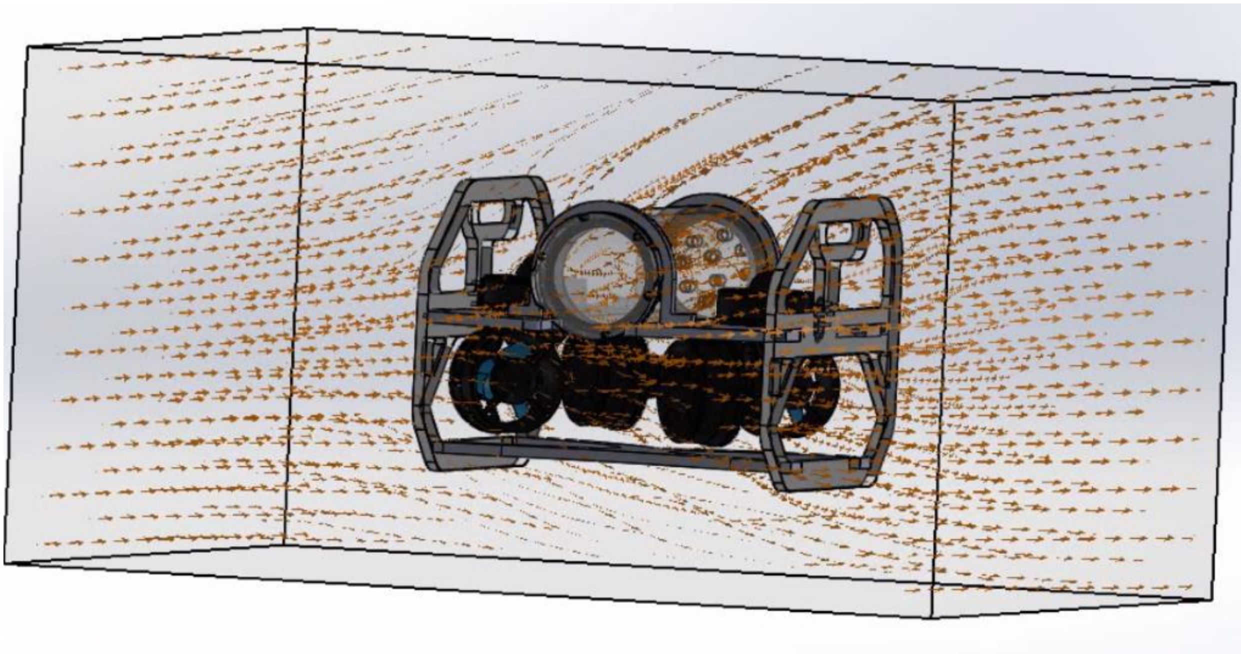

Figure 14 Flow simulation

In choosing the material, we first narrowed down our choices to acrylic, aluminium, other plastics, and HDPE. We eliminated plastic due its high buoyancy, We chose HDPE over acrylic because it matches its strength, but is less brittle than acrylic and therefore much less likely to crack. We chose it as our final material over aluminum after approximating the entire ROV assembly's buoyancy, which would have a had a density and mass too high. With the HDPE, it is minimally positively buoyant, which we counteracted by adding a weight on the bottom of the vehicle, and a little bit of foam on the top, to get an almost neutral buoyancy (very slightly positive, so in any kind of emergency or failure, the ROV will slowly float to the top rather than sink). The added weight at the bottom and foam at the top also contributes to the balance of the vehicle. We used 10mm thick HDPE, which we CNC machined using the solidworks files. Minor tweaks like tapping screw holes for the assembly screws and filing the frame so it wouldn't have any sharp corners was added and we finally had a finished frame ready for having its electronics installed.

