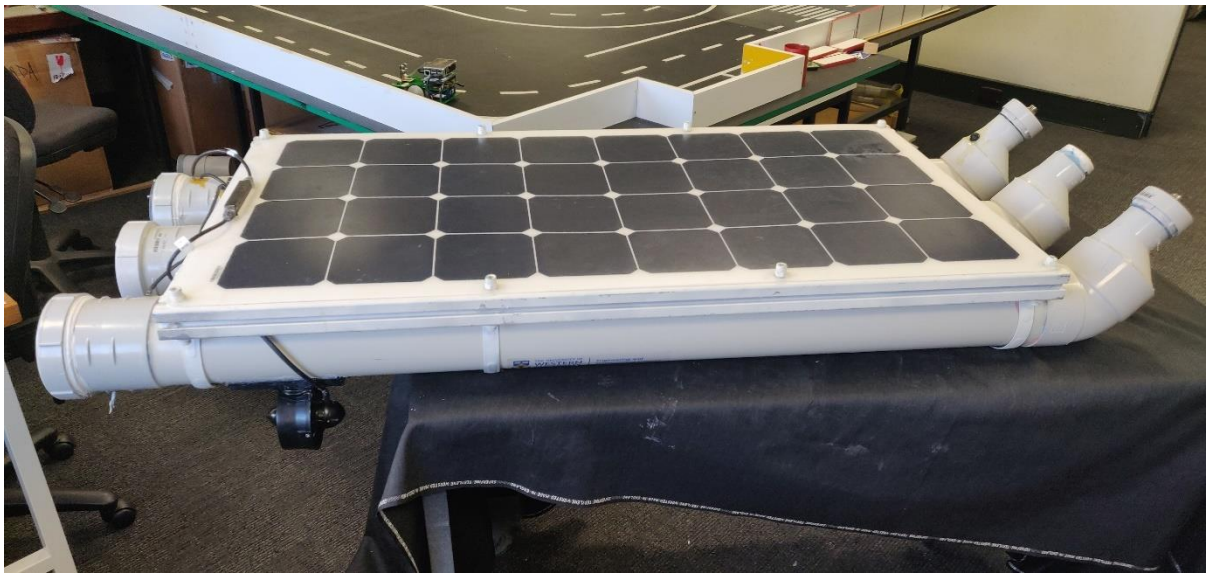
Using a Solar Powered Autonomous Raft to Collect Oceanographic Data

Written by: Wesley Coleman – 21485619

Supervisor: Professor Thomas Bräunl

Word Count: 7489



Previous UWA MPE students have created a Solar Powered Autonomous Raft (SPAR) that can be controlled remotely over a 3G connection. The goal of this project was to explore one of the potential uses of such a platform: the collection of oceanographic data. To achieve this, several sensors were integrated into the SPAR to record ocean temperature and salinity, and to capture still images. The possibility of collecting additional types of data (such as ocean depth and wave height) was considered during the early stages of the project, however upon investigation these plans were discarded due to a combination of cost, impracticality, and time constraints. Additionally, the layout of the existing hardware on board the SPAR was improved to allow room for the new sensors, new motor mounts were designed and installed, and the software controlling the network interface was updated to correspond with an update of the SPAR's server architecture that was conducted as a simultaneous project.

Contents

1. Introduction 3
   1.1 Background 3
   1.2 Initial Project State 4
   1.3 Project Scope 5
2 Sensor Selection 5
   2.1 Temperature 5
   2.2 Conductivity/Salinity 6
   2.3 Ocean Depth 6
   2.4 Wave Height 7
   2.5 Camera Imagery 7
3 Hardware Changes 10
   3.1 Addition of Sensors 10
   3.2 Motor Mount Replacement 13
   3.3 Layout/Wiring Improvements 16
4 Software Changes 21
   4.1 Sensor Manager Class 21
   4.2 Modem Interface Update 22
   4.3 Mavlink Interface Updates 23
5 Testing 24
6 Issues Faced 24
   6.1 Modem Reliability 24
   6.2 Hardware Issues 25
   6.3 Miscellaneous Issues 25
7 Conclusion 26
   7.1 Recommendations for Future Work 26
8 Acknowledgements 27
9 References 27
Appendix A: Bill of Materials 28
Appendix B: Software Resources 28

1. Introduction

The Solar Powered Autonomous Raft project began in 2017 at the University of Western Australia with the goal of producing an autonomous vehicle with the potential to circumnavigate the globe. As a realistic goal, the team aims to complete a circumnavigation of Rottnest Island. Additionally, the team intends to highlight the potential such a platform would have for data-collection by fitting the SPAR with a small array of sensors. If successful, SPAR will demonstrate the viability of low-cost Unmanned Surface Vessels for use in small-scale oceanographic research projects.

1.1 Background

While several groups have built Unmanned Surface Vehicles previously, the platforms have either not been used for data-collection purposes or have been extremely expensive and therefore not suitable for smaller-scale surveys.

1.1.1 SeaCharger

The SeaCharger is a solar powered autonomous boat built primarily by hobbyist Damon McMillan between 2013 and 2016 [1]. Its successful voyage from California to Hawaii was the primary inspiration for the SPAR project, and its failed subsequent voyage to New Zealand heavily informed the decision to use differential drive instead of rudder-based steering for the SPAR project [2]. While the SeaCharger served as a great source of inspiration, it was not used to gather any meaningful oceanographic data, instead focusing on the non-trivial task of crossing the Pacific Ocean [1].

1.1.2 Microtransat

The Microtransat Challenge is an ongoing transatlantic race for autonomous boats, intended to "stimulate the development of autonomous boats through friendly competition" [3]. On the 26th of August 2018 the SB Met became the first vessel to successfully complete the Microtransat Challenge, competing in the Sailing Division [3]; its success shows the potential of solar powered autonomous vessels to carry out significant voyages. Unfortunately for the SPAR project, the majority of Microtransat vessels (including the SB Met) have been autonomous sailing boats, while the SPAR is rotor-powered. As such, the information that could be gleaned from those designs was limited.

1.1.3 Wave Glider

The Wave Glider is an autonomous wave- and solar-powered USV produced by Liquid Robotics [4]. Intended for use in the defence, environmental assessment and offshore energy industries, it provides a robust and versatile platform for a wide array of potential sensors and applications. Wave gliders have been used to study sediment transport in shallow sandbanks [5], record sea surface height [6], and monitor winds and currents inside an active typhoon [7], demonstrating the usefulness of an autonomous USV platform. The Wave Glider has one crucial drawback: its price. Costing approximately $300,000 USD per unit [8], it is not suitable for smaller-scale projects, so a cheaper autonomous sensor platform would likely be of extreme interest to such projects.

1.2 Initial Project State

A functional prototype vessel has been constructed and successfully completed short voyages in a swimming pool [2] and on the Swan River [9]. The boat consists of a solar panel, two thruster motors, and a series of PVC tubes that provide buoyancy and house the electronic components. These electronic components include four lead-acid batteries, a solar controller, a GPS module, and an ArduCopter 2.8 microcontroller [2]. The SPAR is driven by the ArduPilot autopilot software running on the ArduCopter MCU. Using the accompanying APM Planner software, it is possible to upload GPS waypoints to the SPAR, which the SPAR will then navigate to. The SPAR initially used a short-range radio module for wireless communication, but this was not a viable control method for a fully autonomous craft. To fix this, a Raspberry Pi Zero W and a FourFaith 2414 3G modem were integrated into the SPAR's design [10]. The Pi acts as a bridge between the ArduCopter and the 3G modem and allows the boat to be controlled from a web server. An overview of the SPAR's architecture is shown in Figure 1.

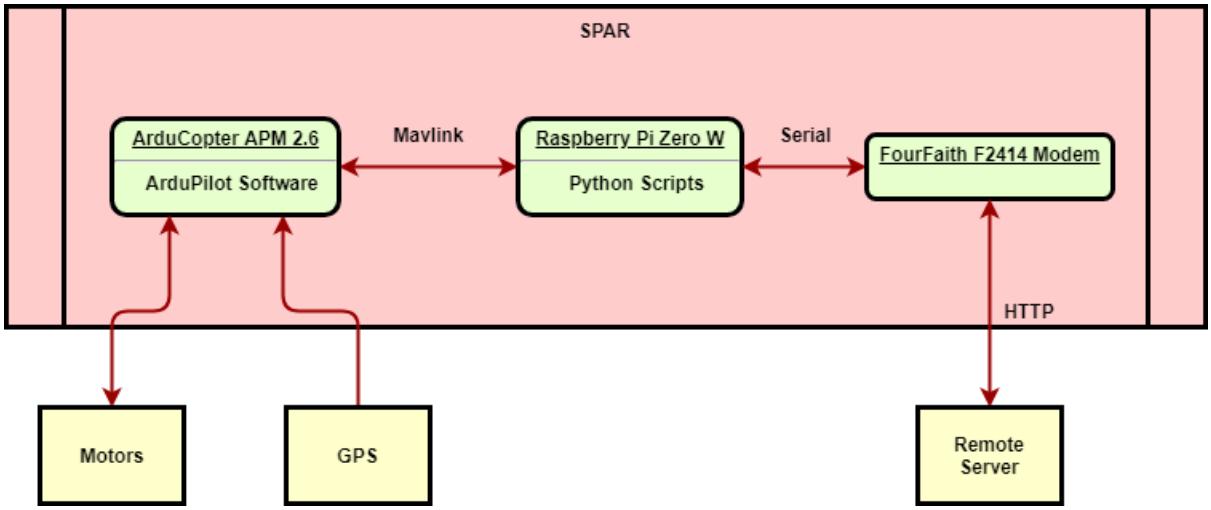


Figure 1: Initial architecture of the SPAR

In addition to the prior work already carried out on the SPAR, an additional project to overhaul the remote server architecture will be carried out simultaneously by Morgan Trench.

1.3 Project Scope

The scope of this project can be broken up into three components. The first and most significant component is the addition of a sensor array capable of collecting oceanographic data to the SPAR. This array will need to be integrated with the existing hardware and software of the SPAR. These sensors will be controlled by Python scripts running on the Raspberry Pi Zero W, and the data they collect will be uploaded to the remote server alongside the SPAR's positional data. An overview of the updated architecture of the SPAR is shown in Figure 2.

