DIY Remotely Operated Vehicle

Design review and recommendations

Authors:
Artur K. Lidtke
Aleksander A. Lidtke

September 3, 2016

Abstract

Need to write

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Aim

The aim of this project was to develop a low-cost remotely operated underwater vehicle (ROV) for operation in calm, shallow waters. Its purpose was to allow principle physical and design challenges of building a robotic underwater platform to be explored first-hand and to highlight possible future paths for the development of a more refined design. The intention of the project was not to deliver a state-of-the art vehicle and instead focus was put on building experience and appreciation for the engineering challenges involved. The ROV was to be manufactured, as much as possible, from easily available and affordable off-the-shelf electronics and components. This would allow the devised solutions to be easily reproducible by anyone with access to the most rudimentary tools and manufacturing processes, making it easier to share the developed experiences. The entire project, including software, CAD models, and design calculations is provided on GitHub at https://github.com/UnnamedMoose/DIYROV.

1.2 Objectives

To achieve the specified aim, several objectives have been specified at the outset of the project:

1. Create a simple software framework allowing basic electronic components to be controlled from a laptop via USB
2. Select a suitable means of propulsion for the ROV
3. Come up with a basic set of sensors which would make it easier to control the vehicle and provide basic feedback to the operator
4. Integrate the selected components into a coherent system and devise an arrangement that fits into a volume as small as possible
5. Design a watertight vessel to house the electronics using off-the-shelf, cheap DIY products
6. Assemble the ROV and test it in a calm environment such as a swimming pool or a towing tank

1.3 Subsequent parts of this document

The following parts of this informal report will address each of the listed objectives, and discuss them it in the context of the ultimate aim. The intention is to convey both the engineering rationale and experience gained throughout the project, rather than keeping the writing strictly technical. The report is accompanied by the complete release of the necessary source code (both on-board microcontroller and shore-based computer), CAD models of the ROV, and design calculations, all of which combined should allow the Reader to reproduce the process undertaken at their own leisure.

Chapter 2: Manufacture and assembly

2.1 Architecture

Overview of the ROV system is shown in Fig. 2.1, while Fig. 2.3 and Fig. 2.2 show the general dimensions of the vehicle and location of its major components. The ROV is based around an Arduino Uno R3, which is controlled and powered through its serial port connected to a laptop. To overcome range limitations of ordinary USB cables, a USB range extender is used, which transmits the USB signal and power over a CAT5 cable.

The ROV houses one camera, which is operated over the same USB bus as the Arduino (the USB signal is split using an on-board USB hub). The camera field of view can be illuminated using three LED lights. Only one additional sensor, a 30 PSI pressure transducer, is present on board.

The ROV carries four 750 kV brushless DC motors driven by four electronic speed controllers (ESCs). These enable forward/backward and up/down motion, as well as give control of yaw and roll. Preliminary study revealed that incorporating two additional motors, which would allow sideways motion, would require too big an investment. It would also increase the ROV displacement, thus making the vehicle larger than desired. The selected ESCs (chosen due to their low cost) can only provide thrust in one direction; a custom printed circuit board (PCB) is used to reverse the thrust of individual motors.

The power to the motors is delivered from a shore-based battery pack via a three-wire power chord, which, together with the CAT5 cable, constitutes the ROV umbilical tether. The signals and power from the umbilical are delivered inside the pressure vessel via a custom water-tight connector. In the current implementation, only the electronics sensitive to water are housed inside the pressure vessel while the brushless DC motors are exposed to water as shown in Fig. 2.4.

The final subsystem of the ROV, not shown in Fig. 2.1, is the external steel structure. It provides a mounting point for the motors and ballast, as well as protects the ROV from hitting the bottom with the pressure vessel. The tether is also attached to this external structure, which can carry more load than the umbilical connector; this becomes important during emergency recovery, when the ROV is lifted out of the water by pulling on the tether.

The remainder of this chapter describes how the mentioned subsystems were manufactured, and how they are integrated into the system shown in Fig. 2.1. Components are grouped into subsystems based on their function. Thus, section 2.2 describes the electronics, which provide the ROV with its top-level functionality. The pressure vessel, made of PVC pipes and providing water-tight volume for the electronics, is described in section 2.3. The vessel is surrounded by an external structure described in section 2.4. Lastly, the umbilical tether is described in section 2.5.

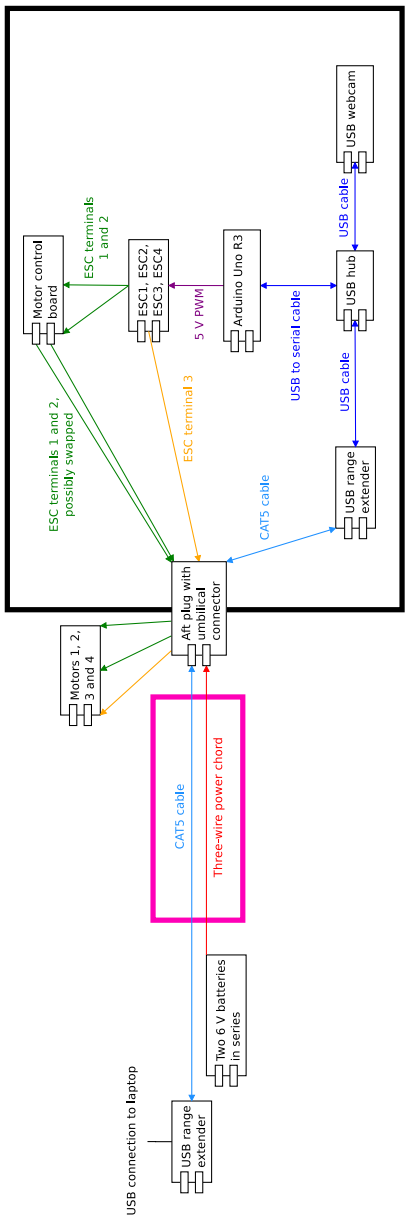


Figure 2.1: The ROV system block diagram. CAT5 cable and Three-wire power cord constitute the umbilical tether (marked with the pink rectangle). The black rectangle marks the extents of the pressure vessel.

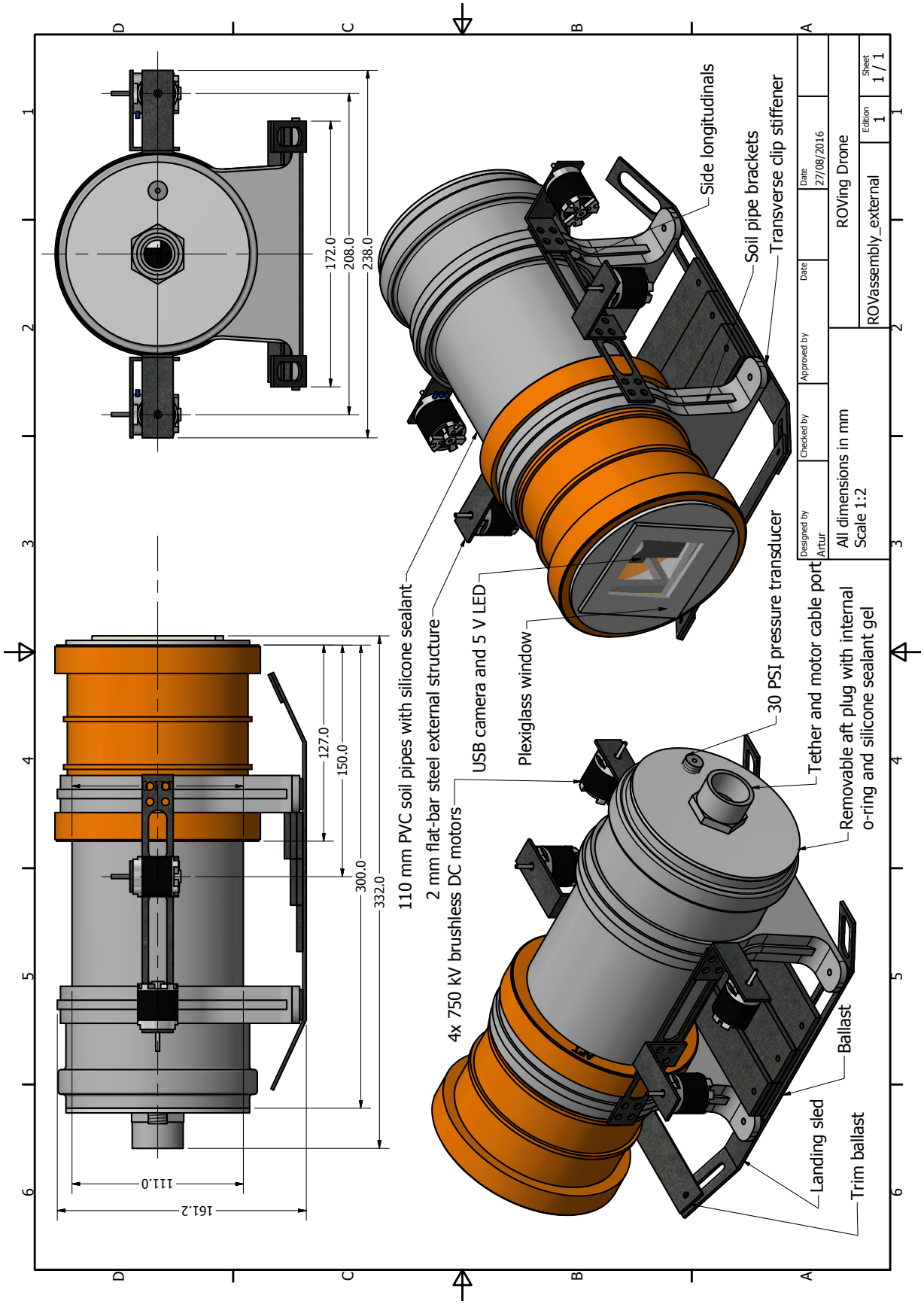


Figure 2.2: External general arrangement of the current version of the ROV. Drawing not to scale, original provided in a separate file.

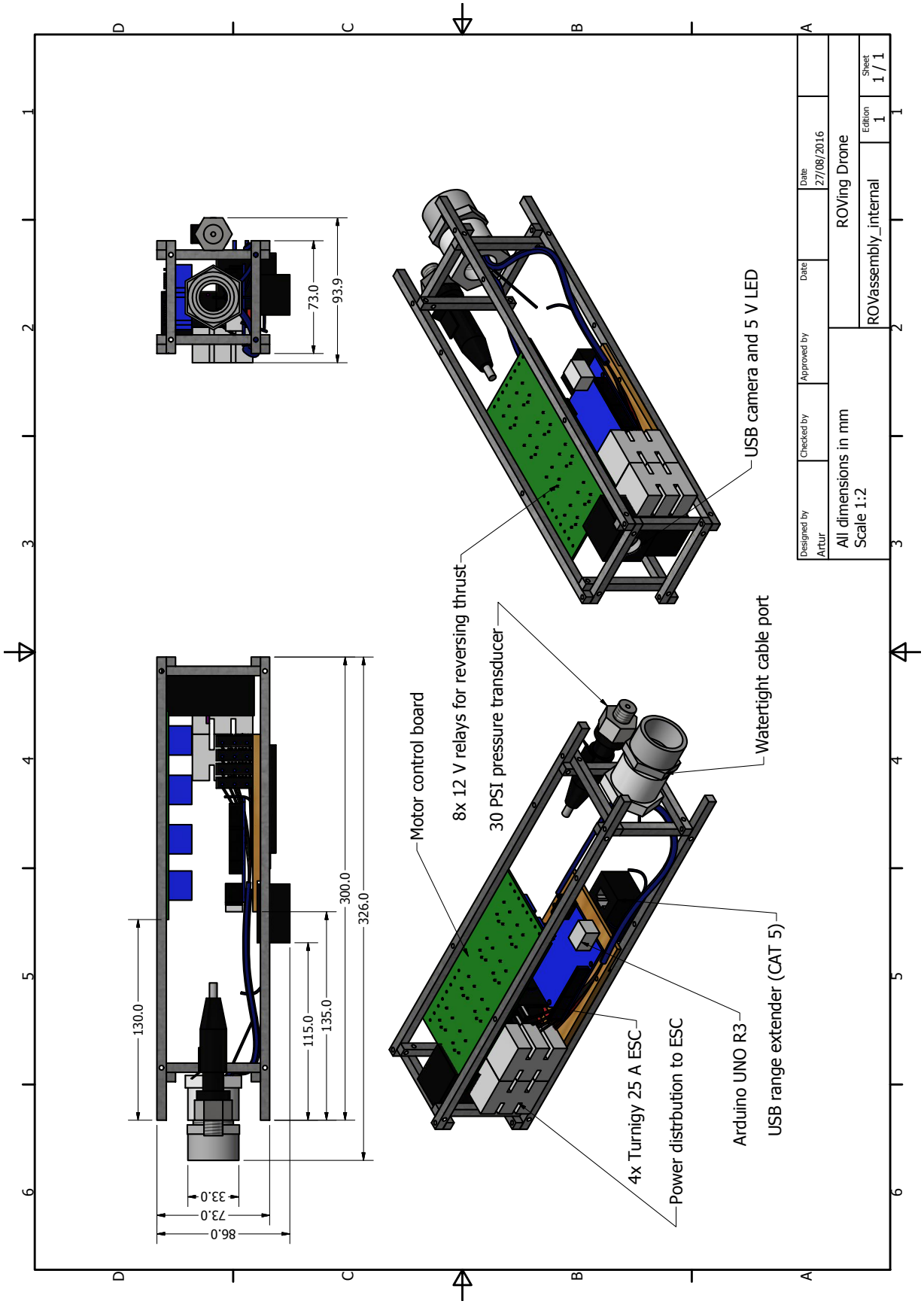
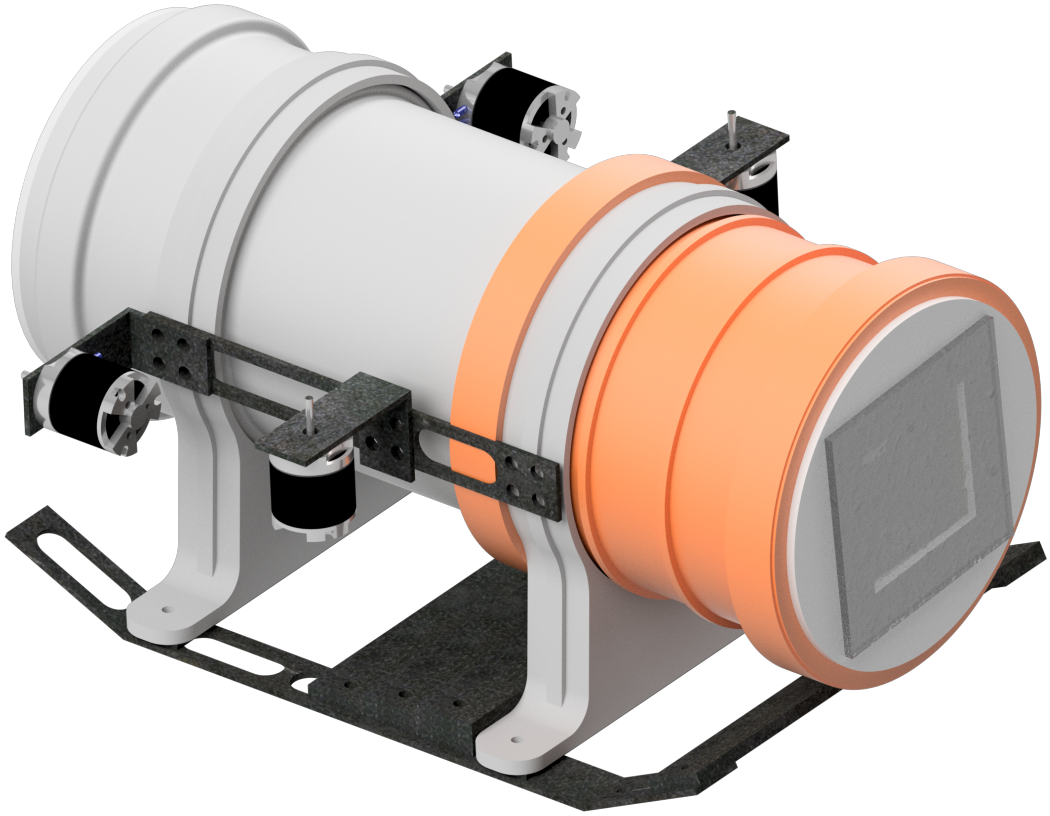


