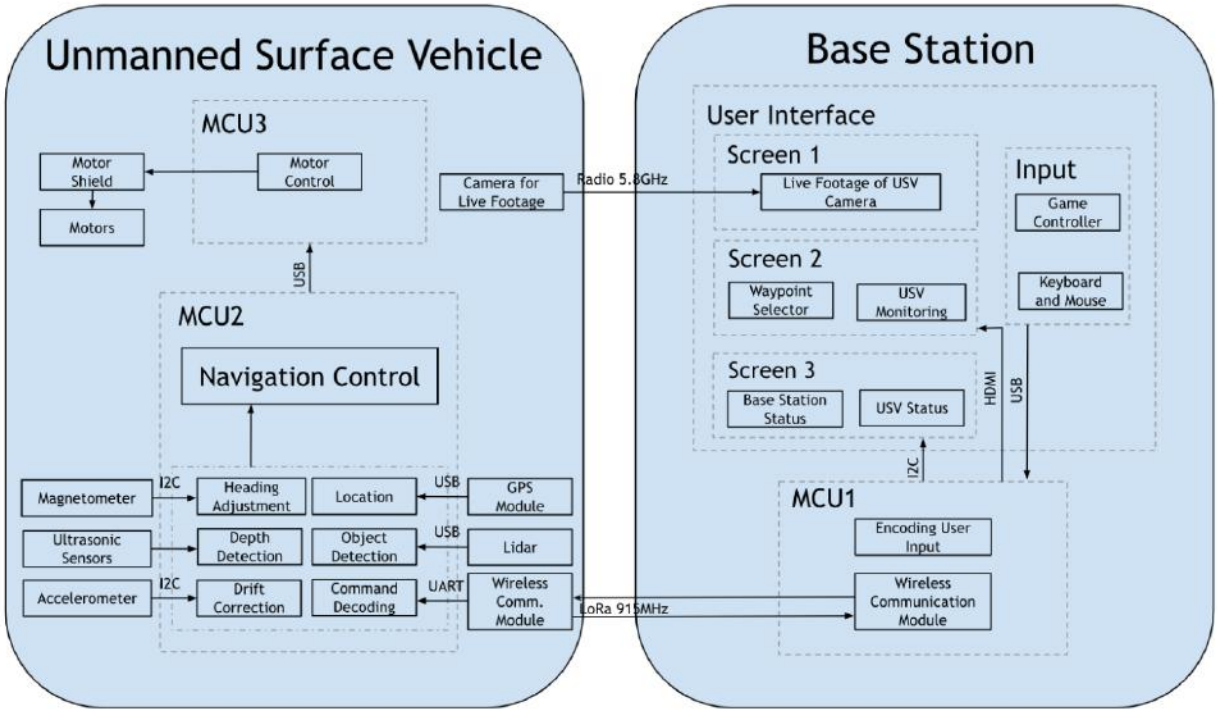
EE175AB Final Report
Unmanned Surface Vehicle

EE 175AB Final Report

Department of Electrical Engineering, UC Riverside

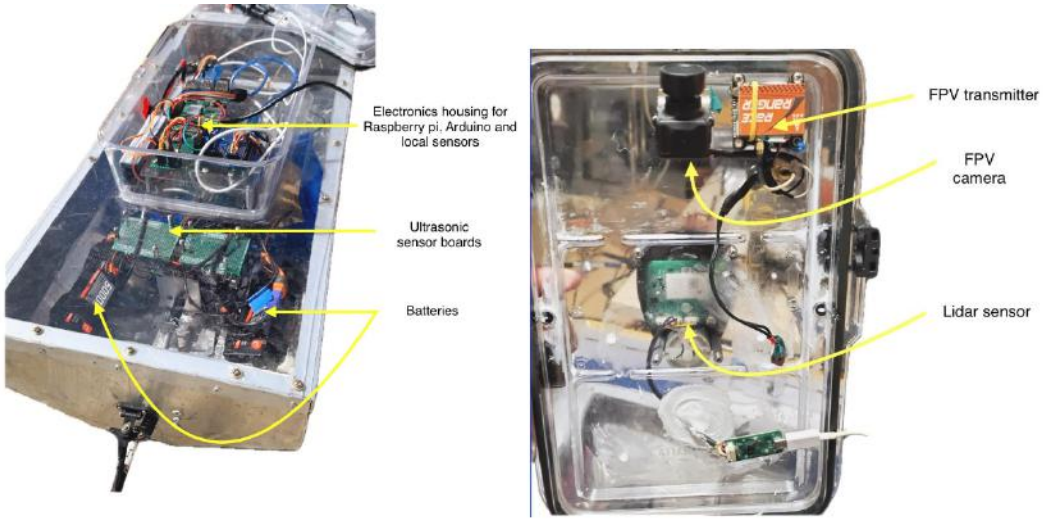
Project Team Member(s)	Maxwell Curren, Daniel Gutierrez, Alyan Gillett
Date Submitted	3/19/2024
Section Professor	Dr. Roman Chomko
Revision	Version 1
URL of Project YouTube Videos, Wiki/Webpage	Playlist of Demo Videos Github Repository
Permanent Emails	gillettab@gmail.com, dasbro73@gmail.com, maxcurren02@gmail.com

System Block Diagram

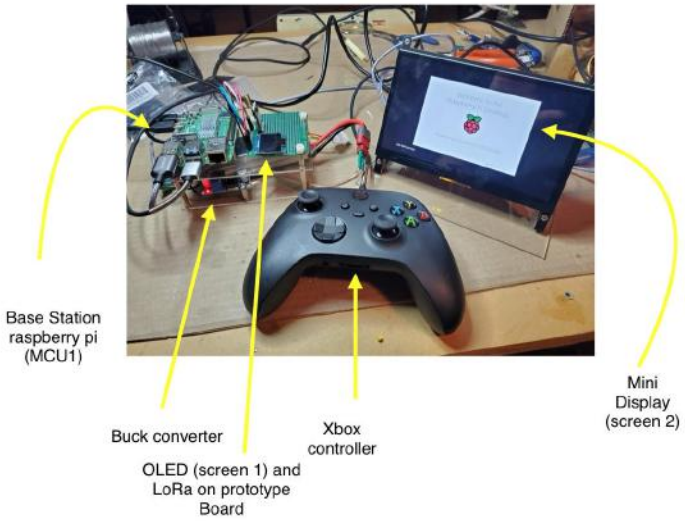


Labeled System

USV:



Base Station:



Revisions

Version	Description of Version	Author(s)	Date Completed	Approval
Version Number	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	Full Name	00/00/00	
1.0	First prototype	Max Curren Daniel Gutierrez Alyan Gillet	02/01/2023	

Table of Contents

1 * Executive Summary

The Unmanned Surface Vehicle (USV) project aimed to develop an autonomous boat capable of navigating to waypoints, avoiding obstacles, and collecting survey data. The project's motivation was fueled by the need for an affordable, safe, and efficient solution for wildlife monitoring, search and rescue, freight transportation, and even data collection.

The design objective and the overall goals included the integration of sensors for perception, location, and communication, the use of microcontrollers for motion control and navigation, and meeting specific operational requirements such as following waypoints, sensing objects, water depth, and operating the USV within specified ranges. Additionally, while it would have been easier to mount the electronics on a boogie board or just attach pool noodles to the tackle box, we aimed for a design that more closely resembled a traditional boat hull with a deck. Nonetheless, this decision was made to ensure that the USV not only functioned effectively but also behaved like a real boat.

Key features of the USV that ensured an efficient and reliable operation first included our autonomous navigation. The USV, while having manual mode, was designed with autonomous navigation capabilities where the user can set predefined waypoints at the base station and observe as it goes to the desired locations. The autonomy was further enhanced by the obstacle detection and response which allows the boat to detect and avoid objects within its path. Moreover, the USV also has long range communication with a base station for data interpretation and signal commands. This feature was crucial for real-time monitoring, controlling the USV, as well as for transmitting collected data back to the base station for analysis. For performance specifications the USV was also designed to have a maximum range of 2 kilometers and provide a clear view in front of the USV up to a range of 1 kilometer. The USV is also capable of operating at a speed of 1 meter per second, ensuring timely and efficient navigation to the waypoints.

Even with a good design we faced many challenges during testing such as waterproofing the electronics, the boat, fine-tuning control algorithms, and ensuring reliable communication in complex environments. Despite these challenges, we successfully demonstrated the feasibility and effectiveness of our USV design.

Our important achievements included integrating various sensors and control systems into a functional USV, optimizing communication between components, and demonstrating the potential for real-world applications in environmental monitoring and data collection. Moving forward, we aim to further develop our USV by incorporating more advanced obstacle avoidance algorithms, integrating machine learning for decision-making, and improving communication models for longer ranges of environmental monitoring.

2 * Introduction

2.1 * Design Objectives and System Overview

The concept and application of the USV are based on our goals of the design being affordable, safe, and efficient, to enable freight transportation, wildlife monitoring, search and rescue operations, and data collection. The USV is intended to be used for tasks that are hazardous or impractical for manned vessels, such as collecting data in remote or dangerous waters, monitoring marine life, and conducting underwater surveys. This project is closely related to many subjects in electrical engineering like control systems, signal processing, embedded systems, wireless communication, and power systems. It involves designing and integrating hardware components such as sensors, microcontrollers, and communication modules, as well as developing software algorithms for autonomous navigation and data collection. The overall goals of the project are to design a reliable and efficient autonomous USV capable of navigating to predefined waypoints, avoiding obstacles, collecting data using onboard sensors, and transmitting data from the sensors, as well as live or recorded video footage of the boat's driving path, back to a Base Station.

The USV consists of an aluminum hull mounted with a brushless motor and servo motor, housing the electronic hardware, including Lidar, GPS, ultrasonic sensors, accelerometer, magnetometer, LoRa, FPV camera, battery, and MCUs. The system interacts with a Base Station for waypoint setting through an LED screen for it to begin calculating its destination path. It operates in a marine environment where it uses Lidar and ultrasonic sensors for obstacle detection and mapping, GPS for location tracking, ultrasonic sensors for close-range obstacle detection, an accelerometer for measuring acceleration forces, a magnetometer for orientation, a LoRa module for communication with the Base Station, an FPV camera for live or recorded video footage, and a battery for power. Microcontroller units (MCUs) process sensor data and user commands, generating motor control signals for navigation and sending sensor readings to the Base Station.

Sensors and components assigned to each team member with quantitative technical design objectives:
Our overall budget is $1000, and our goal is to complete the project within this budget.

Alyan:

- Lidar Accuracy: 95% within a range of 10 meters.
- Ultrasonic Sensor Accuracy: 90% within a range of 0.1778 meters to 3.81 meters .
- Accelerometer Accuracy: 95% for measuring accelerations up to ±2g.
- Magnetometer Accuracy: 95% for measuring magnetic fields in a range of ±2 Gauss.

Max:

- PD Response Time: Less than 10 milliseconds for adjusting motor control based on sensor inputs.
- LoRa Communication Range: Transmission range of at least 2 kilometers.
- LoRa Communication Speed: Less than 3 seconds
- Waypoint Creation/GUI: User-friendly graphical interface for setting waypoints.
- GPS Module Precision: Within 10 meters for location tracking.
- Waypoint Precision: Must be within 10 meters of waypoint before tracking next

Daniel:

- Control Interface Response Time: Less than 50 milliseconds for processing user commands.
- FPV Video Data Transmission: Real-time video transmission with a range ½ kilometer.
- Servo and Motor Control: spec a power system for under 50 watts powerful enough to navigate moderate currents.
- Power Management Efficiency: Ensure power consumption is optimized for long-duration operation.
- Aluminum Hull and Sensor Mounting: Ensure the hull is robust and waterproof, with secure mounting for sensors.

2.2 * Backgrounds and Prior Art

Our literature and internet search revealed several existing products and designs for autonomous Unmanned Surface Vehicles (USVs) with similar objectives to our design. One product is the Sailbuoy, developed by Offshore Sensing AS, which is designed for ocean monitoring and research applications. The Sailbuoy features autonomous operation, environmental sensors for data collection, and satellite communication for remote operation and data transmission. A second product is the "Echoboat" by Echobot which is a small, autonomous boat equipped with sonar systems for bathymetric surveys, water quality sensors for monitoring various parameters such as temperature, pH, and dissolved oxygen, and GPS for accurate positioning. It is designed to be easily deployable and can operate for extended periods, making it suitable for detailed lake surveys. Another example of a USV suitable for lake surveying is the "Self-Driving Boat " project based on ArduPilot Rover. This project demonstrates how to build an autonomous boat using the ArduPilot open-source autopilot system. The boat is equipped with GPS for navigation, a compass for heading control, and sonar sensors for obstacle avoidance.

The advantages of our design compared to existing products include its affordability, as our budget is less than $1000, making it more accessible for research and educational purposes. Additionally, our USV is designed to be modular and customizable, allowing for easy integration of different sensors and components based on specific project requirements.

The shortcomings of our design compared to existing products include lower endurance and a shorter range of communication due to budget constraints. Additionally, our USV does not have the same level of sophistication or sensor capabilities as the commercial products, which can impact performance in certain applications.

2.3 * Development Environment and Tools

The Unmanned Surface Vehicle (USV) project design environment was an electrical engineering maker space studio. This studio provided access to a wide range of tools and equipment necessary for building and prototyping the USV. It had workbenches for assembling the USV, soldering stations for soldering electronic components, and Multimeters for current, voltage, and resistance. Additionally, the maker space would have a variety of electronic components and sensors needed for the USV.

In our USV project, for our hardware, we utilized the Raspberry Pi for high-level control, communication, and data processing. The Raspberry Pi's processing power and ability to run a full operating system

(Rasbian) made it ideal for tasks that required more computational resources, such as managing navigation algorithms, sensor fusion, and communication with the Base Station. Conversely, we used Arduino for low-level control, specifically for sending signals related to direction and propulsion to the motor. Other parts of the hardware were the sensors, motors, communication module, power supply, and hull. The software tools include things like the use of Rasbian for our Raspberry Pi operating system with Python as our preferred programming language and Arduino IDE for Arduino. Libraries include Pigpiod for GPIO access, the SMBus module to read and write data, adafruit RP Lidar for angle and distance values of a space, and the servo library for motor control.

For the process of testing the electronic components of the USV to ensure their functionality and reliability we started by making sure the tackle box containing the electronics was waterproof so we submerged it in water to test its waterproofing. Next, Using a multimeter, we checked the voltage supply from the batteries and used voltage dividers to verify the right amounts were being distributed to the MCUs, sensors, and motors. Depth testing was conducted in a pool to validate the depth sensor's accuracy. In a controlled environment, the obstacle detection system was tested by moving objects in the USV's path. The accelerometer was tested by checking the change in gravity as the sensor tilts in different directions. The compass range of magnetic interference was tested by using a magnet and seeing how close we could get before skewing the data. The GPS was tested by driving to different locations and ensuring it matched the actual coordinates of the location. The servo was tested to make sure it could rotate roughly 180 degrees through software testing as well as the power sent to the motor speed. Finally, the FPV camera and LoRa module were tested for their ability to transmit video feed and data through building structures. These comprehensive tests ensured that the USV's electronic systems met design specifications and were capable of operating reliably in real-world conditions.

2.4 * Related Documents and Supporting Materials

- The Federal Communications Commission (FCC) regulates the use of radio frequency devices, including wireless communication modules like LoRa.
- The use of cameras for surveillance on government-owned property needed approval based on the California Consumer Privacy Act (CCPA).
- The Local Lake Perris Regulations: Specific rules and regulations established by the management authority for Lake Perris, which include restrictions on the operation of the USVs, speed limits, and designated operating areas.

2.5 * Definitions and Acronyms

USV: Unmanned Surface Vehicle

PID: Proportional-Integral-Derivative

PD: Proportional-Derivative

Lidar: Light Detection and Ranging sensor

LoRa: Long Range Communication Protocol

RSSI: Received Signal Strength

GPS: Global Positioning System

USB: Universal Serial Bus

UART: Universal Asynchronous Receiver Transmitter

I2C: Inter-Integrated Circuit

FPV: First Person View

MCUs: Microcontroller Units

GPIO: General Purpose Input/Output

PWM: Pulse Width Modulation

RPM: Revolutions Per Minute

ESC: Electronic Speed Controller

BEC: Battery Eliminator Circuit

Kv: constant velocity per volt applied to a motor. I.E rpm per volt applied

D-Pad: a set of four face buttons on the left-hand side of the controller device

3 * Design Considerations

3.1 * Realistic Constraints

The realistic constraints that needed to be addressed, resolved, and completed, along with other constraints influencing the design process of our USV, included:

- Power Consumption Constraints: Due to the limited power supply on board the USV, we must design the system to operate efficiently and minimize power consumption to ensure at least an hour of run time.

- Voltage/Current Supply Constraints: The USV's components, such as sensors, microcontrollers, and communication modules, must operate within specified voltage and current limits to prevent any damage and return optimal performance.

- Processor Speed/Memory Size Constraints: The processing capabilities and memory size of the microcontrollers used in the USV limit the complexity of algorithms and the amount of data that can be processed simultaneously.

- Motor Speed Constraints: The speed of the motors driving the USV must be controlled within specific limits to ensure safe and efficient operation and to avoid damage to mechanical components.

- Wireless Transmission Power and Range Constraints: The communication range of the LoRa module and FPV camera must provide sufficient range for reliable communication with the Base Station.

- Performance Requirements: The USV must achieve precise waypoint navigation, effective obstacle avoidance, and accurate sensor data to fulfill its intended applications effectively.

- Weight/Size Constraints: The USV's design must consider weight and size limitations to remain portable, transportable, and easily deployable.

- Project Time and Budget Constraints: The project must be completed within a specified timeframe and budget, requiring careful planning and resource management.

- Government Regulations and Legal Constraints: The design must comply with relevant laws and regulations governing autonomous vehicles for approved operation.

- Societal and Environmental Constraints: The USV's design and operation must consider societal and environmental impacts, such as noise levels, emissions, and interactions with wildlife and habitats.

- Requirements of Industry Standards: The design must adhere to relevant industry standards for electronics, and communication systems to ensure compatibility.

- Waterproofing Constraints: The USV's electronics and components must be adequately waterproofed to ensure reliable operation in marine environments.

- Finding Testing Locations Constraints: Suitable testing locations are needed to validate the USV's performance under real-world conditions.

- User Input Constraints: The USV's control interface must be user-friendly and capable of receiving input from operators for manual control or setting parameters.

- Proportional-Derivative (PD) Tuning Constraints: tuning parameters must be optimized to ensure smooth and stable control of the USV's motors and actuators.

- Sensor Calibration Constraints: Sensors must be calibrated accurately to provide reliable data for navigation, obstacle avoidance, and environmental monitoring.