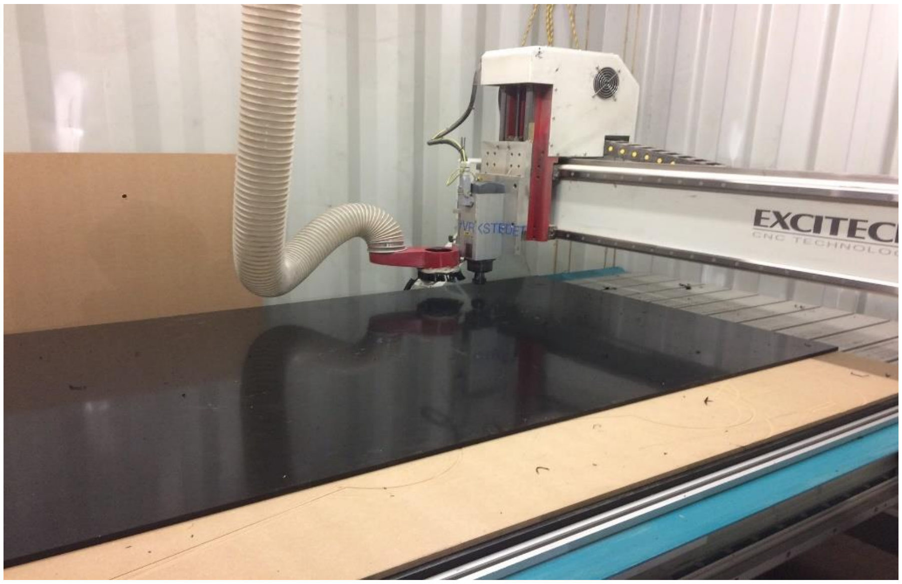

Figure 15 CNC cutting

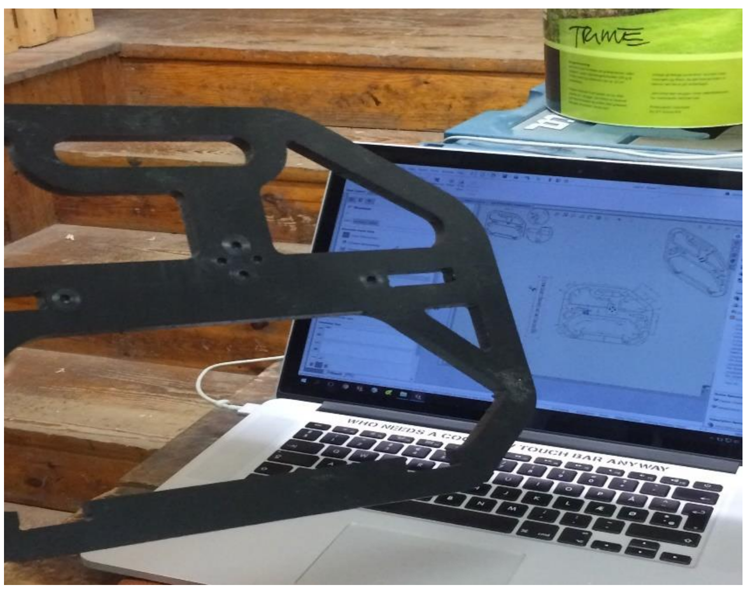

Figure 16 CNC process

Electronics enclosure

For the electronics enclosure we used a waterproof acrylic tube 100mm x 298mm purchased from blue robotics, with an end cap on the back, and a 100mm x 50mm acrylic dome on the front in order to have our camera at the best viewing angle for the tasks given. The tube is connected to the frame using mounting braces also cut from HDPE in order to securely mount the tube though still maintaining a fast way to disassemble the ROV for transportation where minimum space is given.

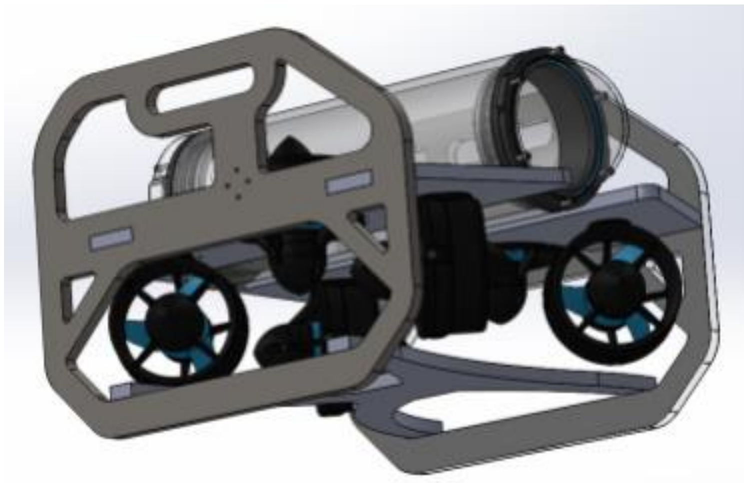

Figure 17 Electronic tube

Thrusters

P100 thrusters with blueESC purchased from Blue Robotics. During our first test run before Christmas, the ESC's on the thrusters burned out. We communicated with Blue Robotics, and learned that the ESC's had a malfunction that was causing them to not run properly. They sent us replacement parts for the thrusters, and we converted them to no-ESC thrusters, and went on to making our own ESC’s.