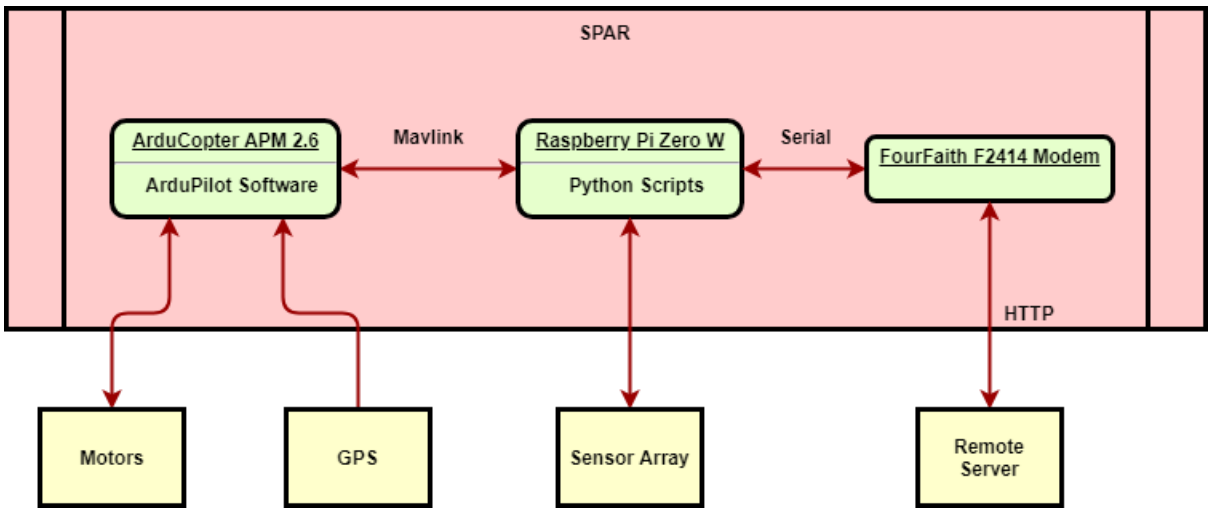


Figure 2: Updated architecture of the SPAR

The second component of this project will be to improve the existing hardware design of the SPAR. The layout of the components within the raft will need to be updated to provide room for the additional sensor hardware that will be added, and the circuit topography of the SPAR will be updated to accommodate the new hardware additions.

The final component of the project will be to update the software running on the Raspberry Pi. Code to manage the new sensor array will be added to the existing software architecture, and the control flow of the software will be altered accordingly. Additionally, the software responsible for the interface with the remote server will be updated to reflect the changes that will be made to the remote server architecture.

2 Sensor Selection

The first step of designing the sensor array was deciding which sensors would be used. A variety of types of sensors were considered for inclusion, but only some of these proved to be feasible.

2.1 Temperature

The first type of sensor considered was a temperature sensor to record the temperature of the water the SPAR travels through. Ocean life is very sensitive to temperature changes [11], so being able to monitor ocean temperatures from an autonomous platform like the SPAR would be highly desirable. Fortunately, temperature sensors are widely used in microcontroller applications, so it is easy to find sensors suitable for this application. The sensor chosen was a waterproof DS18B20 temperature probe. These probes are widely used, very cheap, and are produced by several manufacturers. They communicate using the 1-Wire interface, which is supported by the Pi, and there are Python libraries available for it, making it simple to integrate.

2.2 Conductivity/Salinity

Ocean salinity is a metric that is extremely important to oceanographers, as it is a key driver of ocean circulation [12], so adding a salinity sensor to the SPAR was a high priority. The most practical method of tracking salinity is to measure the conductivity of the seawater, since the conductivity of water is directly proportional to its salt content. Conductivity probes work by attempting to pass current through two electrodes that are a fixed distance apart, and measuring the voltage drop across them. The voltage drop can then be used to find the conductivity (κ) using the following equation [13].

$$\kappa = \frac{Q}{10 \cdot R \cdot |V_{in}|} \times V_{out}$$

In this equation, the vessel constant $Q = L/A$, where L refers the distance separating the two electrodes and A refers to their area [13].

The sensor chosen was the Analog EC Meter, produced by DFRobot. The primary reason this sensor was chosen was cost; measuring conductivity is somewhat niche, so most commercially available sensors are intended for laboratory use and can cost upwards of $1000. Additionally, this sensor shipped with a DS18B20 temperature probe, breakout boards for both the conductivity and temperature sensors, and sample solutions with known conductivities: 1413µs/cm and 12.88ms/cm. This sensor outputs a voltage which can be used in the above formula to determine the conductivity. However, the vessel constant of each specific sensor will vary based on manufacturing irregularities, so the sensor will need to be calibrated. By recording the output of the conductivity sensor for the included standard solutions and linearly interpolating, the relationship between conductivity and output voltage for the specific sensor used was found to be $\kappa = 8.065 \times V_{25°C} - 0.2013$

The conductivity of the water also depends on the temperature, so the voltage measured by the conductivity sensor will need to be corrected before it can be used. This can be done using the following formula [13].

$$V_{25°C} = \frac{V_{measured}}{1 + 0.0185 * (T_{measured} - 25)}$$

This is an analogue sensor, but the Pi only accepts digital inputs, so an ADC is required. The ADC chosen was an Adafruit ADS1115. This sensor accepts up to 5V analogue inputs, and outputs over an I2C bus, which is a protocol supported by the Pi. This, along with the ADS1115's low cost, make it an ideal choice.

2.3 Ocean Depth

Using measurements of the ocean depth to map the sea floor is of key interest to oceanographers. This is especially relevant close to shore, where large autonomous vessels such as the Wave Glider cannot travel. The basic principal would be to use a sonar or laser distance sensor to map the distance to the sea floor over a given area. However, for the results to be meaningful an oceanographer needs to know the raft's location extremely precisely. GPS signals can have up to 0.5m of error due to variations in the transmission speeds of the signals themselves [14]; enough to completely invalidate any results. This problem can be solved using differential GPS; a technique where a stationary base station with a known position is used to correct GPS errors. Unfortunately, differential GPS is very expensive to set up, and as such would not be viable for this project. As a result, ocean depth recording was not implemented into the SPAR.

2.4 Wave Height

By tracking the movement of the SPAR with a six-axis IMU – a measurement unit consisting of an accelerometer and a gyroscope – it would be possible to record the vertical motion of the SPAR and hence derive wave heights. This can be done by using the gyroscope measurements to convert the accelerometer readings from the IMU's body frame to the navigation frame, which is stationary with respect to the Earth [15]. The vertical acceleration can then be double integrated to track the SPAR's vertical displacement, which could be used to calculate wave height and frequency.

The ArduCopter autopilot module includes a TDK MPU-6000 IMU, which could theoretically be used for this application. However, the IMU used needs to be extremely accurate for the results to be meaningful. To gauge the accuracy of the integrated IMU, it was compared to two known-good control IMUs: an LP-Research LPMS-B2 and an Xsens MTi. All three IMUs were placed on a rigid plank, which was then subjected to a series of motions. The accelerometer readings of all three IMUs is shown in Figure 3, and the gyroscope readings of all three IMUs is shown in Figure 4.

Looking first at the acceleration results, the two control IMUs show extremely similar results for all three axes, but the integrated IMU results are extremely noisy. The integrated IMU's z-axis readings are particularly bad, with none of the features visible in the z-axis of the two control IMUs being visible, and a significant amount of drift present. The gyroscope readings show a similar story: the control IMUs provide very similar results while the integrated IMU results are mostly noise. Based on the poor performance of the integrated IMU in this test, using it to attempt to record wave height would not be viable. Neither of the control IMUs could be integrated into the SPAR due to budget constraints, so wave height monitoring was ultimately not implemented in this project.

2.5 Camera Imagery

The final type of sensor considered was a camera to take images during the SPAR's voyages. Since the SPAR already features a Raspberry Pi Zero, the obvious choice for a camera was a Raspberry Pi Camera module. These cameras are relatively cheap and easy to integrate with a Pi. One caveat with the Raspberry Pi camera is that the camera connector on the Pi Zero is different than the connector featured on a normal Pi, so the CSi cable that comes standard with the Pi camera would not be suitable. Fortunately, the Pi Zero obtained in this project came with a suitable cable, which was used.