3.2 System Environment and External Interfaces

The Unmanned Surface Vehicle (USV) must operate in a marine/aquatic environment as it interacts with various hardware and software components. The hardware includes a brush motor and stepper motor for propulsion, a Raspberry Pi for high-level control and data processing, an Arduino for low-level motor control, and electronic components such as Lidar, GPS, ultrasonic sensors, accelerometer, magnetometer, LoRa module, FPV camera, and MCUs. These components must communicate using protocols such as UART for serial communication, PWM for motor control, I2C for sensor communication, and LoRa for long-range wireless communication. The software must include control algorithms for waypoint navigation and obstacle avoidance, as well as interfaces for user input and monitoring. APIs such as RPi.GPIO for GPIO control on the Raspberry Pi, pigpio for advanced GPIO control, Servo and Wire libraries for Arduino, and other custom APIs for sensor interfacing and communication will define how different software components interact with each other and with the hardware components of the USV

3.3 * Industry Standards

Several industry standards and protocols are relevant to the hardware and software design. The LoRa standard regulates the use of the 915MHz frequency band for long-range wireless communication according to the Federal Communications Commission(FCC), which we use for the communication between the USV and the Base Station. The California Consumer Privacy Act (CCPA) governs the collection and use of personal information, potentially impacting surveillance activities conducted by the USV, especially on government-owned property. Additionally, Lakes in California, Regulations specify rules for operating USVs on the water, including speed limits and designated operating areas, which we must comply with. We used the I2C protocol for communication between the Raspberry Pi and sensors like the accelerometer and magnetometer, ensuring proper data transfer rates and packet sizes. USB to UART communication standards are followed for connecting peripherals such as the GPS module and the Lidar to the Raspberry Pi, for proper connection and reliable data processing. UART communication is used for the LoRa module communication between the Raspberry Pi and Arduino, complying with standards. Lastly, safety standards are also considered, including waterproofing the electronics, using safer batteries, and ensuring the safe operation of the USV in various environmental conditions.

3.4 * Knowledge and Skills

The project requires knowledge of electrical engineering, particularly in sensor integration, control systems, communication protocols, and software development, along with skills in system integration, sensor calibration, autonomous navigation algorithms, and practical abilities in soldering and circuit board assembly for hardware implementation.

Alyan:

Electrical Engineering Courses: Control Systems, Communication Systems, Digital Signal Processing, Microcontrollers, Logic Design, Embedded Systems.

New Knowledge and Skills: Lidar sensor integration, Ultrasonic sensor integration, Accelerometer and magnetometer calibration, usage for autonomous navigation algorithms, python programming, Arduino Ide, circuit board design.

Max:

Electrical Engineering Courses: Control Systems, Communication Systems, Digital Signal Processing, Microcontrollers, Logic Design, Embedded Systems.

New Knowledge and Skills: PID tuning, waypoint creation/GUI development, GPS module integration, full software integration, LoRa communication range optimization, python programming, Arduino Ide, circuit board design.

Daniel:

Electrical Engineering Courses: Control Systems, Communication Systems, Digital Signal Processing, Microcontrollers, Logic Design, Embedded Systems.

New Knowledge and Skills: control interface design, LoRa communication range optimization, FPV video transmission, servo, and motor control, power management optimization, waterproofing techniques, sensor mounting on an aluminum hull, python programming, circuit board design.

3.5 * Budget and Cost Analysis

Total Overall Budget: $1000
Total Cost: $961.55

Item	Price
Raspberry Pi 4 model B(2x)	$152.80
Arduino uno(1x)	$20.00
Lidar A1M8(1x)	$100.00
Accelerometer(3x)	$15.61
JSN-SR04T ultrasonic sensor(6x)	$36.62
LoRa Modules(2x)	$41.32
LoRa Module UART to serial Adapter(1x)	$9.79
Xbox Controller(1x)	$59.80
Propeller Parts(1x)	$45.78
Boat Hull Parts(1x)	$180.00
Buck Converter(5x)	$8.00
OLED screen(5x)	$15.00
Heat Sink(1x)	$8.00
Epoxy(1x)	$10.00
FPV Camera(1x)	$137.00
Magnetometer(3x)	$10.00
GPS(2x)	$18.00
Mini Touchscreen Monitor (1x)	$44.83
Cables Wires(2x)	$14.00
Total	$961.55

3.6 * Safety

Safety considerations are an essential part of designing anything in electrical engineering, especially for an Unmanned Surface Vehicle (USV). The reason for this is to ensure the protection of people, property, and the environment that we are operating in. The safety objectives we aimed to achieve first was to prioritize electrical safety by ensuring that all electrical components are properly insulated, grounded, and meet safety standards to prevent electrical hazards. Additionally, we focus on water safety by waterproofing all electronic components and ensuring the hull is sealed to prevent water ingress and damage. Collision avoidance is another critical safety objective, for which we implement sensors and algorithms to detect obstacles and avoid collisions with other vessels or objects. Navigation safety is also a priority, and we aim to develop precise navigation systems to accurately reach waypoint destinations. Furthermore, we implement an emergency stop mechanism that can quickly halt the USV's operation in case of malfunction or danger. To mitigate the impact on wildlife and the environment, we implement measures to minimize noise and pollution emissions. Finally, compliance with relevant industry standards and regulations, such as FCC regulations for radio frequency emissions and safety standards for marine vehicles, is a key focus to ensure the USV's operation is legal and safe.

3.7 Performance, Security, Quality, Reliability, Aesthetics etc.

To ensure meeting performance, security, quality control, reliability, aesthetics, and other requirements, several considerations and processes were implemented in our Unmanned Surface Vehicle (USV) project:

Performance: Propulsion Efficiency was our starting point where we meticulously designed the hull shape and propulsion system for an optimal efficiency to achieve the desired speed and cargo capacity. Power Management was another important aspect that was a concern so we implemented an efficient power management system like the use of voltage dividers and buck converters to ensure the USV could operate for extended periods without frequent recharging or battery changes. Sensor fusion was crucial in improving navigation accuracy and obstacle detection so we tested and calibrated rigorously through different settings to see what form of data our sensors were reading. Additionally, we developed and tested Autonomous Navigation Algorithms for waypoint following and obstacle avoidance by going out to the lake or the pool to detect and record the movement of the USV. Lastly, Environmental Adaptation was considered, and the USV was designed to operate effectively in various environmental conditions, including different water depths and temperatures.

Security: Non-Applicable

Quality Control: Conducted rigorous testing of our components (e.g., sensors, motors, communication modules) to ensure they meet specifications. Tested the integration of all components to check if they function together. Validated the performance of the USV under various operating conditions to ensure it meets design requirements. Through the reading of the serial monitor on the Raspberry Pi, we could continuously monitor the USV's performance and calculations.

Reliability: We did have a bag of extra components to continue operation in case of component failure. We also programmed the USV's software and hardware to be fault-tolerant, capable of detecting and recovering from errors such as missed scans or a bugged value. Lastly, we tested the USV's reliability under various environmental conditions (temperature, humidity, and water conditions).

Aesthetics: For our design approach we wanted the integration of electronic components within the USV's structure to have a visually appealing appearance resembling that of a conventional boat similar to the basement engineering design for their ardupilot rover[3]. The choice of aluminum for the hull was not only guided by its color compatibility but also by its durability, lightweight nature, corrosion resistance, and recyclability. Emphasis was placed on minimizing visual clutter and maintaining a sleek design. The utilization of a tackle box on the hull and the plexiglass deck served dual purposes, aiding in waterproofing and facilitating efficient wire management. Additionally, careful consideration was given to the USV's size to ensure ease of handling without posing risks to individuals or disrupting marine life.

3.8 * Documentation

We utilized a variety of methods to generate and maintain technical and user documentation for the project. One key approach was the use of GitHub, where we maintained a detailed repository containing all relevant documentation. This included step-by-step guides on how to use the project, descriptions of required libraries, and pinout diagrams. Additionally, we implemented weekly reports to track our progress. These reports documented the work completed each week, the progress made, and outlined plans for the following week. This helped us stay organized and ensured that everyone involved was aware of the project's status and goals.

3.9 Risks and Volatile Areas

One notably volatile area of the system is the fully integrated algorithm for autonomous navigation, including waypoint following and obstacle avoidance. This algorithm was crucial for the USV to navigate safely and effectively, but it posed challenges due to the complex nature of marine environments and the need for real-time decision-making. To mitigate risks, we employed several strategies: extensive testing, including simulation and real-world scenarios; implementation of redundant sensors and safety checks; continuous improvement through iterative refinement based on test results and feedback; the addition of manual control at startup and if autonomous mode fails; and integration of fail-safe mechanisms to override the autonomous navigation system in emergencies. These strategies helped ensure the safe and reliable operation of the USV despite the volatile nature of the algorithm.

4 * Experiment Design and Feasibility Study

4.1 * Experiment Design

Since the USV needs to have a sufficiently large enough body of water to test in, it was not possible to leave and set up at the lake for each test. For this reason there needed to be a way to verify that the individual modules were ready for water testing at the lake. For this reason testing was separated into two types of tests: water and bench testing.

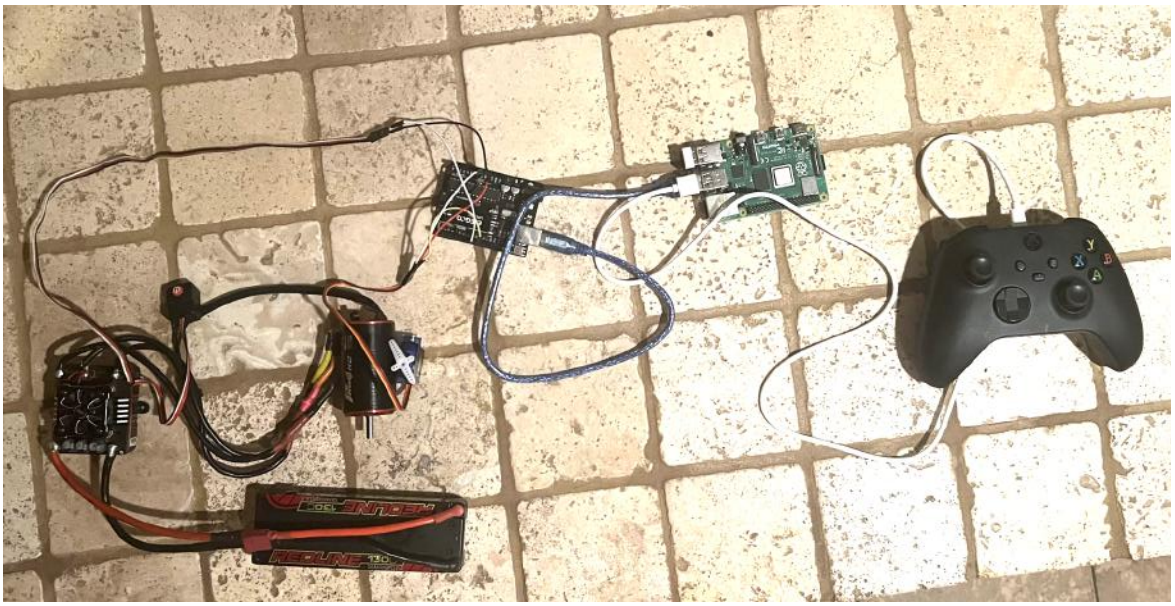
Bench testing

Bench testing was used to test sensors, control parts, and some features of the PD controller to get them ready for water testing. This allowed the team to individually test the components they were working on to speed up design time.

Controller test (Daniel)

To test the controller inputs from the Xbox controller, the remote was wired directly to the USV Raspberry Pi (MCU2) connected to the Arduino (MCU 3). The ESC and servo were then attached to test the control output. This way the control input output system would be tuned prior to implementing the LoRa communication.

Fig 4.1a bench test of controller outputs



Sensor testing(Alyan)

The goal of the sensor testing was to determine if the data outputted from the Lidar and ultrasonic sensors were feasible enough to be used for obstacle detection and avoidance on the USV. The setup I used to test the ultrasonic sensors included first checking if the sensors were waterproof to which I would try it in different aquatic mediums to also determine if the sound would transmit and receive. Finally, I would check the speed in different mediums in order to make changes in my code to the distance as sound travels

faster. Another test was simply to place an object in front of the sensor and see if they could detect that something was there. For the Lidar I used inanimate and moving objects around my room to see if the scans could pick me up. The Lidar was tested also in multiple different rooms to determine if the scans could be blocked by different diffractions whether that be plastic or glass walls. Finally, for the accelerometer I would tilt the sensor in different directions and see if the results would print to the serial monitor the intended values it should be outputting for the x,y,z axes. The expected results of the ultrasonic sensors and the Lidar was that they should accurately detect obstacles within their specified range, they should also be waterproof, and the data should be feasible enough while moving in order to use it for obstacle detection and avoidance. For the accelerometer it should be able to detect and print 1 to -1 g values for the acceleration and the gyroscope should be between 2 to -2 degrees/second values.

LoRa test (Max/Daniel)

To test the LoRa communications a similar setup to the wired controller test was created except the LoRa communication was set up between the Base Station and USV Raspberry Pis (MCU 1 and 2). To simplify testing in the lab, the output of the controller was viewed in the serial monitor rather than setting up the servos and motors each time. These test goals were to ensure reliable communication and input values could be sent wirelessly to the USV to the Base Station and vice versa.

GPS test (Max)

To test the precision of the GPS module we will compare the coordinates measured from the GPS module with the actual coordinates of the current location. This will be done by bringing the module to different locations and using the coordinates from google maps as the actual coordinates.

Water testing

The goal of water testing is to ensure no leakage and control over the USV while on the water. To do this the USV will be tested at home pools for basic control and leakage testing to avoid travelon all the way to the lake. To test leakage, the hull will be weighed down and left in the pool to see if water accumulates inside. Home pool water testing will also allow the USV to be power tested for power consumption while running in the water. For in depth control testing the USV will be tested at a shallow location at Lake Perris where it can be monitored. From there basic control tests of auto and manual mode can be conducted on the lake.

4.2 * Experiment Results, Data Analysis and Feasibility

Carry out the experiments designed above and present experimental data, quantitative analysis of the data, and the conclusion to show the feasibility of your project idea, how the experiments help you decide whether your technical design objectives can be achieved, and how they help you select the best solution to be further developed in the design project.

Power testing (Daniel,Max,Alyan)

USV

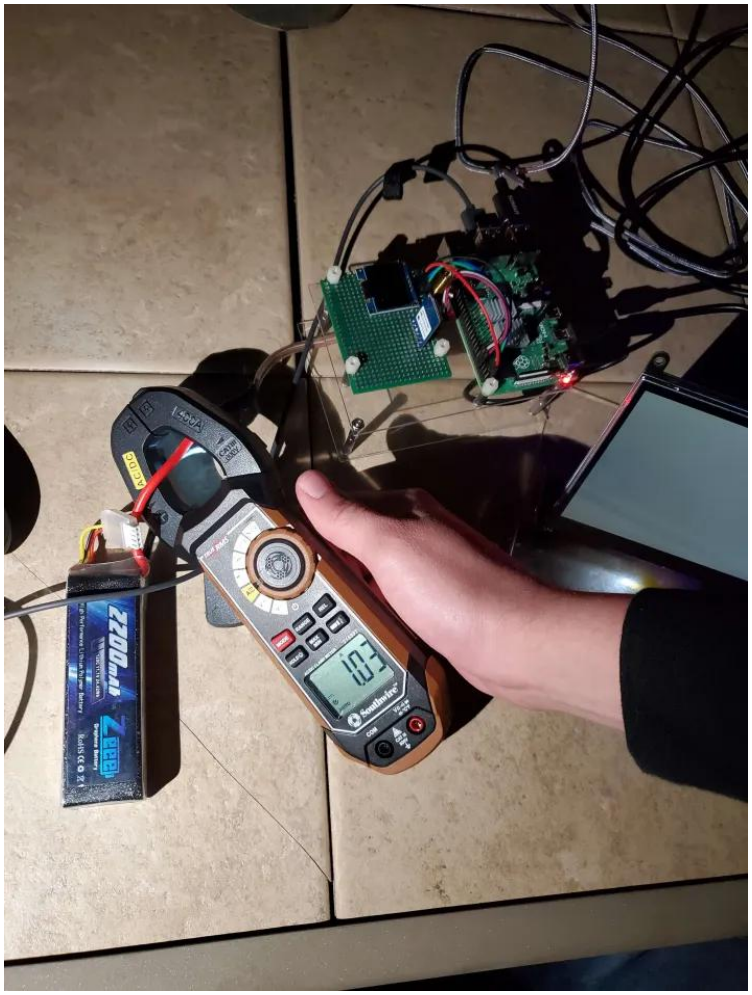
Given the 1 hour minimum run time criteria it was necessary to test the USV power consumption. For this test two separate measurements were taken; one with no power input or idle current draw and one loaded with the USV in water while the motor is running. For the idle current test the batteries were plugged into the USV and the code was started. The current was then measured at the node where the two batteries were connected giving the total current drawn from the system. When measured it was found that the USV had an idle current draw of 430 mA and a loaded current draw of 3.72 amps. This means that the maximum power output of the USV is 41 watts. Given the 111 watt hour battery capacity, the USV should be able to run under power continuously for 2.5 hours.

Fig 4.2a current measurements (left: Idle, Right: loaded)

Base Station

For the Base Station power test, all components including the Xbox controller, mini display, OLED and LoRa were plugged in and running. The ammeter was then set around the positive of the battery to measure the current draw of the system. When measured it was found that the Base Station had a current draw of 1.03 amps. This puts the total Base Station power at 11 watts. Given the 2200 mAh battery supplying the Base Station the Base Station should be able to operate for 2 hours.

Fig 4.2b Base Station current draw test



FPV

Since the FPV camera and transmitter on board the USV runs off its own power supply, current measurements were also taken for this system. For this test the system was plugged in and running and the ammeter was placed around the positive of the battery like the previous tests. When measured the FPV current draw was 870 mA. With the 2200 mAh capacity the FPV should be able to run for 2.5 hours.

Fig 4.2c FPV current draw

Water Testing (Daniel/Max)

For water testing the main goal was to ensure that the hull would not leak and that there was sufficient control of the USV under power. Since finding locations where the USV was allowed to be tested were scarce there were only options of small home pools and large bodies of water like lake Perris to test at. At home the hull was tested for leaks and strength tested with 20 pounds added inside the hull. This ensured that all the base welds holding the hull together provided a water tight seal. To test the full integration of all the systems we brought the USV, Base Station and other equipment for testing to Lake Perris. Here we were able to test the portability of the Base Station as well as the functionality of both the USV and the Base Station. The test location was chosen such that the water was shallow enough to wade into next to the USV in case it turned and headed out to the middle of the lake. This area also sat at the corner of the lake making only one direction where the USV could head out to an unreachable area should anything go wrong. We also noted to stay clear of any fishers to stay out of the fishing line, although some inevitably did get tangled in the propeller, and to be generally respectful to others at the lake. Here we found that manual mode was limited but good enough for maneuvering the USV into position on the water. To test auto mode, the USV was positioned about 30 feet away from the way point and then set in auto mode to head toward that location. The waypoint and start location was set to follow the shoreline and head

toward the dam wall so that if the USV drifted out of reach it could be recovered off the rocks. When testing auto mode, the USV would initially head toward the location before unforeseen calibration issues would send the USV in circles. We attempted to fix the issue onsite using the mini display however the sun, small display, and small workspace made debugging cumbersome. In all the USV was tested at Lake Perris 3 times where the first time only location scouting and a float test were done due to heavy rain. On tests 2 and 3 control testing of the USV in water was performed.