Electronics description

We have spent 4 months designing, testing and making the electronics for our ROV. The most difficult part was designing PCBs which can work on the first attempt - we knew that we won't have time to make them twice. Therefore all the parts had to be designed in such way that even if we make a design fail, we have to be able to modify the part and make the ROV run.

Figure 18 Box with a laptop, ground station and PCB

Our Electronics consists of 3 main parts:

Ground station

It's basically a box with a laptop, Ground station PCB and 2 wires

Tether

CAT 5 Internet cable and 2x 4mm2 stranded copper wire with silicone insulation and balancing rope, 4 wires of the CAT 5 cable are used for establishing internet connection in between Raspberry Pi and Ground station. 2 wires are used for RS-485 UART communication for sending commands to the Arduino, 2 wires are back up

ROV

Contains majority of the electronics, all important sensors, electric drive and control units.

Electric drive

Figure 19 Thrusters T100

Our ROV is using T100 thrusters from the company Blue Robotics. Thrusters have 11,5A current consumption at maximum power at 12V, creating 2,36kg of thrust forward and 1,82kg of thrust backward. We have decided to go with “ready to go solution” because we felt like we are too short on time for developing something which has already been done.

We are using Afro ESC 30A speed controllers with modified software (SimonK) which is allowing us to update power at 1khz rate over I2C bus. We had to re-solder all the connectors on the ESCs, and upload new firmware into them through Arduino board configured as a programmer.

Active current limitation

The specialty of our ROV is the ability to limit the current consumption to the defined value - meaning, we can set up in the code of the Arduino code “how much current” do we want to use at maximum, and the control unit makes sure that the thrusters newer draw more current than it’s allowed. This is achieved by measuring the current more than 10 000 times every second and updating the power output of the ESCs 1 000 times a second. This way we can make sure that our ROV never draws more than maximum allowed 25A. The chosen 25A fuse has never blown off during testing.

PCB Board description

We have designed 3 PCBs for our ROV. All of them contain 6 copper layers. Majority of the components are SMD with the size as little as 0402. We have soldered them under the microscope at the office of the company UAVComponents ApS. They have also helped us with ordering and delivering of the PCBs and components for them. Two of the boards - Arduino Board and Power Board are placed inside of the ROV. Ground station is inside of the control station, connected to the notebook.

Arduino Board

Arduino Board is a place, where all the logic components of the ROV are connected together.

The board contains:

- Arduino Due

2x Power Supply TSR 1-2450 for Raspberry Pi and Camera Servo

- MPU 9250 for stabilization
- Temperature sensor
- Leak sensor
- LED diodes for signalization of operation
- Voltage dividers for monitoring voltage

We have decided to use Arduino DUE since this is one of the most powerful boards from all Arduino boards line-up and runs on 3.3V logic level that is needed for most of our sensors. Arduino DUE uses ARM microcontroller from Atmel that is running at 84 MHz. This allows us to make many operation at once for example when reading the current value.