Figure 2.3: Internal general arrangement of the current version of the ROV. Drawing not to scale, original provided in a separate file.



(a) CAD render

(b) Photo

Figure 2.4: Overview of the ROV.

2.2 On-board electronics

2.2.1 Architecture and physical arrangement

The electronics are mounted to an internal structure, which is made of 4 by 4 mm aluminium bars. The bars are bolted together to form a frame with a square cross-section. The frame, with the electronics attached to it, slides into the pressure vessel as shown in Fig. 2.5. A PVC plug (aft plug from section 2.3.1) then slides in and seals the pressure vessel. The electrical connections between the internal electronics, and the motors and the shore are routed through a connector in the aft plug, described in section 2.5.1.

The Arduino, USB hub and ESCs are mounted onto a plywood bed, which bolts onto the aluminium frame. This plywood bed and the USB hub are visible in Fig. 2.6. The PCB, which enables the thrust of the motors to be reversed (described in section 2.2.4), is also bolted onto the frame, directly above the plywood bed as shown in Fig. 2.7. The camera is integrated into a stand-alone module with three LEDs, as described in section 2.2.5; this module is attached to the aluminium frame using cable ties. The USB range extender and power distribution units (described in section 2.2.2) are also attached to the internal frame using cable ties.

Figure 2.5: Electronics are mounted to the internal aluminium structure, which slides into the pressure vessel.

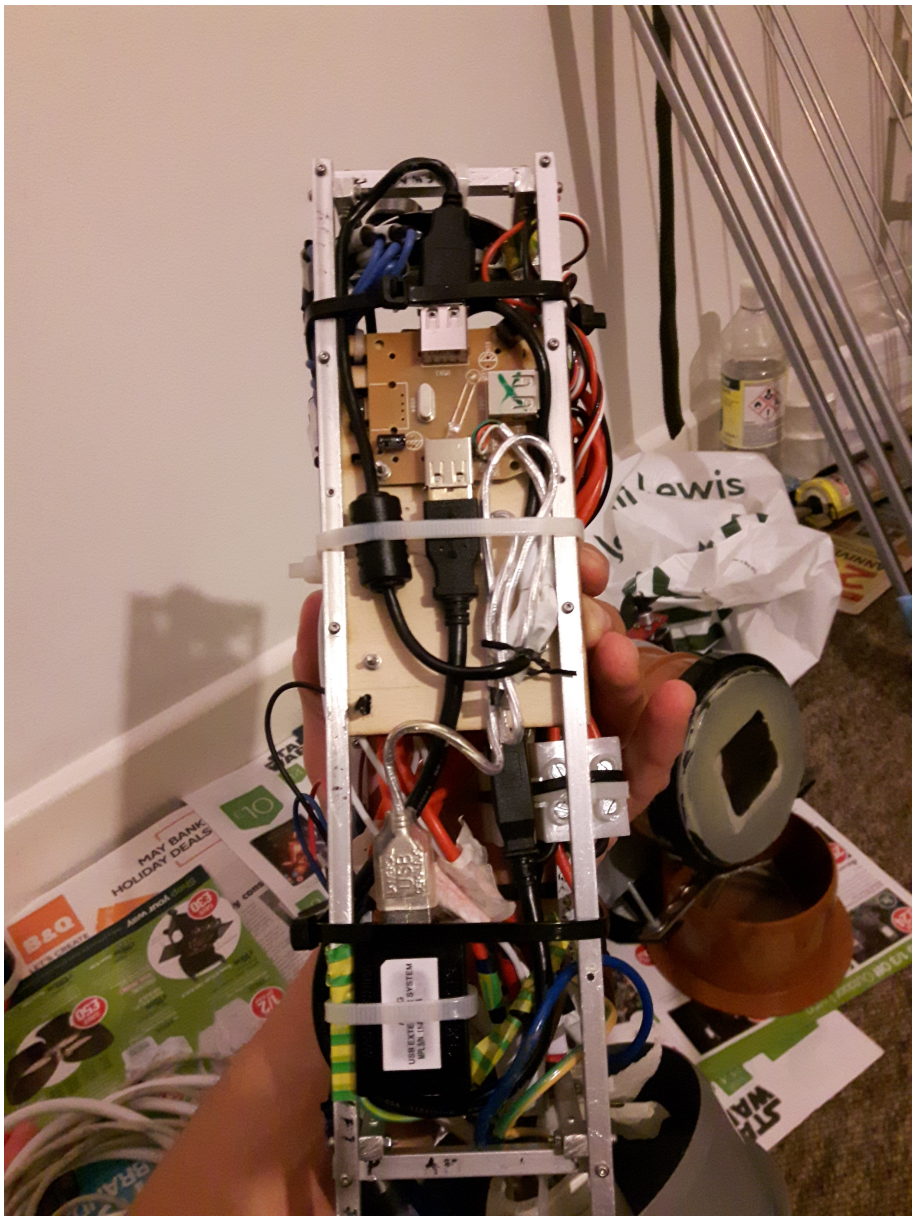


Figure 2.6: USB range extender (black box on the bottom left of the phot) is mounted to the aluminium internal structure with a white cable tie. The USB hub, into which Arduino and the webcam are plugged in, is bolted onto the underside of the plywood bed.

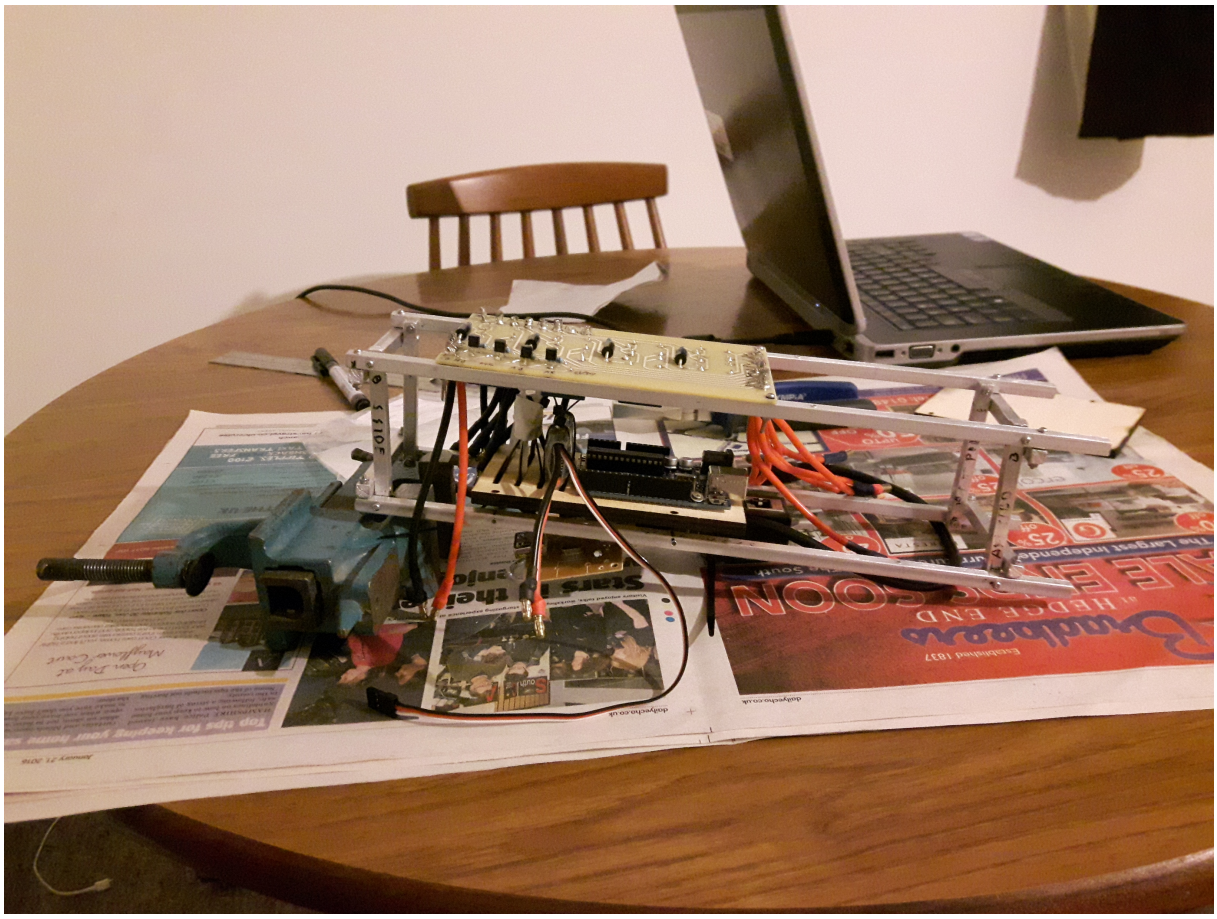


Figure 2.7: General arrangement of the internal electronics. Only one electronic speed controller, and no LEDs and camera are mounted for clarity.

2.2.2 Power distribution subsystem

Block diagram of the on-board power distribution subsystem is shown in Fig. 2.8. The Arduino and the camera are powered directly from the USB bus delivered on board via the CAT5 cable, which forms part of the tether (c.f. section 2.5). The USB bus is changed from the CAT5 cable to a standard USB cable using the USB range extender, and then split by the USB hub to deliver power to both the camera and the Arduino.

The pressure sensor is powered from the Arduino 5 V pin. The LEDs are turned on by setting an Arduino pin high, which provides sufficient power for them to operate. Both the pressure sensor and the LEDs are connected to the Arduino ground, and connections to both peripherals are realised using hookup wires.

The power for the ESCs and the motor control board is delivered from the shore-based battery via the power cable, which constitutes part of the tether. Custom power distribution unit (PDU) and common ground for these five components were manufactured out of automotive cable connectors. These components are shown in Fig. 2.9; they connect the ESCs and the motor control board in parallel.

The ground (GND) of the Arduino is connected to four ESCs. However, these connections are separate from the ones that deliver power to the ESCs (negative battery terminals). Whether the Arduino GND and and battery GND are connected via the ESCs is unknown. The motor control board connects the Arduino and battery GND, as described in section 2.2.4.

Power is delivered to the ESCs and motor control board using gauge 32 wires (thick lines in Fig. 2.8). All other connections in Fig. 2.8 are realised using hookup wires.

The third wire in the power cable (besides battery positive and negative terminals) is used to ground the internal structure. This is intended to protect the operators in case the battery positive terminal short-circuits to one of the internal components or the aluminium structure. The external structure is not grounded.

The shore battery pack constitutes two 6 V batteries connected in series. These are housed inside an aluminium box, which also incorporates a 20 mm fuse, switches and leads that can be used to monitor the battery voltage. Disassembled battery pack is shown in Fig. 2.10.

