Designing the RoboSeals' RobotX 2022 Challenge

John Bowes, Isabelle Colby, Bowen Thomas, Tom Hooper, Shuaipu Zhao, Jeremiah Ye and Adam Jenkins

***Abstract*—The RoboSeals are a newly formed RobotX team combining students and staff from the University of South Australia with mentors and alumni from the Student Robotics Club of South Australia, Inc, with the assistance of industry partners. For their 2022 entry into the competition, the team used a highly distributed and modular system design that allowed the project to be embedded within undergraduate Information Technology degrees at the university. It was found that this approach was viable, despite several challenges, and the resulting entry incorporates ideas and technologies from FIRST Robotics Competition entries with new approaches, including swerve drive, vision processing, and communication over a CAN bus using custom protocols.**

I. Introduction

The Maritime RobotX Challenge is an exciting opportunity for any university student. It involves constructing and programing an Unmanned Surface Vehicle (USV) to perform a series of autonomous tasks using the WAM-V 16 platform. [1] This is challenging, but when combined with a second robotics task – an Unmanned Aerial Vehicle (UVA) – the competition gets even more interesting. In 2021 the University of South Australia and the Student Robotics Club of South Australia, Inc, (SRCSA) along with industry partners, decided to find out if they were up to the challenge and the RoboSeals were formed.

Over the next 18 months a core team consisting of a small number of students from the university and the SRCSA, along with academic advisors, mentors and industry representatives, have been developing "Licorice Sticks", their entry for the 2022 competition. As will be described, the team was much larger than that core group – the project was embedded into undergraduate IT degrees at the university through a capstone project course, and accordingly a large number of students from the university worked on different aspects of the project. This created multiple challenges, most importantly that it required a distributed architecture to be employed on the USV.

This paper explores those challenges, looks at the solutions that were developed, and examines how and why those approaches were taken, before looking at how the team plans to complete the preparation of their entry.

II. Background

UniSA is the largest university in South Australia and has a strong STEM focus, offering degrees in information technology, electrical and mechanical engineering, and mechatronics. Although students at the university have historically competed in small scale robotics competitions, such as the Warman Challenge, [2] the university has had little direct involvement in large scale robotics contests such as that offered by RobotX.

As part of all undergraduate information technology degrees at UniSA, final year students must complete a team-based project through the ICT Capstone Project (ICT) course. These projects are generally sourced from industry partners but getting projects that fit well with student and university requirements can be challenging. RobotX was viewed as providing a means of generating multiple capstone projects for students which can be aligned to specific student capabilities.

The SRCSA, also known as the "RoboRoos", is a community robotics club that has a ten-year track record of student robotics programs covering five different levels. They have competed in the FIRST Robotics competition, primarily in the FIRST Lego League (FLL) for primary and middle school students, and the FIRST Robotics Competition (FRC) for students in middle school through to the end of high school.

Over the years that the SRSCA has been in operation, many of their students have completed high school and gone on to higher education. In some cases, this has been in areas related to robotics, but often they must give up robotics to pursue their degrees. For many years the SRCSA have been looking for a university-level robotics challenge so that alumni can continue to engage with robotics after they no longer qualify to compete in the FIRST competitions.

Combining the two teams brought together the programing and network skills of the students at UniSA with the robotics and competition experience from the SRCSA alumni, along with a large network of mentors and industry partners from both sides. Ultimately, approximately 30 undergraduate UniSA students worked on the RobotX project over 18 months, while 12 SCSRA students were also involved. But bringing together two different groups presented challenges, and while the two organisations worked together, it was necessary to formulate a design strategy that allowed for the needs of the ICT students while still capitalizing on the strengths of the SRCSA members.

III. Design Strategy

Due to the need to incorporate multiple independent teams into the project it was decided to go with a highly distributed and modular system design. Rather than focus on a single high-powered computer, the USV was designed to use multiple low powered embedded systems employing a mix of microcontrollers and Single Board Computers (SBC). Using this architecture, each team or subteam could determine their own hardware and Operating System

requirements, independently generate a solution, and then have that unit incorporated into the overall system structure.

Although this approach creates its own set of problems, one of the advantages of the design is that it allows the USV to employ parts and code from the SRCSA's FRC experience. Accordingly, where possible the intent was to use motors, controllers, electronics, and libraries from FRC, although this limited the robot to the 12 VDC systems employed in that competition. Another advantage to the modular approach was that it permitted multiple teams to work on one subsystem independently. As can be seen in Table 1, GNSS positioning, buoy identification and communications to the Technical Director were viewed as essential subsystems, and thus multiple teams worked on each of these tasks. Each team took their own approach, and the final system will incorporate the most effective design offered. On the negative side, highly distributed systems become heavily dependent on the reliability of the communication network, which requires more attention being paid to onboard communications and the development of internal communication protocols.

One of the goals of the project was to develop an architecture that could be used both in future RobotX competitions and in research projects. This fitted well with the modular design and led to decisions which may not pay off in the first competition, but are hoped to prove valuable in later projects. Although the USV is hoped to be competitive in 2022, it is primarily aimed at future competitions with the 2022 competition employed as an opportunity to learn more about how best to manage the tasks.

Two major limitations had to be considered in the design strategy. The first was that the team has a limited budget, and accordingly price was a significant factor in the decisions. The second was that the UniSA students are exclusively from information technology degrees, so there is a lack of mechanical and electrical engineering students in the team. To some extent the SRCSA members were able to counter this, but nevertheless the overall emphasis was of necessity on software solutions rather than hardware.

Given this situation, the team set out to tackle the competition using three levels of requirements:

- Level 1: "Minimum Viable Product" – a USV that can qualify to enter the competition but will not qualify for the semi-finals.
- Level 2: "Semi-Final Qualifier" – a USV that can show proficiency in at least five tasks.
- Level 3: "Dream Boat" – a UV that has proficiency in more than five tasks.

For level one a set of primary systems were identified:

- Command and control system
- Base station
- Propulsion
- Emergency stop system
- Batteries and power distribution
- Onboard communications
- Shore communications
- Light tower
- Heartbeat (Task 1)
- Basic vision system (identifying buoys)
- Basic navigation (moving between two points).

Assuming that this was successful, this would allow the team to compete. To achieve Level 2, four tasks were identified as viable goals:

- Task 5: Scan the Code
- Task 6: Detect and Dock
- Task 7: Find and Fling
- Task 3: Follow the Path

The first three involve identifying blocks of colour, and prior experience with OpenCV in FRC competitions suggested that this was viable. In addition, FRC competitions regularly involve shooting balls at targets, and while invariably challenging, this meant that there was some experience to build on among the SRCSA team members.