Fig 4.2d variety of water tests for USV




Field control test

https://youtube.com/shorts/URFtzErI4Vo?si=jIMahiunlo9m9Gs4

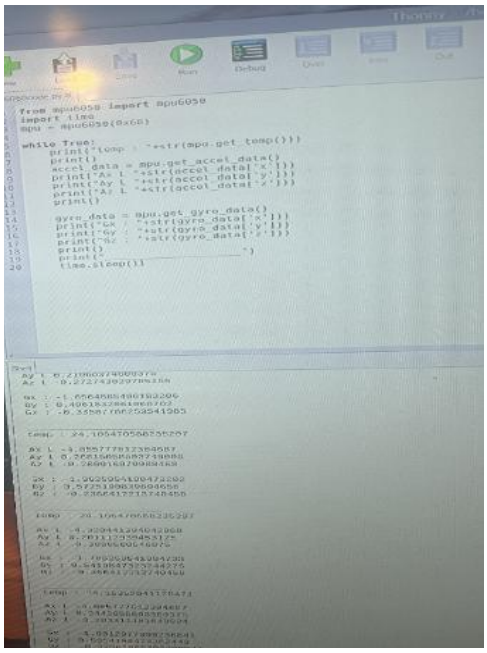
Field Auto test

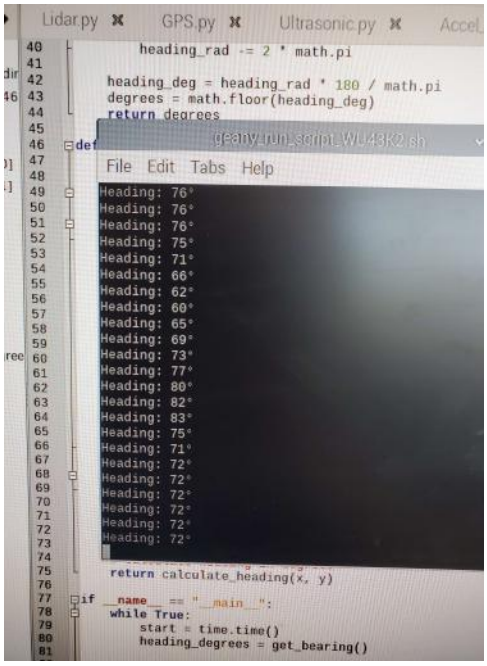
https://www.youtube.com/shorts/euukcnsd_w0

Sensor testing(Alyan)

For the sensor testing the Lidar, ultrasonic sensors, the magnetometer and the accelerometer were plugged into the Raspberry Pi and connected. The I2C devices which are the magnetometer and accelerometer were connected to the SDA and SCL pins while the ultrasonics went to the GPIO and the Lidar was connected via a UART to USB connector. After being plugged in I plotted the data that the Lidar was mapping off the room, and printed the distances from the ultrasonic sensors, accelerometer and magnetometer.

Fig 4.2e MPU6050, QMC5883L sensor data printed, Ultrasonics sensor data printed





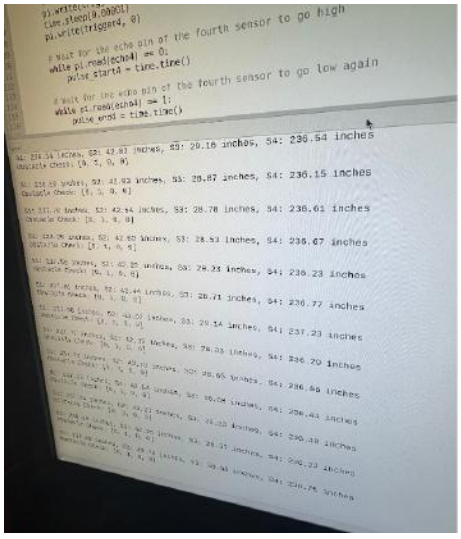
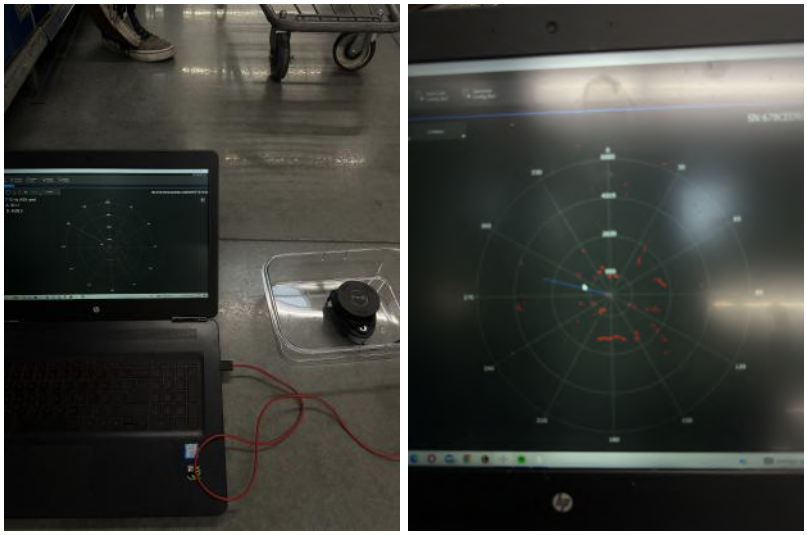


Figure 4.2f Lidar sensor at the store



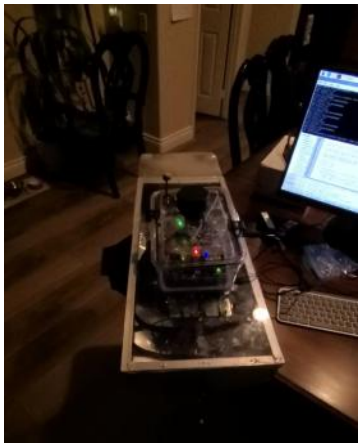
Lidar Plot Demo Video:

https://youtu.be/_pMVb9DJNh4?list=PLDdB8V38alHQ_w6yEoxjNoKn4vi3sJ0ct

PD Testing

For the PD feasibility testing, the USV was placed indoors on a raised platform shown in Fig .4.2g. The object detection has been turned off to isolate the PD functionality. The ideal heading was set to the front of the USV so that the propeller is centered. If the USV is turned counterclockwise the propeller turns to the right. If the USV is turned clockwise the propeller turns to the left. This test was successful. From this test, the feasibility of controlling the heading with a PD controller was proven. A youtube video is linked below of the test.

Figure 4.2g PD Testing Environment



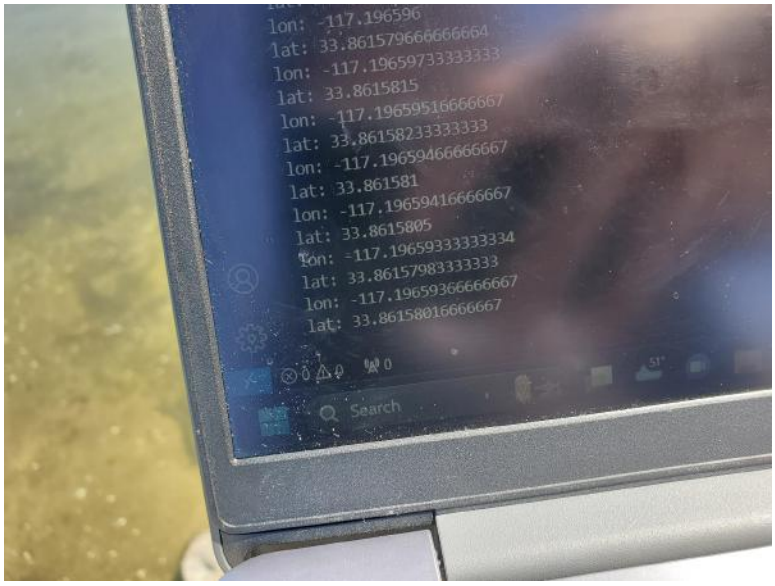
PD Testing Video:

https://youtube.com/shorts/1aUQc_LDcwY?feature=share

GPS Testing

For the feasibility testing for the GPS module, we grabbed the coordinates at different locations to ensure that it was within at 10 meter precision. Some sample measurements are shown in Fig. 4.2h.

Fig. 4.2h GPS Test



5 * Architecture and High Level Design

5.1 * System Architecture and Design

Control: (Daniel / Max)

The control input can be separated into two parts, Manual and Automatic. These modes allow the user to control the USV manually or set desired locations to send it autonomously to a location. These two modes determine how the controls receive input.

Manual Mode (Daniel)

The goal of manual mode is to enable the user to operate the USV from the Base Station. This Base Station requires a user interface to adjust the throttle and steering inputs and to also manually set waypoints for an autonomous mission. These controls will all be operated at a Base Station for the USV in which the user can send and receive data from the Base Station to the USV. To interface the controls the Base Station uses an Xbox controller using the D-Pad for controls. These controls are intuitively up for forward, back for reverse, left for left and right for right. From there to switch to auto mode the Y button presses and to exit auto mode the B button is pressed. These values are then sent via LoRa for processing on board the USV.

Automatic Mode (Max)

The goal of automatic mode is to have the USV navigate fully unmanned to a set location. This uses PD controller code which will navigate from waypoint to waypoint by using longitude, latitude, and heading and adjusting course accordingly. These waypoints are set upon start up on the Graphic User Interface or GUI. The GUI is opened up on start-up of the Base Station where a map of the location pops up and a path can be selected. Once created the USV waits for user input to begin the mission.

Power: (Daniel)

Supply power (Daniel)

The power section of the USV needs to be able to power the controls, on-board computer and all other peripheral electronics on board. These systems need to all run off the main power supply, with exception to the FPV system explained in the communications section. These systems all required different input voltages which will require a step down dc circuit from the manpower supply.

Controls (Daniel)

The controls for steering and propulsion are controlled by a brushless motor and a servo . The brushless motor is connected to an esc which is tasked with switching the high current to the motor. Both the esc and servo are connected to the on board Arduino (MCU 3) which takes the place of a normal rc receiver. The Arduino has a bit bang style PWM signal that receives an input

from the Raspberry Pi (MCU 2) and outputs the appropriate signal for throttle and steering controls.

Sensing: (Alyan/Max)

The USV will need to gather information about its surrounding environment to navigate to waypoints and avoid potential obstacles along the way. This will require a 360 degree field of view around the USV to detect objects as well as locational sensing like GPS location and orientation. These sensors will allow the PD controller to make the appropriate decisions when encountering a new waypoint or obstacle. The Object detection needs to be able to detect obstacles up to 25 feet away from the USV to give ample time for the USV to react. Additionally the USV needs to fetch accurate location and heading information to navigate to waypoints within 25 feet of the location.

Lidar (Alyan)

The Lidar sensor allows the USV to see objects 360 degrees around the vessel. Lidar serves at the USV's primary ranged object detection on board and was selected for its accuracy and range of 10 meters. It looks for objects in the way of the heading of the USV and if one is detected a value is triggered for appropriate action to be taken by the auto controller.

Ultrasonics (Alyan)

The Ultrasonic sensors on board the USV are there as a secondary object detection system. The USV is equipped with four ultrasonic sensors for detecting objects to the left, right, front, and below the vessel. The ultrasonic sensor at the bottom is there to avoid running ground in shallow water due to objects that are submerged or due to water level fluctuations not seen when setting waypoints. The above water ultrasonics provide object detection lower than that of the Lidar in case very close objects are missed by the primary object detection.

GPS (Max)

The GPS module is used to ping the current location of the USV. This value is then compared to the next waypoint location to calculate the appropriate heading to the next waypoint. This value is pinged periodically to continuously adjust the USV towards its desired location.

Magnetometer (Alyan)

The magnetometer equipt on board is used to detect the heading of the USV while navigating from waypoint to waypoint. Like the GPS module it too will be checked periodically to make adjustments to the USV while navigating the waypoints. The magnetometer is necessary because although the location of the USV is known the direction is not.

Accelerometer(Alyan)

The accelerometer plays a crucial role in detecting the USV's acceleration in different directions. It measures the gravitational forces acting on the USV, providing valuable data for motion analysis. By detecting changes in the acceleration, the accelerometer could help the USV in determining its steering and propulsion in order to account for drift. This data is particularly important in more unstable environments with dynamic water/wind movement where the vessel may encounter varying forces.

Communication: (Max/Daniel)

Given the long ranges and variety of tasks the USV will have to perform, it is necessary to have reliable long range communication to send and receive data to and from the USV and Base Station. For this reason LoRa communication was chosen for its long range capability and low power consumption.This was ideal for simple information however, it was also necessary to get video information from the USV as well which LoRa's low bandwidth would not support. For this reason an external one way video feed was added using FPV 5.8 GHz analog video signal for viewing and recording USV missions.

LoRa (Max/Daniel)

Both the USV and the Base Station are equipped with LoRa communication modules. These modules are used to send necessary instructions between the user interface at the Base Station and the USV. The message is encoded with four different values which are request, mode, steering and throttle respectively. These values will always be formatted this way when sending from the Base Station to the USV. Throttle and steering are intuitively the values sent from user input to navigate the USV in manual mode. Mode is the variable that switches states between auto and manual mode which by default is set to 1 for manual mode. Request is a variable for requesting location and heading information while in auto mode. When receiving a request the USV will switch to transmit mode while the Base Station switches to receive mode to fetch information. In all the entire function of the basestation to USV commands can be categorized into these four variables and can read information sent from the USV in three variables for longitude latitude and heading.

FPV (Daniel)

The FPV camera on board the USV is used to give a visual perspective of the USV while on a mission. This allows the user to see what is ahead of the vessel at all times. This system is operated by a separate power supply from the main board and control battery bank. This is to ensure that the FPV camera will always be on even if main power fails or runs out of battery. This acts as a fail safe because there are two separate ways of location the USV in case of wreaking where the user could use GPS location from auto mode or visuals from the FPV camera.

5.2 * Hardware Architecture

Base Station (Daniel)

The goal of the Base Station was to have a compact and portable way to control and receive data from the USV. The Base Station needed ways for users to enter waypoints, control the USV, and a way to view data from the USV.

LoRa

The LoRa module for the Base Station is used for the RF communications needed to communicate with the USV. It is assembled on the Base Station pcb prototype board along with the OLED screen where they bus power and grounds for wire management.

SSD 1306 OLED

The mini OLED screen displays the mode the Base Station is currently in. This display also lets you know when waypoints are being sent to the USV on startup. In auto mode the oled will display the current location of the USV by displaying latitude, longitude, and heading. The OLED was chosen for its small size while also being able to display simple yet critical information.

Xbox controller

The Xbox controller is used for user input for control in manual mode while also used for switching from manual mode to auto mode and vice versa. It is wired to the Base Station via USB and inputs are interpreted by the Raspberry Pi (MCU1). The Xbox controller was chosen for its wide functionality and documentation for custom setup in python in which references were found. It proved to be perfect for the project as it had multiple interfaces to be set up for controls.

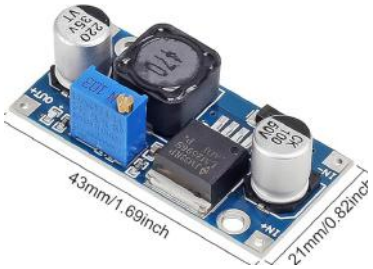
Wisecoco 7 inch Mini Display

The 7 inch mini display is used to view waypoint maps on startup of the USV/Base Station. The mini display also proved critical as it allowed onsite code debugging when at the lake testing. The display uses the Raspberry Pi's Micro HDMI out for video input and is powered off the 5 volt USB power from the Raspberry Pi.

LM2596 buck converter

The buck converter on board Base Station takes 11.1 volt input and steps down to 5 volt to power the Raspberry Pi. This system allows the Base Station to be battery powered and portable to launch the USV at a variety of locations.

Fig 5.2 LM2596 buck converter



FPV receiver

The FPV receiver is used to view live stream video from the USV. This allows the user to see how the USV is positioned when using manual mode or to watch while the USV navigates in auto mode. The FPV receiver also allows recording so that video of the USV mission can be recorded and watched at a later time.

USV

Hull and Electronics Housing: (Daniel)

The goal of the hull and electronics housing was to create a watertight enclosure to hold all the components necessary to operate the USV. The hull also needed to be big enough to take on mildly rough water in case of bad weather or wake from other boats on the water. To counter this the boat hull was built sufficiently large enough to use its mass to its advantage whilst also being small enough to transport and test.

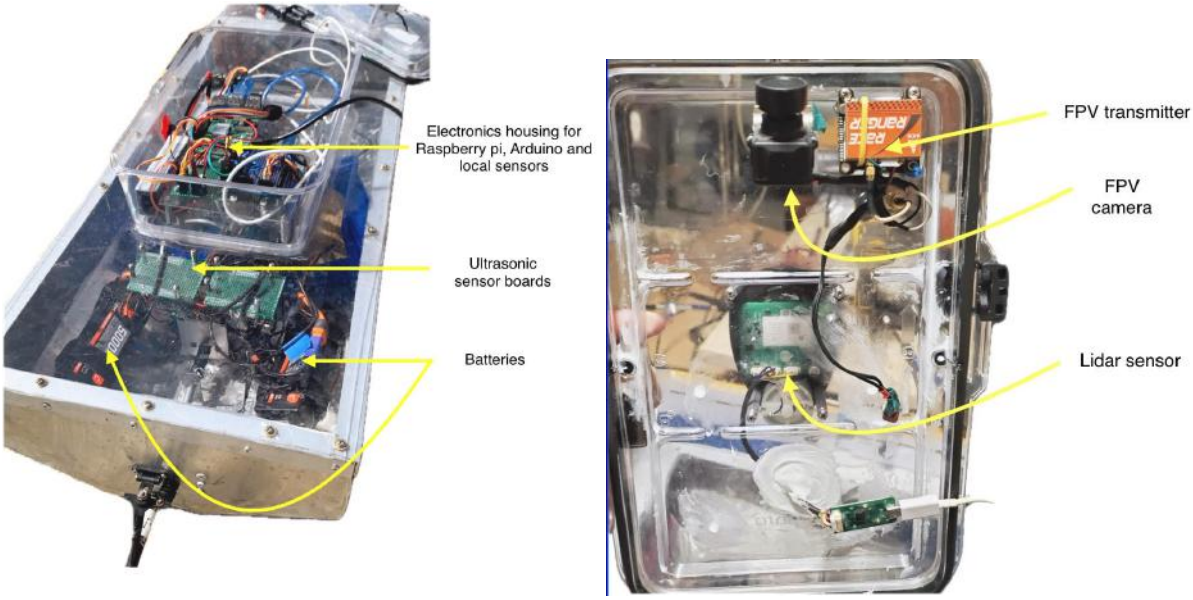
Hull (Daniel)

The boat hull was designed to be 1x.3 meters made from aluminum sheet. It needed to house the motor, servo, batteries, cargo stores, and ultrasonic sensors. Each component needs the necessary brackets or other mounts fitted to hold these components in place. The hull was formed by folding aluminum sheet and then using braze rod to fix separate sheets together. Aluminum rivets are also used to fix the top and plexiglass supports to the hull. Finally JB weld marine epoxy was used to seal the seams of attached parts to ensure it was sealed. For propulsion, drive components from a Traxxas blast rc boat were used for all drive components, and rudder. The seals and propeller are from a Traxxas villain IV and seal the steering linkage and propeller stuffing tube. Ultimately the build came out to the price of an rc boat hull but with way more flexibility and room for the project.