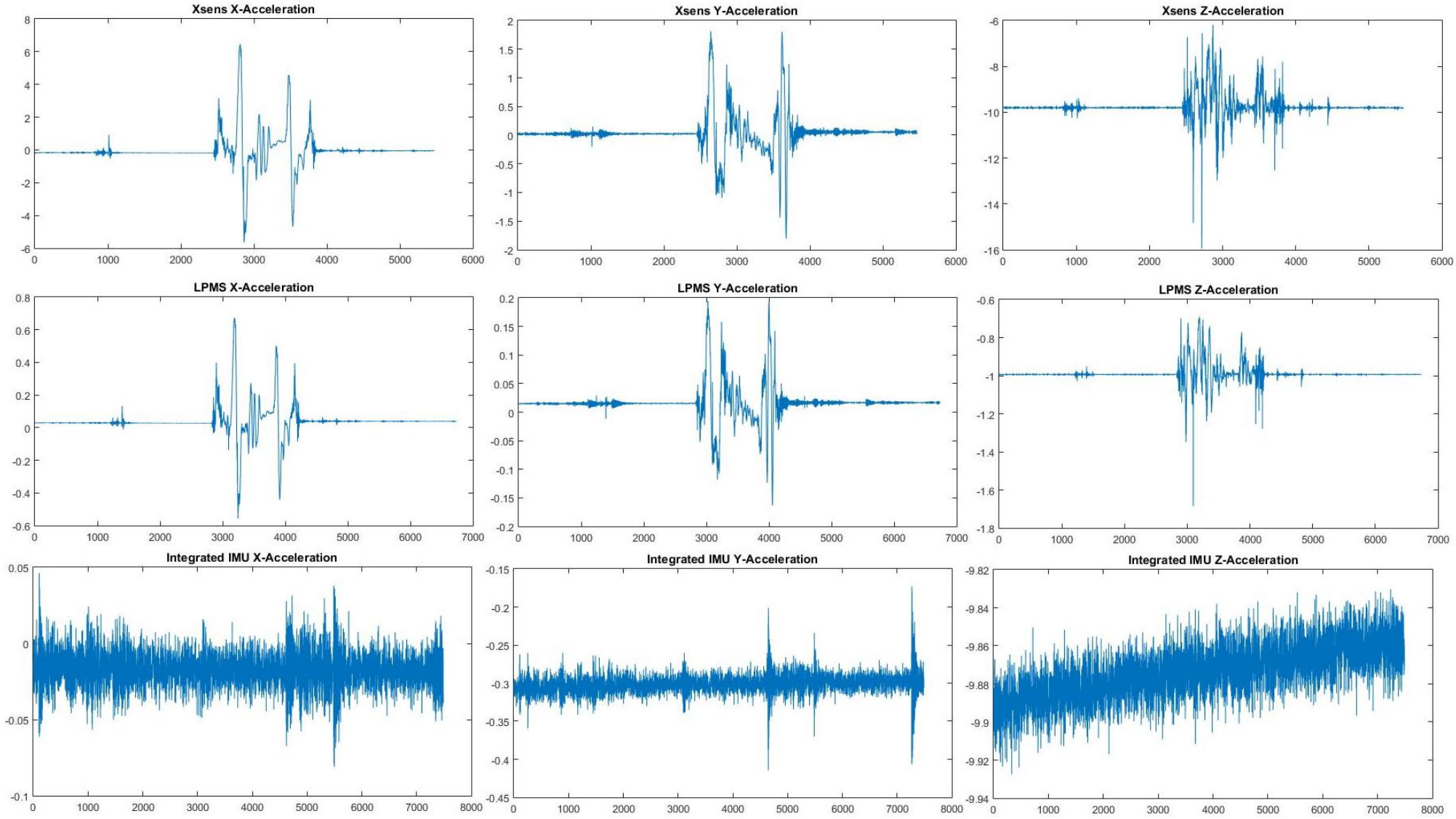


Figure 3: IMU test acceleration results

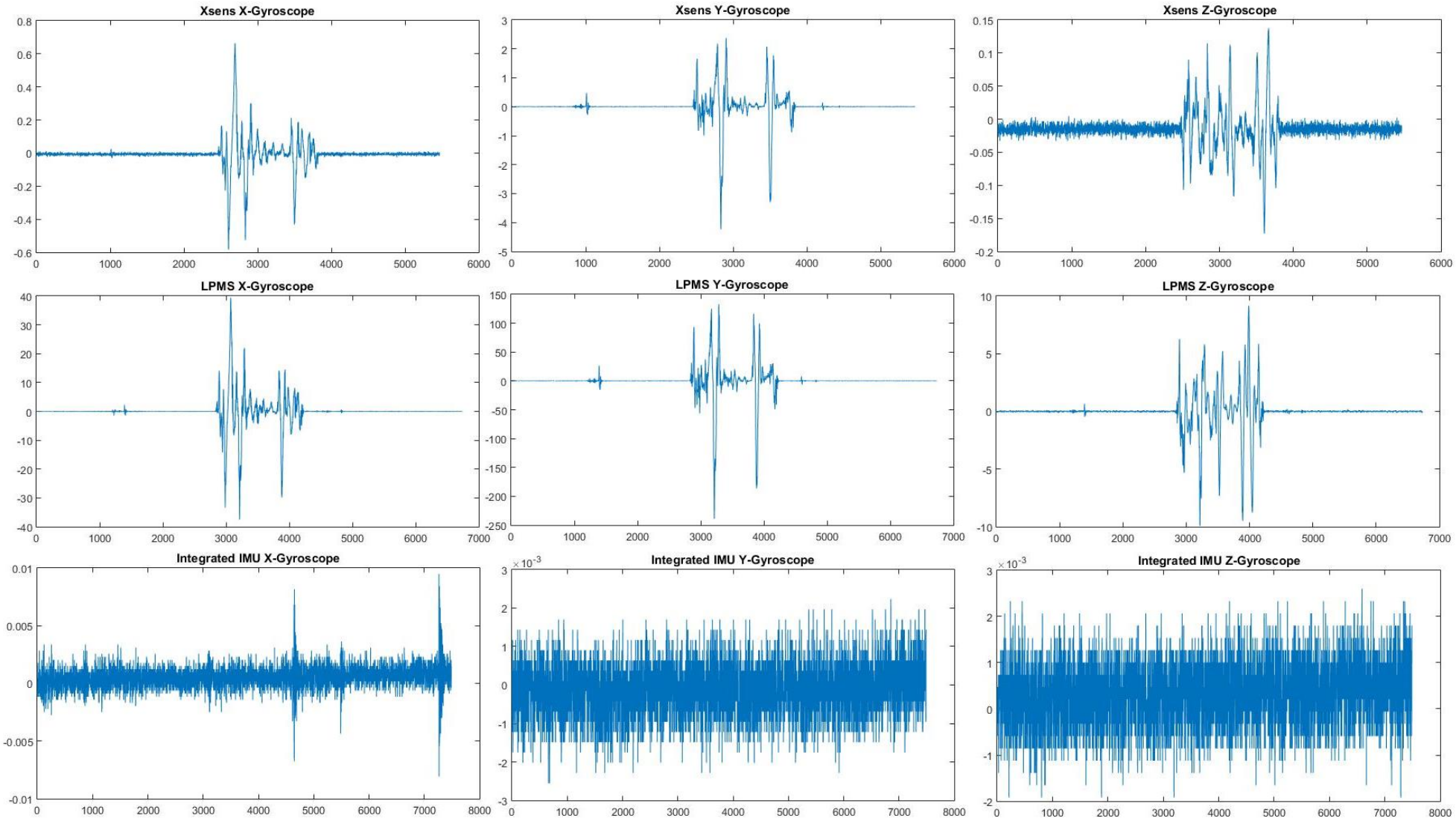


Figure 4: IMU test gyroscope results

3 Hardware Changes

Changing the hardware configuration of the SPAR was a major aspect of this project. In addition to the installation of the sensor array – which was the main goal of the project – the motor mounting hardware was redesigned and replaced, modifications were made to the layout of electronic components located inside the SPAR, and the power distribution circuit was improved.

3.1 Addition of Sensors

Each of the three sensors that formed the sensor array needed to be securely mounted on the SPAR and connected to the Raspberry Pi Zero.

3.1.1 Daughter Board

A 75 x 43mm prototyping PCB was added to the SPAR to act as a daughter board. The ADC was mounted directly to this board, all the power and logic connections for the conductivity and temperature sensors were made via this board. A circuit diagram of the board can be seen in Figure 5, and a picture of the board can be seen in Figure 6.

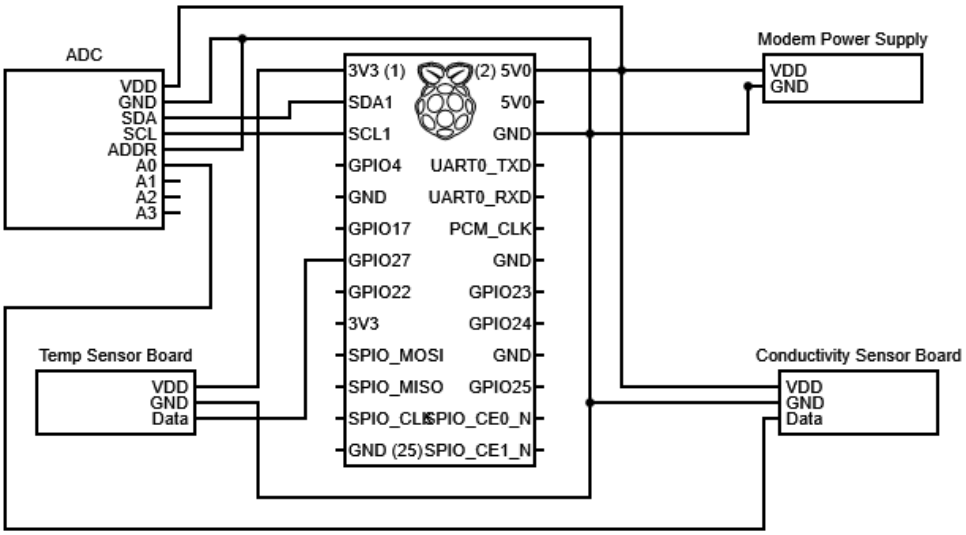


Figure 5: Circuit Diagram of components routed through the daughter board

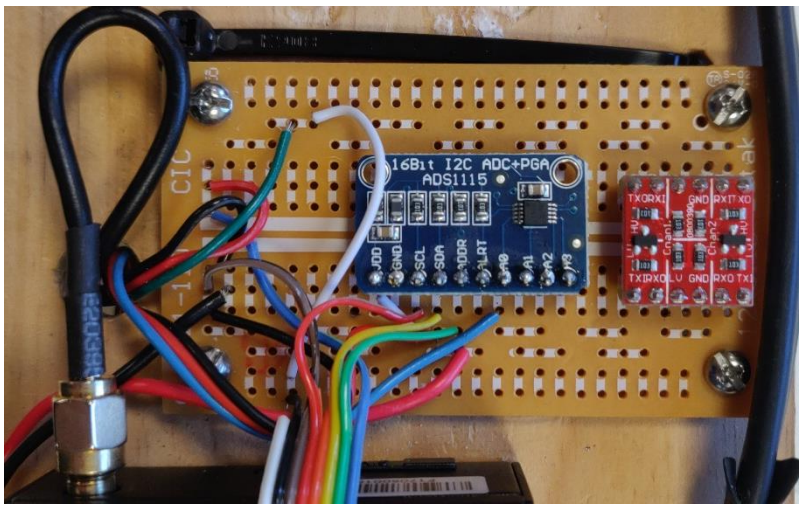


Figure 6: Picture of daughter board mounted on the SPAR

3.1.2 Conductivity Sensor and ADC

The conductivity sensor needs to be fully submerged in water to work, so it was mounted in the base of one of the motor mounts (Refer to Section 3.2). The wire was passed through a hole drilled in the top of the PVC tube, which was then sealed with epoxy glue. Inside the tube the cable was passed through a hole in the electronics plank and connected to a dedicated breakout board. The board converts the BNC signal output by the sensor into individual power, data, and ground lines, which are then soldered onto the daughter board.

The data line of the conductivity sensor is connected to the input pin A0 of the ADS1115 ADC. The SCL and SDA pins are then connected to the corresponding pins on the Raspberry Pi. Additionally, the ADDR pin of the ADC is grounded, corresponding to an I2C address of 0x48.

3.1.3 Temperature Sensor

The temperature sensor also needed to be submerged in the water, so it was mounted alongside the conductivity sensor in a similar manner. As with the conductivity sensor, it was plugged into a dedicated breakout board which ensured that the sensor could be connected to the Pi securely, while still allowing it to be disconnected if the SPAR needed to be unloaded. Additionally, the DS18B20 requires a 4.7kΩ pull-down resistor between the power and data lines to function, but this was integrated into the breakout board. The output wires of the breakout board were soldered directly to the daughter board. The data line needs to be connected to the GPIO pin that is configured to use the 1-Wire interface: by default, pin GPIO4 is used for this purpose. Unfortunately, this caused a problem; the Pi camera module – despite being connected through a dedicated port – uses GPIO4 during its runtime. As such, connecting the temperature sensor to the GPIO4 pin caused the camera driver to crash on start-up, preventing the camera from being used. This was fixed by configuring the Pi to use GPIO27 for the 1-Wire interface by adding the line `dtoverlay=w1-gpio,gpiopin=27` to the end of /boot/config.txt.

3.1.4 Camera

The integration of the Raspberry Pi camera module necessitated a significant number of hardware changes. Firstly, the camera needed to be able to see out of the SPAR, but the construction of the SPAR made this difficult. Since the SPAR curves upwards at the front, and the electronics are loaded from the rear, mounting the camera on the front of the boat would have been very impractical; the camera could not be mounted to the electronics board since it would not be able to see past the aforementioned curve, and it could not be mounted directly to the PVC tube because it would not be possible to plug it into the Pi after the electronics were loaded in. The camera could instead be mounted at the back, but the original design features solid PVC endcaps. To fix this, a PVC endcap with a Perspex window was commissioned from the UWA mechanical workshop, which can be seen in Figure 8.