"Follow the Path" required an accurate vision system which was likely to prove to be more difficult to implement that the colour and basic shape recognition in the first three options. To try and handle this complexity, three teams were assigned to the task: an initial UniSA team in late 2021 using OpenCV, a second UniSA team in 2022 employing OpenCV, and a SRCSA-based team using machine learning techniques.

Level 3 was considered but involved some significant complexities. Task 2, "Entrance and Exit Gates" incorporates two independent systems – a method of identifying an underwater beacon and a method of identifying and traversing the gates. Some team members expressed a particular interest in the second part of this, so they continued down that line. However, this was not considered a high priority.

There was also considerable interest within the team in the UAV-based tasks. A UAV was purchased, and several team members were assigned to the UAV. The biggest problem faced, though, was regulatory rather than technical – under Australian law, members of the team are required to hold a Remote Pilot License (RePL) to operate the UAV. [3] This is underway, but development has been necessarily limited until it can be achieved. Nevertheless, team members have been working with the provided Hyper Spectral Camera (HSI) and with data transmission to and from the drone. If nothing else, this will put the team on to the correct path for future competition and research.

IV. Vehicle Design

Based on the design strategy, teams were assigned modules as described in Table 1. A separate team worked on the mechanical and electrical aspects of the USV, but as the focus was largely on information technology, most of the teams were working on software.

Table I

MODULES

Subsystem	Controller	Team
Command and Control	NI RoboRio	SRCSA 1
TeleOp	NI RoboRio	SRCSA 2
Propulsion Control	NI RoboRio	SRCSA 2
Remote eStop	Arduino Uno	SRCSA 3
Shore Comms	Various	UniSA 1
Technical Director Comms	Orange Pi	UniSA 2
Technical Director Comms	Orange Pi	SRCSA 4
GNSS Positioning	Arduino Uno	UniSA 3
GNSS Positioning	Odroid M1	SRCSA 5
Buoys (OpenCV)	Raspberry Pi 4 8GB	UniSA 4
Buoys (TensorFlow)	Jetson Nano	SRCSA 6
Wind Direction/Strength	Arduino Uno	UniSA 5
Shooter Control	Raspberry Pi 4GB	UniSA 5
Shooter Vision	Raspberry Pi 4GB	UniSA 6
Colour Sequence	Raspberry Pi 4GB	UniSA 6
Docking	Odroid C4	UniSA 6
Gate Recognition	Raspberry Pi 4 4GB	UniSA 2
UAV Control	Odroid M1	SRCSA 4
HSI Camera	Raspberry Pi 4 4GB	UniSA 7

The core part of the USV is the RoboRio, which serves as the basis for communications back to the shore and as a command-and-control system for the rest of the modules. The navigation subsystem pulls data from a LiDAR, GPS, and vision. Separate vision systems are employed for colour recognition when docking and identifying the light sequence, and the shooter has its own vision subsystem to identify the target and to instruct the RoboRio on how best to position the USV. The weather module feeds data into navigation, docking and shooting. The heartbeat and the gates were treated as independent systems due to the importance of the heartbeat and the perceived complexity of navigating through the gates. A high-level view of these systems is included in Figure 1.

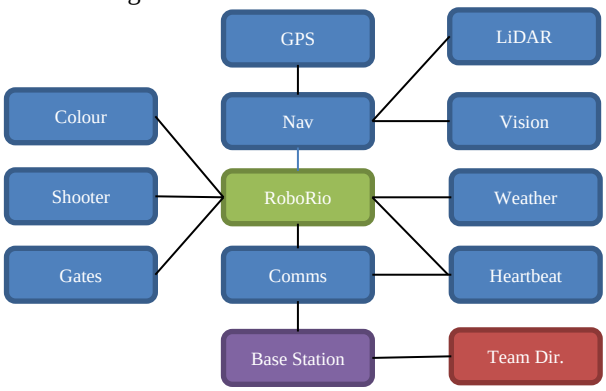

Fig 1. High level system diagram.

A. *Propulsion*

In examining entries from previous years three basic propulsion models stood out: differential drive, where two fixed aft-mounted motors use power control to adjust the direction and speed of the USV; differential drive with bow thrusters, where the bow thrusters are employed for fine control of direction; and omnidirectional drives, where three or four motors can push the robot in any direction. As noted by the Embry-Riddle Aeronautical University and the Queensland University of Technology teams in their 2018 technical reports, position holding and maneuverability with only a differential drive had proven to be insufficient. [4] [5] This matched well with the team's prior experience in FRC where omnidirectional drives are common, and the SRCSA members had recently been involved in designing and programming an omnidirectional drive for that competition. Although the transition from land-based to a water-based solution was unlikely to be trivial, it provided for a solid base from which to start.

There are a number of different omnidirectional designs used in FRC, but many of these cannot make the transition to water. The two best candidates were an "X" drive, where four wheels (or in this case, outboards) are in fixed positions at each corner of the robot, positioned at 45 degrees to make a "X" pattern. This has been employed in RobotX by teams such as the University of Florida. [6] [7] X drives are mechanically straight forward, but they sacrifice power for maneuverability, as by necessity the motors always push against each other when the robot is in motion. An alternative, albeit more complicated solution, is to independently rotate all four motors to change the direction of thrust. This is commonly referred to as "swerve" or "crab" drive. Swerve drives allow the robot to account for current and wind by automatically adjusting power and direction of each motor to maintain a target vector. Given the team's prior experience with swerve drive, it was felt that the added mechanical and software complexity would be countered by the increased maneuverability and no significant loss of power.

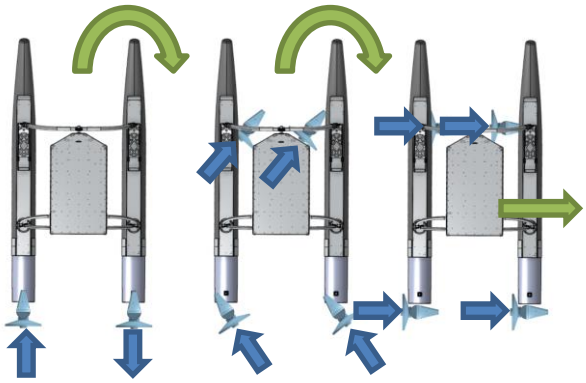

Fig 2. Propulsion systems. On the left is a differential drive. Adjusting power to the two motors allows the robot to turn. The centre and right illustrations are of a swerve drive. By directing the thrust of the four motors the boat can be made to turn or move sideways.