Electronics housing (Daniel)

The electronics housing was made from a clear tupperware container to seal the electronics from outside elements. The clearinghouse was chosen so that the FPV camera could be housed inside. All electronics, with exception to the ultrasonic sensor boards and esc, are mounted on standoffs in case any water enters the housing so the boards do not sit in water. The FPV electronics and Lidar are mounted to the lid of the tupperware container for easy maintenance and testing. The Tupperware container is secured to a plexiglass sheet and sealed with silicone around the edges where the tupperware meets the plexiglass sheet. On the underside of the plexiglass sheet the ultrasonic sensor boards and esc are mounted. These sit upside down to ensure if any water enters the hull the electronics stay out of the water.

Fig 5.2 USV hull and electronics box



5.3 * Software Architecture (only required if your design includes software)

<u>Control</u>: (Daniel / Max)

USV (Max)

This flowchart (Fig. 5.3a) describes the high level software design for the USV. On startup the USV will first read the GPS module and store its starting coordinates. It then enter a loop where it waits until it has received the longitude and latitude values for each waypoint from the Base Station. Once it has received those values it will default to manual mode. In manual mode the throttle and steering values are parsed from the command sent from the Base Station and sent to the MCU 3. MCU 3 gives the throttle and steering values to the brushless and servo motors. In auto mode the data from the Lidar, ultrasonic sensors, magnetometer, and GPS module are read. Next the distance from the USV's current location and the next waypoint is calculated. If the USV is within 10 meters of its next waypoint, it switches to the next. If the USV is within 10 meters of the final waypoint it will calculate the heading towards the starting location (Fig. 5.3b). If the USV is not at the final waypoint it will calculate the heading towards the next waypoint. After calculating the next heading, the steering and throttle values are sent to MCU 3 to be sent to the brushless and servo motors. The throttle is always set to the same number to maintain a constant speed during auto mode. If the request flag is set to one in the command sent from the Base Station the USV LoRa module will be set to transmit and will send its current location and orientation to the Base Station LoRa module and begin to parse the next command. If there is no data request, the next command from the Base Station is parsed.

Fig. 5.3a Auto Mode Flowchart

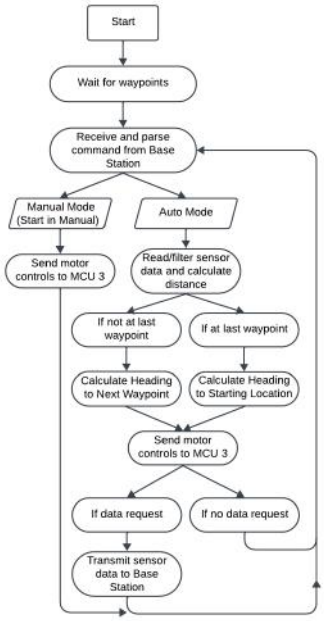
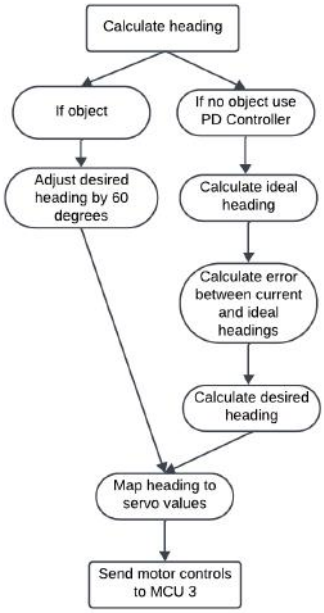


Fig. 5.3b Calculate Heading Flowchart

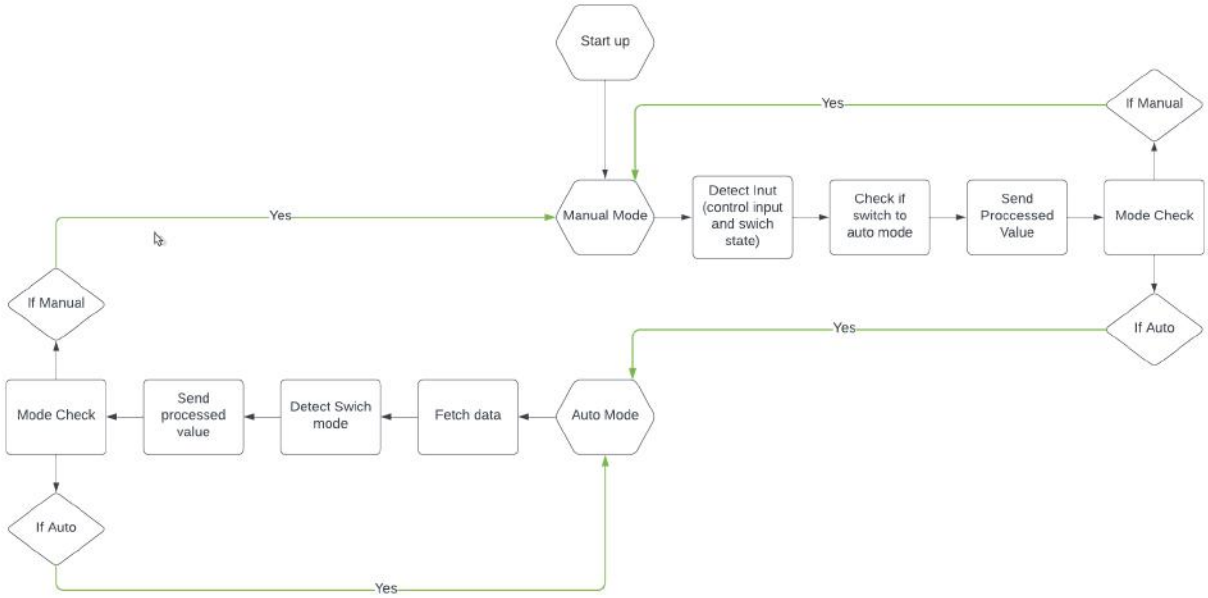


This flowchart (Fig. 5.3b) describes the algorithm for calculating the desired heading of the USV. If the Lidar or ultrasonic sensors detect any objects then the heading for the USV is adjusted by 60 degrees. If there is no object detected, the PD controller handles the calculation. In the PD controller the ideal heading is calculated by comparing the coordinates of the USV and the waypoint it is currently following and calculating the angle between them. The error between the current heading of the USV and the ideal angle is calculated and is stored for the next calculation of the desired heading. The PD output is used in the calculation of the desired heading. Once we have the desired heading in degrees it is converted to the range of values used by the servo motor. These values are sent to MCU 3 to be sent to the servo and brushless motors.

BaseStation (Daniel/Max)

This flowchart (Fig. 5.3c) describes the high level software design for the Base Station. The Base Station is tasked with setting waypoints, giving user interface for manual control and waypoint selection, and user ability to switch from auto to manual mode. On startup the Base Station will open the GUI to select GPS coordinates on a map and make an array of waypoints to follow. After transmitting the waypoints to the USV, the Base Station will send a command to enter manual mode and begin transmitting user input from the controller. This loop will run continuously until the user makes the switch to auto mode. In auto mode the Base Station switches to a transceiver mode where it will continuously fetch location data from the USV while also checking if a mode switch occurred and if that value needs to be sent to the USV for switching.

Fig. 5.3c Base Station flow chart



5.4 * Rationale and Alternatives

For the project there were many alternatives to the systems we set up for the USV and Base Station. For the Base Station for example, a laptop could have been used to perform the same functions as the Base Station designed; however, the Raspberry Pi was chosen for its cheap cost and versatility with other components. This goes for The USV as well considering the variety of sensors needed. Additionally there were many options when looking for RF communication modules. LoRa was chosen for its superior range however in hindsight should be swapped due to its slow transmission rate. When choosing a motor for the USV there were a variety of options including stepper and brushed dc motors. The USV ended up being fitted with a brushless DC motor for its isolated design and high torque needed for aquatic environments.

6 Data Structures (include if used)

A description of all data structures including internal, global, and temporary data structures. State clearly who is responsible for which module/task

Not Applicable

6.1 Internal software data structure

Data structures that are passed among components the software are described.

6.2 Global data structure

Data structured that are available to major portions of the architecture are described.

6.3 Temporary data structure

Files created for interim use are described.

6.4 Database descriptions

Database(s) created as part of the application is(are) described.

7 * Low Level Design

7.1 Description, Schematic and Diagrams

USV(Group):

Power (Daniel) #1

The USV has three separate power voltages to power the motor, servo, and onboard computer. To manage the power to each system on board a power harness would need to be made utilizing the same LM2596 buck converter the Base Station uses to step the 11.1 volt main power supply down to 5 volts to USB for the on board computers. The harness uses two IC5 battery connectors to wire the two 3 cell 5000mAh lithium polymer batteries in parallel giving the USV the total power capacity of 111 Watt hours. The 11.1 volt battery power is directly connected to the esc using an XT90 connector and will supply the brushless motor with power. By using different connectors prevents user error when setting up the wiring harness. The servo is powered off the esc's internal BEC to send 6 volts for powering the servo. It is to be noted that the ESC and servo share a common ground with the Arduino to process signals that can be sent to each control. Figure 7.1a shows the schematic for all the wiring for the control and power circuits needed to power the USV.

Fig 7.1a (power management circuit)

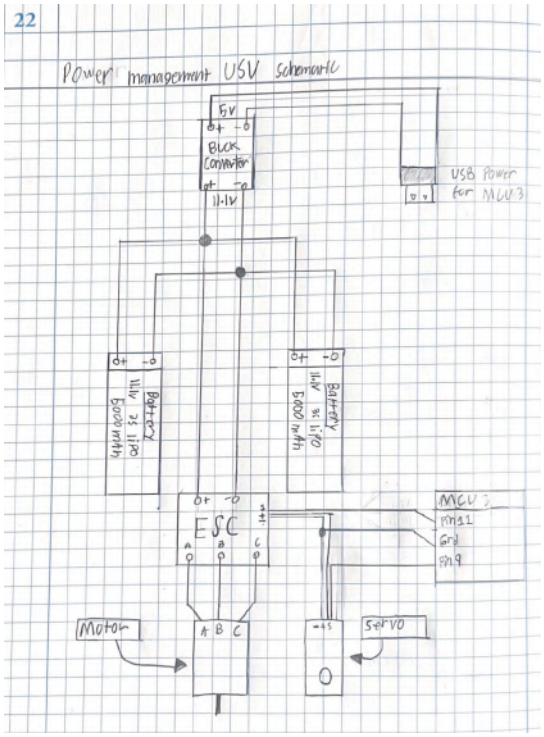
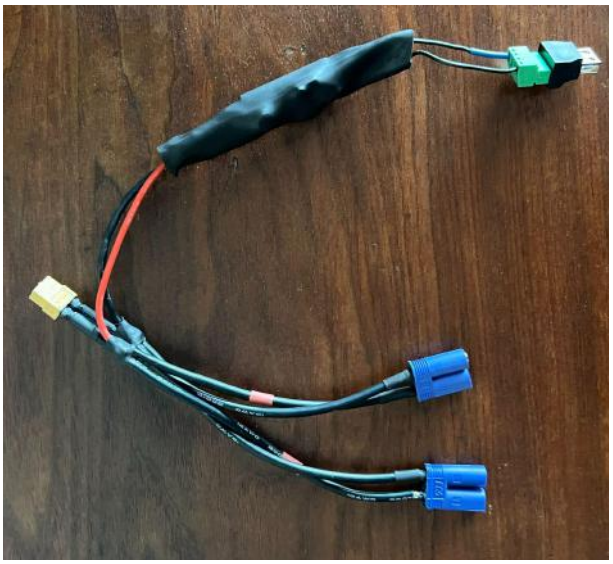


Fig 7.1b (assembled power harness)

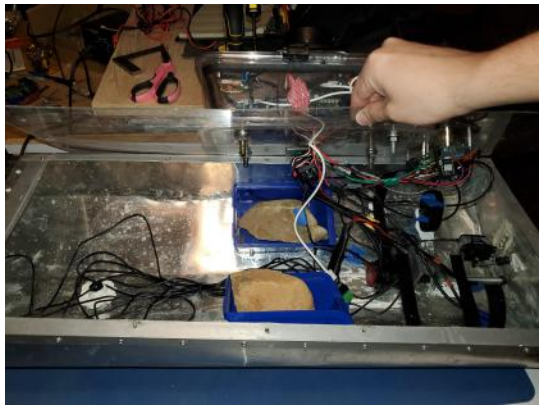


Control (Daniel)

ESC (Daniel) #2

The ESC specced for the USV is a Quicrun 10BL120. It is a 120 amp continuous rated waterproof brushless speed controller and can take inputs from 2 to 4 cell LiPo batteries. This ESC far exceeds any reasonable conditions for power consumption the USV will demand but was chosen for its rated waterproofing, combo with the motor, and relatively cheap price compared to other options on the market.

Fig 7.1c (ESC and motor combo used in USV)



Motor (Daniel) #3

The motor specced for the USV is the Quicrun 3652 3250 Kv brushless motor. This motor was included as a set with the ESC and was chosen for its relatively low Kv value to supply low end torque needed for marine applications. Given the contactless nature of the brushless motor it makes for the ideal choice for a motor subject to moisture.

Servo (Daniel) #4

The servo used by the USV is the PowerHobby 209 mg servo. At 209 oz/inch of torque this servo far exceeds the strength needed for the USV. Although overpowered, the servo was chosen because it is waterproof and the team already owned one.

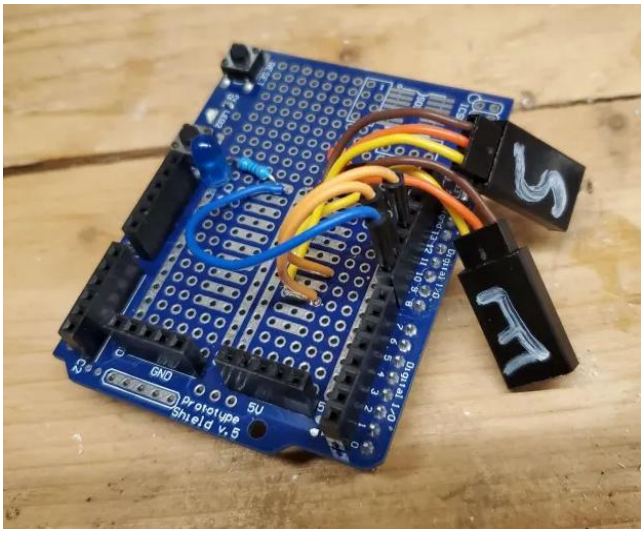
Fig 7.1d Servo and motor components

Arduino (Daniel) #5

The Arduino on board is tasked with receiving values 0→1023 and assigning the values accordingly to the ESC and servo. This is accomplished by using a simple bit bang style PWM signal to time the on off pulses. These outputs use pin 9 for the servo signal wire and pin 11 for the ESC signal wire. The ground wires from all components are bussed and the 6 volt BEC output from the ESC goes to servo power. All these connections are hard soldered to an Arduino prototype board to make clean connections for all the wires to the Arduino. This hat board also has an LED indicator light to show when the board is powered. When the Arduino receives throttle and steering values it will then output the correct 50Hz duty cycle for appropriate steering angle and throttle position. The values are distinguished by even and odd values where even values are steering values and odds are throttle values. Since these values are from 0 to 1023, this allows for 512 steering and throttle positions while also making minimum processing time to output positions.

Fig 7.1e Arduino bit bang code and output hat

```
int remainT1=0;
int remainT2=0;
int EscInput=511;
int ServoInput=510;
void setup() {
  Serial.begin(1200);
  pinMode(9,OUTPUT);
  pinMode(11,OUTPUT);
  digitalWrite(8, HIGH);      //Led check light
}

void loop() {
  //int timeS = millis();  // for timing setup

  if (Serial.available() >= 1) {   // for reading and determining if throttle or steering value
    int receivedValue;
    int lastVal=0;
    Serial.readBytes((byte*)&receivedValue, sizeof(int));
    Serial.flush();
    Serial.println(receivedValue);

    if(receivedValue>1024 || receivedValue<=0){
      receivedValue=lastVal;
    }

    else{
      if (receivedValue % 2 == 1) {
        EscInput=receivedValue;
      }
      if (receivedValue % 2 == 0) {
        ServoInput=receivedValue;
      }
      lastVal=receivedValue;
    }
  }

  int throttle = map(EscInput, 0, 1024, 1100, 1800); //maps input 0-1023 to output
  int steering = map(ServoInput, 0, 1024, 1000, 2000);
  // entire loop always takes 20ms or 50hz
  //delays ajusted acordingly to produce PWM signal
  // ESC
    digitalWrite(11, HIGH);
    delayMicroseconds(throttle);
    digitalWrite(11, LOW);
    remainT1=10000-throttle;
    delayMicroseconds(remainT1);
  //servo
    digitalWrite(9, HIGH);
    delayMicroseconds(steering);
    digitalWrite(9, LOW);
    remainT2 = 10000 - steering;
    delayMicroseconds(remainT2);
```

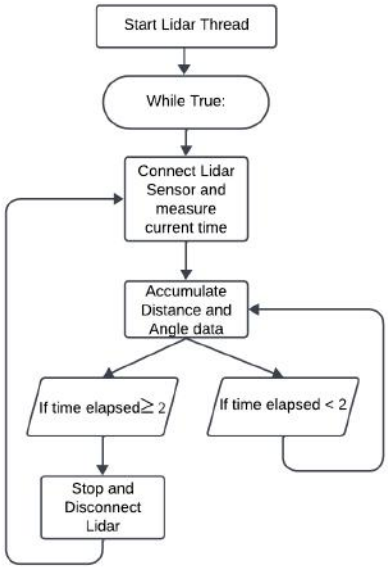


Lidar Sensor (Alyan) #6

The Slamtech RPLidar A1 [6] sensor is used for object detection on the USV. To retrieve data from this sensor we used the adafruit-rpLidar library. A good example of the results of how the Lidar was suppose to operate can be seen in geo BruceBruce instructable[4] where he uses a RPLidar to control what led lights turn on. The software was integrated into the full USV script using threading. We did this to allow the Lidar to retrieve data without halting the rest of the USV script. Fig. 7.1f shows a flowchart of the algorithm for retrieving the angles and distances from the Lidar. The Lidar thread is started before the execution of the main while loop for manual and auto mode. In the Lidar thread there is an infinite while loop with a try except block. In the try block, we call a function called scan_Lidar. This function connects the Lidar and takes note of the current execution time. Two nested for loops are used to fill an angle and distance array. In the second for loop the elapsed time is measured. If the elapsed time is greater than 2 seconds the for loops are exited and the distance and angle arrays stop accumulating values. This is done so that we can ensure that the distance and angle arrays will always have at least one full rotation of data when used in the object detection algorithm. If the time elapsed is less than 2 seconds, the angles and distances measured by the Lidar are appended to the angle and distance arrays.