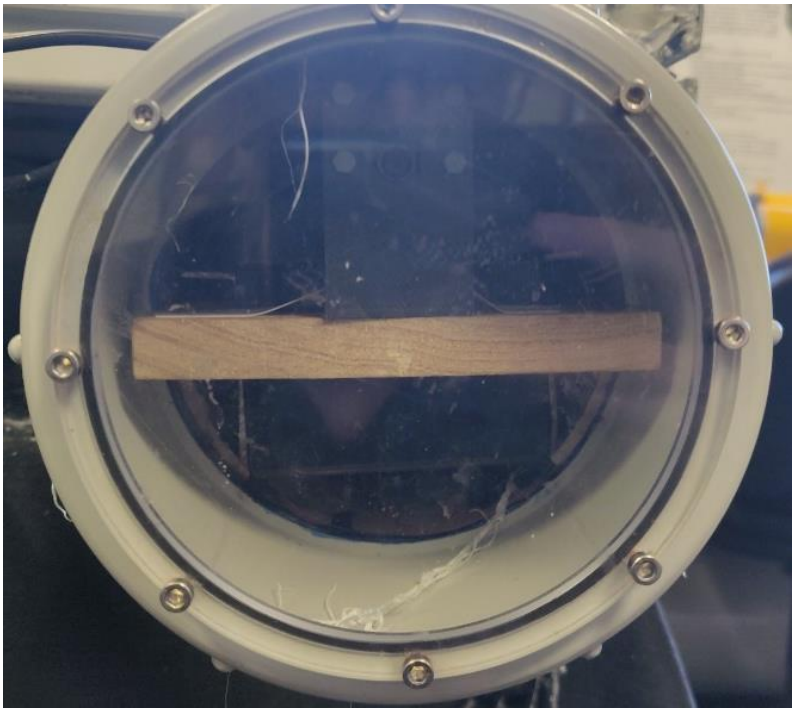


Figure 8: Perspex endcap installed on SPAR

Next, the camera itself had to be mounted, but the dimensions of the SPAR made this difficult. There was only 50mm of vertical clearance between the top of the electronics plank and the top of the PVC tube, and only 26mm of clearance between the end of the plank and the edge of one of the SPAR's batteries. Rather than trying to find an existing Pi camera mount that would fit these specifications, a custom camera mount was designed, and then 3D printed in ABS plastic. The mount was modelled using the Fusion 360 3D CAD/CAM software produced by Autodesk, and 3D printed by students working with Thomas Bräunl on other projects. The design is shown in Figure 7. The camera is mounted on the rear of the design to make it easier to connect to the Pi, with the sensor itself protruding through a special aperture. The mount itself is bolted to the plank using a pair of M2 screws.

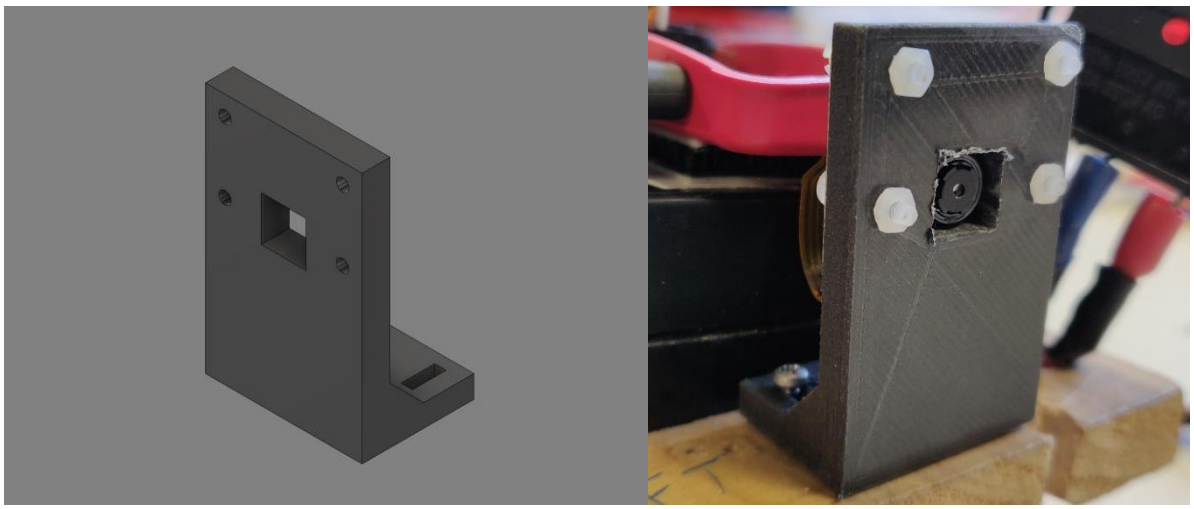


Figure 7 Left: 3D model of camera mount design. Right: 3D printed model of camera mount, with camera installed

3.2 Motor Mount Replacement

Due to outside circumstances, the SPAR's motor mounting hardware was damaged halfway through the project. The motor mounts were 3D printed components that allowed the SPAR's two electric motors to be attached to the PVC tubes that make up the SPAR's body. The damaged sustained can be seen in Figure 9 and Figure 10.

Figure 9: Damaged SPAR motor mount

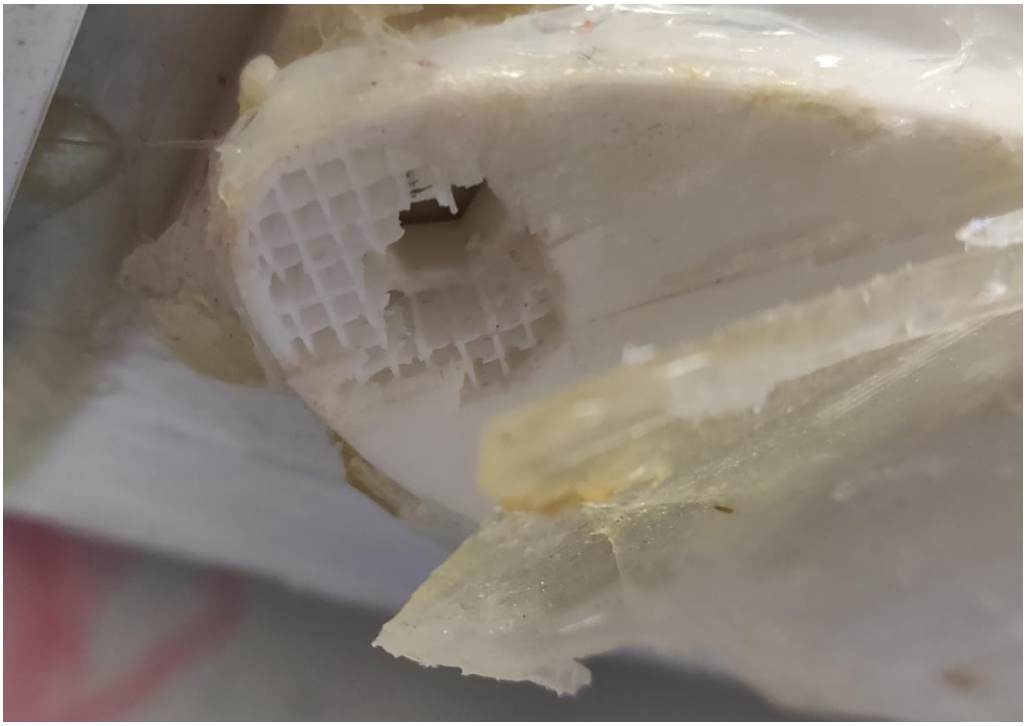


Figure 10: Close-up of damaged SPAR motor mount

Due to this damage, replacements for the motor mounts were designed. The replacement mounts were modelled and printed in the same fashion as the camera mounts. The first version of the replacement design was modelled after the original motor mount hardware and can be seen in Figure 11.

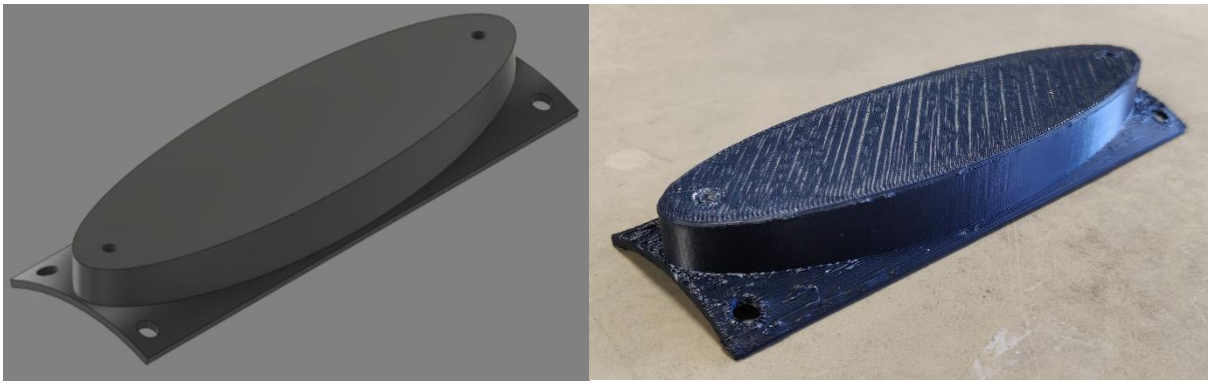


Figure 11 Left: 3D model of first motor mount design. Right: 3D printed model of first motor mount design

Unlike the first design which was attached to the PVC tubing using glue, this design was intended to be bolted through the PVC. However, due to concerns of compromising the integrity of the SPAR's hull, it was decided that the replacement mounts should be attached with epoxy glue like the originals. Additionally, the students who assisted with 3D printing the original mount provided feedback that the thin double curves present in the base of the mount were poorly suited to the 3D printing process and were a major structural weak point.

These factors prompted a complete redesign of the mounting hardware. The update design eliminated the weak double curves and was made significantly wider to provide more surface area for the epoxy glue. This design can be seen in Figure 12.

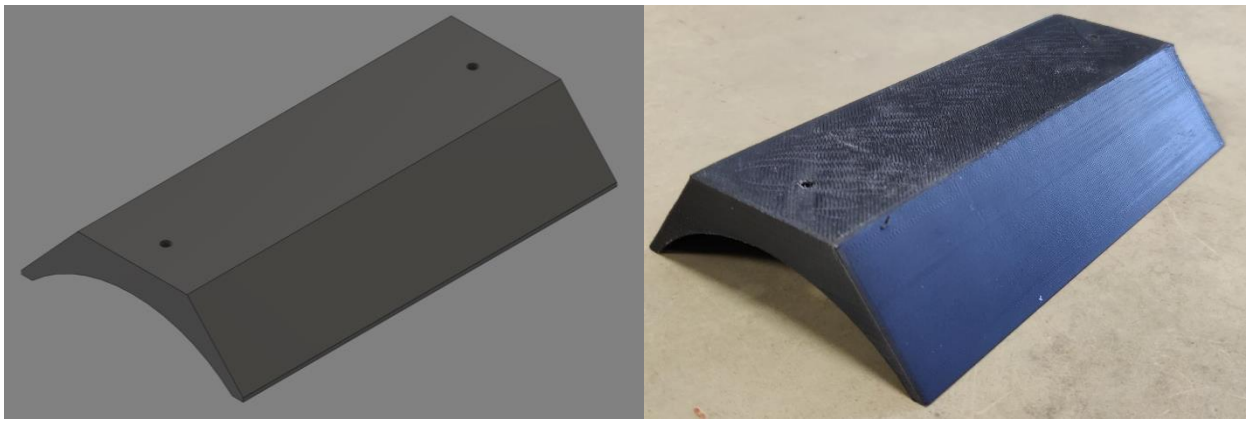


Figure 12 Left: 3D model of second motor mount design. Right: 3D printed model of second motor mount design

The redesign of these mounts provided an opportunity to solve another problem that needed to be addressed. The conductivity and temperature sensors needed to be installed on the outside of the boat, since they both need to be submersed in water to function as intended. However, this was not factored in to the original design of the SPAR, and as such a mounting solution for these components needed to be found. The ability to custom design the motor mounts allowed the mounting solutions for these components to be integrated directly into one of the motor mounts itself. A channel for the temperature sensor was integrated into the top of the mount, and a hole for the conductivity sensor was placed on one side of the mount. This version of the design can be seen in Figure 13, and the final mount with both sensors installed can be seen in Figure 14.