An examination of the motor configurations by prior competitors led to the conclusion that 200lbs thrust would be a good target for the USV. In particular, it was noted that the Harbin Engineering University team, who competed in 2018, had found 120lbs of thrust to be insufficient and had changed to 240lbs. [8] In combination with the decision to use swerve drive and the 12 VDC power limitation, this led to three main options: four Minn Kota RT 55 motors; three Torqeedo

Travel 1103 motors in a triangular configuration; or three Watersnake Advance Brushless Motors with a similar layout. Given budgetary considerations, the three Watersnake motors appeared to provide the best power to price ratio. Unfortunately, ongoing supply shortages meant that these proved to be unavailable, so a fourth option – four Watersnake Venom SXW 54 motors – was selected instead. This met the targeted maximum power of 200lbs but required some modifications to the USV's design.

In FRC swerve drives generally rotate 360 degrees, but that requires the use of a slip ring (as used by the FRC team "Dropbears" [9]) or a bevel gear with the drive motor mounted on top of a vertical shaft (as employed by Swerve Drive Specialties [10]). Neither solution was practical with the Watersnake motors, so an alternative approach has been employed. Rather than spinning the full 360 degrees, each motor is limited to 180 degrees of rotation and the prop rotation is reversed to cover the remaining 180 degrees. This produces less thrust in some directions but was deemed to be an acceptable compromise.

Control of the motors is handled through the RoboRio using a similar approach to what had been employed on the FRC robots. On the hardware level, the manual speed controller was replaced with a Talon SX, which can be controlled through both CAN and PWM. (Figure 3 below). This was like the approach employed by the University of Newcastle in 2018, who used Arduino microcontrollers to control the power output on an otherwise manual system. [11] Rotation was through a Neo 500 brushless motor managed by a Spark Max motor controller. The Neo 500, Spark Max and Talon SRX are all regularly employed in FRC competitions, and the SRCSA team members have considerable experience using all three. (Programming of the RoboRio to manage the swerve drive was completed by one of the current SRCSA students).

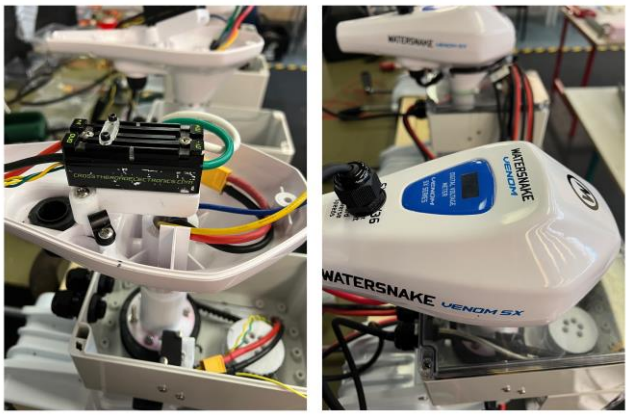

Fig 3. Talon SRX mounted into a Watersnake Venom SWX 54 motor enclosure. By connecting the 12v output from the Talon to the 12v input on the Watersnake the motor was quickly converted from manual to remote control. With the cover replaced, the voltage window allows the operator to see the LED indicator lights on the Talon. Also visible is the 3D-printed pulleys for direction control.

In competition two limit switches at 0 and 180 degrees allow the motor to be automatically calibrated on startup, and when triggered during robot operation they reset the calibration settings to counter any encoder drift or potential slippage – both being significant issues with drives in FRC.

Comments by other teams on the RobotX forums led to the decision to further manage propulsion by ramping the power to the motors, rather than simply changing speed and direction immediately.

B. Control and Power Unit

Power is distributed through two main channels – a custom system to provide 12 VDC to each of the four main motors at a maximum draw of 50A, and a Rev Power Distribution Panel (PDP) to cover everything else. Power to both systems is through two automatic over-current circuit breakers rated to 120A at 12 VDC. Power to the motors is routed through a relay, which can be shut down by any of four emergency stop switches located on the boat or by a remote switch located on shore. Power to each individual motor is then run through individual 50A circuit breakers. The speed controllers for each of the motors was software-limited to 30A, but 50A was the maximum rating.

All other power passes through the PDP. The PDP is the same model used in FRC competitions and includes a CAN connector to report on power usage. The PDP provides multiple 12 VDC channels at 20A and 30A. When 5 VDC is required, it is provided through a Voltage Regulator Module (VRM) connected to the PDP.

An initial electrical configuration was tested on the first of the sea trials, but several concerns emerged. Most notably, the design provided insufficient space for some the planned modules, and the wiring required extensive setup on-site, reducing the time the USV was able to be in the water. Accordingly, much of the electrical system has been redesigned.

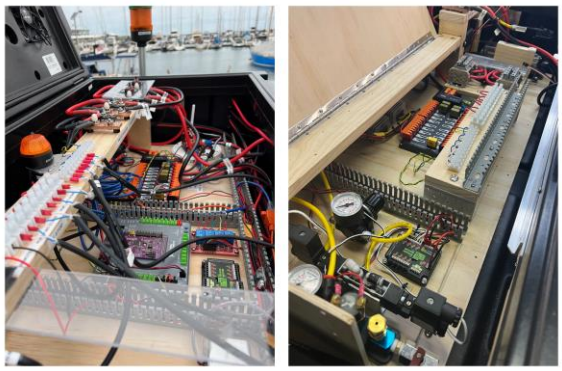

Fig 4. Initial electrical configuration (on the left) with the redesigned electrical system on the right. The new design allows for more components and easier access. The RoboRio and other devices which need to be regularly inspected are mounted on the top of the fold-down shelf and are immediately apparent when the unit is opened.

Power is provided through eight 280Ah 3.2 VDC lithium ferro-phosphate (LiFePO) batteries in two groups of four, providing an overall 12 VDC at 2240Ah. These were coupled with battery controllers capable of providing 150A per set. LiFePO batteries were selected due to the improved safety they offered over LiPo batteries. [12] Due to concerns about

possible supply shortages, the batteries were selected prior to the team gaining a full understanding of the power requirements for the USV. However, it was calculated that the propulsion would need a maximum of 60A per motor, so the batteries were selected to provide sufficient power for the propulsion and should cover any additional electrical systems. In practice, the software limiting of the speed controllers reduced the draw to 40A per motor.