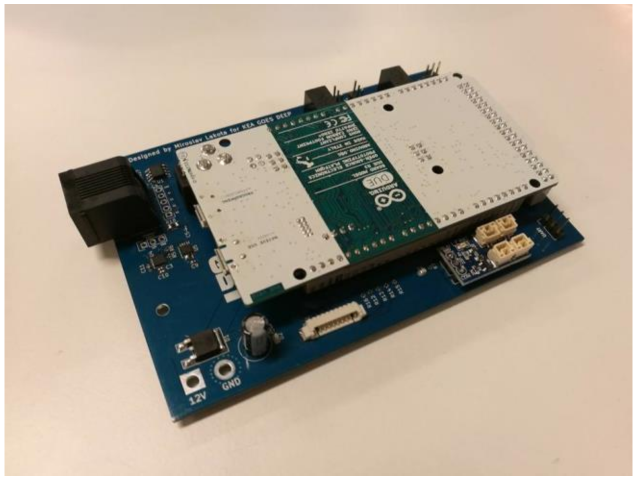

Figure 20 Arduino board

The whole Arduino code consists of two parts - setup and loop. Setup is run just in the beginning of the code. All the initializing of gripper, thrusters, UART communication, analog-digital converter (ADC), camera servo as well as serial communication is done in setup function. Second much bigger part of the code, loop, is the part where the main code is run. to read current value as reliable as possible we read it every time the loop function is executed. This allows us to have precise value of current flowing into thrusters at all time. besides reading current value there are 2 loops running inside of the loop function one with frequency of 1kHz and second one with frequency of 50Hz. Faster loop is used mainly for updating thrusters. First thing to be done in 1 kHz loop is reading average current value. Afterwards appropriate values of power to thrusters are calculated based on input from controller. Then the current limiting algorithm runs to prevent high current draw. Later these values are set to ESCs. At the end of this loop all the servo values are calculated by the controller value and sent to servos. The slower loop is used for reading data from sensors.

Power Board

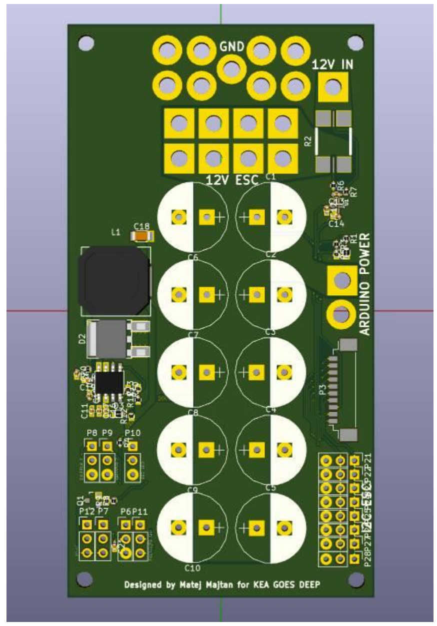

Figure 21 Power board

This board was designed to be a kind of power hub distributing power to every part of ROV. Power board also serves as a breakout for other connectors for gripper servos, light, ESC, red LED and pressure sensor. The board incorporates a current and voltage sensor for reading values of current and voltage used in power circuit. The power connection is established by a cable with Anderson connector soldered right on the board. Afterwards the current flows through 1mΩ shunt resistor and the voltage drop over this resistor is later amplified by INA199 and sent through data bus to arduino to get analyzed. afterwards the current flows straight to into ESCs minimizing flow of high current through the board. In parallel to power connection to ESC are connected 10 capacitors (each 2200µF) to ensure that voltage in the circuit won't drop when spikes are occurring in current draw. For powering gripper servos there is 5V power supply (TPS54560) on the board. The Board has also power and data input/output which is going to the Arduino board

Ground Station

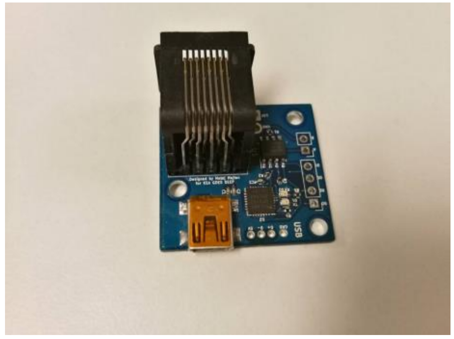

Figure 22 Ground station board

Ground station board serves as a converter of signal. It converts USB protocol to serial RS-485 protocol. For this we use intermediate level of conversion that is UART. So the whole conversion starts at USB protocol that is converted to UART by CP2102. This UART signal is later connected through 2 LED indicating data transfer to SP485CN that converts the data to RS-485. For ease of use RS-485 chip was set up just as a transmitter therefore it is not possible to receive any data sent to computer. The whole board is powered by USB connection thus it is running at 5V.