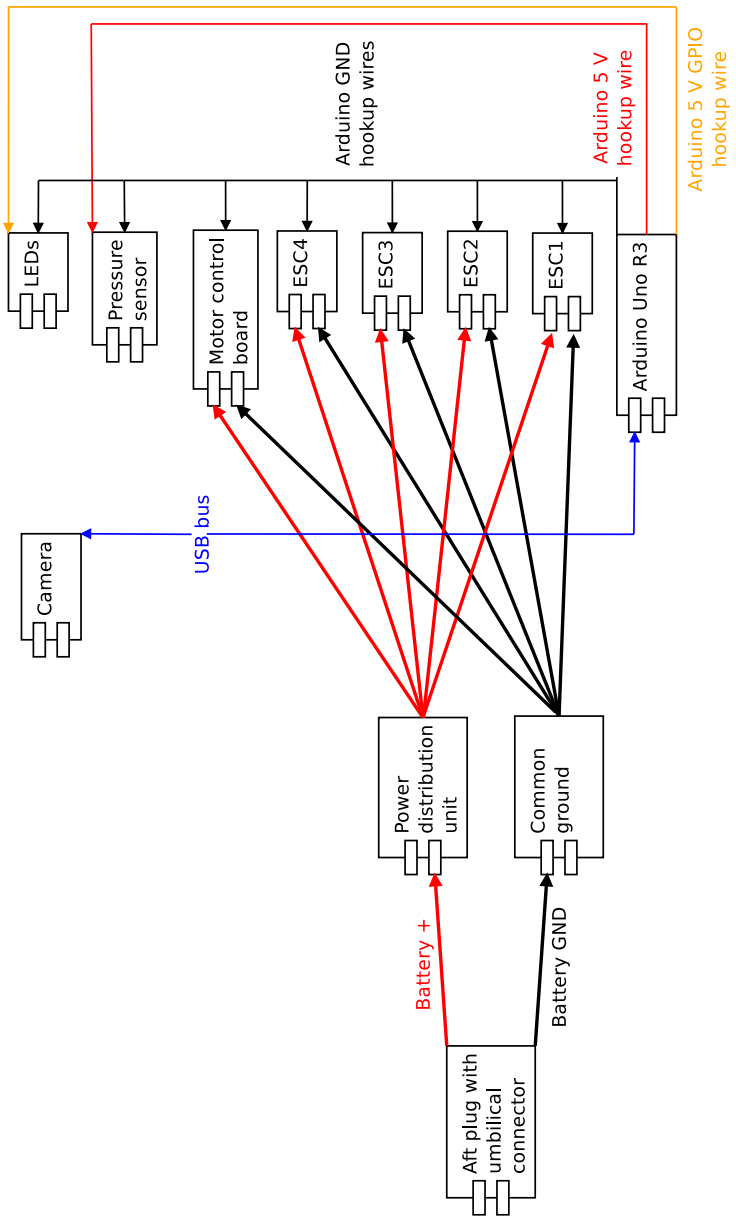
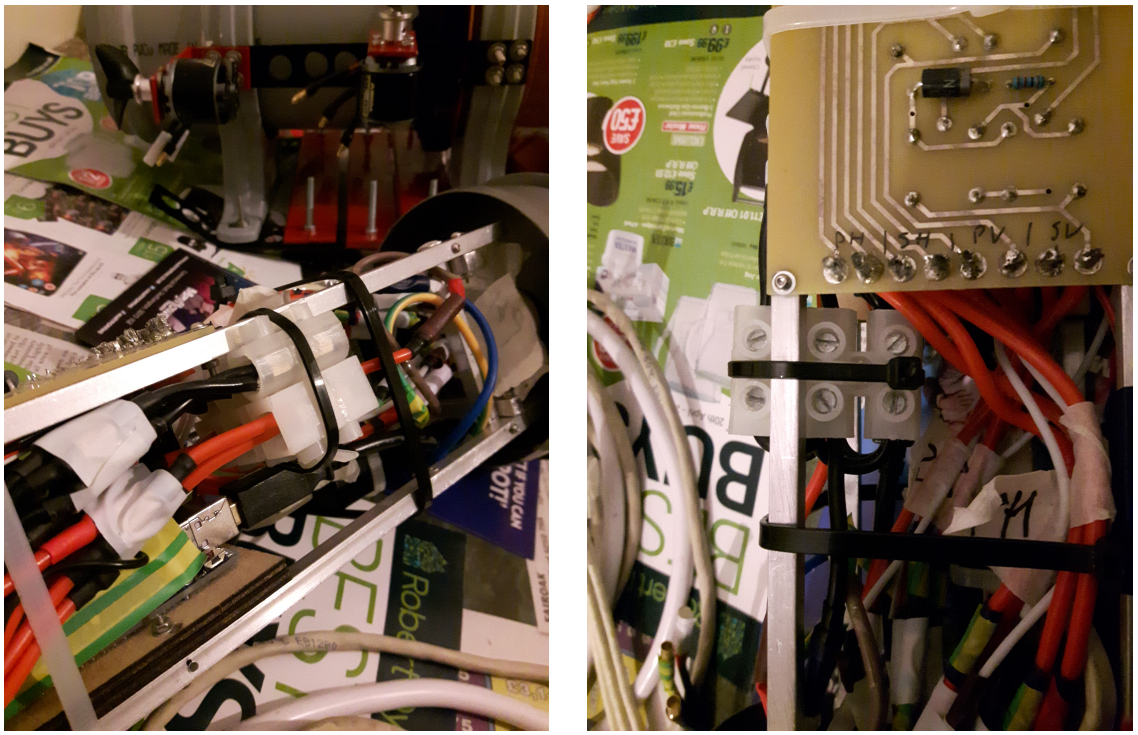


Figure 2.8: Block diagram of the power distribution subsystem. Power is delivered to the aft plug using the umbilical tether, which is connected to the battery pack.



(a) Side (b) Top

Figure 2.9: Side and top views of the common ground and common power lines made out of screw automotive cable connectors and mounted to the aluminium internal structure with cable ties.

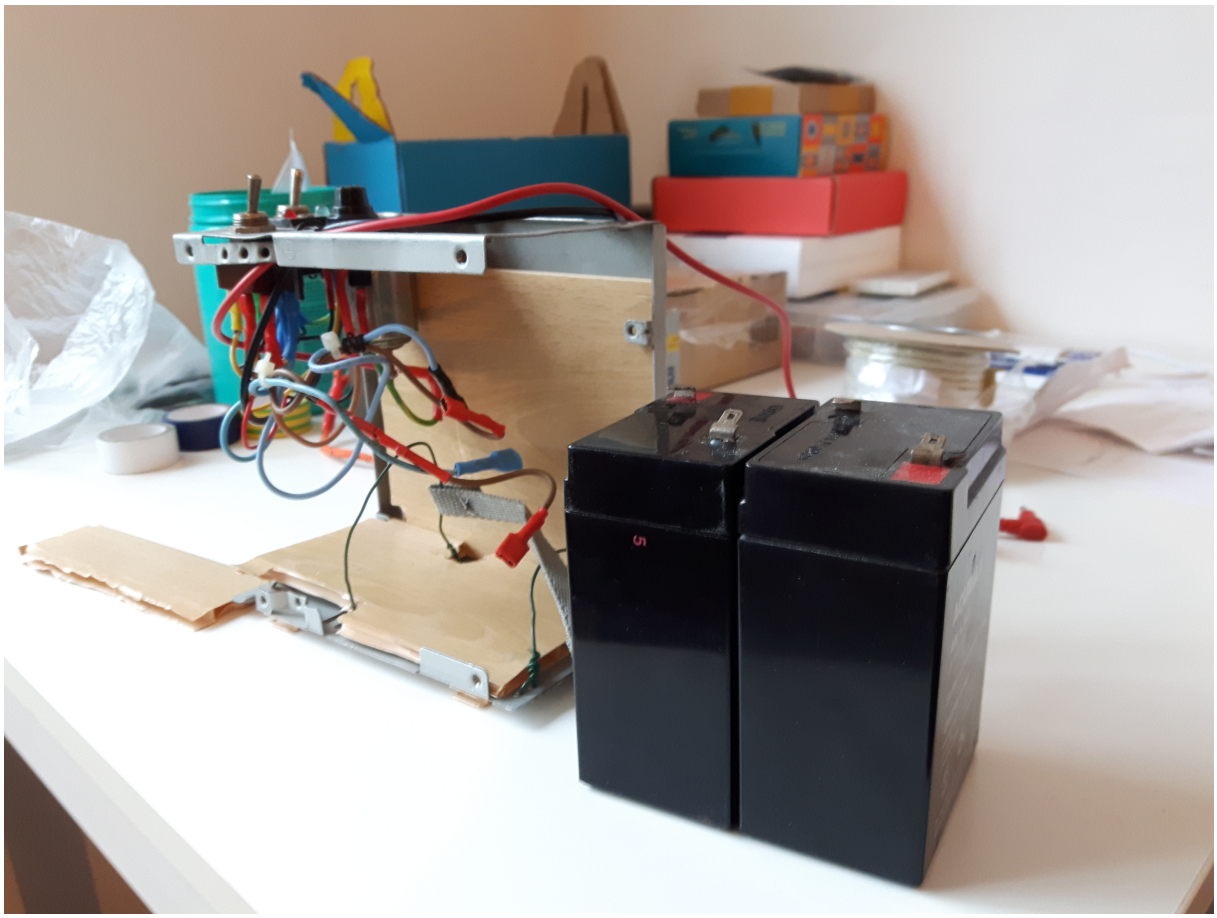


Figure 2.10: Disassembled battery pack. One switch enables the batteries to be connected in parallel or in series (series configuration is used for the ROV project) while the other one switches the power on or off. The output of the battery pack is protected from excessive currents using a 20 mm fuse.

2.2.3 Propulsion subsystem

Block diagram of the ROV propulsion subsystem is shown in Fig. 2.11. It is based on four brushless DC motors that are driven by four commercial, off-the-shelf electronic speed controllers. Every ESC converts a pulse width modulation signal (PWM) from the Arduino into pulses of power delivered to every of three terminals on the brushless motor. Changing the frequency of the PWM changes the frequency of the power pulses, which changes the rotation frequency of the motor and so also its thrust. This PWM signal is delivered from the Arduino pins using hookup wires. The ESCs draw power, which is then delivered to the motors in pulses, directly from the shore-based batteries, which is described in section 2.2.2.

In order to reverse thrust of the motor, it suffices to swap two out of three signals output by the ESC. This is achieved by a custom printed circuit board, which is described in more detail in section 2.2.4. Two out of three outputs of every ESC (terminals 1 and 2 in Fig. 2.11) are routed through this board, before they are delivered to the corresponding motors using the connector in the aft plug using red wires in Fig. 2.5. The third signal of every ESC (terminal 3 in Fig. 2.11) circumvents the motor control board and connects to the corresponding motor only through the connector in the aft plug (white wires in Fig. 2.5). The commands to swap the thrust of a given motor are issued to the motor control board via hookup wires connected to the Arduino pins.

The ESCs are mounted to the plywood bed ahead of the Arduino using cable ties (the aft-most ESC is visible in Fig. 2.7). The ESCs connect to the motor control board, the aft plug connector, and the motors using gauge 32 wires.

2.2.4 Brushless-motor control board

As explained in section 2.2.3, two signals from every ESC have to be swapped in order to reverse the thrust of a motor. The schematic of the circuit, which achieves this, is shown in Fig. 2.12. Nominally, terminals 1 and 2 of every ESC (e.g. ESC1T1 and ESC1T2 for ESC no. 1) are connected to terminals 1 and 2 of the corresponding motor (e.g. M1T1 and M1T2 for motor no. 1). When a command to reverse the thrust of a given motor is issued by the Arduino by setting one of its pins high (e.g. pin 2 for motor no. 1), an n-type mosfet activates two switching relays. This connects terminals 1 and 2 of the ESC with terminals 2 and 1 of the motor, respectively, thus reversing the thrust. When the command pin is set low, the relays switch back to the nominal configuration and thrust is no longer reversed.

The power output of the Arduino is insufficient to activate the relays - relatively large relays, which can withstand currents output by the ESCs, have to be used. Thus, the power to the relays is delivered by the shore-based batteries. Every ESC-motor pair requires two switching relays, and every two relays have one flyback diode.

A two-layer PCB houses the entire circuit from Fig. 2.12; its layout is shown in Fig. 2.13. The motor control PCB is manufactured with 1 oz Cu and has trace widths of at least 1 mm rated to at least 2.5 A. The design of this board is open-source, and can be accessed in its repository at: www.upverter.com/AleksanderLidtke/7eccd1441d1097f8/BrushlessMotorControlBoard/.