C. Shore Communications

Communications between the robot and the base station are a core requirement. A team of UniSA final year Networking and Cybersecurity students were assigned to the task under the supervision of an industry partner.

It was determined that the networking solution must incorporate 2.4ghz connection and a 5ghz connection. The 5ghz antenna must remain within the Effective Isotropic Radiated Power (EIRP) of 23dBm (200mW) with the 2.4ghz limited to an EIRP of below 36dBm (4W). Accordingly, three antennas were shortlisted for the USV: the Ubiquiti AM-2G16-90, the AM-5G17-90 and the UMA-D. Based on these options, three configurations were examined, outlined in Figures 5, 6 and 7.

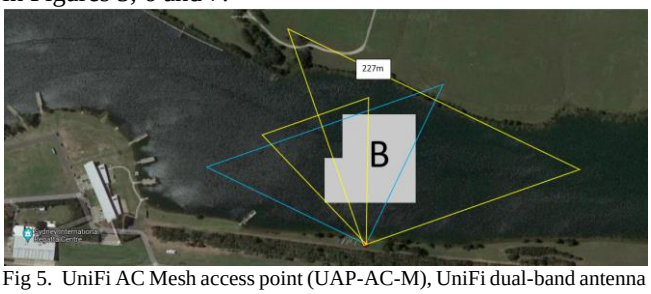

Fig 5. UniFi AC Mesh access point (UAP-AC-M), UniFi dual-band antenna (UMA-D), UniFi Rocket M access point (M5) and airMAX Sector antenna (AM-5G17-90).

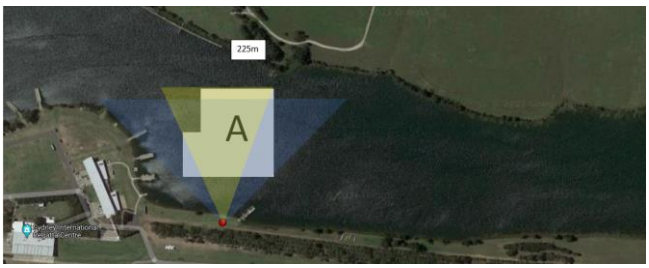

Fig 6. UniFi AC Mesh access point (UAP-AC-M) coupled with UniFi dual-band antenna (UMA-D).

Fig 7. Two UniFi Rocket M access points (M5) with two airMAX sector antennas (AM-2G16-90 and AM-5G17-90).

The chosen solution was the approach in Figure 8: two AirMAX Sector antennas (AM-5G17-90 and AM-2G16-90). this allowed for sufficient coverage of the competition fields and provide 17dBi and 16dbi signal strength. [13]

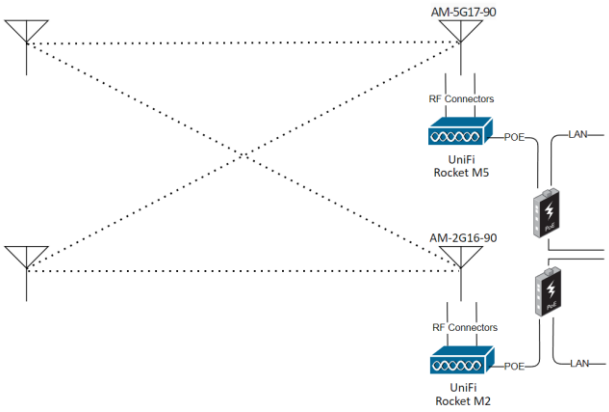

Fig 8. Final networking diagram.

One of the requirements for the competition is that a remote emergency stop (eStop) system be installed. The specific requirements included:

- The eStop must be a physical switch located onshore
- It must transmit to the boat without interference
- It must shut down all power to the motors when triggered
- It needs to function without line-of-sight for 500m
- It must comply with Australian RF transmission regulations

The Semtech SX1276 IC 915MHz LoRa long Range RF Wireless Transceiver module was determined to meet the requirements. Two LoRa shields and two Arduino microcontrollers were sourced. Testing proved that they had the required 500m range without line of sight. In the final system, the shore-based transmitter sends a ping to the receiver at a preset interval. If no pings are received the eStop activates and power to the motors is cut. Alternatively, if a "kill" message is sent by the transmitter (triggered by an emergency stop button) the receiver cuts all power to the motors. To return power a "reset" message needs to be sent from the transmitter before normal operation is resumed.

To reduce possible conflicts, a communication protocol was developed for the LoRa system, with each packet containing a two-byte destination code and a two-byte sender code. If the packet does not include a recognized sender and is not addressed to the receiver it is ignored.

After the first round of sea trials, it was realized that the operators needed to have information about the state of the eStop without needing to inspect it on board. The transmitter was modified as per Figure 9 to have two sets of indicator LEDs. One set indicates the state of the transmitter (sending, in reset mode, or stopped) and the other indicates the state of the receiver (stopped, resetting, running). To facilitate this the receiver on board the USV responds to pings with a status update.

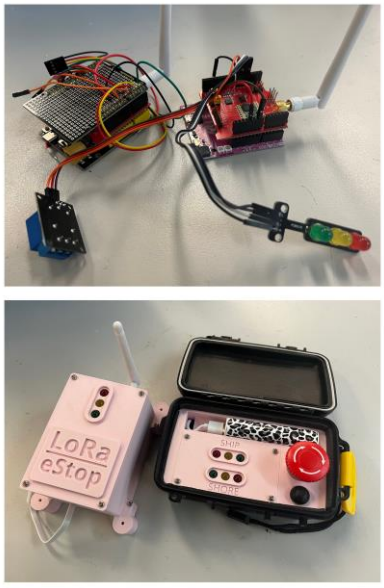

Fig 9. Remote eStop. The top image displays the transmitter (on the left) and the receiver (on the right). Each consists of an Arduino Uno, a LoRa shield, and an assortment of indicator lights, buttons, and a relay. The final product can be seen below, with the receiver (left) to be installed on the USV, while the transmitter (right) is kept on shore.