Layout of the components in the enclosure

Laying out components inside of enclosure was quite of a challenge since the space inside of it is limited. We started by designing mounting board. The board is held by walls of enclosure so it doesn't need any kind of support and that leaves more space for electronics components. In the front of the enclosure there is the camera mount with servo to turn it. Right after it is Arduino shield board that has connector for camera mount conveniently placed right next to the camera mount. after arduino shield there is the power board. This board is connected with Arduino shield by a 10 wire data bus cable and by power connection. These were designed to be on the same side of the boards to eliminate long cables running in enclosure. Power board has all the small connectors for components outside of enclosure as close to end of it so the cables are going straight out. Raspberry Pi is mounted under Arduino shield so it has easy access to camera cable as well as power from arduino shield. Last part of components inside of the enclosure are ESCs. They are mounted under power board so that the power wires and I2c data cables are as short as they can be. ESCs were re-soldered and designed the way so they are easy to swap since it is the most unreliable part of ROV. All cables coming out of ESC were made to have connectors that are easy to unplug and the whole mounting is made by Velcro. So in the need of replacing ESC we will just upload new code to ESC with the same address as the last one and simply replace it. It's good to mention that to prevent any shorts in the case we used connectors that are not able to be swapped. We also used number and color marking to indicate ESC and thruster wires so we are always sure we connect it well.

Raspberry Pi

When the raspberry pi was delivered we installed the newest version of Raspbian which worked perfectly with the Raspberry pi 3. Raspberry pi 3 Model B has a 4 core 1,2Ghz processor and 1GB of RAM. Raspberry Pi Camera uses 8Mpx chip.

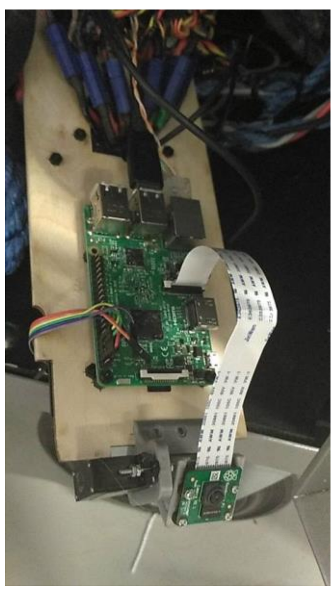

Figure 23 Raspberry PI Raspberry pi 3 Model B has a 4 core 1,2Ghz processor and 1GB of RAM. Raspberry Pi Camera uses 8Mpx chip

Raspbian allowed us to use VNC and SSH for configuration. At first we programmed through a remote laptop using SSH program that allowed us basic communication and controls over the raspberry. Later we used VNC for further controls and file transfer. On the Raspberry Pi we installed Arduino IDE for reading, writing, compiling and uploading code to the Arduino Due inside the ROV which is connected to one of the USB ports on the Raspberry Pi. For video recording we used a raspberry pi cam that is part of the raspberry kit so we avoided software and hardware conflicts. The video is first recorded in a H264 format then compressed and sent to the ground station laptop through the GStreamer which is supported by both Raspbian and Linux. Transmitting the video could be done through VMP or the GStreamer which cut the 3-5 sec. streaming delay of VMP down to less than a couple of milliseconds.

QT Application

In order to have full control of the ROV, to be able to maneuver in tight spaces one does need a user-friendly and adaptable program which allows him to adjust the controls to his very needs. In the beginning we didn’t have a clear vision of what our application should be able to do. The only thing we knew was that we wanted the ROV to be controllable with a joystick. As a result of that our tasks were simple and random at the start. The programming language of our choice was java since some of us were just in the first semester of a computer science course learning java and had no experience with manipulating hardware. The first attempt to do so was to try to make a connection with an arduino and manipulate its LED. That was done pretty quickly. The next goal was to track the position of a joystick. Before we made any attempt to communicate with the joystick we first created canvas which tracked the position of the mouse cursor. After that we began searching for a suitable library which would manage the input of any USB-connected devices. Meanwhile we received a donation in the form of an original Xbox Controller. It got modified to fit into a standard USB-A port. After spending countless hours of trying to make windows detect the input through JXInput and JInput we just gave up because obviously 32 bit libraries are not supported on 64 bit windows machines.