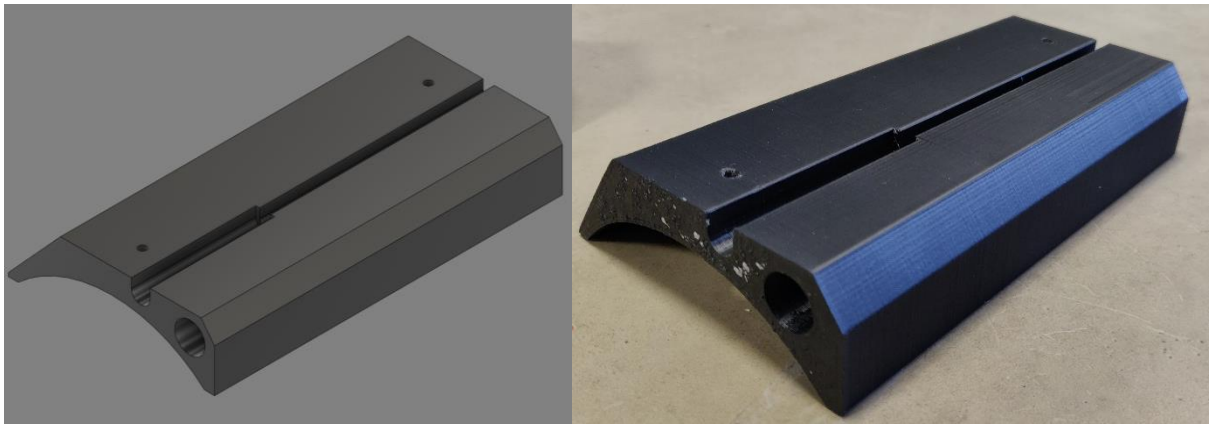


Figure 13 Left: 3D model of motor mount with sensor integration. Right: 3D printed model of motor mount with sensor integration

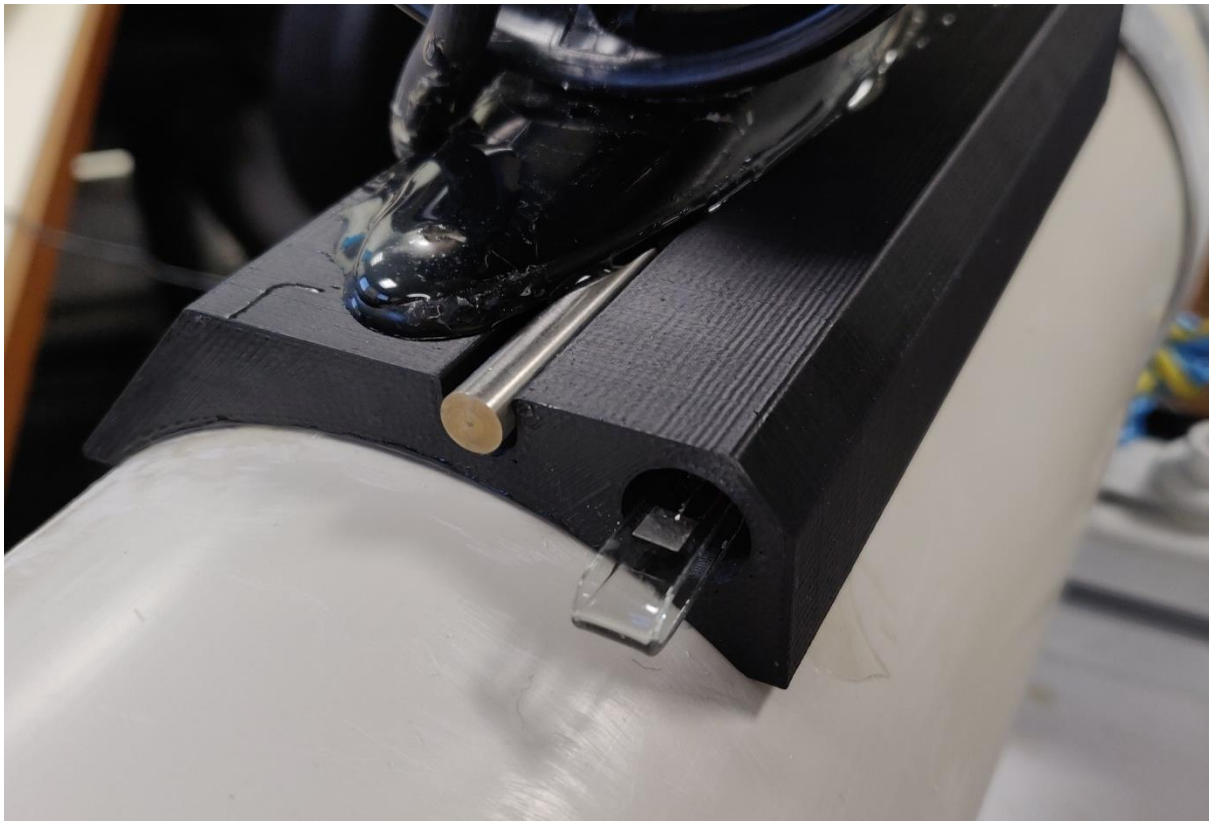


Figure 14: Motor mount with sensors installed

3.3 Layout/Wiring Improvements

While the primary focus of this project was on the addition of new hardware, certain modifications to the existing hardware of the SPAR proved necessary. These changes were made with two objectives: to fix flaws with the existing design, and to accommodate the new hardware.

3.3.1 Layout Update

The original layout of the electrical components of the SPAR is shown in Figure 15 and Figure 16. This layout had several flaws. The original location of the Raspberry Pi made the installation of a Raspberry Pi camera module very difficult and made it awkward to test the sensor array. There wasn't any space remaining for the installation of new components on the top of the board, and many of the connections were very unreliable. The connections made to the Raspberry Pi in particularly were made with daisy-chained Dupont wires, which were prone to disconnecting. Additionally, the poor cable management meant that cables on the plank were prone to getting snagged on the cables that pass through the hull of the SPAR while the plank is loaded into the PVC tube. A variety of changes were made to the layout of the SPAR to fix these issues.

Firstly, the Raspberry Pi was moved from the underside of the plank to a position on top of the aft battery. While mounting the Pi on a battery is not ideal, placing it there made it significantly easier to connect the camera, and left enough room on the top of the plank to install the daughter board. The Pi was mounted inside its case, which was secured to the battery using Scotch adhesive fasteners; this held the case firmly in place while still allowing it to be removed and reattached as necessary.

Initially, the Pi was connected to the ArduCopter MCU using Dupont wires connected to UART pins on both devices, with a 5V/3.3V level shifter in between. This connection was replaced with a micro-USB to USB cable, which also prompted the addition of a 4-way USB hub. The hub was installed on the underside of the plank, and it allowed both the ArduCopter and the modem to connect to the Pi Zero – which only features a single micro-USB port – over USB serial. The homemade RS232 to USB converter was moved from the underside of the plank to the top because strain on the serial cables was causing them to disconnect from the RS232 connector.

Most cables were moved to the underside of the plank, where they were then secured with cable ties, and holes were drilled in appropriate places to allow each cable to reach the appropriate components. As part of this process, the voltage regulator was also moved to the bottom of the plank. To fix issues with components getting caught during loading, the 3G antenna was firmly secured to the front of the plank using duct tape, and additional plastic protective sheathing was added to the plank. During this process, it was discovered that the female spade connectors on the battery leads were too large for the terminals on the batteries themselves, which caused them to come loose very easily; these connectors were replaced with connectors of the appropriate size.

The motor power connectors, which originally were spread over a significant portion of the plank, were moved to a more compact configuration to provide enough room for the installation of the daughter board. The daughter board itself was mounted using M3 screws, with nuts placed underneath the board to elevate it off the wooden plank. Finally, the breakout boards for the conductivity and temperature sensors were installed on the underside of the plank, since there was not enough room on the top.

The final layout of the SPAR is shown in Figure 17 and Figure 18.

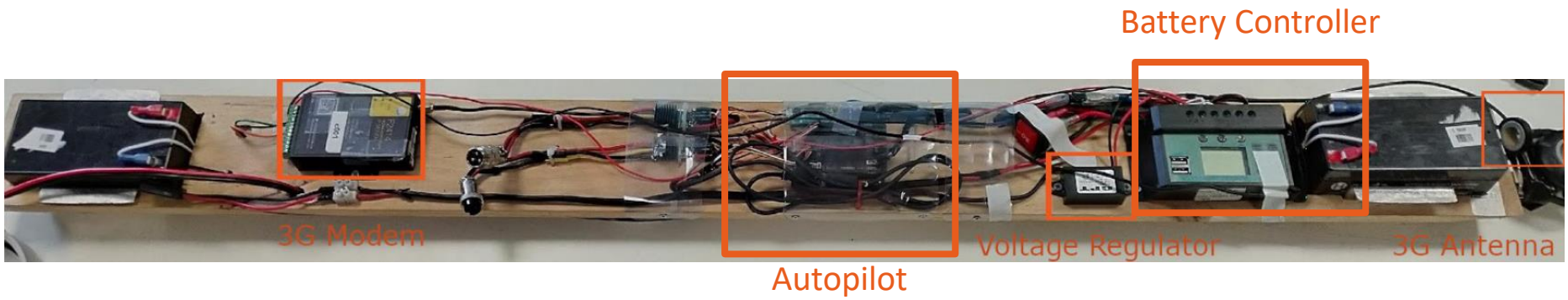


Figure 15: Original layout, Top side

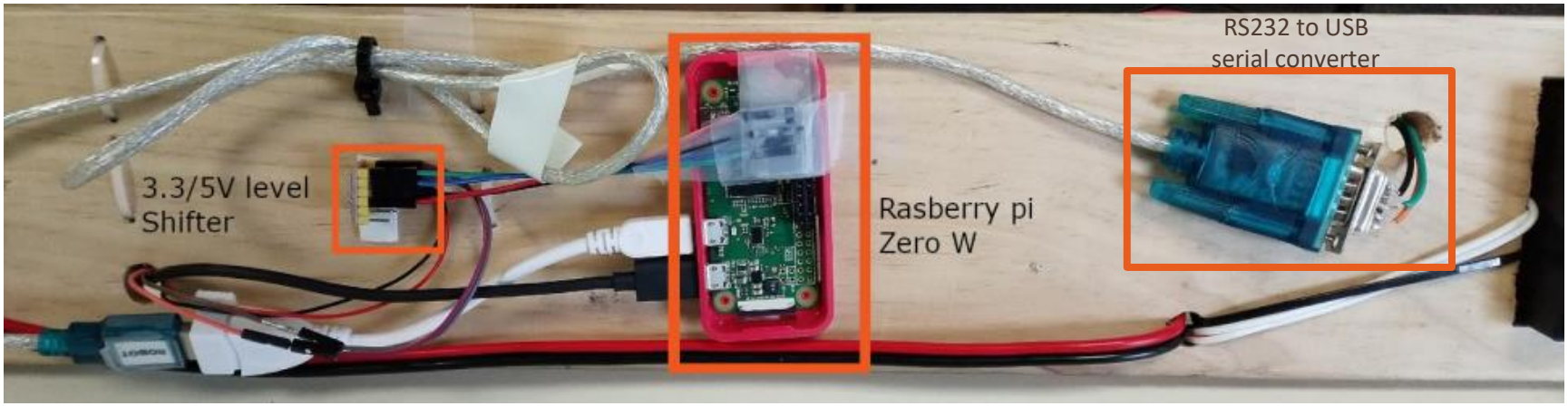


Figure 16: Original layout, Bottom side

Figure 17: Updated layout, Top side [10]