D. Positioning

Two teams have been working on positioning. One has been trialing GPS modules working with Arduinos and SMBs, while the other has been working to integrate an Emid Edge kit into the USV. [14]

The custom modules have had a degree of success. The initial testing for this task was done using a u-Blox Neo-6MV2 module and an external UFL ceramic antenna. A popular choice amongst hobbyists, it requires relatively low power (100ma) while boasting a horizontal position accuracy of 2.5m. [15] The first prototype was built using a Raspberry Pi 3, a bi-directional logic level converter, an LCD module, and the GPS module. It was powered remotely with a 10000mAh power bank on the underside of the prototype.

Suitable testing locations with wide open areas and a good viewing angle of the sky were selected. The testing device was then placed on the ground for a period of 20-30 minutes, and the data recorded to check for drift. For the first test location an open field was used, and the device placed in the centre of the field. Drift proved to be a significant problem, with the recorded position moving over 40m during the 30-minute test period.

Due to concerns about possible interference from nearby buildings, a second series of tests were conducted over water. These proved to be significantly more reliable, with the drift ranging approximately 5m from the initial position during the test periods. (Example results are shown in Figure 10).

The second tests showed promise, but the drift remained greater than was considered viable for the competition. Three more GPS modules have been sourced and are currently undergoing further testing.

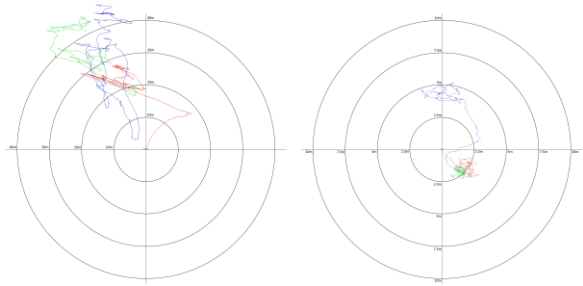

Fig 10. Drift from the custom GOS solution. The image on the left shows the extensive drift that occurred within built-up areas. The colours represent different times during the test and show that the drift did not decrease over time. The second image is the same test device used on water and demonstrates the much more accurate readings that were gained.

The Emid Edge product has provided a different set of problems. While it appears to have greater accuracy than the custom solutions, getting data off the Emid Edge has been a challenge. Currently a SMB is reading data using a CAN bus via the MAVLINK protocol, but more work remains to be done.

E. Onboard Communications

Communications onboard the USV is primarily over the CAN bus. The CAN bus is a common protocol in both maritime and automotive applications and has been part of the FRC competition in recent years. It provides for a robust, easy to implement system consisting of two wires which are connected to each device in sequence. A CAN bus was required to communicate with some of the FRC components on the USV, as well as to address the Emid Edge that was proposed for navigation. Given that a CAN bus was going to be necessary anyway, the decision was made to use it for most onboard communications.

Initially the intention was to use the MAVLINK protocol over the bus, as that was required to integrate with the Emid Edge. However, the FRC components use their own protocol, and running two different protocols over the same bus risked problems. Accordingly, two CAN busses were installed – one for navigation and one for the remaining onboard communications, with a custom CAN router to negotiate the movements between the two. As can be seen in Figure 1, the navigation modules (GPS, LiDAR and one of the vision systems) access the rest of the USV via a navigation module. The navigation module manages data transfer between the two CAN buses.

The use of the FRC protocol means that the priority setting can be retained. Nevertheless, it is only a partial solution, as it has previously been found that it is possible to flood the CAN bus such that collisions can occur with a high frequency. To get around this, all modules listen to the CAN bus, but they only transmit after the Command and Control unit has sent a “start” message, and stop once their role is over. Thus, the shooter module does not transmit unless that task is underway, and it ceases to transmit when shooting is complete. A simple state machine in the Command and Control module manages these activities.

F. Vision: Colour recognition

Three of the tasks in the competition involve colour recognition: Task 3 “Scan the Code”, Task 4 “Detect and Dock” and Task 5 “Find and Fling”. This was tackled by employing OpenCV in Python 3 and was implemented on a Raspberry Pi 4.

The colour recognition code uses HSV thresholding to make it easier to differentiate between colours (red, green, and blue). Converting the pixel colours from RGB to HSV allows the intensity to be separated from the colour components which is much more suitable for computer vision (easier handling of shadows and changes in lighting).

To identify the correct colours, HSV ranges for each target colour must be set. Any HSV values outside of these ranges are to be ignored. The result of thresholding is a black and white image where the coloured object is shown as white, and the surrounding space black. The program is then able to identify the colour based on the white pixels and form contours around each cluster of white pixels.

Like colour recognition, rectangle recognition was developed using OpenCV in Python. The process of rectangle recognition relies on the colour recognition system to identify contours within the video or image. A polygon approximation algorithm is then applied to each of these contours to identify which contours are rectangles. The rectangle with the largest area is selected as it has the highest probability to be the Shape of Interest (SOI).

Given that the SOI will have known width and height it is possible to calculate the distance between the SOI and camera. This is accomplished by calculating the distances relative to the closest (perceived as longest) vertical side and the furthest (perceived as shortest) vertical side of the SOI, and an average is taken. This will be the distance relative to the centre of the SOI. Calculating the angle at which the SOI is being viewed allows the robot to identify its position relative to the SOI. When attempting to centre the camera directly in front of the SOI, if the angle is negative, the robot must shift to the right, and if positive, the robot must shift to the left. The SOI is only visible within +/- 90 degrees from zero (Directly in front of the SOI where its perceived width is maximum); therefore, to calculate the angle, the formula is:

$$\theta = 90 - \frac{percived\ width}{\left(\frac{perceived\ height}{\frac{H}{W}}\right)}{90}$$

Where H and W are the known height and width of the SOI respectively. Perceived height and width use pixels as the unit of measurement. The result of this formula is a positive value, so the negative is applied to the value if the length of the left vertical side is longer than the right.

In testing of this system, it soon became evident that the calculated angle was only accurate when the SOI was in the very centre of the camera frame along the x-axis (y-axis does not have a significant effect on viewing angle). To mitigate this issue, the SOI position from the centre of the frame is calculated, this value indicates to the robot which direction, and how far, it must rotate to keep the SOI centre of frame.

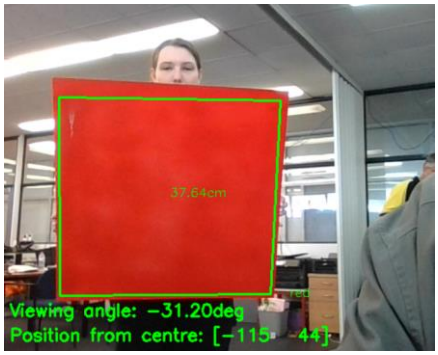

Fig 11. Rectangle recognition.