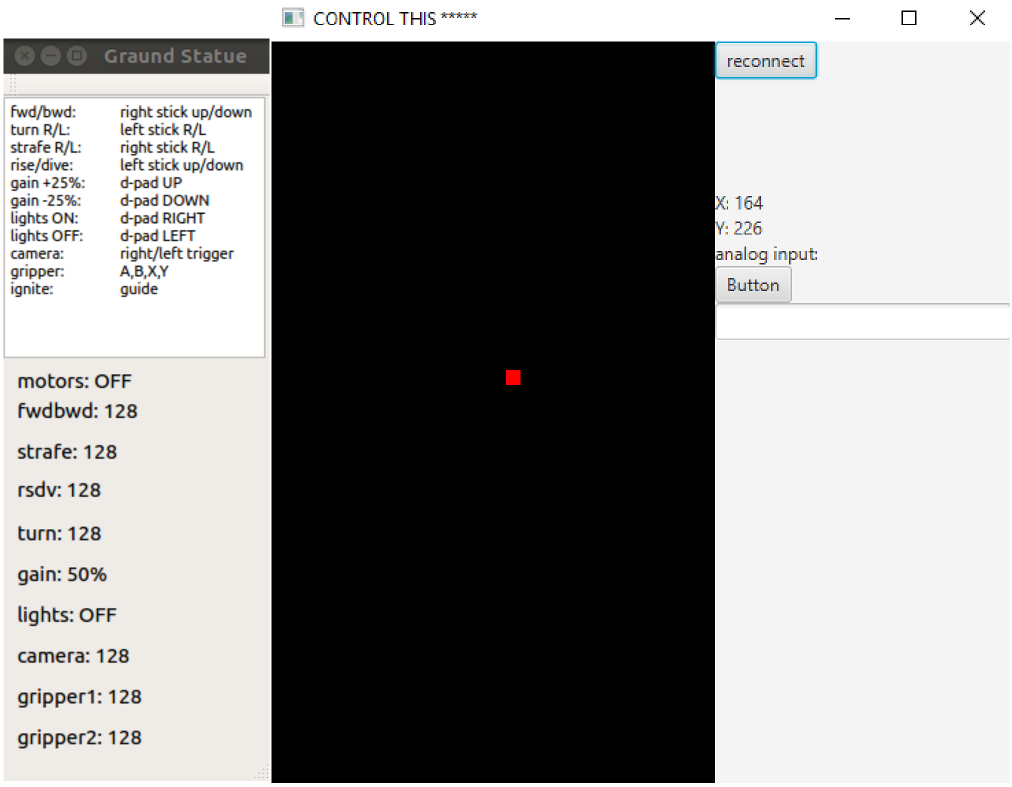

Figure 24 QT Application

Ultimately we began coding in Qt following several of the QGamepad examples and then we made an application that takes in all inputs from an Xbox 360 Controller, modifies them to our needs and sends them through a serial port into the ROV. Also the whole program was moved to a linux machine because windows just cannot do any of the good stuff. There was a complication with the serial port because sometimes a zero value was sent in the middle of a QByteArray and it caused a disorder in the receiving port but we managed to solve this at a cost of losing some of the resolution. The application uses a GUI to display all of the output values to the driver. The movement is controlled by analog sticks with code-added blind areas because the sticks never align dead in the center when let go. The thruster output is also secured by a safety lock mapped to one of the buttons to prevent unwanted propelling. The computer is running on Intel® Core™ i5 CPU M 480 @ 2.67GHz × 4 chip operated by ubuntu 16.04 LTS. The program has been programmed in Qt Creator 4.2.2 based on Qt 5.8.0.

Possible future development of the electronics

Electronics hardware inside of the ROV is really advanced and currently it's limited by our software and frame design.

We are currently working on implementing:

-stabilization in all controllable axis, including depth stabilization, we would be able to achieve full 3D stabilization in 8 thruster configuration

-feedback from the Arduino Due to the control station through Raspberry Pi

-10W LED lights for low light level conditions vision

If we were designing the electronics again, we would have used STM32 microcontroller and made the Arduino board way smaller and cheaper.