The distance and angle arrays are used in the object detection algorithm. These two arrays work alongside data from our ultrasonic sensors to raise a flag if something is detected too close to the USV. If the length of the angles array is not zero then the Lidar has detected an object less than the minimum distance threshold. If 330 <= angle <= 360 or 0 <= angle <= 30 for angle in angles then an object is detected in the front of the USV and the object detected flag is set to 1.

Figure 7.1f Lidar Scan Flowchart



GPS Sensor (Max) #7

The GoouuuTech GT-U7 [5] GPS Sensor is used to provide the position of the USV. Data is retrieved from the sensor using the pySerial library and the stream of data is parsed using the re library. This is what the pre-parsed data from the module looks like:

$GPGSV,3,3,11,32,52,197,38,46,49,200,30,48,50,193,*47

$GPGLL,3353.35925,N,11728.05471,W,063946.00,A,D*74

$GPRMC,063947.00,A,3353.35920,N,11728.05470,W,0.058,,230224,,,D*60

$GPVTG,,T,,M,0.058,N,0.107,K,D*2D

$GPGGA,063947.00,3353.35920,N,11728.05470,W,2,06,1.51,241.0,M,-32.3,M,,0000*69

$GPGSA,A,3,18,24,32,27,10,23,,,,,,,3.61,1.51,3.28*01

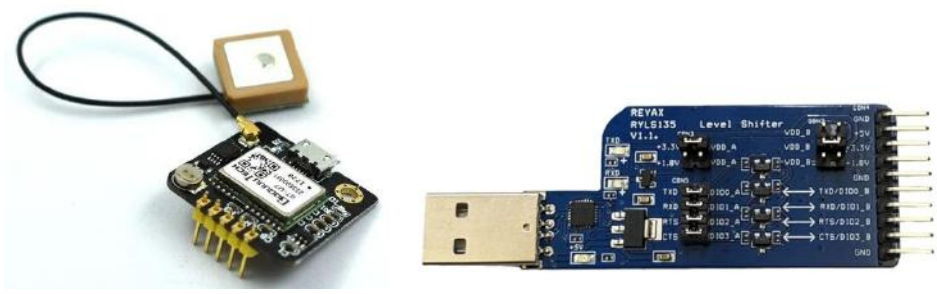
The line starting with $GPGLL contains the longitude and latitude values. In this sample output, latitude is 33 degrees 53.35925 minutes North and longitude is 117 degrees 28.05471 minutes West. These are converted to a latitude of 33.8893208° and a longitude of -117.4675785°. The conversion for latitude is latitude degrees + latitude minutes/60. Latitude is positive if it is North and negative if it is South. The conversion for longitude is longitude degrees + longitude minutes/60. Longitude is positive if it is East and negative if it is West.

To parse through the stream of data from the module, we filter out all lines that do not start with “GPGLL”. The sentence used by the re library is: '\$GPGLL,(\d+\.?\d*),([NS]),(\d+\.?\d*),([EW]),(\d+\.\d+),A.*' The re library splits the sentence into groups. The first group is longitude degrees and minutes. In the example output from the GPS module above this is 3353.35925. The third group is the longitude degrees and minutes. In the example output above this is 11728.05470. The second group is [NS]. If it is an ‘N’ the latitude is North and if it is an ‘S’ the latitude is South. The fourth group is [EW]. If it is an ‘E’ the longitude is East and if it is a ‘W’ the longitude is West.

Limitations and Solutions:

The Raspberry Pi 4 has two UART ports but they are both assigned to the same GPIO Pins (14 and 15). Since we are using the pins for our LoRa module, there is nowhere to put to GPS module. To work around this, the Reyax RYLS135 UART to USB converter (shown in Fig. 7.1.g) is used to interface the module with the Raspberry Pi. Since the USB on a Raspberry Pi provides 5V, the 5V to 3.3V step down on the converter is used for safe use with the GPS module.

Fig. 7.1.g GPS Module and UART to USB Converter



Ultrasonic Sensors (Alyan/Daniel) #8

The JSN-SR04T ultrasonic sensors [1] on the USV are essential for obstacle detection and avoidance during the operation. These sensors use the sending and receiving of sound waves in order to measure distances in front, right, left and at the bottom of the USV. The system uses the GPIO pins of the Raspberry Pi (MCU 2) using I2C connection in order to transmit trigger signals and receive echo signals. The code first works by initializing all the pins to their respective trigger and echo and begin the pulsing every second in the detObj() function. After getting all the raw distance values and converting it from centimeters to inches it runs it through Filter_Values() to filter all the values. For the front, left and right sensor if the object is over 78 inches it ignores it and a zero is stored in our object array. If the distance value is within 0-78 inches then a 1 is stored which means there is an object in front of the sensor. for the depth sensor(bottom sensor) we check if it is greater than 30 inches then we store a 0 for no object else if it is less then we store a 1 in the object array. We then return this object array to the PD controller for simplicity for it to determine what its intended path will be and how it plans on avoiding this object.

Fig 7.i snippet code for ultrasonic sensor and board

```python
import pigpio
import time

# Define GPIO pins for the sensors
trigger1, echo1 = 17, 18
trigger2, echo2 = 22, 27
trigger3, echo3 = 23, 24
trigger4, echo4 = 8, 25

def detObj():
    global trigger1, echo1, trigger2, echo2, trigger3, echo3, trigger4, echo4
    # Initialize pigpio
    pi = pigpio.pi()

    # Set trigger pins as output and echo pins as input for all sensors
    pi.set_mode(trigger1, pigpio.OUTPUT)
    pi.set_mode(echo1, pigpio.INPUT)
    pi.set_mode(trigger2, pigpio.OUTPUT)
    pi.set_mode(echo2, pigpio.INPUT)
    pi.set_mode(trigger3, pigpio.OUTPUT)
    pi.set_mode(echo3, pigpio.INPUT)
    pi.set_mode(trigger4, pigpio.OUTPUT)
    pi.set_mode(echo4, pigpio.INPUT)

    try:
        # Send a short pulse to trigger the first sensor
        pi.write(trigger1, 1)
        time.sleep(0.00001)
        pi.write(trigger1, 0)

        # Wait for the echo pin of the first sensor to go high
        while pi.read(echo1) == 0:
            pulse_start1 = time.time()

        # Wait for the echo pin of the first sensor to go low again
        while pi.read(echo1) == 1:
```

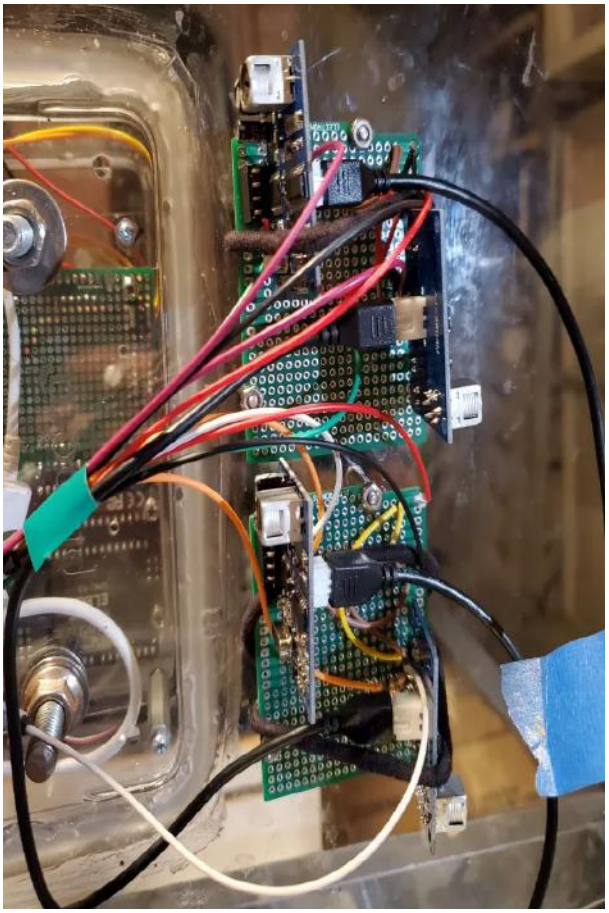
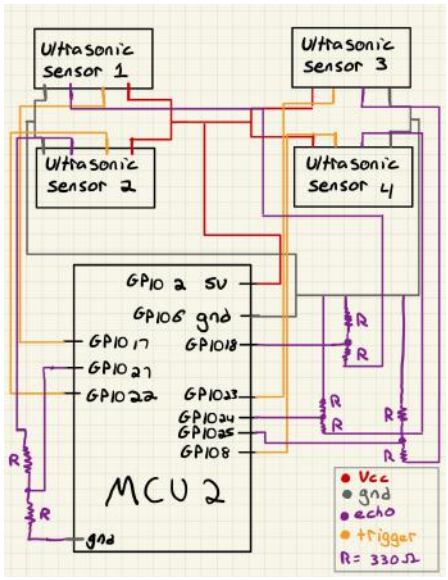


Fig. 7.1j Ultrasonic Sensor board Schematic



Accelerometer(Alyan) #9

The accelerometer on the USV is essential for our drift detection algorithm as it gives us the acceleration forces acting on our USV at any initial period. The datasheet that helped get all the addresses was made by Invensense[9] which helped speed up the process of figuring out how to code it in Raspberry Pi. In the code how we approach getting these x,y,z axes to be used for our analysis is by first calling our MPU6050 class which initializes the MPU6050 sensor. This sensor is both an accelerometer and gyroscope combined. Inside this class the read_raw_data method reads the raw sensor data from the specified register address and combines the bytes to get the complete 16-bit data value. If the value is greater than 32768, it subtracts 65536 to get the correct signed value. The read_accelerometer_data method is then called alongside the read_gyroscope_data which reads the raw data from the accelerometer registers and gyroscope registers then converts the acceleration data to gravitational units (g) using the sensitivity scale factor (16384 LSB/g) and the gyroscopic values to degrees per second using a scale factor of (131 LSB/°/s) . Finally it returns this data every second and is now up to the USV's PD algorithm to determine further instructions based on the given data.

Fig 7.k snippet code for MPU6050 sensor

```
class MPU6050:
        return value

    def read_accelerometer_data(self):
        acc_x = self.read_raw_data(self.ACCEL_XOUT_H)
        acc_y = self.read_raw_data(self.ACCEL_YOUT_H)
        acc_z = self.read_raw_data(self.ACCEL_ZOUT_H)
        Ax = acc_x / 16384.0
        Ay = acc_y / 16384.0
        Az = acc_z / 16384.0
        return Ax, Ay, Az

    def read_gyroscope_data(self):
        gyro_x = self.read_raw_data(self.GYRO_XOUT_H)
        gyro_y = self.read_raw_data(self.GYRO_YOUT_H)
        gyro_z = self.read_raw_data(self.GYRO_ZOUT_H)
        Gx = gyro_x / 131.0
        Gy = gyro_y / 131.0
        Gz = gyro_z / 131.0
        return Gx, Gy, Gz



def getAccel():
    Ax, Ay, Az = mpu6050.read_accelerometer_data()
return Ax, Ay, Az

def getGyro():
    Gx, Gy, Gz = mpu6050.read_gyroscope_data()
    return Gx, Gy, Gz

```

Magnetometer (Alyan) #10

The magnetometer on the USV is crucial for determining the orientation of the vehicle relative to Earth's magnetic field. It provides heading information that is essential for navigation and control. The data sheet that helped a lot was this was made by qstcorp[14] where it labeled the addresses needed to get our raw data. The magnetometer is interfaced with the Raspberry Pi (MCU 2) using the I2C protocol which allows it to communicate with the Pi and provide real-time orientation data. This code calculates the heading in degrees using a magnetometer, which measures the magnetic field's strength and direction. The magnetometer provides three-axis measurements (x, y, z) of the magnetic field. In this code, only the x and y components are used to calculate the heading. The x and y values are normalized to ensure they are within a specific range then scaled and offset using predefined scaling and offset factors. The scaled and offset x and y values are used as inputs to the calculate_heading() function where it calculates the headings in radian which is then converted into degrees. This is then sent to the USV for further calculations while the tracking of the minimum and maximum values of x, y were intended to be used for calibration purposes.

Fig 7.lL snippet code for QMC5883L sensor and picture of magnetometer and accelerometer

```
    # Initialize the I2C bus
    i2c = smbus.SMBus(I2C_BUS)
    data = array('B', [0] * 6)

    # Configure the sensor
    i2c.write_byte_data(0x0D, 0xB, 0x01)
    i2c.write_byte_data(0x0D, 0x9, 0x011101)

    # Scaling and offsetting factors for x and y
    SCALE_X = 1
    SCALE_Y = 1
    OFFSET_X = 0
    OFFSET_Y = 0

    # Main loop
    while True:
        try:
            sleep(0.2)
            for i in range(6):
                data[i] = i2c.read_byte_data(0x0D, 0x00 + i)

            x = (data[1] << 8) | data[0]
            y = (data[3] << 8) | data[2]
            z = (data[5] << 8) | data[4]

            x = x - (1 << 16) if x & (1 << 15) else x
            y = y - (1 << 16) if y & (1 << 15) else y
            z = z - (1 << 16) if z & (1 << 15) else z

            # Track minimum and maximum values of x and y
            X_MIN = min(x, X_MIN)
            X_MAX = max(x, X_MAX)
            Y_MIN = min(y, Y_MIN)
            Y_MAX = max(y, Y_MAX)

        except KeyboardInterrupt:
            print('Got ctrl-c')
            break

        # Scale and offset x and y
        x = x * SCALE_X + OFFSET_X
        y = y * SCALE_Y + OFFSET_Y

        # Calculate heading in degrees
        heading_degrees = calculate_heading(x, y)
```

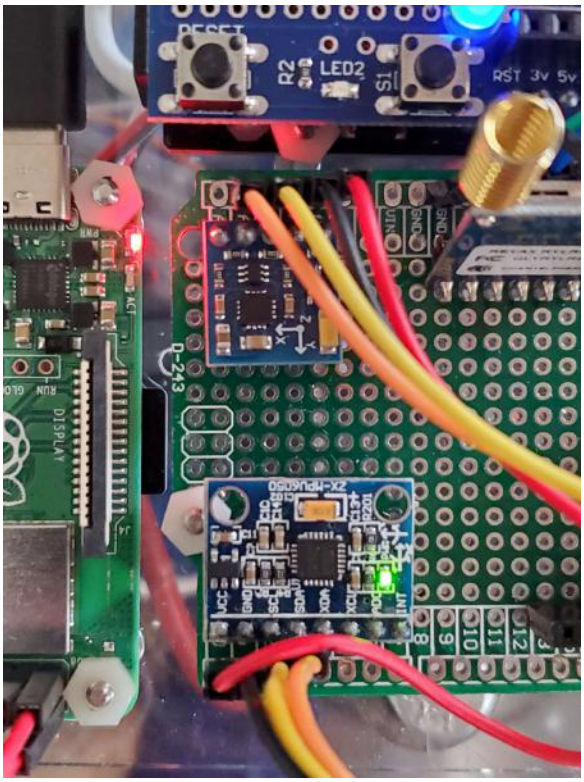
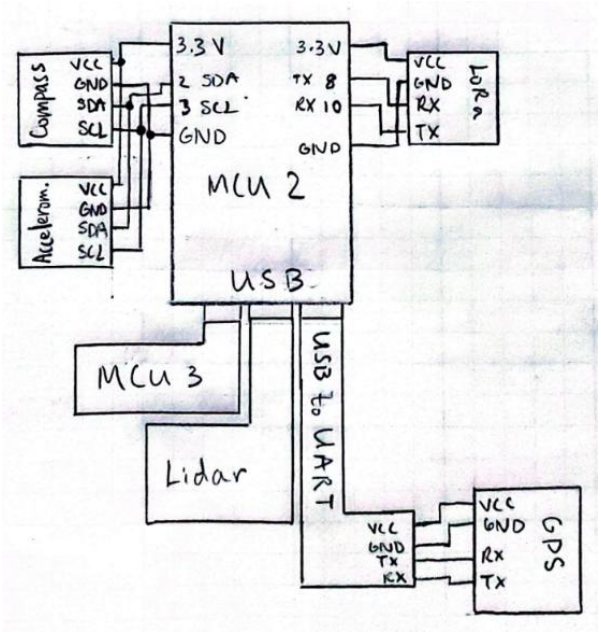
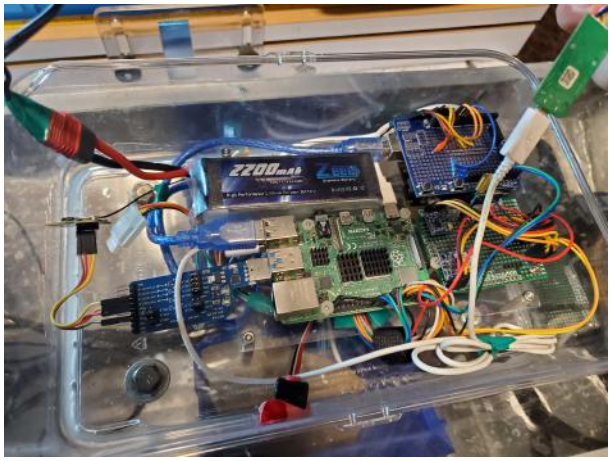


Fig. 7.1m USV Sensor and MCU schematic





PD control Logic (Max) #11

The USV relies on a PD controller when it is in Auto mode. It calculates the differences in longitude and latitude from the USV to the targeted waypoint and the angle between the resulting vector. This angle is set as the ideal heading of the USV. To ensure that the ideal heading is in the range 0 to 360, I add 360 to the difference and mod by 360. The difference between the ideal and current heading of the USV is calculated and set as the error. The proportional term is calculated as Kp * error where Kp is 0.5. The derivative term is calculated as Kd * (error - previous error) where Kd is 0.1. The PD output is calculated as the proportional plus the derivative terms. While not used by the script, the next heading of the USV is calculated as the current heading of the USV plus the PD output for debug purposes. This value is mapped to the range of 0 to 360 by adding 360 and modding by 360 and printed to the terminal. The PD output is mapped to even integers from 2 to 1022 so that it can be interpreted by the servo. This is done by multiplying the PD output by the ratio (1022-2)/360. The center servo value (510) is added to the resulting operation and is rounded to the nearest integer. If the value is odd, 1 is subtracted to make it even. When designing this PD controller I referenced an IEEE article with a similar project [10].

Limitations and Performance Issues:

The Auto Mode code runs every time that it receives a command from the Base Station over LoRa. This means that the PD controller is dependent on receiving a command over LoRa. Originally, the code was run on a timer on the USV but there were performance issues switching from Auto Mode back to Manual. This needed to be changed to allow for the reliable switch from Auto Mode to Manual. The command from the Base Station to switch to Manual Mode would be missed due to the acquisition of sensor values taking too much time. To solve this, the Auto Mode code now runs when it receives a command from the Base Station to do so.