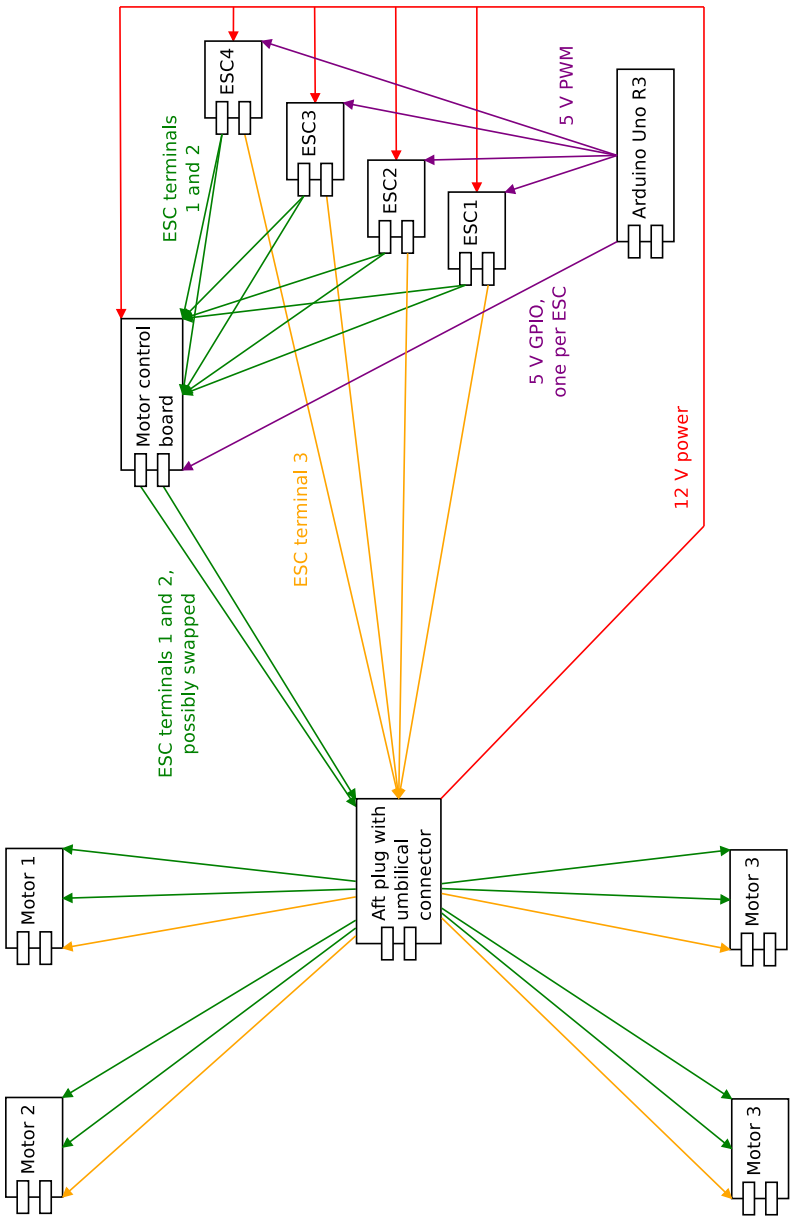


Figure 2.11: Block diagram of the ROV propulsion subsystem. Connections from the Arduino are realised using hookup wires. All other signals (power, and ESC signals 1, 2 and 3) are carried using gauge 32 wire.

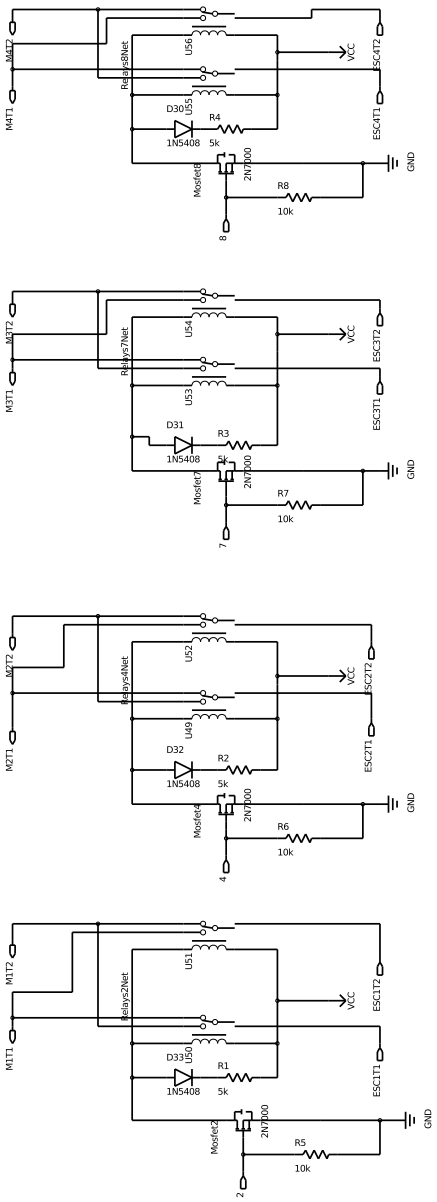


Figure 2.12: Schematic of the brushless motor control PCB. The board connects terminals of ESCs to motor terminals via pairs of switching relays.

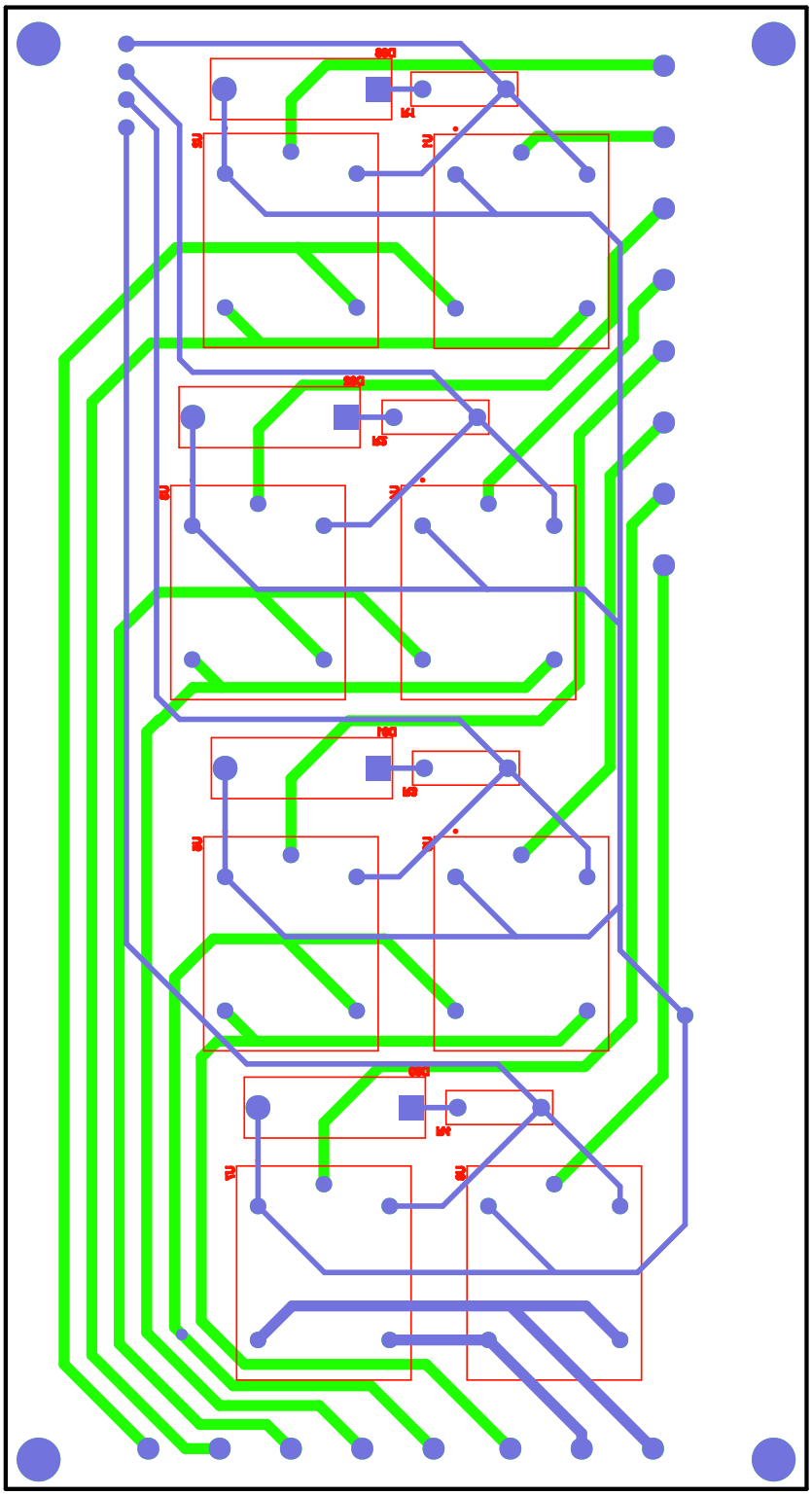
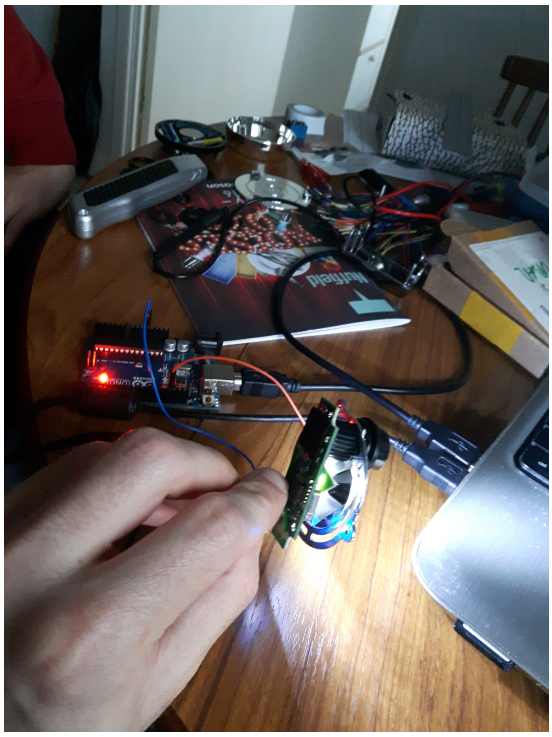


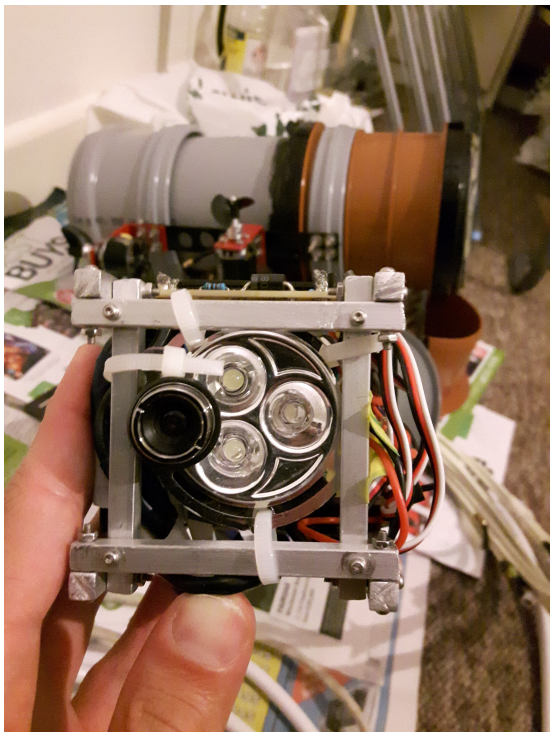
Figure 2.13: Layout of the brushless motor control PCB. Bottom copper marked with green, top with blue, and component outlines with red. Switching relays are mounted on the top, and resistors, diodes and n-type mosfets on the bottom of the board. The PCB is actually mounted upside down in the vehicle, so that the bottom copper layer is facing the top of the ROV.

2.2.5 LEDs and camera

A small, one-layer PCB, which houses three white LEDs connected in series, is mounted to a reflective plastic surface. A hole is drilled in the plastic, through which a disassembled USB webcam is pointed in the direction of the LED lights. These LEDs and camera form a stand-alone module, which is shown in Fig. 2.14a. The module is mounted to the front of the internal aluminium frame of the ROV using cable ties, as shown in Fig. 2.14b. A translucent window is placed in the pressure vessel in front of the camera, which is described in section 2.3.2. As shown in Fig. 2.8, the LEDs are activated and powered directly from a GPIO pin of the Arduino, whereas the camera is powered and operated via the USB bus.



(a) LED and camera module



(b) Module mounting

Figure 2.14: The module, which houses the LEDs and the camera, and its mounting in the vehicle.

2.2.6 Pressure sensor

A 30 PSI pressure transducer, depicted in Figure 2.15, is mounted in the aft plug, which is a modified PVC socket plug shown in Fig. 2.16. A hole was punched in the socket plug, into which the pressure sensor was screwed in. The connection was sealed by applying PTFE tape on the thread of the pressure transducer and inserting rubber washers on either side of the socket plug. The entire assembly was then tightened using a nut and a washer placed on the external side of the socket plug. The inlet of the pressure sensor, brass nut and a steel washer, under which a rubber washer is placed, can be seen in Fig. 2.17. The transducer is powered from the Arduino 5 V pin through a hookup wire. It is connected to the Arduino GND and an analogue input pin using hookup wires as well.

Figure 2.15: Overview of the pressure transducer.

2.2.7 Ardunio pinout

Table 2.1 describes all of the Arduino connections used by the ROV. Thrust direction pins are responsible for switching the relays on the motor control board via mosfet transistors, and throttle pins are connected to the ESCs directly and control motor rpm using PWM signals (see section 2.2.4 for details).

Table 2.1: Pinout showing Arduino connections.

Pin no.	Connect net
2	Motor port horizontal - thrust direction
3	Motor port horizontal - throttle
4	Motor starboard horizontal - thrust direction
5	Motor starboard horizontal - throttle
6	Motor port vertical - throttle
7	Motor port vertical - thrust direction
8	Motor starboard vertical - thrust direction
9	Motor starboard vertical - throttle
12	Forward LED on/off
A0	Pressure transducer analogue input
5 V	Pressure transducer VDD

2.3 Pressure vessel

2.3.1 Aft plug

The aft plug is made of a PVC socket plug shown in Fig. 2.16. Two holes were punched in it - one, into which the pressure transducer is screwed in, as described in section 2.2.6, and one, which houses the umbilical connector described in section 2.5.1.