The colour sequence is captured by identifying the colour displayed on the SOI in each video frame at 10 frames per second to ensure that the colour is captured within the one second that it is showing. This will result in consecutive occurrences of the same colour. These are omitted from the data and the indexes of all blanks (no colour displayed) are retrieved. The sequence will be the data that was captured between the first and second blanks. If the system has captured a sequence that does not have a length of 3, the system will retry capturing. In the “Scan the Code” task, the LED panel will not be a solid colour as it is made up of a matrix of LED lights. To ensure the colour and shape can be recognised, the Gaussian Blur mask may need to be adjusted to ensure the individual LED lights are blended to appear as a solid colour.

G. Vision: Gate recognition

The gates in Task 2: “Entrance and Exit Gates” were treated as a separate task to navigating the buoys, and thus a separate team was assigned to the problem. Like the design for tasks 3, 4 and 5, a colour detection algorithm was selected to determine the relative position of the gates. The gate vision system (which used OpenCV in Python) went through a range of design choices and changes. It started with having a stream class, a vision class and a main class to run the actual program.

The stream class is tasked with handling different type of streams, such as for using a camera, playing a video or loading an image.

Initially, the vision class was to be tasked with handling all the vision system functionality as an entire system on its own, including a colour detector, edge (and contour) detector for shape classification and an object tracker. Though it was later discovered that these subsystems needed to be preloaded as a set of configurations in a specific order, instead of immediately loading them individually, as they each affect the output of the current stream. For example, the colour detector must be loaded first before the edge detector to allow the program to classify the objects appropriately.

While a monolithic system was first considered, it was determined that the design should be adjusted to be more flexible and customizable for the user in the consideration of this program being applied to future endeavours. Therefore, the monolith vision system class was scrapped, and instead a composite based design was implemented, where each vision subsystem (such as the colour detector, edge detector, object tracker etc.) would be separate functional objects. This allows the user to load individual vision subsystems in any order that they desire, allowing for different combinations of configurations.

However, this created considerable complexity, and thus it was decided to further simplify the design and scrapped the conventional composite design in replace of a single class that would be tasked with handling everything related to the stream's configuration, while also preserving customization and flexibility with an unconventional composite design. This class (called stream settings) includes all the vision subsystems as separate functions (and preloads them into a set order) but it also allows the user to configure everything that would assist both users of the system and future developers that may improve on it. Stream settings with the assistance of a threshold class allows the user to set specific colour thresholds, edge thresholds, blur, dilation and erosion and area thresholds, camera and resolution settings, as well as also provides the option to enable or disable trackbars for testing purposes (which provides developers the ability to manually adjust thresholds until the object that they desire to identify is aligned).

The colour detector went through a range of adjustments. It was first tested with household objects with the goal to make it a universal detector that could identify each colour, and this was done with pixel based colour detection. However, this was inefficient and highly inaccurate since if the object that required detection was not completely in the center of the camera, the colour could not be detected correctly.

Therefore, a more refined solution was designed to combine the lower and upper thresholds that defined each colour into a single mask - this would allow the entire stream to ignore all other colours that do not meet these thresholds and proved to be a viable solution in comparison to the original pixel based colour detection. However, when applying this to detect images of the gates it did not perform as expected and so the colour thresholds had to go through further adjustments. It is believed that similarities in the colour hues of the buoys, glare and shadows may be the cause of these inaccuracies.

False positives were further reduced by passing the output of the new colour detector through the edge detector and then the contour detector. The contour detector was tasked with identifying contours of the objects detected by the colour detector and drawing boxes over the objects that were identified, and then also calculating the area, perimeter, and number of points that the object contained by using the x, y, width and height values of the object. This created the ability to single out objects that did not meet the area (and number of corner points detected) criteria of the buoys.

Further problems occurred with the object tracker due to how it is implemented in OpenCV. In OpenCV, the tracker requires the manual use of a user to draw a box over the object that they want to track, and this not only interrupts the rest of the program but also does not fulfill the use case in that the system is supposed to be completely autonomous. Furthermore, when the object that was being tracked went off screen, the tracker could not relocate it when put back on screen. Thus, a second approach was implemented to use the detected boxes from the contour detector as the objects to track. Unfortunately, this strained the performance of the program and there was still a problem of losing track of the object when sent off screen. At the current moment in time, the object tracker is inactive and will remain so until a distance estimator has been implemented to assist with the autonomous tracking process.

To account for the drift of the boat when it is immobile at the start of a gate, a boundary-based position holding implementation is currently being tested. This approach focuses on the idea that further boxes can be drawn over the detected objects to act as a boundary (to provide object avoidance, collision protection and position holding), and the center of the gate, (which the boat must align with), can be identified through the center pixel of the camera. If this focal point moves into the boundaries of the objects, then we can assume that the boat is not centered and therefore we can move it back to the correct position.

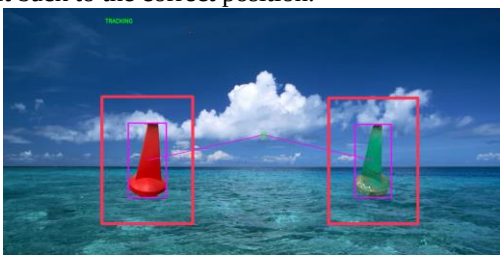

Fig 12. Two buoys identified in OpenCV. By keeping the buoys centered in the image it is hoped that the boat can achieve a high degree of accuracy when autonomously holding position.

If deemed appropriate, it is also planned to use this implementation to measure the distance between the objects and the focal point, to provide a viable distance estimator and therefore also re-implement the object tracker into this subsystem, if required.

H. Buoy Recognition

Although still at an experimental stage, two alternative approaches are being trialed for buoy recognition. One is to use machine learning via TensorFlow on Jetson Nanos. The software has been installed, but this is currently untested.

The second, last minute, option was to include a LiDAR. Given the complexity of using LiDARs a relatively simple model was selected, the RPLIDAR S1-360 Degree TOF Laser Scanner. LiDARs such as the S1-360 use a rotating laser-based time-of-flight sensor to identify objects around

the USV. High quality LiDARs with multiple channels can provide very accurate data, but managing that data adds considerable complexity. Simple LiDARs, on the other hand, can use a single channel and may have difficulty identifying round objects and objects with a low reflectivity, such as black-colour buoys as used in the competition. However, there is considerable promise when combined with a functioning vision system (a LiDAR on its own would be unable to detect the colour of the buoys). For initial testing the LiDAR was fitted to an existing FRC robot at the same height it was intended to be used on the USV, and current testing involves both indoor and outdoor applications.