Fig. 7.1n PD controller Code

```
def pd_controller(current_heading, waypoint_lon, waypoint_lat, current_lon, current_lat):
    global prev_error, Kp_yaw, Kd_yaw

    # Calculate the differences in x and y coordinates
    delta_lon, delta_lat = waypoint_lon - current_lon, waypoint_lat - current_lat

    # Calculate the yaw angle using arctangent
    target_heading = math.atan2(delta_lon, delta_lat)

    # Normalize the target_heading to the range [0, 360)
    target_heading_degrees = (math.degrees(target_heading) + 360) % 360
    print(f"Ideal Heading: {target_heading_degrees}")
    # Calculate the error
    error = target_heading_degrees - current_heading

    # Calculate the PD components
    proportional = Kp_yaw * error
    derivative = Kd_yaw * (error - prev_error)

    # Calculate the PD output
    pd_output = proportional + derivative
    # Update previous error
    prev_error = error

    # Calculate next heading
    next_heading = current_heading + pd_output
    # Normalize the next_heading to the range [0, 360)
    next_heading = (next_heading + 360) % 360
    print(f"Desired heading: {next_heading}")
    # Calculate the proportional servo value based on the PD output
    proportional_servo = pd_output * (1022 - 2) / 360

    #print(f"Desired heading: {pd_output}")

    # Calculate the final servo value
    final_servo_value = int(round(510 + proportional_servo))
    final_servo_value -= final_servo_value % 2  # Adjust to the nearest even number

    # Ensure the servo value stays within the valid range
    #print(f"Mapped Steering: {final_servo_value}")
    return max(min(final_servo_value, 1022), 2)
```

Base Station (Daniel/Max):

Controller and interpretation (Daniel) #12

The controller input to the Base Station is imputed using an Xbox one controller. To interpret the data from the xbox controller the Base Station uses a python library called EvDev which can decode a variety of controller inputs not exclusive to the xbox controller. This library and example code was found on the Core Electronics website [7,15] which described set up and how the inputs are interpreted, link found in references. The controller is set up for 4 separate user inputs. In manual mode the controller D-Pad is used for the user to input throttle and steering input to the USV using the intuitive forward left and right controls. To enter Auto mode the user will press the Y button where the Base Station will sense the input and switch both the USV and Base Station into auto mode. To exit from auto mode back to manual mode, the user presses the B button on the controller to set the Base Station and USV back to manual mode for user control. In manual mode the absolute inputs, i.e non single state inputs like D-Pad, joystick, and triggers, are set up to map their values.

Limitations:

For the final version only the D-Pad was used due to slow transmission rates however, the joysticks and triggers were implemented during wired testing confirming they work and why their setup still exists in the code.

Fig7.1o EvDev controller setup and Interpretation code

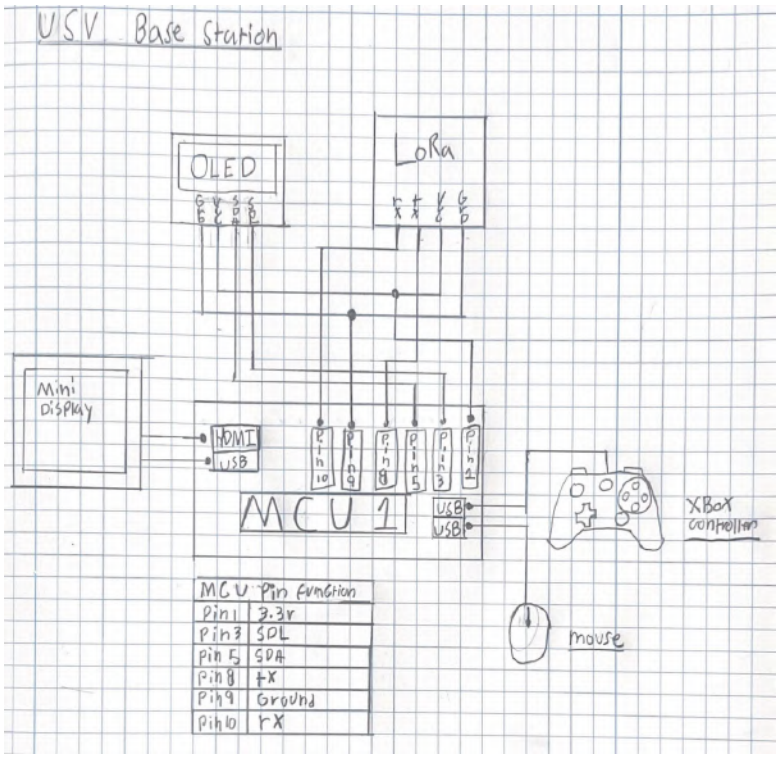
```
from evdev import InputDevice, categorize, ecodes
```

```
def Manual():
    global mode
    global req
    global steering
    global throttle

    axis = {
        ecodes.ABS_X: 'ls_x' ,    # 0-65,535 input
        ecodes.ABS_Y: 'ls_y' ,
        ecodes.ABS_RX: 'rs_x' ,
        ecodes.ABS_RY: 'rs_y' ,

        ecodes.ABS_RZ: 'rt',      #0-1023 trigger input
        ecodes.ABS_Z: 'lt',

        ecodes.ABS_HAT0X: 'dpad_x', #-1 0 1 input
        ecodes.ABS_HAT0Y: 'dpad_y'
    }

    center = {
            'ls_x': max3s/2,
            'ls_y': max3s/2,
            'rs_x': max3s/2,
            'rs_y': max3s/2
        }

    last = {
            'ls_x': max3s/2,
            'ls_y': max3s/2,
            'rs_x': max3s/2,
            'rs_y': max3s/2
        }
```

```
def Auto():
    global mode
    global Manual
    global req
    global controller
    global dPrntA

    dPrntA()

    last_a_state = 0  # Initialize the last A button state (0 for up, 1 for down)
    last_a_press_time = 0  # Initialize the last A press time

    while mode == 0:   # make void function for auto mode
        for event in controller.read_loop():
            if event.type == ecodes.EV_KEY:
                if categorize(event).keycode[0] == "BTN_B":  # press B to exit
                    mode = 1
                    Manual()

while mode==1:
    current_time = time.time()

    for event in controller.read_loop() :
        if event.type == ecodes.EV_KEY:

            if categorize(event).keycode[0] =="BTN_WEST":               # press y button to exit manual
                mode=0
                throttle=511
                steering=511
                test_data = '*' + str(req) + 'k' + str(mode) + '^' + str(throttle) + '%' + str(steering) + '+\n'
                transmit_lora(test_data)
                Auto()
```

In manual mode when D-Pad input is detected, the Base Station will send the processed values to the USV via LoRa. The D-Pad input has 3 values associated with each axis giving 0 for neutral, 1 for right on x axis and up for y axis, and -1 for left on x axis and down on y axis. The Base Station code will check all of these conditions and send the appropriate steering and throttle value associated with the input. In manual mode the throttle input is limited to 585 or about 20% throttle to ensure safe operation.

Fig 7.1p schematic and assembled view for Base Station



Oled/Mini Display(Daniel/Max) #13

The Base Station has two screens for user output where the mini OLED displays mode, sending waypoints on startup, and current GPS location and heading in auto mode. The Mini display is used for setting waypoints on the GUI map on startup.

OLED

For the Base Station the ssd 1306 OLED screen was used for its small size, cheap cost, and versatility. Since the screen uses I2C communication this also reduced the amount of wires needed for communication further simplifying the Base Station wiring. The ssd 1306 OLED was used with the circuit python library adafruit ssd-1306. For a reference there was a Youtube video [2] found that showed how to set up the ssd1306 OLED to display things like cpu temp and IP address. Although these features would not be implemented by the USV, this example code gave setup and basic understanding needed to use the OLED for the base station display. This made setup and displaying information easy while also helping with debugging as values could be checked while running the program.

Fig 7.1q OLED display functions

Mini Display

For the mini display for the Base Station, it was necessary to have a USB powered monitor for portability. For this we used the Wisecoco 7 inch mini display which uses the Raspberry Pi's Micro HDMI output and 5 volt USB power for video signal and power. This allows the user to select waypoints for the USV on site.

Fig 7.1r Mini display

LoRa Module (Wireless Communication Procedures) (Max/Daniel) #10 The LoRa module on the USV is always receiving commands from the Base Station except when the Base Station sends a request for sensor data while in Auto Mode. Since we are using one module on the USV and one on the Base Station, we have a half-duplex system. As seen in Fig. 7.1f, the LoRa modules uses 3.3V and its Rx and Tx pins connect to the UART Tx and Rx pins on the USV Raspberry Pi (MCU2) and Base Station Raspberry Pi (MCU1). The datasheet for the LoRa module [12], its AT Command Guide [11], and a youtube tutorial on AT Commands [8] were referenced while writing the software for this module.

Before the main loop in the USV code, the the array of waypoints from the Base Station are received and the longitudes and latitudes are parsed. The waypoints are sent in two waves. The first wave sends the longitude values for the waypoints three times. The seconds wave sends the latitude values for the waypoints three times. The waypoints are sent in two waves to decrease the size of the string transmitted over LoRa. They are sent three times to lower the chances that the USV misses them. The longitude string's format is: "_longitude1?longitude2?longitude3?longituden!". In this format the longitude values are separated by question marks, the list begins with an underscore, and is terminated by an exclamation point. The number of longitude and latitude values depend on the number of waypoints selected, n. The latitude string's format is the same as the longitude except the character starting the list is an equals sign.

In the main loop, the messages sent that are in the format: "+RCV=ADDRESS,MESSAGE LENGTH,*request&mode^throttle%steering+" are parsed. +RCV means that the LoRa module received this message. ADDRESS is the address of the module that transmitted the message. MESSAGE LENGTH is the number of characters in the message string. Mode is a flag that is set by the Base Station. If mode is 0 then the USV switches to auto mode and if mode is 1 then the USV switches to manual mode. Throttle is the value determining the speed of the brushless motor. Steering is the value determining the angle of the servo motor. Request is a flag set by the Base Station. If it is one, the LoRa module is set in transmit mode and sends the current coordinates and orientation of the USV to the Base Station LoRa module. It sends message in the format: " *longitude&latitude^orientation+".

When the Base Station sends a request for sensor data in auto mode, the Base Station LoRa module is put in receive mode and waits 4 seconds for the message to come back. When the message is received it parses through the string and stores the coordinates and orientation so that they can be displayed on Screen 2 and 3. If no message is received, the LoRa module will attempt to receive the data 15 more times before it times out by repeating the steps above.

The messages received by the USV and Base Station are parsed using the re library from the Python standard library. This library parses strings using its match functionality, sectioning it into groups. For the receive functionality in the main loop, group 3 is the request flag, group 4 is the mode flag, group 5 is the throttle value, and group 6 is the steering value. If the LoRa module receives a noisy message where one of the separating characters (*,&,^,%,+) are not present, the match functionality of the re library will be unable to parse the string. In this case, we keep the previously received values. For the waypoint acquisition the re library was not used. This is done to account for a different number of waypoints on each run. This implementation splits the string based on its separating, starting and delimiter characters. If the received string starts with an underscore, the values are stored in an array for longtitudes. If the received string starts with an equals sign, the values are stored in an array for latitudes. The messages received by the Base Station are parsed using the re library. If the received message does not contain the separating characters (*,&,^,+) then the previous values are stored for the USV's location and orientation. The libraries that we used for transmitting and receiving with the LoRa modules is pySerial.

Limitations and Design Constraints:

When selecting the wireless communication protocol we had to find a balance between the range and latency of the communication. LoRa has very long range but has a high latency. While a low latency would be preferred, the range offered by LoRa made it a requirement to successfully meet our design constraint of 2km. To account for the innate latency with using this module the values in the commands are stored and used until the next command is received. If this was not implemented, the USV be at a standstill for most of it's runtime. The amount of noise in the signal of the received message varied with location. When testing indoors with a line of sight, the signal to noise ratio in a message was about 40. When outdoors, the signal to noise ratio was about 70. Since the parsing method used ignores a command if any separating character is not transmitted, about 3 out of 10 messages are missed when outdoors. Since we store the previous command, missing a command is not that critical. The live footage from the FPV camera can always be used to see if a switch in modes is successful.

8 * Technical Problem Solving

8.1 * The communication delay with controller problem

Problem(Daniel): The communication delay between the controller and the USV made the process of having real-time control very difficult. This is because it impacted the user's ability to navigate the vessel smoothly. What made this problematic is the delayed responses will risk unsafe navigation in dynamic aquatic environments.

Identification(Daniel): This issue became apparent during initial testing sessions, where it was observed that the controller inputs were not being immediately translated into corresponding actions by the USV.

8.2 * Solving the Communication Delay with Controller Problem

Solution(Daniel): To address this problem we first identified the root cause of the delay, which was attributed to the complexity and communication speed of the LoRa device used. To mitigate this delay, the controller was configured to send fixed values representing full left, full right, and full throttle, simplifying the control scheme to ensure more predictable behavior. Additionally, we optimized the communication protocol between the controller and the USV to include error-checking mechanisms to ensure accurate data was being transmitted and received. We also swapped over from the joystick which uses a potentiometer and used the D-Pad instead which is better for high low values. One of the skills learned was finding a balance between how easy it was to control the values and how functional the inputs were going to be. Nonetheless, also learning how to optimize a communication module and produce a response time capable of operating the boat safely.

8.3 * The Communication Delay between Base Station and USV Problem

Problem(Daniel, Max): The communication delay between the Base Station and the USV caused the autonomous missions to not work ideally. This delay due to the LoRa module increased the risk of the USV deviating from its intended path and potentially causing safety hazards. Moreover the communication delay between the controller and the USV made the process of having real time responses very difficult. This could be seen when the USV would stop moving while waiting for new base commands or even during the startup where the waypoints couldn't be received from the USV.

Identification(Daniel, Max): The issue was identified during the integration testing where we saw that the commands from the Base Station were not reaching the USV in a timely manner. This can be seen on the USV screen printing that no commands have been received.

8.4 * Solving the Communication Delay between Base Station and USV Problem

Solution(Daniel, Max): In order to address the delay we optimized the communication protocol between the Base Station and the USV. This included implementing a repeat transmission, and an error checking loop to send the same information multiple times from the Base Station until the USV receives it. This ensured the timely start of its route. Some takeaways from this experience include learning the importance of robust communication protocols in ensuring reliable data transmission, especially in environments where signal interference or packet loss can occur. Additionally, we gained insights into the complexities of coordinating between the Base Station and the USV, highlighting the need for clear and efficient communication strategies.

8.5 * The waterproofing electronics problem

Problem(Daniel, Alyan): The need to waterproof all electronics on the USV arose to protect them from water damage, ensuring the system's reliability in aquatic environments. Failure to waterproof the electronics could lead to system failure which would compromise the entire mission.

Identification(Daniel, Alyan): This requirement was recognized early in the design phase, highlighting the importance of sealing all electronic components from water ingress.

8.6 * Solving the X waterproofing electronics problem

Solution(Daniel): To achieve waterproofing, epoxy was used to seal the electronics which were then placed in a plastic tackle box. Mounting them on perforated boards helped organize the wiring and sealing the plastic deck with adhesive further enhanced the waterproofing measures. Even the Lidar was water resistant by adding epoxy to the circuit board making it at least splash resistant. This solution ensured that the electronics remained functional and protected from water damage throughout the USV's operation. Through this process, we learned valuable lessons about the importance of waterproofing electronics in marine environments and the techniques required to ensure their protection.

8.7 * The magnetic interference from motors problem

Problem(**Daniel, Alyan, Max**): Magnetic interference from the motors was affecting the accuracy of the magnetometer, impacting the USV's ability to maintain correct heading information. This interference could lead to navigation errors and pose a safety risk, especially in environments where precise heading information is crucial.

Identification(**Daniel, Alyan, Max**): This issue was discovered during system testing when the magnetometer readings were found to be inconsistent and fluctuating, correlating with the operation of the motors.

8.8 * Solving magnetic interference from motors Problem

Solution**(Daniel, Max)**: To mitigate the interference, the magnetometer was carefully placed furthest away from the magnetic field generated by the motors. Additionally, proper mounting techniques were employed to reduce the impact of interference on the sensor. This solution improved the accuracy of the magnetometer readings, however due to calibration problems the sensor was still not accurate enough to be feasible. Through this experience, we learned about the effects of magnetic interference on sensor accuracy and the techniques to minimize its impact, enhancing our understanding of sensor calibration and mounting in electromagnetically noisy environments.

8.9 * The UART communication problem

Problem (**Daniel,Max**): When testing LoRa communications between the Base Station and the USV we noticed the code would continuously jam despite receiving correct inputs verified through the Raspberry Pi (MCU2) and the serial monitor from the Arduino (MCU 3). This problem persisted occurring randomly during testing and took a while to trace where the issue in communication was. Initially the LoRa module was the assumed culprit however we saw that transmission was successful so the serial plot was checked once again to confirm the Arduino was receiving correct values. We found the correct value was sent to the Arduino however, the code would freeze after a few iterations.

8.10 * Solving UART communication problem

Solution: (**Daniel,Max)**: To find the issue with the freezing code the first step taken was view the inputs from both the Raspberry Pi and Arduino. After verifying that the sent values were correct and tracing back the communication path it was found that the bug was somewhere in the UART communication between Arduino and Raspberry Pi. After testing a variety of print statements to find where in the code the issue lied it was found that printing to the serial monitor itself was the issue. By removing all serial print statements from the Arduino code the UART communication worried flawlessly. We presume that the UART communication was jamming when trying to send and receive the control values and the serial monitor information.

9 User Interface Design

9.1 Application Control

Waypoint Selector: (Max) The main screen of the Base Station will be used to display the Waypoint Selector and terminal output from the Base Station script. The Waypoint selector opens on startup of the Base Station code. A new window appears with a map of the current testing location. If the user left clicks on the map, a waypoint will be created with a designated number. This number increases with the amount of waypoints the user creates. Once the user right clicks the map the waypoints will be submitted and transmitted to the USV. The general process is described in Fig. 9.1a below.