The external side of the aft plug (the wet side of the ROV), with the pressure transducer and umbilical connector mounted, is shown in Fig. 2.17. Its internal side is shown in Fig. 2.18; the body of the pressure

sensor, CAT5 cable, power chord, and 12 motor wires (three signals per each of the four motors) can be seen on this side. The plug can be removed from the main pressure vessel to enable inspection and modification of the interior of the ROV. In order to ease the process and improve the water-tightness, silicone lubricant from Fig. 2.19 is applied on the aft plug before it is slid into the main pressure vessel. Once in place, the aft plug is secured using a water-resistant duct tape.

2.3.2 Forward plug

The forward plug was manufactured from a socket plug identical to the one used for manufacture of the aft plug (shown in Fig. 2.16). The forward plug has to enable the camera to photograph the space in front of the ROV to enable remote operation. To this end, a hole was cut in the socket plug as shown in Fig. 2.20. A window was cut out of a 4 mm thick plexiglass, which was glued onto the outer side of the socket plug using a two-part epoxy glue. The complete forward plug is fixed in the pressure vessel using silicone; this provides both structural strength and makes the connection water-tight. The complete forward plug, mounted into the pressure vessel, can be seen in Fig. 2.22.

2.3.3 Main vessel

The main pressure vessel is made of two PVC pipes, shown in Fig. 2.21. The orange double-sided socket is located on the bow of the ROV. The grey socket pipe was cut to the right length (to provide sufficient displacement), slid into the orange double socket, and secured using silicone. This connection is water-tight, and provides structural rigidity and strength. The assembled pressure vessel is shown in Fig. 2.22.

Apparently, PVC pipes of the same dimensions can have different masses, depending on their manufacturer. The ones used for this ROV were bought at Wickes, a UK DIY supermarket.

Figure 2.16: Socket plug used to manufacture the aft and forward plug.

Figure 2.17: External side of the aft plug with the pressure sensor and umbilical connector mounted.

Figure 2.18: Internal side of the aft plug with the pressure sensor and umbilical connectors mounted. The CAT5 cable is plugged into the USB range extender, and all motor wires are connected to the ESCs and the motor control board.

Figure 2.19: When the aft plug is slid into the main pressure vessel, this silicone lubricant is applied onto it to ease the assembly and improve the seal.

Figure 2.20: Cutting a hole in the socket plug to provide a viewing window for the camera.

Figure 2.21: The pressure vessel consists of two main parts - an orange double-sided socket and a grey socket pipe.

Figure 2.22: Assembled pressure vessel.

2.4 External structure

For the naming convention and location of the components of the external structure, please refer to Fig. 2.2.

2.4.1 Transversals and longitudinals

Side longitudinals, trasnverse clip stiffeners, and landing sleds were made of 2 mm thick, 20 mm wide rolled steel. In order to reduce their mass, M10 holes were drilled along their lengths. The structure is bolted together using M3 bolts. All the metal parts are painted to reduce corrosion.

2.4.2 Pressure vessel mounting

In order to mate the external metal structure with the pressure vessel made of PVC pipes, PVC socket brackets are used. The landing sleds and transversal clip stiffeners are bolted onto the default mounting holes in the brackets, which can be seen on the bottom of Fig. 2.24a. In order to mount the side longitudinals and motors, additional M3 holes were drilled in the socket brackets, as shown in Fig. 2.23 and Fig. 2.24a.

The steel structure was bolted together and onto the socket brackets, but one side of every bracket was left free (one bolt going through the landing sled and transverse clip stiffener was not inserted). Then, the pressure vessel was inserted into the brackets, and the loose end of every bracket was fixed in place around the pressure vessel by bolting it onto the sled and the transverse clip stiffener.

To prevent the steel structure from slipping along the pressure vessel, the bow bracket was placed in the groove in the double socket pipe, as shown in Fig. 2.23. To stop the brackets from rotating around the pressure vessel, both of them were glued to it using a two-part epoxy glue.

2.4.3 Motor mounting

Motors are bolted onto custom brackets made of bent steel (2 by 20 mm section) using M2.5 bolts. The motor manufacturer does not provide a detailed drawing that specifies the location of the mounting holes. Therefore, the location of the holes was marked on the brackets by using the motors themselves as a guide. To ensure that the motors rotate freely, they are separated from their brackets using two washers, as shown in Fig. 2.24b.

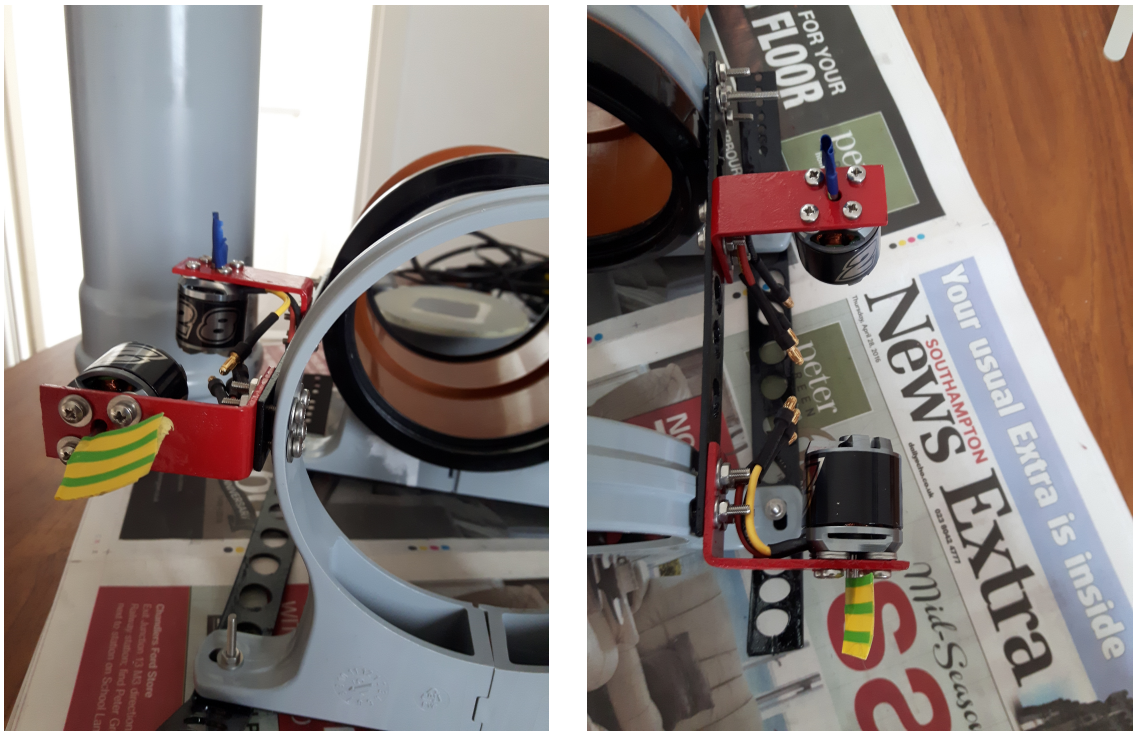
The vertical motor brackets are located in the approximate longitudinal centre of buoyancy (centre of the pressure vessel) to reduce the pitching moment that the vertical motors impart on the vehicle. Similarly, the horizontal motor brackets are mounted at the vertical centre of buoyancy. To reduce the number of the holes in the socket brackets, both sets of motor brackets are bolted onto the side longitudinals, which are then bolted onto the socket brackets, as shown in Fig. 2.24.

2.4.4 Ballast

The ballast consists of two sets of elements: 4 by 30 mm rolled steel flat-bar section cut to the length of the transverse clip stiffeners, and a 186 mm long, 2 by 20 mm steel bar. The latter was a left-over from the steel bought to manufacture the external frame.

Figure 2.23: Using socket brackets to mount the steel structure onto the pressure vessel. One of the brackets is placed in the grooves in the double-socket pipe to stop them from slipping along the ROV.

Five of the 4 by 30 mm sections are mounted close to the centre of gravity in order to compensate for the excess net buoyancy available from the hull. They are placed slightly forward to balance out the moment generated by the heavy wires and cables placed at the aft end. The smaller ballast bar is placed at the forward end of the sled and its exact position was adjusted to achieve an even trim. This was possible thanks to there being a series of holes on either side of the sled, which allow fine-tuning of the ballast position. The arrangement of the ballast is indicated in Fig. 2.2.



(a) Back (b) Top

Figure 2.24: Details of mounting the motors onto the external steel structure. Views from the top and back of the vehicle.

2.5 Umbilical

2.5.1 Connector on the ROV

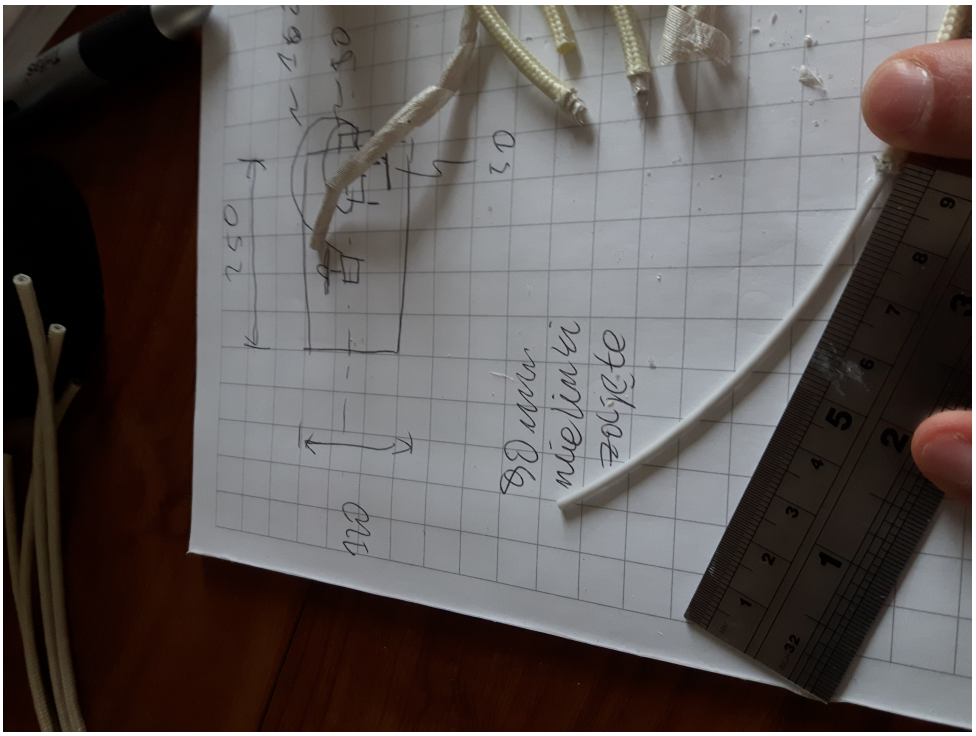
The connector enables the CAT5 cable and the shore power cable, which form the tether described in more detail in section 2.5.2, as well as the wires carrying signals from the ESCs to the motors described in section 2.2.3, to penetrate inside the pressure vessel without compromising water tightness.

The connector is based on a threaded PVC connector, the male part of which is shown in Fig. 2.25. All the mentioned cables and wires were threaded through the male connector, which was then filled with a two-part epoxy resin. All the wires were stripped out of their outermost covers as shown in Fig. 2.26 to allow the resin to penetrate as deep into the wires as possible, to reduce the possibility of water dripping into the pressure vessel between the resin and the wires' cores. For the same reasons, the wires were separated from each other when applying the epoxy, as shown in Fig. 2.27. This ensured that some resin was present between all the wires.

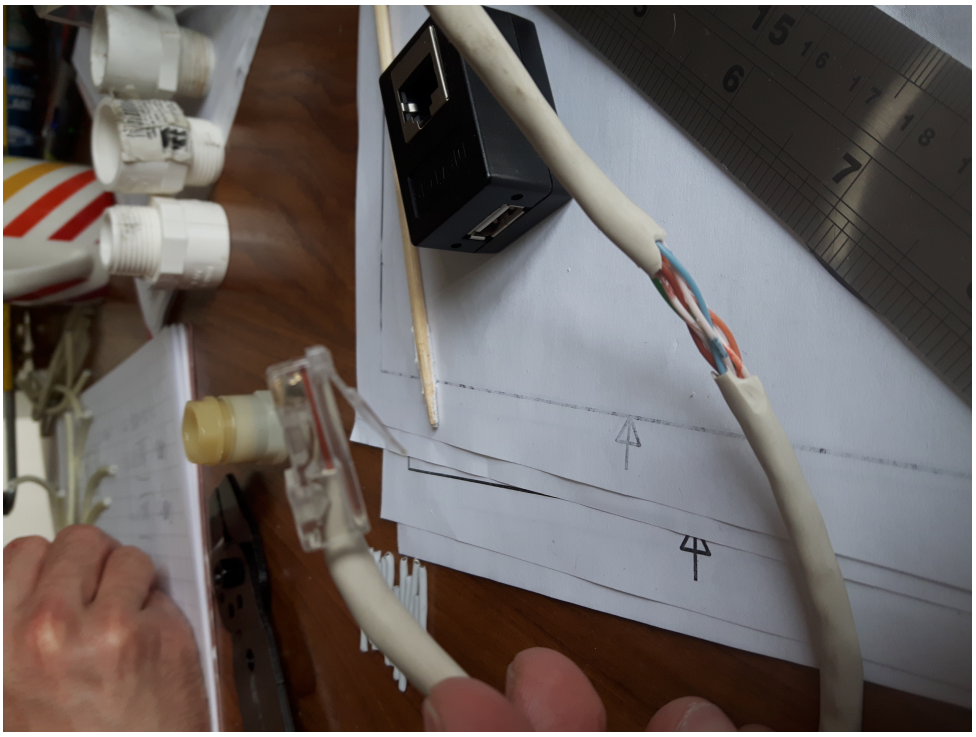
Once complete, the male connector with wires laminated-in was screwed into the hole punched in the aft plug from its outer side, with PTFE tape used to seal the thread. Two rubber washers were placed on the connector, on both the inner and outer sides of the aft plug. A complementary, female part of the PVC connector was then screwed onto the male counterpart to remove the voids between the washers and the PVC parts. Lastly, silicone was applied to the outer side of the aft plug to further seal the connector. The complete aft plug, with the connector screwed in and silicone applied, is shown in Fig. 2.17.