I. Shooter

Task 7, "Find and Fling", requires the robot to aim and shoot a ball at a goal identified by a coloured square. Shooting has been a common task in FRC competitions, where two primary approaches have emerged:

- Single wheel hooded shooters
- Double wheel shooters

Prior experience with double wheel shooters suggested that they were complex and required extra maintenance yet provided for faster exit speeds. This made aiming easier, but at the same time they tended to have poorer accuracy. Alternatively, hooded single-wheel shooters were simpler to build and sacrificed exit speed and ease of aiming for increased accuracy and topspin. In addition, the SRCSA members have had more experience with hooded shooters, and thus the shooter could be developed more efficiently by building on prior experience. A hooded shooter design was selected.

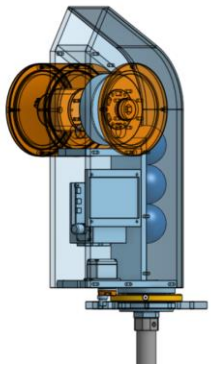

Fig 13. Hooded shooter design as incorporated onto the robot. It uses a pneumatic cylinder to push the balls into the shooter mechanism, which consists of a single wheel driven by a Falcon brushless motor. The design is self-contained and incorporates a Raspberry Pi for shooter control and image recognition, and a small LCD screen to assist with debugging.

J. Drone

The Splash Drone 4, a novel water-friendly drone, was chosen for the Unmanned Aerial Vehicle (UAV) to address concerns that, should control of the UAV be lost, it would not survive contact with water. The Splash Drone 4 provides an API for accepting TCP communication packets over WIFI to the onboard computer. While a consumer-based application does exist for controlling the Splash Drone 4, it was necessary to construct a custom system to issue TCP commands to the drone as required. Unfortunately, the Splash Drone 4 documentation was not polished, having been recently released, obscuring areas of the internal implementation with missing/out-of-date information. An initial investigation into the drone's API involved writing a basic client application to listen for broadcast byte-stream output from the Drone's TCP server and print it to the console, giving a visual representation of drone interaction. Strings of bytes extracted from the console were then cross-referenced with the Splash Drone 4 documentation, verifying that the drone was indeed working. The initial investigation was conducted in both Java and Python but it was quickly decided that building packets to interact with the drone system was much simpler through a low-level language, for which C++ was chosen. Despite an established connection to the drone, it was determined that the received status-packets were incompatible with the updated API documentation and that the drone's firmware needed updating.

What started out as a simple testing program — sending packets to the drone — rapidly morphed into a code library for building and receiving packets from the drone, including unit tests verifying that each packet was built as expected. QT, a library that intends to supplement the C++ standard library, provided memory safety and made the handling of TCP easier. Development, however, was slowed by a lack of access to the drone which was only available on the premises. Consequently, a simple dummy server was written in Java to provide a testing method for those times when the drone was inaccessible.

Despite a (presumably) functional library being built to interact with the UAV, a major requirement for using the drone is a drone pilot license. Once a license is acquired, the framework will be tested against the drone in the field.

K. Heartbeat and Reports

Finally, a heartbeat signal needs to be sent by the robot to the Technical Director (TD) Network along with various reports. The original plan was to employ a dedicated Raspberry Pi 3 to the task, but ultimately an Orange Pi was used due to supply shortages. Given the importance of this task two teams were assigned to it: one developed code using C++, while the other went with a Python solution. Both are still being finalized at the time of writing, but the most reliable of the two solutions will be installed on the robot

With the lessons learnt from sending packets to the UAV, it was realised that the Heartbeat and Messaging System (HMS) should function in a similar fashion, building packets (as defined by the RobotX documentation) and sending them over a TCP client. Since the HMS system would need to forward the UAV and USV statuses (along with other meta-data such as GPS coordinates), it would have to read packets from the rest of the system, develop an internal model of the system's state, and then forward that state to the technical director in the form of heartbeat messages.

The C++ approach was developed after consideration of its similarities to the UAV's control system, as it was thought that the HMS could also leverage C++ and the QT Frameworks that would give low-level control of reading and writing packets and use a test-server, written in Java, that acts as a dummy Technical Directors network to drive rapid development and testing.

L. Experimental Results

Testing to date has involved three different approaches: unit testing, where each module is tested individually under controlled conditions; land testing, where the module is fitted to an existing FRC robot; and on-water testing, where the module is tested as part of the full system. On-water testing to date has been limited due to the complexity of setting up sea trials, but the changes to the electrical system made after the first on-water test will hopefully streamline that process.

Unit testing in controlled environments is good for confirming the behaviour of software, but real-world light conditions can significantly affect the accuracy of data from OpenCV. The team has encountered these problems in FRC competitions, where vision systems work perfectly before the competition, but fail completely once you get there. Nevertheless, they provide a good starting point, and all modules are tested extensively in controlled environments.

One of the strengths that the SRCSA could bring to the project was the ability to trial modules on FRC robots. As both the USV and the FRC robots use swerve drive, a CAN bus and a RoboRio for control, it is possible to get good data from those tests. The most recent testing using this model has involved the LiDAR, which was fitted to an FRC robot to get data as it drove around buoys. This is continuing as the team gets closer to the competition date.

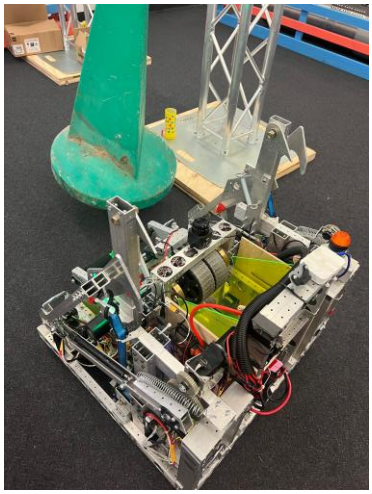

Fig 14. FRC robot with LiDAR fitted for testing. Buoys were positioned at different locations around the room, and the LiDAR data was compared against the actual environment.