Figure 9.1a High Level Waypoint Selector Flowchart

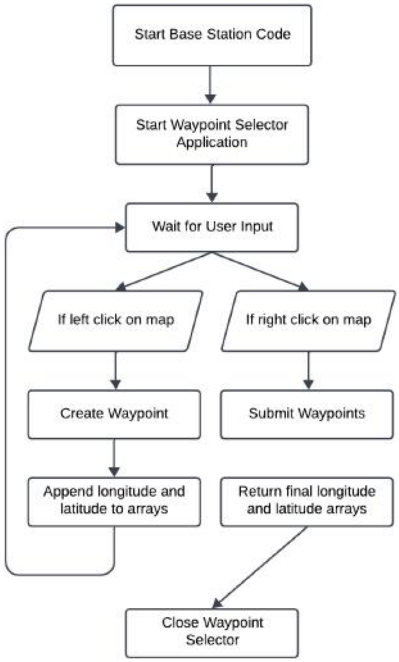


Figure 9.1b Waypoint Selector

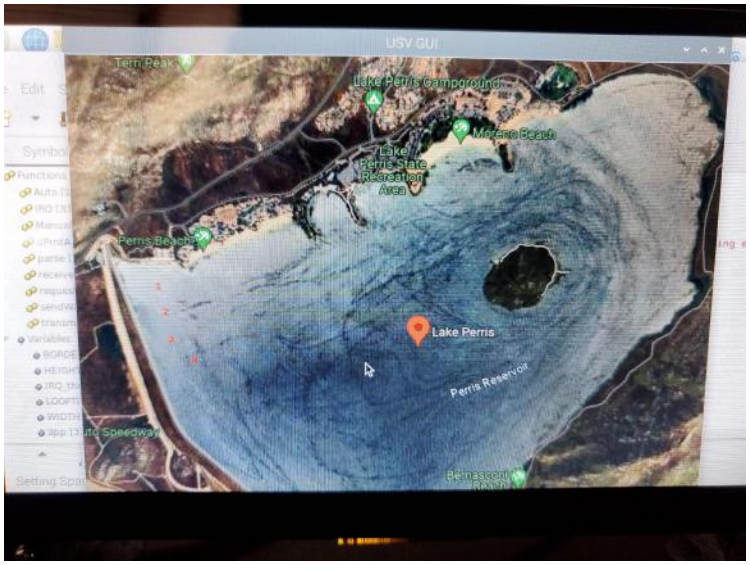


Figure 9.1c System Status Updates

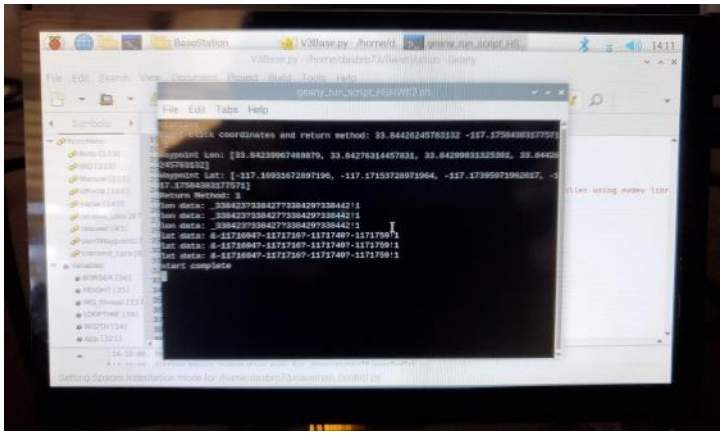


Figure 9.1d OLED Screen Messages

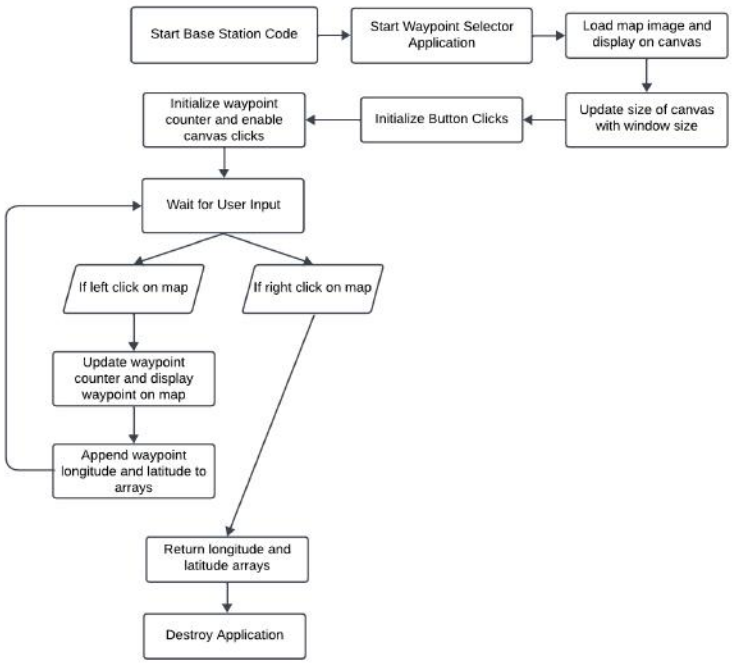
9.2 User Interface Screens

FPV Screen: (Daniel) The FPV Screen (Screen 1) shows the live footage received from the FPV transmitter on the USV. There is a button on the screen to start and stop recording. There is a power button to turn on and off the screen.

Waypoint Selector: (Max) The Waypoint Selector will be displayed on the main screen of the Base Station (Screen 2) as described in Section 9.1. The low level logic for the process carried out by the Waypoint Selector is shown in Fig. 9.2a below. The screen for the Waypoint selector is also used to monitor the USV's current mode, throttle, and steering values. It also shows the coordinates and orientation of the USV during auto mode. This screen is shown in Fig. 9.1c above.

Status Screen: (Daniel) A small OLED screen (Screen 3) is on the Base Station that displays the status of the system and coordinates and orientation of the USV. When the Base Station code first starts, the OLED displays a "Retrieving Waypoints" message. Once the waypoints are selected in the Waypoint Selector, a "Sending Waypoints" message appears. Once the Base Station is done transmitting the waypoints, the OLED will display "In Manual Mode" and "Press Y to Exit" if the system is in manual mode. If the system is in auto mode the OLED display will show "In Auto mode" and "Press B to Exit." Once the coordinates and orientation of the USV are received in auto mode the screen will also show the message "Lat: ", "Lon: ", "Mag: " where Lat is the latitude, Lon is the longitude, and Mag is the orientation. The messages displayed on the OLED screen are shown in Fig. 9.1d above.

Figure 9.2a Low Level Waypoint Selector Flowchart



10 * Test Plan

10.1 * Test Design

Test #1 control bench test (Daniel)

1. Objective: test controller input to make sure input is correctly interpreted and correct servo or motor action is taken.
2. Set up the USV control boards and directly wire and set up the Xbox controller for wired input for controls.
3. Input commands for steering and throttle and look at serial monitor outputs to confirm values are correct and actions are performed correctly
4. If set up correctly the controller inputs shout output the desired throttle speed and steering angle.

Test #2 PD test (Max)

1. Objective: Test proportional and derivative coefficient strengths to allow a smooth change in heading in auto mode.
2. Place the USV in water and select a waypoint in front of it. Select small coefficients at first.
3. Set USV to auto mode and see if it moves straight forward. If it does not then lower the proportional coefficient until it does. When updating coefficients do so in very small increments. Once the USV moves straight, test a waypoint to the right or left. If the USV overshoots the turn lowers the proportional coefficient. If it undershoots, raise the proportional coefficient. If it seems like changing the proportional coefficient is no longer improving the turning then do the same procedure above with the proportional coefficient with the best result while modifying the derivative coefficient.
4. The PD controller needs to be tuned so at first the USV will not behave correctly at all. After testing and modifying the proportional and derivative coefficients the USV will be able to accurately track waypoints.

Test #3 Ultrasonic Sensor Test (Alyan)

1. This test is to ensure accurate distance measurements from our ultrasonic sensors in different mediums.
2. Start code for retrieving distance measurements from ultrasonic sensors.
3. Place an object at a known distance from the sensors then read and record the distance measurements from the ultrasonic sensors. Repeat this test in different mediums(water, juice, lake). Compare the measured distances with the actual distance. If the measurements are inaccurate, adjust the sensor angles/positions or method of calculation.
4. The expected result in air is that the ultrasonic sensors should accurately measure the distance within a small margin of error. In the water the ultrasonic sensors distances should read a higher value due to speed of sound traveling faster in water. Adjustments to the sensor's calculations should be necessary. In juice and in the lake water similar results should be expected.

Test #4 Object detection(Alyan/Max)

1. This test is to make sure that the Lidar and Ultrasonics can consistently detect an object within a certain distance of the USV.
2. Set up USV in an indoor environment so that it is easier to test the Lidar and Ultrasonic sensors.
3. Put the USV in auto mode and check if Lidar detects nothing when nothing is in range of the USV. If Lidar detects something then lower the range so the test can be more reliable. Do the same for the Ultrasonics. Once the Lidar and Ultrasonics are not detecting anything when nothing is within the range, put something in the range and see if an object is detected. Once this works, increase the distance thresholds back. The object detection will now function properly.
4. The Lidar and Ultrasonic sensors should be reliable in detecting an object.

Test #5 Hull water leakage test (Daniel)

1. Test the hull for any leaks or pin holes that would make the hull take on water.
2. The boat hull base would be weighed down in the water with 20 pounds and allowed to float for an hour to test bonded seams
3. The hull was wiped clean and dry to make sure any water or dust was removed prior to testing.
4. If sealed properly the hull should have little to no water accumulate inside after the test. It is ok for a little water to enter so long as it does not persist and sealing is adequate to avoid sinking or damage to internal components.

Test #6 GPS positioning (Max)

1. This test is to measure the precision of the GPS module.
2. Plug the GPS module into the laptop with UART to USB converter and begin reading coordinates.
3. Drive to different locations and compare the GPS module readings with the actual coordinates of your current location. If within 10 meters then stop testing. If not within 10 meters, check calculations in software.
4. The GPS module will have a precision of 10 meters if the calculations are correct in software.

Test #7 Compass test (Alyan/Max)

1. Objective of this test is to get accurate readings from our magnetometer.
2. Start code for retrieving orientation of compass.
3. Compare the current orientation of the compass with the actual orientation. If the orientations are more than 10 degrees off, calibrate the compass. Repeat this until the compass gives accurate data.
4. The compass will need to be calibrated before use. This procedure will make sure that it gives accurate readings, applying the hard iron scalars and offsets to the compasses raw data.

Test #8 LoRa test (Max/Daniel)

1. This test will measure the latency and noise level for communication with the LoRa modules.
2. Set one LoRa module to constantly receive. Set the other to transmit a test string. Do this in an inside environment with a line of sight between the modules.
3. Measure the amount of time that it takes for the receiving LoRa module to get the string. Display the received string. The message will contain the Received Signal Strength Indicator (RSSI) of the received message. Repeat this test outdoors with a line of sight between the modules and see how the latency and RSSI changes.
4. The LoRa modules will have a lower latency outside since the RSSI will be much higher. If the LoRa modules are not in a line of sight, the message will probably not be received.

Test #9 Manual Test (Daniel)

1. This test will be done to ensure the USV can be controlled manually from the Base Station.
2. Run USV Code and place in water. Run Base Station code.
3. First test the thruster of the USV. If the D-Pad up arrow is pressed and the USV does not start moving forward then something is wrong. If the D-Pad down arrow is pressed and the USV does not slowly reverse then something is wrong. Then test the steering of the USV. If the left D-Pad arrow is pressed and the USV does not turn left then something is wrong. If the right D-Pad button is pressed and the USV does not turn right then something is wrong.
4. If the D-Pad up arrow is pressed the USV should move forward. If the D-Pad left arrow is pressed the USV should turn left. If the D-Pad right arrow is pressed the USV should turn right. If the D-Pad down button is pressed the USV should reverse.

Test #10 Auto Test (Max)

1. This test will be done to ensure that the USV can use the PD controller if no objects are detected and adjust the heading if there is an object detected.
2. In auto mode the USV maintains a constant throttle value so it can be disabled for this test. This test will be completed outside of water for ease of controlling the USV's current heading. Place the USV on a flat table so that the propeller hangs off the side. Start the USV code and Base Station code. Select test waypoints on the Base Station and put the USV in auto mode. Face the USV in the direction of the first waypoint. Make sure that no objects are being detected when starting.
3. Since the propeller is facing it's next waypoint, the propeller should be centered. If the propeller is not in the center and no objects are detected then there are incorrect calculations in the PD controller. Place an object in front of the Lidar or ultrasonic sensors and the propeller direction should adjust. Remove the object and the propeller should re-center. Turn the USV clockwise and counterclockwise and observe the propeller's direction. Place an object in front of the USV when it is in a different orientation and observe the resulting propeller direction.
4. If there are no objects detected, the propeller should move left if the USV turns clockwise. If the USV is turned counterclockwise the propeller should move right. As you return the USV to the starting position where it is facing the first waypoint the propeller should return to the center. If at any time during the test an object is placed in the view of the Lidar sensor or ultrasonic sensors the heading should adjust accordingly.

Test #11 Full System Test (Max)

1. This test will be done to ensure smooth transitions between the manual and auto modes of the USV. If this transition is not reliable this can lead to problems with retrieving the USV or positioning it to a desirable location.
2. Place the USV on a flat table so that the propeller hangs off the side. The auto mode throttle value should be set so that the propeller does not spin so that it can be tested out of water. Start the USV code and Base Station code. Select test waypoints on the Base Station. Ensure the USV is in manual mode.
3. Press left or right on the D-pad to ensure that the propeller changes direction with User Input. Switch to Auto Mode. Test auto mode using the same procedure in Test 9. Press B on the controller to switch the USV to manual mode. Press left or right on the D-pad to determine if the switch was successful.
4. Auto and manual mode should function the same as they did while running individually. The transition from manual to auto and back should be reliable.

Test #12 Waypoint Following Test (Max)

1. This test will show that the USV can direct itself toward waypoints.
2. Start the USV code and place it in the pool. Start Base Station Code and select one waypoint about 30 meters away in the direction that the USV is currently facing.
3. Put the USV in auto mode. Observe how the USV corrects itself to head toward the waypoint. Reset the USV and select a waypoint that is 30 meters to the right of it. Put the USV in auto mode and observe the behavior. Reset the USV and select a waypoint that is 30 meters to the left of it. Put USV in auto mode and observe the behavior.
4. For the first waypoint in this procedure, the USV should travel straight since it is currently at the correct heading. For the second waypoint in the procedure, the USV should gradually turn right to orient itself towards the waypoint. For the third waypoint in the procedure the USV should gradually turn left to orient itself towards the waypoint.

10.2 * Bug Tracking

GPS (Max):

The coordinates that were received after parsing the longitude and latitudes from the raw data outputted by the GPS module were extremely off. While testing at home, our parsed longitude and latitude values were a few cities away. This was caused by improper handling of the coordinates output from the module. If the GPS module outputted this for it's GPLL line:

$GPGLL,3353.35925,N,11728.05471,W,063946.00,A,D*74

We would originally take 3353.35925 and 11728.05471 and divide them both by 100 to get the longitude and latitude values. The correct way to process those two numbers is to read it as 33 degrees 53.35925 minutes North and 117 degrees 28.05471 minutes West. The minutes values are divided by 60 and added to the degree to get the correct longitude and latitude values.

Magnetometer (Max/Alyan):

When testing the magnetometer while mounted in the USV, we found that there was magnetic interference from the permanent magnet in the motor used to spin the Lidar. While we were not able to solve this problem cleanly, we were able to create our own offsets that would get the compass within 10 degrees of the actual measurement. If time allowed, this would have been solved by moving the magnetometer away from the Lidar module.

Lidar (Max/Alyan):

When running the Lidar scan in a separate thread alongside the rest of the USV code, sometimes we would get an error. This error stated that the starting bits did not match when trying to scan. This error was caused by a messy termination of the Lidar when halting the USV code. The Lidar would not disconnect which doesn't allow it's memory to be wiped and causes the error. This was fixed by proper termination of the Lidar module when stopping the USV code.

Controller Debugging (Daniel):

When first setting up the controller inputs for the base station, there were occasional garbage values that would lead to random outputs. This took some time to filter and get rid of these values to achieve smooth control over the USV throttle and steering outputs. To debug the inputs were sent through a wired connection to the USV control board. Then control inputs were sent and verified all the way down the communication chain to check if the value was correctly communicated.

Reliable Mode Switching (Max):

When the USV is in auto mode, the command from the Base Station to return to manual mode was often missed due to sensor data acquisition time. It takes about 2 seconds to get all of the sensor data. This was fixed by making the auto mode code only run whenever it receives a command from the Base Station. The Base Station sends a command to run the auto mode code periodically. This allows enough time for the PD controller, object detection algorithm, and sensor data acquisition functionalities to finish running before a command is received.

10.3 * Quality Control

Control Bench Test:

Test 1:

Date: 11/16/2023

Result: Pass

Note: bse controller code worked for servo control and for ESC controller for operation the brushless motor

PD Test:

Test 1:

Date: 03/03/24

Result: Pass

The propeller moved to the left when the USV was rotated counterclockwise and moved to the right when the USV was rotated clockwise. When facing the ideal heading of the USV (towards the waypoint) the propeller was centered.

Ultrasonics Test:

Test1:

Date: 10/29/23

Result: Pass

The ultrasonics successfully provided accurate distance measurements in all the different mediums with slight calculation adjustments to the sensor placed in the water. Since the speed of sound was about 4x faster in water than air the ultrasonic used for the depth sensor took into account this speed ratio.

Object Detection Test:

Test 1:

Date: 03/03/24

Result: Pass

Object was successfully detected by both Lidar and Ultrasonics and the propeller moved to steer out of the way. When removing the object, the propeller went back to starting position.

Hull Water Leakage:

Test 1:

Date: 11/26/2023

Result: Pass

Note: very minimal water found inside the hull due to small leakage around the propeller drive shaft and stuffing tube.

GPS Positioning:

Test 1:

Date: 01/25/24

Result: Failure

The coordinates would say that the module is a few cities over.

Test 2:

Date: 02/05/24

Result: Pass

Fixed the software issue where longitude and latitude were not calculated correctly. Readings now have a 10 meter precision.

Magnetometer Test:

Test 1:

Date: 11/20/2023

Result: Fail

No matter what the readings from the compass were not accurate.

Test 2:

Date: 02/25/2024

Result: Pass

After calibrating the magnetometer the compass was still off due to the Lidar motor. This was fixed by adding offsets to the raw values which gave an acceptable degree of precision.

LoRa Test:

Test 1:

Date: 01/25/24

Result: Pass

The LoRa modules were able to successfully send a test string.

Test 2:

Date: 02/15/24

Result: Pass

We were also able to get the LoRa modules to have two-way communication.

Manual Mode Test:

Test 1:

Date: 03/08/2024

Result: Pass

Note: the USV was able to be controlled and positioned using manual mode on the lake. The system was not perfect as the LoRa latency made control of the USV in the water a little clumsy however it was able to position the vessel.

Auto Mode Test:

Test 1:

Date: 03/08/2024

Result: Fail

The USV was positioned 30 feet from a waypoint and set to sail to the location. It initially headed toward the waypoint however, began running in circles due to calibration errors for the compass. This heading miscalculation ended up running the auto mode code into a loop causing the USV to constantly circle.

Test 2:

Date: 3/09/2024

Result: Pass

The USV was able to successfully track the waypoint set in the pool. When the waypoint was in front of the USV, the USV went straight but was drifting from left to right. When the waypoint was to the right of the USV, the USV turned right.

Full System Test:

Tested 03/08/2024 - 03/09/2024

Test 1:

Date: 3/08/2024

Result: Fail

The USV was able to switch between manual and auto mode, was able to be controlled in manual mode, but the auto mode did not work as intended. The USV began running in circles due to calibration errors from the compass.

Test 2:

Date: 03/09/2024

Result: Pass

The USV was able to switch between manual and auto mode, was able to be controlled in manual mode, and auto mode correctly adjusted the heading of the USV so that it moved toward the waypoint. When the USV was about to run into the wall of the pool it adjusted its heading to try and avoid it.