Banana connectors were soldered onto the motor wires. The motor wires were then connected to the motors, and the ESCs and the motor control board (described in section 2.2.3) such that forward and upward thrust would be produced without the relays on the motor control board being activated. The wires were colour-coded to ease re-assembly in this configuration. Finally, after the final assembly, the wires were fixed using white electrical tape, which also isolates the banana connectors from each other. The colour coding of the motor wires and the white electrical tape can be seen in Fig. 2.18.

Figure 2.25: Male part of the PVC connector through which the umbilical tether is threaded and then laminated.



(a) Motor wires



(b) CAT5 cable

Figure 2.26: External cover of the wires is stripped so that less water drips through the umbilical connector once the wires are laminated in place.

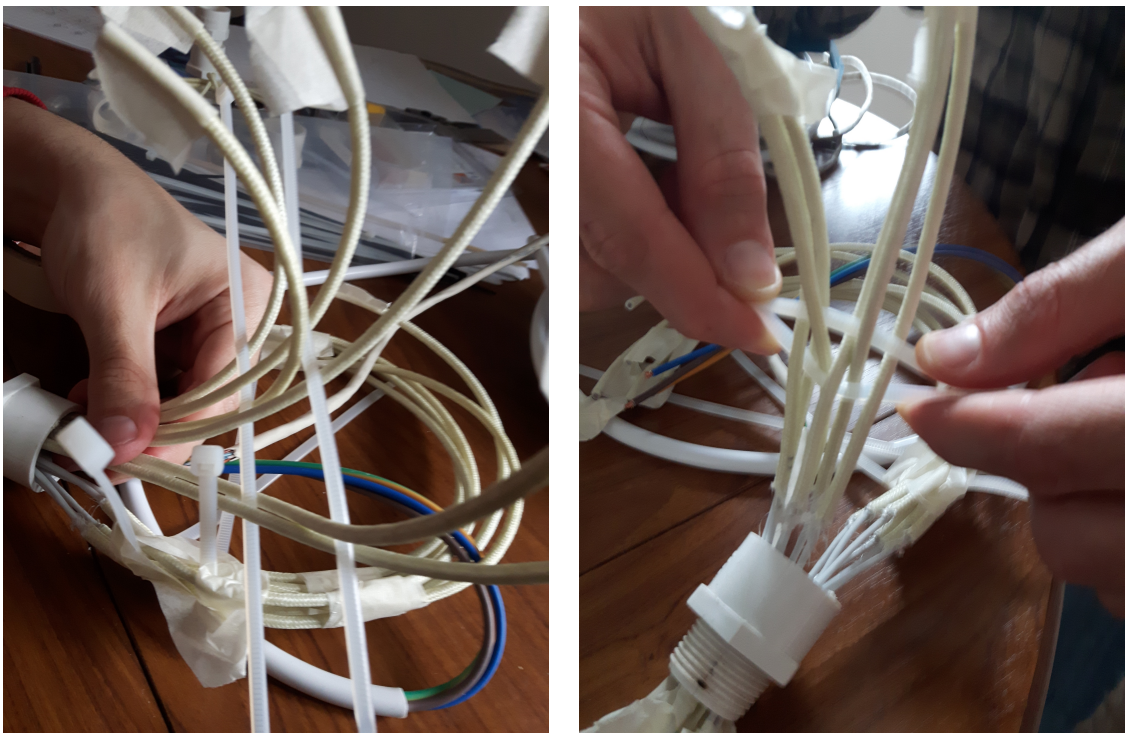


Figure 2.27: Before laminating the wires into the PVC connector, they are separated to make sure some resin surrounds every wire and makes a seal.

2.5.2 Tether

The umbilical tether consists of a CAT5 cable, which extends the USB bus from the shore-based laptop to the ROV, as well as a 16 A-rated, three-chord power cable, which delivers battery power to the vehicle and grounds its structure. These two cables are held together by a 6 mm diameter nylon rope, which is tied around both cables at intervals of approximately one metre. The nylon rope can also be used to lift the vehicle or recover it from the water in case of failure. To this end, the nylon rope is mounted to the external steel structure of the ROV close to its centre of buoyancy. The steel structure provides a strong mounting point, and the attachment location minimises the moments that the tether imparts on the vehicle, which makes it easier to operate. The location of the tether and the way it is attached to the ROV is shown in Fig. 2.28.

During the initial towing tank test, described in section 4.2, it was discovered that the tether was too heavy and thus caused the ROV to sink, even though it was first made neutrally-buoyant on its own. To remedy this, bubble wrap was tied around the tether to increase its buoyancy. The bubble wrap was fixed with water-proof duct tape, as can be seen in Fig. 2.28.

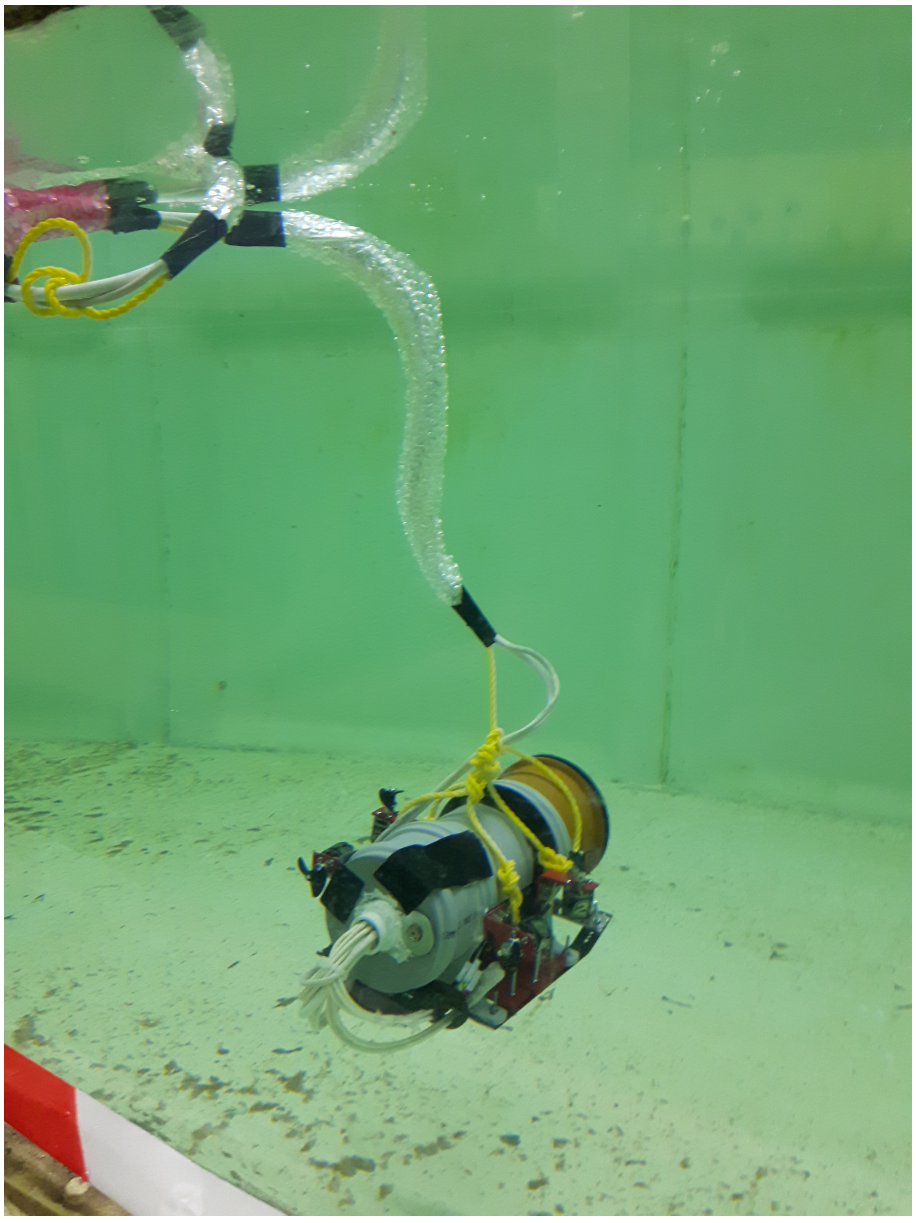


Figure 2.28: Overview of the tether structure together with its attachment to the external steel structure.

Chapter 3: Software design

3.1 Overview

The ROV software can be split into two distinct parts - the code present on the Arduino, which translates the operator commands to physical actions of the vehicle, as well as the shore-based computer part, which collects the user inputs and sends the corresponding commands to the Arduino. The remainder of this chapter covers both parts of the software separately. The computer part of the code is intended as cross-platform because its based on Python 2.7 and its open-source libraries. However, it has been developed and tested exclusively on Ubuntu 12 and 14.

3.2 Micro-controller

3.2.1 Program structure

The Arduino only receives and executes operator commands, which means its functional flow is rather simple, as shown in the flowchart in Fig. 3.1. The `execute every instruction in series block` might involve arming the ESCs, sending readings of all the sensors, switching the forward LEDs on or off, and setting thrust levels and directions of all the motors.

3.2.2 Hardware interfaces

In order to interact with each piece of hardware, such as a sensor or electric motor, or in order to implement simple functionality, a special class `Module` has been implemented. This is holds fields such as `identifier` and `currentValue`, the combination of which is used to control a single aspect of the Arduino operation. For example, a `Module` object called `refreshRate` holds the loop frequency the micro-controller operates at. Such basic objects typically get used directly in the `loop()` method and are treated as simple storage containers. For instance, at the end of each `loop()` call a new delay gets set as `delay(refreshRate.getValue());`, which allows setting the delay value using the same command protocol as used with more complex `Module`s.

Key methods of the `Module` class are the virtual `getValue()`, `setValue(int newValue)`, and `arm()` functions. Their default implementation in the `Module` class allows the user to read or set the value the object holds, for instance the refresh rate of the Arduino, or to initialise the object at the beginning of the execution.

Each of the modules gets created upon initialisation of the Arduino code and pointers to them are collected in arrays `Module* actuators[]` or `Module* sensors[]`, depending on their functionality. This arrangement allows the user to create classes derived from `Module` and overriding the functionality they execute when given or asked for a new value, at the same time allowing the same code in the main program loop to be used to interact with each object. A depiction of the currently used classes is shown in Figure 3.2. This approach is used primarily in the `parseInput()` method which is responsible for communication between the micro-controller and the PC or the `sendSensorReadings()` method

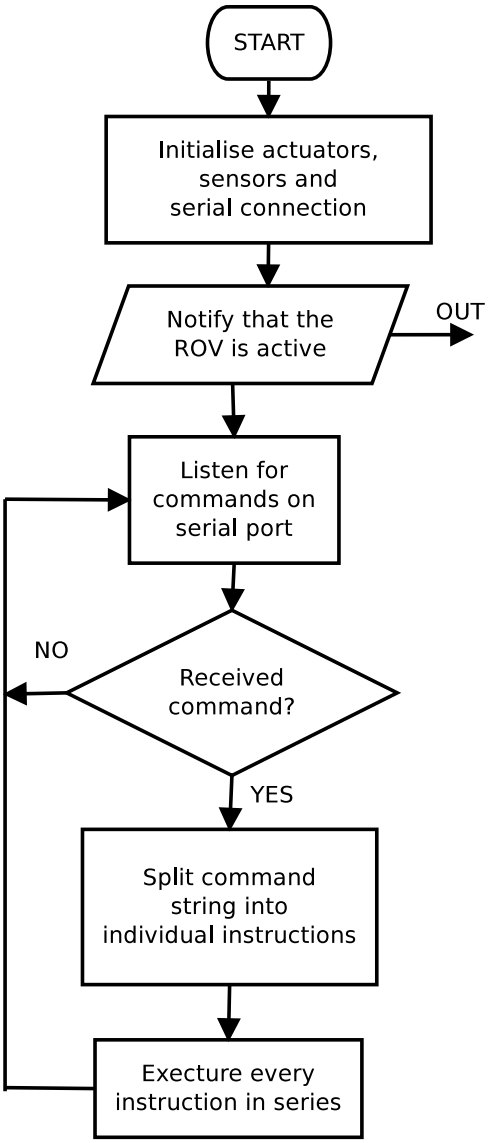


Figure 3.1: Block diagram showing the flow of the Arduino `setup()` and `loop()` functions.

which calls the `getValue()` function on each sensor module and encodes their readings into a string message which gets sent to the PC.