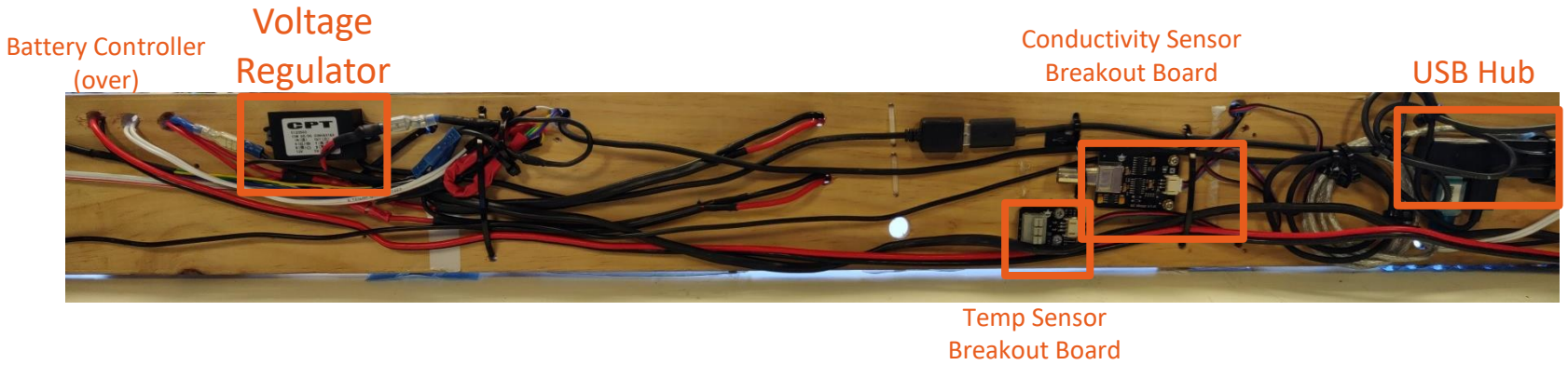


Figure 18: Updated layout, Bottom side [10]

3.3.2 Load Circuit Topography Improvement

While testing the power distribution circuit of the SPAR it was discovered that the initial design was seriously flawed. A simplified overview of the topography of the power distribution circuit is shown in Figure 19. The intent of this circuit was that a load switch – located at the rear of the vessel – could be used to control power to the motor, while not shutting off power to the modem or Pi.

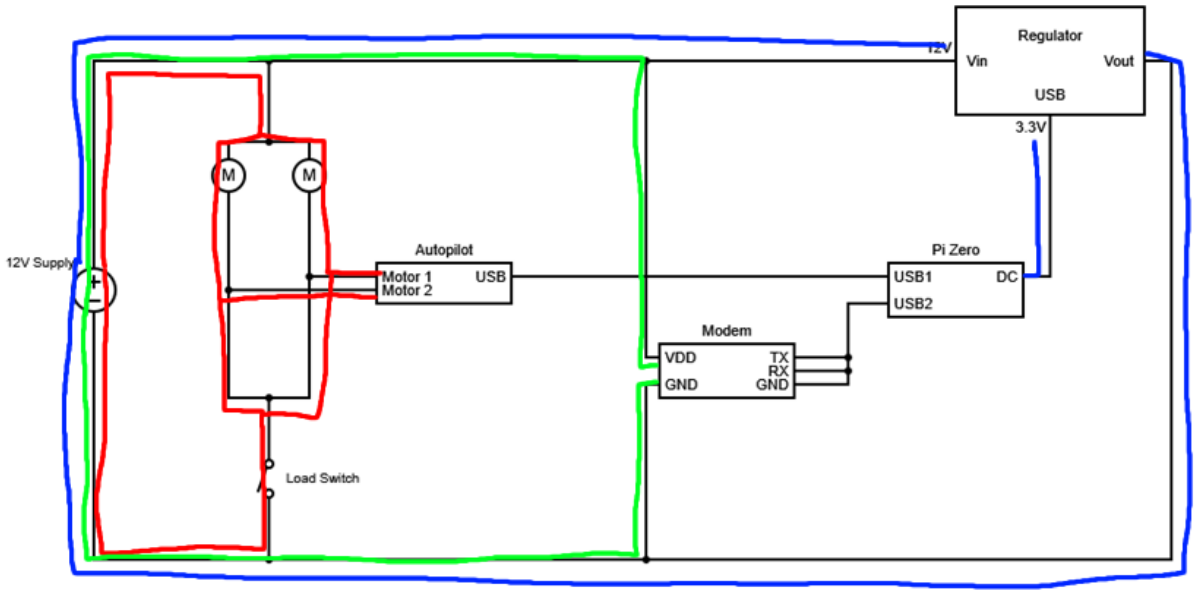


Figure 19: Simplified topography of SPAR power distribution circuit, with intended conduction paths illustrated

However, this circuit topography was severely flawed. As highlighted in Figure 20 and Figure 21, current could flow through the Ardupilot, over the USB connection into the Pi, and then either through the 3G modem or through the voltage regulator to reach ground and effectively bypass the load switch. This posed a safety hazard, as the only switch that could turn off power to the motors was located at the front end of the electronics plank and not easily accessible during loading.

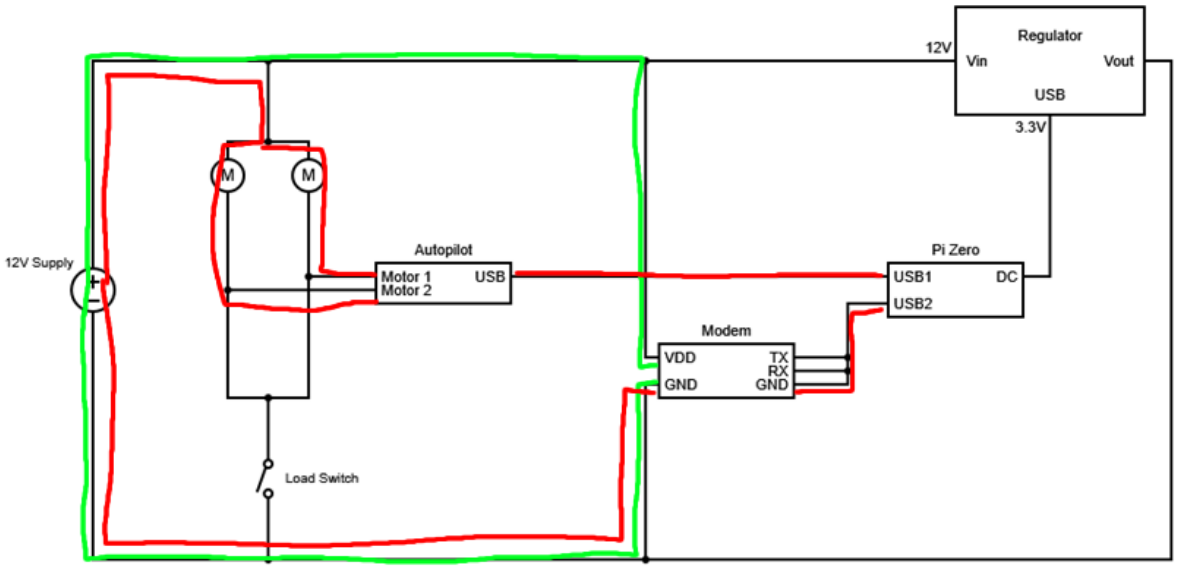


Figure 20: Switch bypass conduction path, grounding through the 3G modem

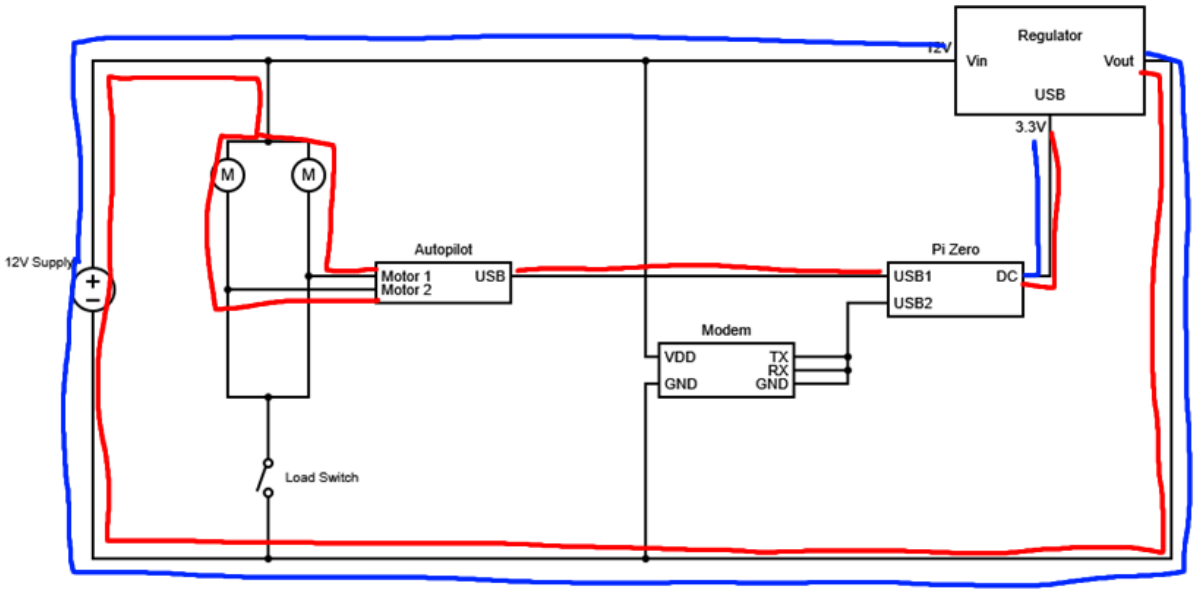


Figure 21: Switch bypass conduction path, grounding through the voltage regulator

This issue was fixed by reconfiguring the circuit topography. The modem – which was originally supplied 12V directly from the battery controller – was changed to be supplied with 5V from the Raspberry Pi, and the ground line of the voltage regulator was moved upstream of the load switch. This configuration can be seen in Figure 22.