The most effective testing, though, has been on-water testing. It is not possible to test the propulsion system on land, and the movement of a USV is difficult if not impossible to replicate in land-based robots. Although both the FRC robots and the USV use swerve drive, the behaviour of that drive on the two systems differs considerably. To address this, the first on-water testing was focused on propulsion. As a result of this test several changes were made to the USV:

- The electrical system was redesigned to reduce setup time for future trials
- Additional indicator lights were fitted to provide the team with data about the USV status, as it was not always clear when the USV was enabled
- Cameras were fitted to assist operators when moving the USV under human control
- The swerve drive was modified so that motors would not automatically calibrate on startup, and could instead be calibrated by operator command
- Mechanical changes were made to improve the fit of the motors on the USV

On the positive side, the USV moved as hoped – there was sufficient thrust to push both the USV and a second boat tether to the side; the swerve drive provided precise control of direction and speed; and power consumption was well within predicted levels.

The remaining on-water trials are to test position holding, navigation and the vision system.

Fig 15. USV undergoing on-water testing at the Royal Yacht Squadron of South Australia. The swerve drive, which consists of the four outboards and the associated turning systems, met all performance targets.

M. Conclusion

The RoboSeals would like to take this opportunity to acknowledge the support from both the University of South Australia and the Student Robotics Club of South Australia, Inc. We would also especially like to thank Intrusion Inc for their support, the Royal South Australian Yacht Squadron for providing a place in which to test the USV, and mentors Jeff Jenkins, Sam Koulianos and Adrian Johnson for the many, many hours they put into helping us to reach this point. Finally, we would like to acknowledge the assistance of Peter Ryan-Kane from the SRCSA and Richard Bowyer from Flinders University, whose encouragement led us down this path.

V. References

[1] RoboNation, "RobotX Challenge: 2022 Team Handbook," 2 2022. [Online]. Available: https://robonation.org/app/uploads/sites/2/2022/03/2022-RobotX_Team-Handbook_v2.pdf. [Accessed 8 10 2022].

[2] Warman Design and Build Challenge, "International Final," 2022. [Online]. Available: https://warmandesignandbuild.org.au/national-final/. [Accessed 6 10 2022].

[3] "Operator accreditation," Civil Aviation Saftey Authority, [Online]. Available: https://www.casa.gov.au/drones/get-your-operator-credentials/operator-accreditation. [Accessed 5 10 2022].

[4] J. E. Barnes, N. D. Bloom, S. P. Cronin, G. C. Delp, J. L. Halleran, M. R. Helms, J. J. Hendrickson, N. R. Middlebrooks, N. D. Moline, J. B. Near III, J. S. Romney, M. A. Schoener, N. C. Schultz, D. J. Thompson, T. A. Zuercher, C. F. Reinholtz, E. J. Coyle, P. N. Currier, B. K. Butka and C. J. Hockley, "Design of the Minion Research Platform for the 2018 Maritime RobotX Challenge," 2018. [Online]. Available: https://robonation.org/app/uploads/sites/2/2019/09/ERAU_RX18_Paper.pdf. [Accessed 5 10 2022].

[5] L. Stanislas, K. Moyle, E. Corser, T. Ha, R. Dyson, R. Lamont and M. Dunbabin, "Bruce: A system-of-systems solution to the 2018 Maritime RobotX Challenge," 2018. [Online]. Available: https://robonation.org/app/uploads/sites/2/2019/09/QUT_RX18_Paper.pdf. [Accessed 5 10 2022].

[6] D. Frank, A. Gray, K. Allen, T. Bianchi, K. Cohen, D. Dugger, J. Easterling, M. Frank, A. Gray, K. Allen, T. Bianchi, K. Cohen, D. Dugger, J. Easterling, M. Peterson, D. Soto, F. Voight, D. Volya, T. Williams, E. Schwartz, C. Crane, I. Hill, S. Ridgeway, "University of Florida: Team NaviGator AMS," 2016. [Online]. Available: https://robonation.org/app/uploads/sites/2/2019/09/UF_RX16_Paper.pdf. [Accessed 5 10 2022].

[7] V. D., M. Griessler, A. Kevin, K. Cohen, A. Albritton, N. Suhlman, D. Zobel, J. Mejia, R. Fabien, M. Rawson, J. Brown, R. Pendon, D. Olis, M. Jones, E. Schwartz, C. Crane and S. Ridgeway, "NaviGator AMS 2018," 2018. [Online]. Available: https://robonation.org/app/uploads/sites/2/2019/09/UFlorida_RX18_Paper.pdf. [Accessed 5 10 2022].

[8] G. Su, B. Zhou, J. Feng, C. Zhu, K. Jiang, Z. Yuan, W. Zhang, S. Zheng, L. Zhang, B. Wang and J. Zhuang, "Design and Implementation of HEU Heading for the 2018 Maritime RobotX Challenge," 2018. [Online]. Available: https://robonation.org/app/uploads/sites/2/2019/09/Harbin_RX18_Paper.pdf. [Accessed 5 10 2022].

[9] "Our Custom Swerve Drive Module," The Drop Bears, [Online]. Available: https://www.thedropbears.org.au/swerve-drive/. [Accessed 5 10 2022].

[10] Swerve Drive Specialties, "MK4 Swerve Module," 2021. [Online]. Available: https://www.swervedrivespecialties.com/products/mk4-swerve-module. [Accessed 5 10 2022].

[11] J. A. Coller, M. J. Sypnewski, S. B. Taylordean, C. J. Goodrum and D. J. Singer, "The Design of an Autonomous Surface Vehicle for the 2018 Maritime RobotX Challenge," 2018. [Online]. Available: https://robonation.org/app/uploads/sites/2/2019/09/UMich_RX18_Paper.pdf. [Accessed 5 10 2022].

[12] J. Hu , W. Huang , L. Yang and F. Pan , "Structure and performance of the LiFePO4 cathode material: from the bulk to the surface," *Nanoscale*, vol. 12, pp. 15036-15044, 2020.

[13] Ubiquiti Networks, "airMax Sector: Datasheet," 2018. [Online]. Available: https://dl.ui.com/datasheets/airmaxsector/airMAX_Sector_Antennas_DS.pdf. [Accessed 4 10 2022].

[14] MapGear, "Emlid Edge Kit with Wi-Fi," [Online]. Available: https://www.mapgear.com.au/product/emlid-edge-kit-with-wi-fi/. [Accessed 3 10 2022].

[15] u-blox, "NEO-6," 2011. [Online]. Available: https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf. [Accessed 7 10 2022].