Currently implemented classes derived from `Module` are:

- `BrushlessDCMotor` - accepts negative and positive demand on the motor rps into the `setValue` function, treating negative values as reverse. Uses two assigned output pins in order to control the direction (binary digital output) and speed (PWM) of the motor via the ESC.
- `LEDModule` - the value passed to the `setValue` method is treated as a binary switch telling the module to set the output pin either to HIGH or LOW state, thus controlling the LED.
- `DepthSensor` - implements the `getValue()` method in order to return the current depth. Currently, this is a **placeholder only**.

3.2.3 Communications

The ROV communicates with the computer via a serial port using the USB connection. The process involves encoding messages into one-line strings and dumping them into the buffer for the other side to see. This imposes a need for strict formatting of the message packets, which is described in Table 3.1. Each message consists of a string flag denoting an action or a subsystem, for instance ”forwardLED” or ”sendSensorReadings”, followed by an integer value. An arbitrary number of commands may be sent in any one packet although the length of the entire message is limited by the character buffer size on the Arduino.

During each loop, the micro-controller attempts to read data from the serial port. If any data is found, the `parseInput()` method gets called. This is responsible for checking whether the passed information conforms to the message encoding system. If yes, the string gets unwrapped into a series of flags and corresponding values. Certain flags do not require the value and so an arbitrary integer may be passed, typically a 1 indicating a boolean true.

Each flag is then compared to the names of the registered `Module` objects until a match has been found. The value corresponding to the flag gets then passed using the virtual `setValue(int value)`. This is implemented by each class derived from the baseline `Module` class if a special action is required when a new value gets passed. For instance, the `BrushlessDCMotor` class converts the integer throttle demand into values understood by the ESC and adjusts the PWM signal, flipping the thrust direction relay if necessary. Other modules holding simple integer or boolean values do not re-implement the `setValue` method.

Table 3.1: Description of the special characters used to format string messages sent over the serial port.

Character	Meaning
<	beginning of message to PC
>	beginning of message to Arduino
,	delimiter
;	end of message

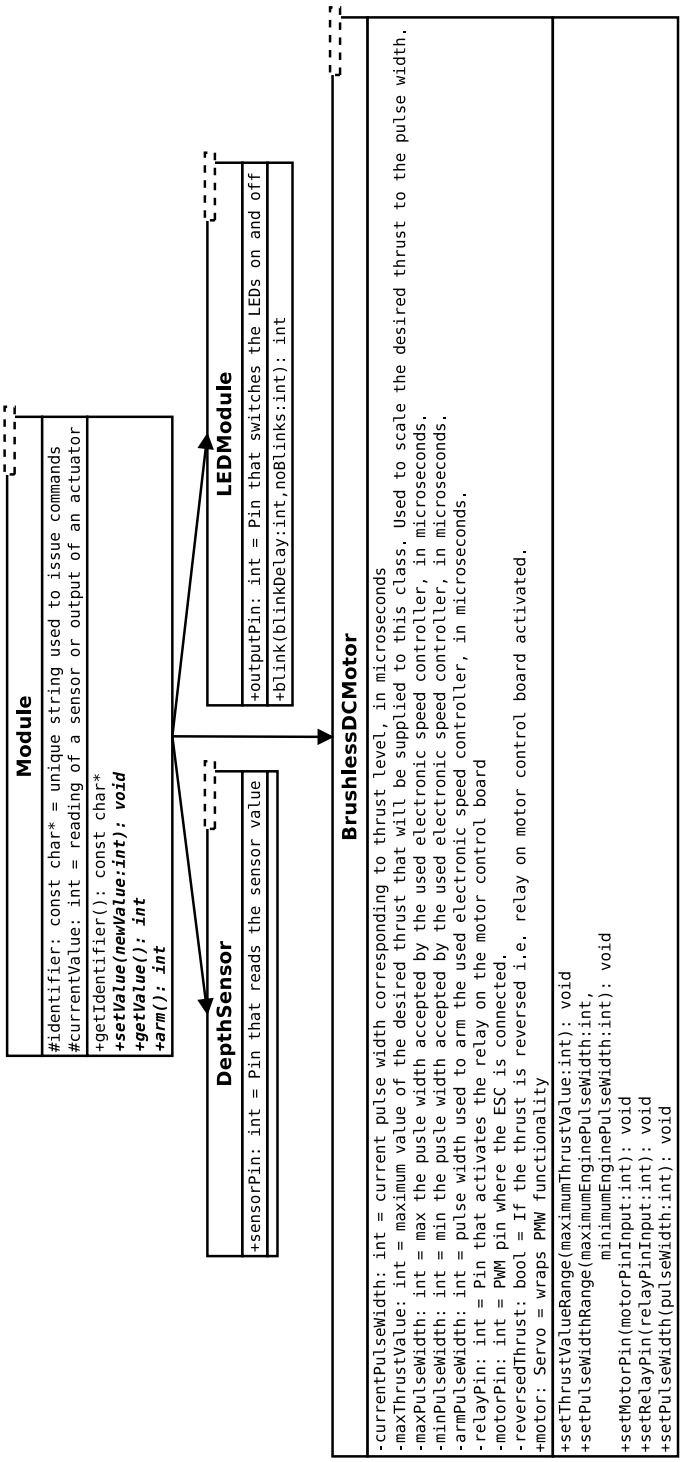


Figure 3.2: Block diagram showing the class hierarchy of the `Module` classes used to control Arduino functionality.

3.3 Computer

3.3.1 Program structure

The key piece of code that runs as an executable is the `ROVgui.py`. It defines a class `rovGuiMainFrame`, which then gets executed inside a wxWidgets application. The class itself is derived from a base class, which defines the basic functionality of the graphical user interface alone (discussed in more detail later) but all of the code responsible for the operation of the ROV itself is contained in `ROVgui.py`. The class contains several timers which execute specific methods at fixed intervals, for instance they ask the mirco-controller to send the sensor readings or pass demands for the electric motors. Each of the key aspects of the code will be discussed in more detail in the following parts of this chapter.

3.3.2 Communications

The protocol used to communicate with the micro-controller via the serial connection utilises the `serial` Python library. This simply opens a port and treats it as a buffer to read and write from. All of the parsed messages adhere to the structure used by the Arduino side, namely begin and end with specific characters and each element is separated by a known delimiter. The following code listing provides a minimum example showing how a known port is opened, written into and read from.

```
import serial
import time

BAUD_RATE = 19200
REFRESH_RATE = 100
PORT = "/dev/ttyACM0"

# open the connection
serialConnection = serial.Serial(PORT, BAUD_RATE, timeout = 2)

testMsgGood = True
try:
    serialConnection.inWaiting()
except:
    testMsgGood = False

if not serialConnection or not serialConnection.readable() or not testMsgGood:
    raise IOError("Connection is FUBAR!")

# keep looping and printing whatever we get from the serial
message = ">sendSensorReadings,1;"
while True:
  serialConnection.write(message)
    line = serialConnection.readline().replace("\n","")
    if line:
        print line
    time.sleep(1./REFRESH_RATE)
```

3.3.3 User input

Interface to the Xbox controller (Figure 3.3) has been created using pyGame 1.9.1, an open-source library for game development. The interface relies on pyGame joystick module capturing all the inputs provided by the user via the controller as events. These are stored in a buffer and then the main GUI function periodically calls the `parseEvents()` method implemented in the `controller` class. This retrieves the most recent set of events, clears the buffer, and updates the values of axes and button positions held inside the `controller` object. From here they are accessed from the main GUI each time a new set of input parameters gets sent to the micro-controller. A minimum working example which captures the events and prings the current axes positions in the terminal is provided

```
import pygame
from pygame.locals import *

pygame.init()
clock = pygame.time.Clock()

joysticks = []
for i in range(0, pygame.joystick.get_count()):
        joysticks.append(pygame.joystick.Joystick(i))
        joysticks[-1].init()
        print "Detected joystick '",joysticks[-1].get_name(),"'"
while 1:
        clock.tick(60)
        for event in pygame.event.get():
                if event.type == QUIT:
                        print "Received event 'Quit', exiting."
                        return
                elif event.type == KEYDOWN and event.key == K_ESCAPE:
                        print "Escape key pressed, exiting."
                        return
                elif event.type == KEYDOWN:
                        print "Keydown,",event.key
                elif event.type == KEYUP:
                        print "Keyup,",event.key
                elif event.type == MOUSEMOTION:
                        print "Mouse movement detected."
                elif event.type == MOUSEBUTTONDOWN:
                        print "Mouse button",event.button,"down at",pygame.mouse.
    get_pos()
                elif event.type == MOUSEBUTTONUP:
                        print "Mouse button",event.button,"up at",pygame.mouse.
    get_pos()

                elif event.type == JOYAXISMOTION:
                        print "Joystick '",joysticks[event.joy].get_name(),"' axis",
    event.axis,"motion."
                        if event.axis == 2:
                                print ' axis 2', event.value
                        elif event.axis == 5:
```

Figure 3.3: USB Xbox controller used to provide control inputs to the ROV.

```
                    print ' axis 5', event.value
            elif event.type == JOYBUTTONDOWN:
                print ("Joystick '",joysticks[event.joy].get_name(),
                    "' button",event.button,"down.")
            elif event.type == JOYBUTTONUP:
                print "Joystick '",joysticks[event.joy].get_name(),"' button
",event.button,"up."
            elif event.type == JOYHATMOTION:
                print "Joystick '",joysticks[event.joy].get_name(),"' hat",
event.hat," moved to",event.value
```

3.3.4 Video acquisition

In order to acquire imagery from the on-board USB camera, the code uses OpenCV 3.0.0-rc1, a cross-platform, open-source digital imaging library. It is written in C++ but provides a straightforward interface for use in Python. In short, capturing a single frame is achieved as

```
import cv2, wx
# create a camera
cameraID = 1
cameraCapture = cv2.VideoCapture(cameraID)
# get a picture and set up colours
ret, picture = cameraCapture.read()
height, width = picture.shape[:2]
picture = cv2.cvtColor(picture, cv2.COLOR_BGR2RGB)
# convert to wxWidgets bitmap for use inside the GUI
bmp = wx.BitmapFromBuffer(width, height, picture)
```

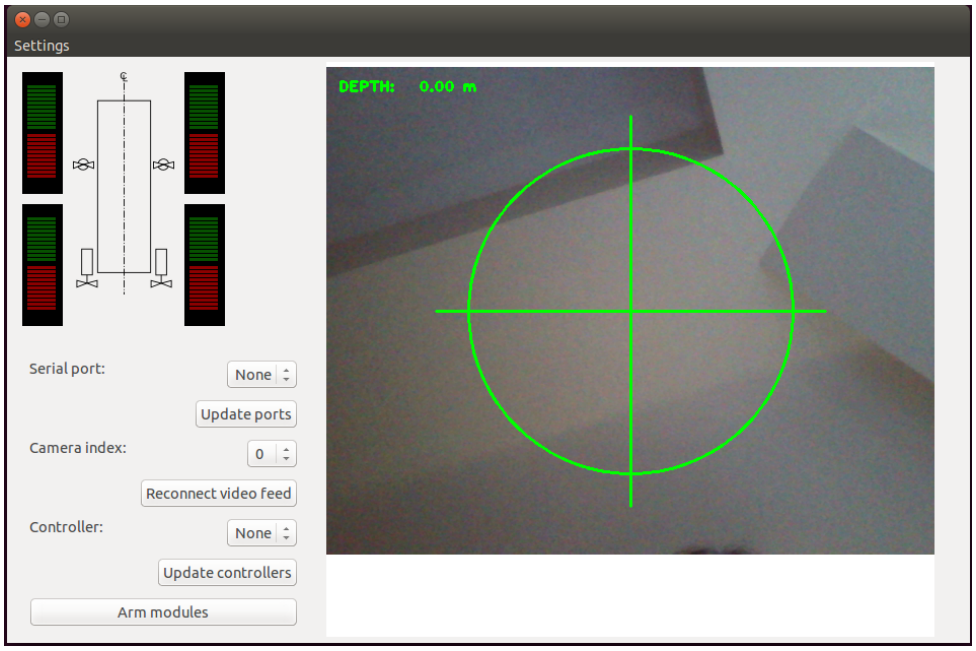


Figure 3.4: Graphical user interface (GUI) used to control the ROV from the laptop.

3.3.5 Graphical interface

Figure 3.4 shows the layout used to provide a graphical interface for controlling the ROV. Live feed from the on-board camera is provided on the right-hand side while the left is devoted to controlling the top-level functionality of the vehicle. The top left-hand corner shows the thruster demand of each motor. Below, controls are provided for the serial port connection to the Arduino, the video feed of the camera, Xbox controller and arming of the electronic speed controllers on board of the ROV. There is also a pop-up Settings window which allows the user to adjust the Arduino loop frequency, as well as frequencies for sending the control inputs, receiving sensor readings, and camera frame rate. The entire GUI runs on wxWidgets 2.8.12.1 and has been designed using wxFormBuilder, both of which may be readily integrated with Python and are open-source.