SID

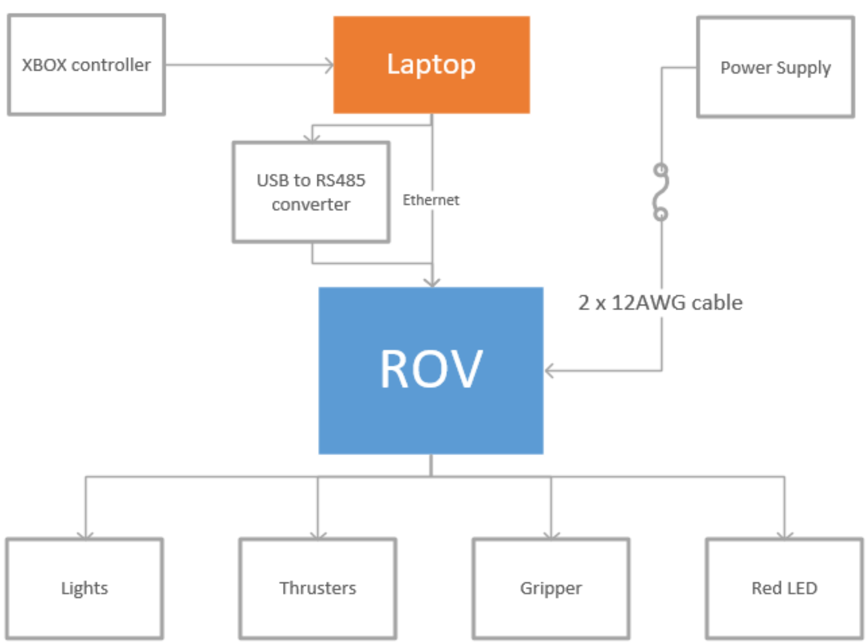

Figure 25 An individual electronics parts connected

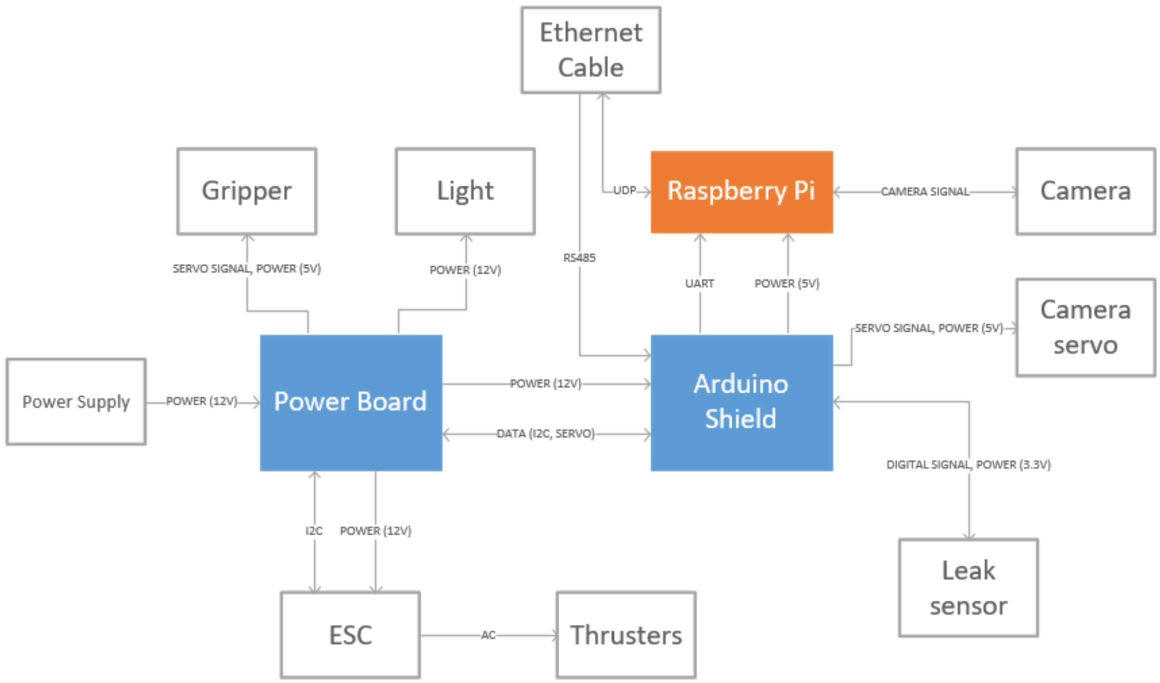

Figure 26 Connections and components inside of the ROV

Safety

During the design process of the frame and electronics we've strived to achieve a design where all parts moveable parts like thrusters and the gripper are shrouded and or covered. Our frame design is based on rounded corners and natural curvatures. While constructing the frame we've been using CNC machines as laser cutters and routers. Working with these requires extreme caution and use of necessary safety equipment like goggles and hearing protection. During the design process safety has been taken into consideration. In depth safety regarding the electronics and MECH standards is addressed in our safety report.