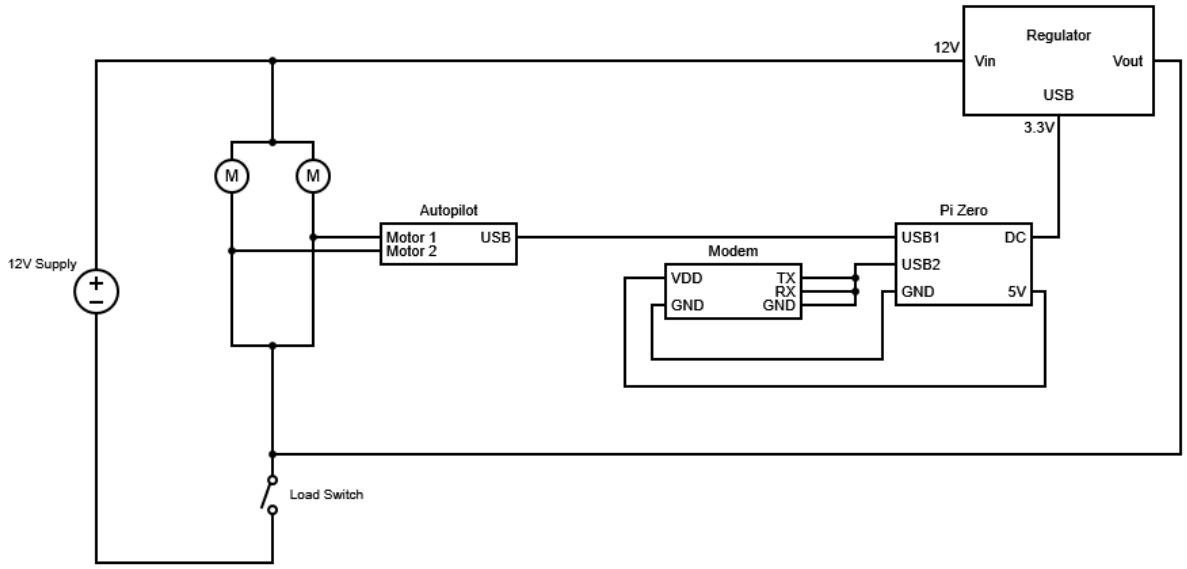


Figure 22: Simplified topography of updated SPAR power distribution circuit

4 Software Changes

The final aspect of this project involved updating the Python scripts running on the Raspberry Pi Zero. These updates focused on integrating the new sensors, updating the interface between the Raspberry Pi and the remote server, and updating the interface between the Pi and the Ardupilot software.

4.1 Sensor Manager Class

The addition of a sensor array to the SPAR prompted the addition of a new Python class to handle it. The updated architecture of the software running on the Raspberry Pi is shown in Figure 23.

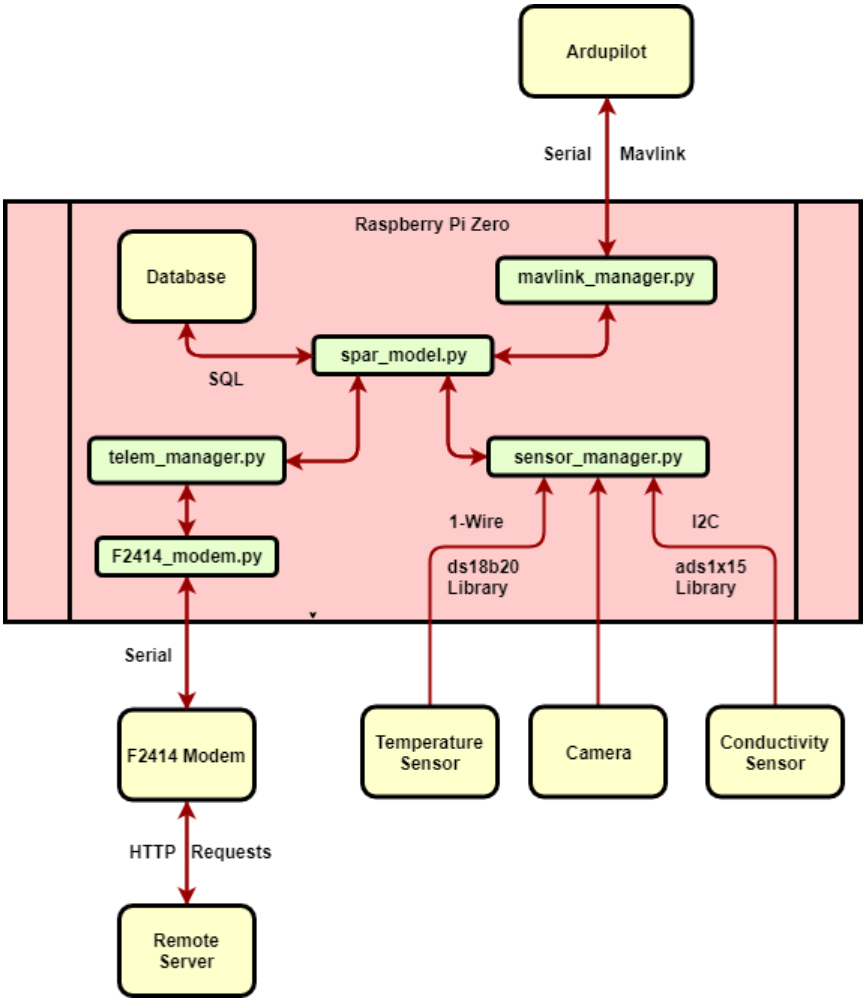


Figure 23: Updated Python software architecture

The sensor manager class contains functions to control the two sensors and the camera. Controlling the temperature sensor is straightforward; the ds18b20 library provides a function to automatically read the current temperature reported by the sensor in degrees Celsius. Once the reading has been taken, it is stored in the spar_model class, which is used to store information that is relevant for the control of the SPAR itself. The function to update the temperature reading is called by the telemetry manager immediately before uploading telemetry data to the server, which is done at approximately 1-minute intervals.

Controlling the conductivity sensor is more involved. The adafruit-ads1x15 library is used to obtain the voltage output of the conductivity sensor via the ADS1115 ADC. One issue with the conductivity sensor is that the conductivity readings are only meaningful if the sensor's electrodes are fully submerged, but given the mobile nature of the SPAR platform it is possible that they might not the case for any specific reading. To compensate for this, multiple readings are taken in quick succession, and the median value of these readings is used, mitigating the effects of any outlier readings. Next, the voltage recorded must converted into a conductivity reading. This done using the temperature compensation and conversion formulas outlined in Section 2.2. If the voltage recorded is outside the measurement range of the sensor (0.15V to 3.3V), the conductivity value is set to 0 and -1 respectively. The conductivity is then stored in the spar_model class. As with the temperature sensor, the function to update the conductivity reading is called by the telemetry manager immediately before uploading telemetry data to the server.

The camera is controlled using the picamera library. To reduce the storage overhead of the recorded images, the camera is set to a resolution of 320x240. Each image is stored in the Raspberry Pi's home directory as image_[N].jpg. The number N is determined by an image counter variable stored in spar_model. This counter is initialised to 0 and is incremented every time a new image is taken. Once the counter reaches 1000 it is reset to 0, limiting the number of images that can be stored at any given time. Accompanying the camera control method is a method which converts an image to a base64 encoded string. The base64 string can then be uploaded to the remote server and decoded to extract the initial image. Unfortunately, due to issues with the reliability of the F2414 modem, the functionality to upload images to the server was not implemented. Unlike the other sensors, the image-taking function is called by a Python scheduler at 15-minute intervals, independent of the other operations of the SPAR.

4.2 Modem Interface Update

The modem interface manages the data flow into and out of the SPAR's 3G modem. The client program communicates with the server by sending HTTP requests at regular intervals and decoding the packets returned by the server to extract waypoint data. Most of the changes made to this interface were concerned with the contents of the data transmissions, leaving the control flow mostly unchanged. These changes were necessitated by the changes to the server made as part of a simultaneous project.

The outgoing HTTP headers were updated to point to the new server backend: POST requests are now sent to http://therevproject.com/solarboat/api/data.cgi, and GET requests are sent to http://therevproject.com/solarboat/api/command.cgi. The JSON packets containing telemetry data were also updated to contain a timestamp (obtained from the SPAR's GPS), a unique source ID, and the latest temperature and salinity data recorded by the sensors. The format of the JSON packet is as follows

```
{
    "msg_type": ,
    "source_id": ,
    "timestamp": ,
    "latitude": ,
    "longitude": ,
    "temperature": ,
    "salinity":
}
```

Additionally, the way packets received from the server are parsed was updated. The original server would only provide a single waypoint at a time, whereas the new server sends all pending waypoints at once. If the packet contains a successful HTTP response, as indicated by the header string `HTTP/1.1 200 OK`, the client converts the remainder of the message into JSON, and then checks the message type field. If the message type indicates that the message contains a list of waypoints, the waypoint list is extracted from the JSON, sorted by the ID field to ensure the waypoint order is correct, and then the GPS coordinates are stored in spar_model.pendingWaypoints.

4.3 Mavlink Interface Updates

The Mavlink interface manages the communication between the Raspberry Pi and the Ardupilot software, which it does using the Mavlink communication protocol. This interface has two primary functions: upload the waypoints the SPAR receives from the remote server into the Ardupilot, and extract location and GPS time data to be uploaded to the server. The original configuration was designed around receiving each waypoint individually and relying on the server to not send duplicate data. When a waypoint was received, it would be appended to the current list of waypoints in the Ardupilot. This approach would not work with the updated backend – which sends a list of all pending waypoints with every transmission – and so was updated.

In the revised approach, the SPAR maintains two lists of waypoints which are both stored in spar_model: a list of the waypoints sent by the server, and a list of waypoints currently stored by the Ardupilot. Every five seconds, these two lists are compared to check whether there are any differences. Initially, this was done using a basic equality test, but it was found that the limited precision of the Ardupilot meant that coordinates were slightly altered when sent to it. As such, each coordinate pair is compared manually, and if there is a difference of more than 0.00001° between the server and Ardupilot coordinates, the two lists are considered different. If there is a discrepancy, the SPAR will assume that the server is correct, clear the Ardupilot's waypoint list, and upload all the waypoints from the server list. The process for uploading the waypoints was established as part of a previous project [10], but a diagram of the messaging sequence is shown in Figure 24.

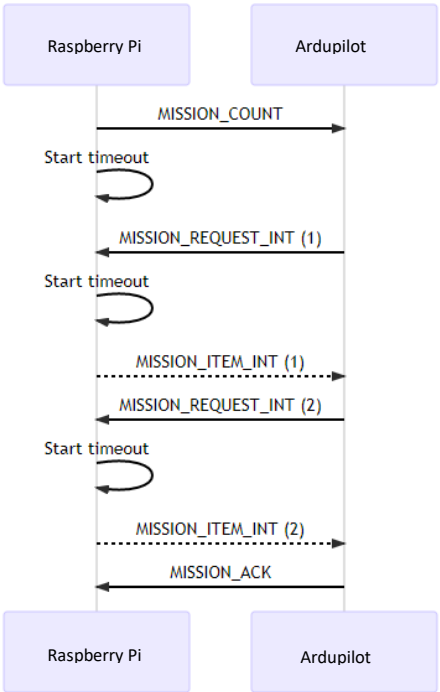


Figure 24: Communication sequence to upload waypoints to the Ardupilot over Mavlink [17]

5 Testing

While a number of issues encountered towards the end of the project meant that the SPAR was not tested as thoroughly as planned, the tests that were conducted showed that the fundamentals of the SPAR are functional. Most testing was done in a dry lab environment, to validate the control flow of the Python scripts running on the Pi, and the server backend. These tests revealed the biggest issue facing the SPAR currently; the 3G modem is extremely unreliable. Only approximately 5% of packets sent to the server resulted in a successful response, while the rest of the time the SPAR would receive a HTTP 400 error (indicating a bad packet) or no response at all. Despite the low success rate of the modem, the testing was able to confirm that all elements of the SPAR acted correctly when the connection did work; telemetry data was successfully uploaded to the server, and waypoints were successfully retrieved and uploaded to the Ardupilot.

A single water test was conducted, but this test was unsuccessful: the motors of the SPAR cut out upon being placed into the water. This was a result of loose battery terminal connections; the jolt of the boat being lowered into the water dislodged these connections and the SPAR lost power. Following this test, the battery connectors were replaced with ones of the correct size, and changes were made to protect the wiring of the SPAR from getting dislodged during the loading process.