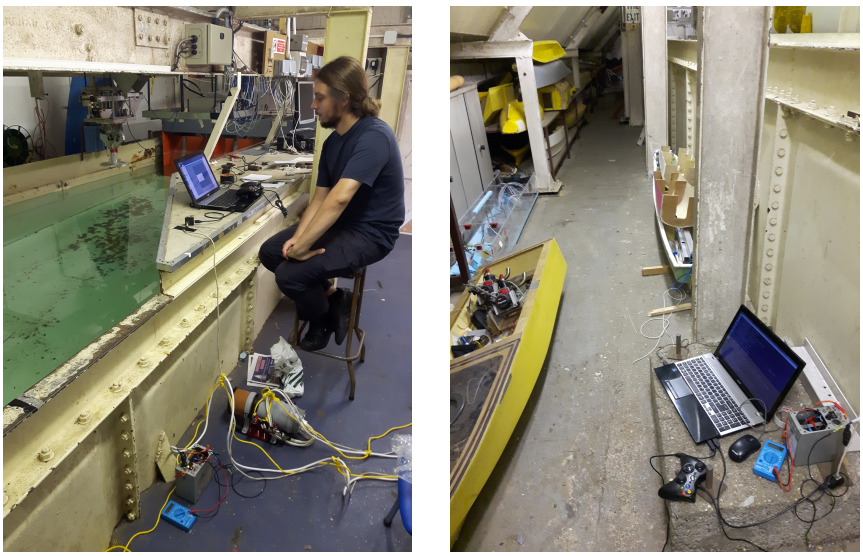
Chapter 4: Testing

4.1 Procedure

The first stage of in-water tests was carried out in the Lamont towing tank of the University of Southampton. Figure 4.1 shows the experimental set-up used in the described tests. The towing tank provided a calm, controlled environment without currents or wind. The water depth was under 2 metres, which was deemed sufficient for the initial test of the pressure vessel. Furthermore, the window in the side of the towing tank allowed easier observation of the robot from the side.

There were three main tests to be carried out during the first series of testing:

1. Testing of the power output of all the motors and making sure they can drive the vehicle in a steady and controlled manner. If not, adjusting the control inputs to compensate for any asymmetries
2. Verifying that the provided visual cues, proposed control interface, and available motor arrangements are suitable to allow a new user to operate the vehicle efficiently and perform simple tasks, such as inspection of underwater objects
3. Submerging the vehicle to progressively larger depths to test the water- tightness of the outer hull



(a) Dry and manoeuvring tests (b) For observation through side window

Figure 4.1: View of the testing set-up in the Lamont towing tank.

In order to comply with the health & safety regulations, a strictly defined procedure of using the ROV was put in placed and adhered to throughout the testing:

1. Plugging the ROV into a laptop via an Ethernet cable using a USB converter

2. Powering-up the motors using a 12 V DC power supply placed on-shore
3. Testing the camera system and ensuring each of the motors is operational
4. Placing the vehicle in the water using an insulated rod and an attachment point provided by the main tether
5. Performing the required tests and tasks
6. Surfacing the vehicle close to the operator and unplugging the power supply
7. Recovering the vehicle by hand and finishing the test

4.2 Manoeuvring tests

Before the towing tank tests it had been recognised that despite all electric motors and electronic speed controllers being identical, the same demand signals from the Arduino board made each of the motors rotate at a slightly different speed. This had been identified by setting them to the highest RPM in pairs and listening to the sound they produce while adjusting the tuning constants in order to eliminate the beat frequency (much like tuning a guitar). One of the vertical thrusters proved to behave particularly differently than the others. Its tuning parameters were thus adjusted in the Arduino script before the tests, in order to achieve approximately identical RPM as the other vertical motor with the same thrust demand.

Once the vehicle was placed in water its balance, initially tested in a small water container, was verified. The predicted centre of gravity location agreed well with the expectations, yielding neutral trim. At the outset of the tests the vehicle was also neutrally buoyant and kept a steady altitude just below the water surface. Placing the tether attachment at the centre of gravity negated any moments and prevented the ROV from adopting excess trim angles. At first, however, the tether was not as buoyant as initially hoped and was dragging the robot down to the bottom. An impromptu solution was employed whereby sheets of bubble wrap were attached at even intervals to the tether, making it neutrally buoyant.

Then, several basic manoeuvres were attempted: moving in a straight line horizontally and in the vertical plane, orienting the ROV in a particular direction, and turning while under way. It has been found that the used motors and propellers provide more than enough thrust to make the vehicle agile and faster than expected. The control interface also fulfilled its purpose well and allowed the operator to convey commands to the ROV with ease. An example of a successful descent manoeuvre is seen in Figure 4.2. Due to the linear mapping of the joystick displacement to thrust demand it was rather difficult for the operator to execute very precise control over the vehicle, however. Such precise control requires very low levels of thrust, which can only be obtained with minimal joystick displacements in the current implementation.

It was also found that the tether, in particular the power cable, was so stiff and twisted (it had spent most of its lifetime on a reel prior to deployment) that it made it difficult for the ROV to move freely. This is because the twist in the tether exerted a constant moment on the robot, which in turn rotated the ROV around the tether. This meant that adopting certain headings was impossible without constant actuation of the thrusters, for example.

It had also been expected that the load on the motors would increase significantly when they operated in water instead of air, which was indeed observed. At first, the battery system had been fitted with 1 A fuse, which was sufficient to operate four motors at their maximum thrusts in air. In water, however, the 1 A fuse allowed the user to only operate the motors at very low RPMs and made it difficult not to exceed the current limit, thus requiring a fuse replacement. In practice, only a small deviation in the joystick position would lead to burning the fuse. The internal electronics were capable of sustaining higher currents and the 1 A limit had been chosen somewhat arbitrarily for increased safety. The safety concern was relaxed during the test and a 4 A fitted instead. This allowed the user to operate the ROV without any restraint while still having a safeguard built into the system.

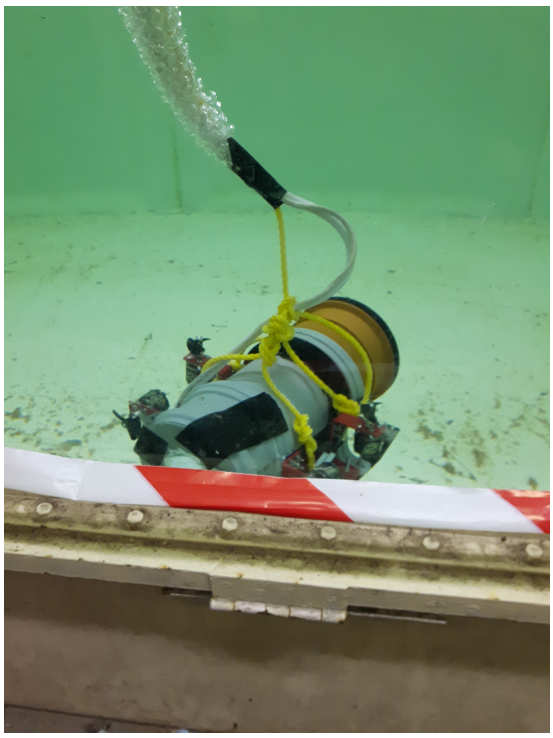


Figure 4.2: ROV floating near the bottom of the towing tank.

4.3 Blind control tests

Outside of control environment, the operator would not be able to see the ROV from their workstation and instead they would have to rely solely on the visual and other cues fed back to them from the on-board sensors. The next stage of tests focused on simulating this scenario by placing the operator so that the ROV was outside their field of view. They were then tasked with performing the same tasks as before (simple manoeuvres - straight line, turning circle, ascent/descent, keeping a constant heading) when relying only on the video feed from the USB camera fitted to the ROV.

It has been found that latency present in the visual system was detrimental to the user's ability to manoeuvre the vehicle easily. They were able to establish the general direction the ROV was facing in relation to the main features of the nearby environment. Doing so required them to put in considerable effort and was further made difficult by the unpredictable forces and moments exerted by the tether. The

exact reason for the existence of latency in the video feed is unclear - it could be due to the camera itself, due to the same CAT5 cable being used to transfer video and serial commands to the Arduino, due to some issues with the OpenCV library and its implementation in the current code, or due to limitations of the current graphical user interface design.

Furthermore, it has been found that the 640x480 resolution of the camera is not sufficient to provide detailed view of the ROV surroundings when the vehicle is put in water with particulates in it. This made it difficult to judge the distance of the vehicle from the obstacles ahead by using visual cues only. Nonetheless, enough detail could be captured to allow key environmental features to be distinguished, as seen in Fig. 4.3.

Lastly, the camera has a considerable zoom. This makes approaching a target at a close distance dfficult because it seems to be much larger and thus closer than it in fact is. At present, none of the ROV components are visible in the camera field of view. Changing this, for example by using a camera with a wider field of view and moving it further away from the pressure vessel surface, could help the operator judge proximity to obstacles. This is because both the visible ROV components and the obstacles would be distorted by the camera in the same way, and the relative distance could be assessed.

(a) ROV on the free surface

(b) Wavemakers at the end of the tank

(c) Operators seen through the side window of the tank

(d) The far end of the tank

Figure 4.3: Video frames captured by the on-board camera during towing tank tests.

4.4 Water-tightness tests

Before the tests in the towing tank, the ROV had been submerged in a small water container (i.e. a bath tub) for the period of several hours to verify the efficacy of seals and connections. No water ingress had been observed even for overnight immersion tests at the depth of approximately 30 cm.

The towing tank tests lasted approximately 3.5-4 hours, during which the ROV was removed from the water several times. After the initial several seconds after the first immersion no visible water bubbles were seen escaping from the hull, indicating the hull remained watertight. Throughout the tests, however, the vehicle was observed to gradually lose its neutral buoyancy. Towards the end of the trials, the ROV would linger close to the tank bottom, and making it ascend using the thrusters alone became difficult. After the final test run, the aft plug was removed and a small amount of water was seen to leak from inside the pressure vessel. The exact volume was not verified due to lack of experience of the operators. It was sufficient to adversely affect the neutral buoyancy of the vehicle but did not reach the electronics, which suggests the leak was small. It is believed the water ingress took place through the aft seal between the hull and removable aft plug as it was at this location that small air bubbles were seen escaping at the beginning of each series of tests. The aft cable port could also be the culprit, although this is seen as less likely.

Chapter 5: Conclusions

5.1 Summary of the design

It has been demonstrated that even within a limited budged of less than £300 and access only to basic tools, a fully-functional remotely operated underwater vehicle can be designed, manufactured and tested. The overall design concept using only four degrees of freedom control and employing a compact vehicle shape has proven to provide sufficient performance that would be expected of a low-cost sensor platform for use in shallow waters. Several specific aspects of the design should be refined but a precious few have also proven to be quite successful. Each of these groups will be discussed in more detail in the subsequent parts of this chapter.

5.2 Successful design features

Several of the chosen engineering solutions have been found to work particularly well during the tests. These were:

- Using the on-board Arduino micro-controller to allow control of the ROV from a standard laptop has proven to be a very flexible and low-cost solution
- The X-Box controller used to collect input data has been found to provide a robust, easy-to-use interface
- Overall general arrangement of the vehicle has been successful and integrating all of the electronics into a removable internal frame allowed ease of access during assembly and disassembly
- Placing the motors so that their thrust passes through the centre of gravity has allowed problem-free operation in the 4 DoF framework
- Use of an external power source allowed the vehicle to be compact and lightweight, with the complete system fitting into an above-average sized rucksack and being easily transportable by hand.

5.3 Recommendations for future designs

Despite the general success of the project, several aspects of the design could, and a few certainly should, be improved if a next iteration or a brand new vehicle were to be designed. These summarise the lessons learned through the development and testing of the ROV:

- Making the aft plug removable without the use of bolts or other permanent connectors made it very easy to assemble the vehicle on-shore but led to the vehicle not being completely watertight. This is a major issue which could be addressed by i) using a threaded aft plug with a better seal ii) use of a slot-in plug secured with nuts and fitted with additional o-rings iii) manufacturing the pressure

hull from a less compressible material (steel or aluminium) and desigining bespoke watertight connections

- Removing pre-twist from the tether and making it float would make manoeuvring the vehicle far easier
- Fixing the video latency issue would be necessary for operation in a real environment. This may require using a dedicated VGA cable, a different camera, or better implementation of the current software
- At present the motors are placed in water without waterproofing which would not be acceptable in the long term in real environments such as seas or rivers. A watertight enclosure would be necessary, most likely utilising a magnetic coupling, to prevent corrosion of the motors.
- Being able to redirect a time trace of control inputs and sensor readings to a file would make analysing the performance of the vehicle easier
- In the present design only the internal structure was grounded, extending this to include all steel elements would add an important safety feature
- Using non-linear mapping or having two sensitivity settings for the controller should make it easier to perform precise manoeuvres
- Adding a sensor measuring power output of the battery would be an important indicator of the efficiency of the vehicle and would allow planning missions more accurately
- At present, each of the power wires in the connector in the aft plug (section 2.5.1) ends with a banana connector. Grouping the cables by the corresponding motors and using a single connector for each motor would streamline the ROV assembly process significantly.
- At present, the external steel frame is difficult to disassemble or even detach from the central hull due to the way the bolts are situated. At the time it was not considered crucial to allow this but having such a possibility would allow modifications to the design to be introduced more easily.
- The electrical system has been significantly over-designed in most places in terms of current ratings. However, there are several segments with much lower safety factors, which effectively dictate the maximum safe power output from the battery. A careful choice of suitable current ratings for each segment of the design would allow a much more efficient overall system design.
- Currently used off-the-shelf electronic speed controllers are straightforward to utilise but require a large amount of wiring to operate, which takes up a considerable amount of internal volume of the ROV. Using PCB-mounted variants or even controlling the motors directly from the Arduino by pulsing current using transistors could allow for a much more compact design. Ultimately, a small PCB for a single motor or a pair of motors could be devised. The PCBs could be connected using stack-through connectors, thus eliminating most of the wiring used in the current design. This would make the system more modular, and lower-mass and thus also smaller and more agile.

- Having the tether attach to the inspection hatch (aft plug) poses certain difficulties during disassembly as often multiple connectors have to be disconnected and reconnected just to allow a simple task to be accomplished. Moving the cable port to a stationary part of the hull could simplify these operations.
- Adding controller calibration constants to the GUI settings window would make it more easily adjustable by the user.