None of the changes made to the SPAR during this project would have impacted the core functionality of the ArduCopter module, and dry tests showed that the interface between the MCU and the server was functioning correctly (aside from the connectivity issues). Given this, and the successful tests conducted in the past [2] [9] [10], it is believed that once the connectivity issues are resolved the SPAR should be fully functional, but unfortunately a lack of testing meant that this could not be properly verified.

6 Issues Faced

This project encountered several issues, and unfortunately some of these issues were not able to be resolved within the timeframe of the project.

6.1 Modem Reliability

The most significant issue facing the SPAR is the unreliability of the FourFaith F2414 Modem. This modem is designed for direct M2M communication over TCP/IP, but it also supports a telnet mode that enables the use of HTTP requests, which were needed for communication with the remote server. Unfortunately, this mode has no documentation surrounding it [16], and the modem itself does not provide any feedback to the client during operation; this made it extremely difficult to diagnose the problem. The specific modem used in the SPAR was swapped for an identical model to rule out hardware faults, but the ultimate cause of the issue was unable to be determined. The most likely explanations are that either the modem is not able to send the data to the server quickly enough (resulting in the server timing out and ending the connection prematurely), or that the modem is not able to properly receive the data sent to it over the serial connection. With no documentation and no feedback, this issue was not able to be resolved, but a significant amount of development time was spent troubleshooting the issue, negatively impacting the rest of the project. The best solution to this problem would be to replace the F2414 modem with a different modem that doesn't experience the same issues – a good candidate might be the Luat 4G Air 720 modem, which members of the REVSki project were able to use successfully after experiencing similar issues with their original modem – but there was not enough time remaining after reaching this conclusion to source and then integrate a replacement modem within the timeframe of the project.

6.2 Hardware Issues

Several hardware issues had negative impacts in the project. Chief among them was the damage sustained to the motor mounts, which required them to be replaced. This was one of the key factors that limited the amount of testing that could be done; the replacement mounts went through a series of revisions, and the final designs were not received until quite late into the project. Additionally, the sensors could not be mounted onto the SPAR itself until the replacement mounts were successfully installed, which further compounded the issue. This combined with the modem connectivity issues heavily limited the scope of testing.

There was also a string of minor issues related to the original hardware design of the SPAR. There is very little clearance inside the PVC tubes that house the electronics, which complicated the process of adding and rearranging components on the electronics plank. To further compound this issue, there are five wires that are passed through the top of the PVC tube, which are connected to the motors, the solar panel, and to the batteries located inside the other main tube. These wires would constantly get snagged on various electrical components, causing internal connections to break on multiple occasions. While this issue was mostly resolved, the clearance issue persists and will complicate any future attempts to expand on the SPAR.

Finally, there was an issue with the power supply of the Raspberry Pi. The ArduCopter MCU can receive power via its connections to the motors, but when it is connected to the Pi over UART the Pi attempts to power it. The Pi is powered by a 15W voltage regulator, which is enough to power the Pi, the modem, and the ArduCopter, but there isn't enough power overhead to power both the ArduCopter and the Pi camera at the same time. Testing revealed that the Pi camera and its associated code were fully functional while the ArduCopter was disconnected from the Pi, but plugging in the Ardupilot meant that the Pi's camera driver could no longer initialise the camera. This issue was also present when attempting to control the Pi camera directly from the command line using the `raspistill -o test.jpg` command. This issue could be fixed by replacing the 15W voltage regulator with a 25W model, but unfortunately this problem was only discovered at the very end of the project, and such a regulator could not be sourced in time. As an interim solution, the code associated with the camera has been commented out of the sensor manager so that the other parts of the SPAR can function as intended.

6.3 Miscellaneous Issues

Another issue which complicated the process of conducting open water testing was the need for a retrieval plan. Since the SPAR is autonomous, there is the possibility that if something goes wrong during a test the SPAR could either become stranded or lose control and start driving constantly in a random direction. In either case, the SPAR would need to be retrieved manually, but this would require the use of an ocean craft. The possibility of borrowing a kayak from the UWA Sports department was investigated, but ultimately proved impossible. Following this, a partnership with the UWA Oceans Institute was considered, but never materialised due to the other issues preventing open water testing.

The final issue facing this project was time management. Due to commitments with other units during the first semester of the project, progress on the sensor array during this time was limited. Additionally, as mentioned a significant amount of time was spent trying and failing to resolve the modem connectivity issue, which had a detrimental effect on the development of other aspects of the SPAR.

7 Conclusion

The success of this project was ultimately hampered by the inability to conduct full, open water testing. While the functionality of the SPAR was verified in dry lab tests, further work will need to be done to fix the connectivity issues and subject the SPAR to proper full-scale tests.

7.1 Recommendations for Future Work

The most important priority for future work is fixing the connectivity problems. The best way to do this would be to replace the FourFaith modem entirely. This could be an excellent opportunity to upgrade to SPAR to an Iridium modem, which would allow the SPAR to make intercontinental voyages. Iridium modems are quite expensive however, so a modem such as the Luat 4G Air 720 might be a more appropriate solution. Additionally, the voltage regulator that supplies power to the Raspberry Pi should be upgraded so that the Pi is supplied with enough power to run the camera and the ArduCopter module simultaneously.

The sensor array could potentially be expanded. Two key metrics which could be recorded are the load voltage and load current at the output of the battery controller. Measuring and uploading these parameters to the remote server could allow for an estimate of the battery charge level, which is highly desirable for a remote, autonomous vessel. Another type of sensor that could be added is a PH sensor, to improve the SPAR's water quality measuring capabilities. A further improvement to the hardware would be the addition of a switch to the outside of the SPAR capable of shutting off the SPAR's motors.

Lastly, the software running on the Raspberry Pi could be extended to allow for improved control of the SPAR from the remote server. The Mavlink protocol supports functions such as waiting at a certain point for a specified time, which could be utilised to allow the SPAR to monitor water quality at a fixed location for an extended period. Additionally, kill-switch functionality could be added such that a remote user could command the SPAR to shut down in the event of a malfunction. Furthermore, the Mavlink control flow could be improved with the addition of a proper timeout system to handle potential communication problems between the Ardupilot and the Pi, and between the Pi and the remote server.

8 Acknowledgements

I would firstly like to thank my supervisor Thomas Bräunl for organising the entire SPAR project as well as fellow project member Morgan Trench. I would also like to thank REVSki project members Dylan Leong and Marti James Leven, who experienced similar issues, and shared their experiences dealing with them, as well as Kai Li Lim, who provided excellent advice for working with the F2414 modem. I would like to thank Marti James Leven (again) and Marcus Pham, for their help with the process of modelling and 3D printing the parts I needed. Lastly, I would like to thank the members of the UWA Electrical workshop, who were able to help with certain aspects of the project I did not have the resources to complete on my own.

9 References

[1] D. McMillan, “Seacharger,” [Online]. Available: http://www.seacharger.com/. [Accessed 12 September 2018].

[2] J. Hodge, “Project Report: Autonomous Solar Powered Boat,” 28 October 2017. [Online]. Available: http://robotics.ee.uwa.edu.au/theses/2017-Solarboat-Hodge.pdf. [Accessed 9 September 2018].

[3] Mircrotransat, “The Microtransat Challenge,” 26 August 2018. [Online]. Available: https://www.microtransat.org/index.php. [Accessed 13 September 2018].

[4] Liquid Robotics, “Wave Glider: Overview,” 2018. [Online]. Available: https://www.liquid-robotics.com/wave-glider/overview/. [Accessed 9 September 2018].

[5] V. V. Lancker and M. Baeye, “Wave Glider Monitoring of Sediment Transport and Dredge Plumes in a Shallow Marine Sandbank Environment,” *PLoS ONE,* 12 June 2015.

[6] N. T. Penna, M. A. M. Maqueda, I. Martin, J. Guo and P. R. Foden, “Sea Surface Height Measurement Using a GNSS Wave Glider,” *Geophysical Research Letters,* vol. 45, no. 11, pp. 5609-5616, 2016 June 2018.

[7] S. Mitarai and J. C. McWilliams, “Wave glider observations of surface winds and currents in the core of Typhoon Danas,” *Geophysical Research Letters,* vol. 43, no. 21, pp. 11,312-11,319, 28 October 2016.

[8] D. Takahashi, “Liquid Robotics launches new generation of wave glider ocean robots,” VentureBeat, 8 April 2013. [Online]. Available: https://venturebeat.com/2013/04/08/liquid-robotics-launches-new-generation-of-wave-glider-ocean-robots/. [Accessed 12 September 2018].

[9] J. Borella, “Solar Powered Autonomous Raft (SPAR) Final Year Research Project Thesis,” 28 May 2018. [Online]. Available: http://robotics.ee.uwa.edu.au/theses/2018-Solarboat-Borella.pdf. [Accessed 9 September 2018].

[10] A. Goldsworthy, “Remote control of Autonomous Surface Vessels,” 31 October 2018.

[11] National Geographic, “Sea Temperature Rise,” *National Geographic,* 27 April 2010.

[12] NASA, “Salinity,” [Online]. Available: https://science.nasa.gov/earth-science/oceanography/physical-ocean/salinity. [Accessed 14 May 2019].

[13] DFRobot, “Analog EC Meter SKU DFR0300,” [Online]. Available: https://wiki.dfrobot.com/Analog_EC_Meter_SKU_DFR0300. [Accessed 14 May 2019].

[14] Racelogic, “How Does DGPS (Differential GPS) Work?,” 22 August 2018. [Online]. Available: https://racelogic.support/01VBOX_Automotive/01General_Information/Knowledge_Base/How_Does_DGPS_(Differential_GPS)_Work%3F. [Accessed 16 May 2019].

[15] M. Kok, J. D. Hol and T. B. Schön, “Using Inertial Sensors for Position and Orientation Estimation,” *Foundations and Trends in Signal Processing,* vol. 11, pp. 1-153, 2017.

[16] FourFaith, “F2X14 Series IP MODEM User Manual,” 24 October 2016. [Online]. Available: https://en.four-faith.com/uploadfile/2016/1116/20161116060649631.pdf. [Accessed 18 May 2019].

[17] Mavlink, “Mission Protocol,” [Online]. Available: https://mavlink.io/en/services/mission.html. [Accessed 5 May 2019].

Appendix A: Bill of Materials

A list all items purchased for this project is shown in Table 1.

Table 1: Bill of Materials

Item	**Quantity**	**Cost**
Analog Electrical Conductivity Meter (with Temperature Compensation)	1	$125.35
Waterproof DS18B20 Digital temperature sensor	1	$10.08
ADS1115 ADC	1	$11.95
75 x 43mm DIP Spacing Prototyping PCB	1	$2.50
J Burrows 4 Port USB 2 Hub	1	$12.00
35ml Selleys Super Strength Araldite Epoxy Adhesive	2	$19.75
8x Zenith M3X12 Marine Grade Stainless Steel Bolts and Nuts	1	$3.54
Total		$204.92

Items such as the 3D printed components, the Raspberry Pi camera module, and miscellaneous cables and fittings were obtained without cost through the University, or through other students working with Thomas Bräunl; these items are not included in Table 1.

Appendix B: Software Resources

The python code that is installed on the Raspberry Pi can be found at https://github.com/21485619/pyspar

The MAVLINK protocol set can be found at https://mavlink.io/en/messages/common.html