Waypoint Following Test:

Test 1: 03/08/2024

Result: Fail

Similar to Test 1 for the Full System Test, the USV functioned properly until it got to auto mode. In auto mode the USV would spin in circles.

Test 2: 03/09/2024

Result : Pass

The USV successfully oriented itself towards the waypoint and maintained a constant heading until reaching the end of the pool.

10.4 * Identification of critical components

During testing the most critical component is the waterproofing of the USVt. Although the USV is waterproofed the state of the USV must be checked after every test in the water to ensure no water leakage. This is essential as water could short the expensive sensors or boards in the USV or the batteries might cause a fire. Worst case scenario the USV could sink and become unrecoverable. For this reason, waterproofing procedures were a huge consideration when designing the USV.

Another critical component of the system is the communications on both the USV and base station. These are critical because all user commands from the base station to the USV rely on wireless communications. Without reliable communications the user would have no control over the USV which could be dangerous.

10.5 * Items Not Tested by the Experiments

Not Applicable. All functions and processes were tested.

11 * Test Report

11.1 Control Bench Test

- **Test result (performed by Daniel):** The control test was conducted using the wired setup to the USV board as stated in section 10. This tested all user inputs in manual mode for steering and throttle values to see if the correct outputs occurred.
- **Comparison with expected results:** After setting up and debugging the code, the wired control test performed as expected giving correct outputs for steering and throttle input.
- **Analysis of test results:** Although the test was successful, it was noticed that the total power output of the motors was way too high so it was limited in the code to reduce the full throttle power.
- **Corrective actions taken:** To limit the throttle output, the max throttle value in the base station d USV code was limited to ensure no damage to drive components would happen.

11.2 PD Test

- **Test result(performed by Max):** The propeller moved to the left when the USV was rotated counterclockwise and moved to the right when the USV was rotated clockwise. When facing the ideal heading of the USV (towards the waypoint) the propeller was centered.
- **Comparison with expected results:** The test coincided with the expected results. It fits the Design Constraint.
- **Analysis of test results:** The PD controller correctly calculated the desired heading of the USV when the orientation of the USV changes with respect to the waypoint. The proportional and derivative coefficients were changed to 2 and 0.5 respectively.
- **Corrective actions taken:** The proportional and derivative coefficients were updated to have a better tuned PD controller.

11.3 Ultrasonics Waterproof and measurements Water Test

- **Test result (performed by Alyan):** The ultrasonic JSR sensors were tested in air, water, juice, and at the lake. The sensors maintained functionality and provided accurate distance measurements even when submerged which indicated that the waterproofing was also effective.
- **Comparison with expected results:** The results aligned with our expectations as the sensors datasheet claimed them to be waterproof and able to operate in aquatic environments.
- **Analysis of test results:** The sensor's performance in the different mediums demonstrates it was suitable to be used for aquatic environments and that it was feasible for the purpose we needed it to be used for on the USV. We can see this during the data measurements as they aligned with the corresponding metrics from a ruler that was used to test the distance.
- **Corrective actions taken:** There were no corrective actions taken based on the test results.

11.4 Object Detection Test

- **Test result (performed by Max, Alyan):** The object detection test was successful. The Lidar and ultrasonic sensors consistently detected objects within the specified distance range.
- **Comparison with expected results:** The comparison with the expected results was almost identical as for the most part the sensors when an object was placed in front would read that an object was detected and when removed the sensor shouldnt detect anything.

- **Analysis of test results:** The Lidar and Ultrasonic sensors were set to detect a range of distances and when no objects were present, both sensors correctly did not detect anything. In addition, when objects were placed within the sensors' range, both sensors successfully detected them.
- **Corrective actions taken:** There were no corrective actions taken based on the consistent readings of the two detection modules.

11.5 Hull Water Leakage Test

- **Test result (performed by Daniel):** When testing the hull for leaks, the hull was filled with 20 pounds of weight and allowed to float in the pool for an hour. This allowed pressure to build underneath the hull to test the seams where the hull was joined.
- **Comparison with expected results:** After the hour of testing the Hull leakdown was as expected with only about half a teaspoon worth of water accumulating inside. This occurs due to some small leaks around seams and around the drive shafts.
- **Analysis of test results:** This test went as expected as the hull had minimal leakage despite holding more weight than its final launch weight. This test proved that the hull itself was seaworthy and ready to fit sensors and electronics inside.
- **Corrective actions taken:** Extra epoxy was added around seams to ensure they were sealed.

11.6 GPS Test

Test 1:

- **Test result (performed by Max):** The code that I was using to process the GPS raw data gave latitude and longitude values that were extremely far off.
- **Comparison with expected results:** The expected results were a precision of 10 meters and the actual results were extremely far off.
- **Analysis of test results:** The processing of the raw data from the GPS module is not done correctly
- **Corrective actions taken:** Adjusted the way the parsed data calculates the longitude and latitude values. The minutes are separated from the degrees and divided by 60 and added to the degree values for longitude and latitude.

Test 2:

- **Test result (performed by Max):** Drove around with GPS module to test precision at different locations.
- **Comparison with expected results:** Each comparison with the actual coordinates of the location and the processed values from the GPS module are about 10 meters of precision.
- **Analysis of test results:** The GPS module and processing code now provides the correct longitude and latitude coordinates.
- **Corrective actions taken:** No further actions need to be taken in order to meet the design constraints

11.7 Magnetometer Magnetic Test - Magnetic Interference

Test 1:

- **Test result (performed by Alyan, Max):** The magnetometer was tested in an environment with no magnetic interference and with interference from an iphone magnet. The sensor provided did not provide accurate heading measurements.
- **Comparison with expected results:** The results aligned with our expectations as the magnetometer should be calibrated each time to provide accurate heading measurements without any external interference from any magnets.
- **Analysis of test results:** The magnetometer's performance in the absence of magnetic interference demonstrates its ability to provide some heading information but it was too far off to be feasible and with magnetic interference it made it completely impossible.
- **Corrective actions taken:** There were corrective actions taken based on the test results which included calibrating the compass and moving it further away from any magnetic interference.

Test 2:

- **Test result (performed by Alyan, Max):** The magnetometer was tested while installed on the boat. The sensor exhibited deviations in heading measurements, indicating that the magnetic field generated by the boat's components, specifically the Lidar motor was affecting its readings.
- **Comparison with expected results:** The results were not expected, as we anticipated that the positioning of the sensor in contrast with it being in the tackle box that the magnetic field generated by the boat's components would not influence the magnetometer's readings.
- **Analysis of test results:** The magnetometer's performance on the boat showed us that we needed an offset to incorporate for the heading being off by the Lidar in order to account for the slight degrees off.
- **Corrective actions taken:** To address the magnetic interference from the boat's components, we wanted to explore alternative sensor mounting locations but for now we calibrated the compass and added offsets to the raw values which gave us an acceptable level of accuracy.

11.8 LoRa Communication Testing

- **Test result (performed by Daniel, Max):** Evaluate LoRa communication range and reliability in an indoor and outdoor environment.
- **Comparison with expected results:** The latency and RSSI were similar to the expected. When testing in Winston Chung Hall at the University of California Riverside we experienced a very low RSSI and an extra high latency. This was not the case when testing outside.
- **Analysis of test results:** The LoRa modules must be operated with line of sight and they do not perform as well inside as they do outside.
- **Corrective actions taken:** Since the LoRa modules are used outside with line of sight no corrective actions needed to be taken.

11.9 Manual Test

- **Test result(Daniel):** Manual mode was tested on the lake and was able to get the USV positioned on the water.
- **Comparison with expected results:** The USV maneuvered as expected however, lag from the LoRa modules made the control less than ideal. The alignment of the mechanical components also contributed to some drift.
- **Analysis of test results:** After testing manual mode some small adjustments were made to the drive components in an attempt to correct the drift.
- **Corrective actions taken:** Controls limited to reduce speed
- Link to video https://youtube.com/shorts/eYrNMEMWscw?si=drgkWX5L1MKJRkpE

11.10 Auto Test

Test 1:

- **Test result(Max):** The USV was positioned 30 feet from a waypoint and set to sail to the location. It initially headed toward the waypoint however, began running in circles.
- **Comparison with expected results:** The USV was supposed to track the waypoint and maintain a constant heading towards that waypoint.
- **Analysis of test results:** The USV failed to follow the waypoints due to calibration errors for the compass. This heading miscalculation ended up running the auto mode code into a loop causing the USV to constantly circle.
- **Corrective actions taken:** The compass was calibrated and additional offsets were added to raise its accuracy.

Test 2:

- **Test result(Max):** After magnetometer calibration, the USV
- **Comparison with expected results:** The USV was able to successfully track the waypoint set in the pool. When the waypoint was in front of the USV, the USV went straight but was drifting from left to right. When the waypoint was to the right of the USV, the USV turned right.
- **Analysis of test results:** The USV now meets the design requirements of being able to follow waypoints. The calibrated magnetometer works well enough to maintain a steady heading.
- **Corrective actions taken:** No corrective action needed to be taken.

11.11 Full System Test

Test 1:

- **Test result(Max):** Tested both manual and auto mode in the water.
- **Comparison with expected results:** The USV was able to switch between manual and auto mode, was able to be controlled in manual mode, but the auto mode did not work as intended. The USV began running in circles due to calibration errors from the compass.
- **Analysis of test results:** The magnetometer needs to be calibrated in order for the auto mode to function properly.
- **Corrective actions taken:** Calibrated the magnetometer.

Test 2:

- **Test result(Max):** Tested manual and auto mode in water again.
- **Comparison with expected results:** The USV was able to switch between manual and auto mode, was able to be controlled in manual mode, and auto mode correctly adjusted the heading of the USV so that it moved toward the waypoint. When the USV was about to run into the wall of the pool it adjusted its heading to try and avoid it.
- **Analysis of test results:** The USV now successfully runs in auto and manual mode.
- **Corrective actions taken:** None needed

11.12 Waypoint Following Test

Test 1:

- **Test result (Max):** Set a waypoint and put the USV in auto mode. The USV should attempt to reach the waypoint.
- **Comparison with expected results:** Similar to Test 1 for the Full System Test, the USV functioned properly until it got to auto mode. In auto mode the USV would spin in circles.
- **Analysis of test results:** The magnetometer needed to be calibrated in order to fix the USV auto mode issues.
- **Corrective actions taken:** Calibrated the magnetometer

Test 2:

- **Test result(Max):** Set three waypoints for the USV. One in front, one to the right, and one to the left. Put the USV in a pool and tested the tracking towards those waypoints.
- **Comparison with expected results:** The USV successfully oriented itself towards the waypoint and maintained a constant heading until reaching the edge of the pool for each waypoint. The heading swayed left and right of the ideal heading as it was heading towards the waypoints.
- **Analysis of test results:** The USV sways left and right while heading towards the waypoint but this is a common characteristic of a PID controller boat.
- **Corrective actions taken:** No corrective actions taken.

12 * Conclusion and Future Work

12.1 * Conclusion

Despite not accomplishing all of our initial project goals and technical design objectives, the USV project has been a tremendous learning experience for our team. Even though we've made great strides and accomplished a number of important benchmarks, there have been certain gaps in our performance and technological difficulties that have affected the project's overall success.

Control (Daniel / Max):
Manual Mode (Daniel): We successfully implemented manual control mode, allowing the user to operate the USV from the Base Station using an Xbox controller. This mode enables users to adjust throttle, steering inputs and manually set waypoints for autonomous missions. The underlying problem was the delay between the controller sending input commands using the LoRa and instead of having adjustable values as you toggle a button we just have all the way left, all the way right, and full-speed. We saw this delay through testing as the controller code worked very responsive when plugged in with a USB to the Raspberry Pi (MCU2). As a result, we learned that selecting a more optimal communication protocol can ensure faster real-time responsiveness.

Automatic Mode (Max): The automatic mode that was designed for the USV did achieve full autonomy. However, due to sensor calibration and LoRa inconsistency, the USV wasn't able to operate to our intended goal. The readings from the sensors being inaccurate at times also made it difficult to write a good PD algorithm to run the boat long distances in automatic mode. One of our most successful goals was the adjustments of the motor to turn towards the desired waypoint and the obstacle avoidance that uses a set value to turn around the item. In the future, we will explore more advanced control algorithms and sensor fusion techniques to improve autonomous navigation capabilities.

Power (Daniel):
The power section of the USV functioned according to plan, providing power to the controls, onboard computer, and peripheral electronics as intended. All systems ran off the main power supply, with a step-down DC circuit effectively managing different input voltages. This successful implementation ensured the reliable operation of the USV's electronics throughout the project.

Controls(Daniel): The steering and propulsion controls, operated by a brushless motor and servo, faced no major issues and performed as expected, with the Arduino managing throttle and steering controls effectively.

Sensing (Alyan / Max):
The Lidar sensor that is serving as the USV's primary ranged object detection system was effective in detecting obstacles 360 degrees around the vessel with an accurate range of 10 meters but we did run into some issues. initially, we faced challenges with the code running without error and stopping the code execution. To address this issue, we modified the code to fetch data from the Lidar sensor every 3 seconds instead of printing data for every rotation. This adjustment ensured that the code ran smoothly and allowed us to integrate the Lidar sensor effectively into the USV's sensing system. In the future, we would

consider using ROS for visualization and object detection. A key takeaway from this experience was learning how to optimize sensor data collection to balance accuracy with computational efficiency.

Ultrasonics (Alyan): The ultrasonic sensors, serving as a secondary object detection system, did achieve its goals of providing additional coverage for detecting objects to the left, right, front, and below the vessel.

GPS (Max): The GPS module successfully pings the USV's current location, allowing for continuous adjustment towards desired waypoints.

Magnetometer (Alyan): The magnetometer provided valuable heading information for the USV project, but its accuracy was compromised by calibration issues and magnetic interference from the boat's motors. The sensor's readings were consistently off by approximately 10-15 degrees, affecting the overall performance of the navigation system. This experience highlighted the importance of proper sensor calibration and the importance of sensor accuracy for a PD algorithm. In future projects, we will prioritize thorough calibration procedures and consider shielding the sensor from external magnetic interference to improve accuracy and reliability of the magnetometer.

Communication (Max / Daniel):
LoRa (Max / Daniel): The LoRa communication modules effectively sent necessary instructions between the Base Station and the USV, however, there were some limitations due to the transmit speed and reliability. There were instances where it would take up to 10 seconds to send a command, and sometimes the command wouldn't send at all. This issue could be attributed to interference, signal loss, or other environmental factors such as different mediums affecting the communication link. To address this challenge, we implemented error-checking mechanisms and optimized the communication protocol to keep sending data until it receives it in order to improve transmission speed and reliability. One key lesson learned during the selection of a good communication module is the importance of finding a good balance between latency and range.

FPV (Daniel): The FPV camera performed exceptionally well, providing a clear visual perspective of the USV's surroundings during missions

Drift Correction (Alyan):
The MPU6500 accelerometer provided accurate acceleration data for use in drift correction algorithms, however, with such limited water testing for the USV we weren't able to develop use for it in the autonomous mode calculations.

12.2 Future Work

The USV we designed has the potential for significant expansion and improvement in the future. The autonomous navigation algorithms and sensor fusion techniques developed for the USV could be further refined to enhance performance in complex environments, increasing navigation accuracy, long-range communication, size/power reduction, and obstacle avoidance capabilities. Integration of advanced machine learning algorithms could improve decision-making processes, enhancing the USV's adaptability and intelligence. Moreover, research focusing on improving communication systems, such as LoRa and FPV, could extend the range and reliability of data transmission, which is crucial for long-range monitoring and surveillance applications. This project demonstrates the feasibility and effectiveness of autonomous USVs, making it an attractive option for commercial and research applications, and suggesting potential for partnerships and funding opportunities. Its innovative design and functionality make it a compelling candidate for further exploration and development by future students or interested parties.

12.3 * Acknowledgement

Faculty in Charge: Roman Chomko
Instructor Teaching Assistant: Cody Simons
Faculty Advisor: Manglai Zhou
Online Tech Creators

13 * References

[1] Admin, "JSN-SR04T Waterproof Ultrasonic Sensor with Arduino Guide," *How To Electronics*, May 15, 2023. https://how2electronics.com/jsn-sr04t-waterproof-ultrasonic-sensor-with-Arduino-guide/

[2] "Add an OLED Stats Display to Raspberry Pi OS Bullseye," *www.youtube.com*. http://www.youtube.com/watch?v=lRTQ0NsXMuw&list=LL&index=9&t=2s.

[3] B. Engineering, "Building a Self-Driving Boat (ArduPilot Rover)," *Instructables*. https://www.instructables.com/Building-a-Self-Driving-Boat-ArduPilot-Rover/

[4] geo bruceBruce is on fire, "How to Use the RPLidar 360° Laser Scanner With Arduino," Instructables. https://www.instructables.com/How-to-Use-the-RPLidar-360-Laser-Scanner-With-Ardu/

[5] GoouuuTech,"GT-U7 GPS Modules," no datasheet number

[6] "Home - SLAMTEC Global Network," Sep. 07, 2023. https://www.slamtec.ai/.

[7] "How to use Bluetooth Controllers with Python on Raspberry Pi," *www.youtube.com*. http://www.youtube.com/watch?v=F5-dV6ULeg8&list=LL&index=14&t=405s.

[8] "LoRa - Reyax RYLR896 MODULE - GETTING STARTED WITH AT COMMANDS," *www.youtube.com*. https://www.youtube.com/watch?v=7Yqobsuzkn4&ab_channel=SaravananAL.

[9] "MPU-6500 | TDK." https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6500/

[10] "Real-Time Implementation Of PID Control For Unmanned Surface Vessel: A Practical Approach," *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/document/10234784.

[11] Reyax Technology Corporation Ltd., "LoRa AT COMMAND GUIDE," 56322E32 datasheet, Jun. 2022

[12] Reyax Technology Corporation Ltd., "RYLR896," 56312E37 datasheet, Nov. 2021

[13] Reyax Technology Corporation Ltd., ”USB to 1.8V/3.3V/5V UART bridge with 4-Bit Bidirectional Voltage Level Shifter,” 56312E36 datasheet, Nov. 2022

[14] “QMC5883P-QST CORPORATION LIMITED,” *www.qstcorp.com*. https://www.qstcorp.com/en_comp_prod/QMC5883P.

[15] “Using USB and Bluetooth Controllers with Python - Tutorial Australia,” *Core Electronics*. http://core-electronics.com.au/guides/using-USB-and-bluetooth-controllers-with-python/.

14 * Appendices

Appendix A: Parts List

- ☐ Raspberry Pi 4 Model B
- ☐ Arduino Uno
- ☐ Lidar sensor (Adafruit)
- ☐ Ultrasonic sensors
- ☐ Accelerometer
- ☐ GPS module
- ☐ Magnetometer
- ☐ FPV camera
- ☐ Brushless motor
- ☐ Servo motor
- ☐ ESC (Electronic Speed Controller)
- ☐ Xbox controller
- ☐ Aluminum hull
- ☐ Tupperware container (for electronics housing)
- ☐ Various electronic components (wires, connectors, etc.)
- ☐ perforated PCB boards
- ☐ Batteries

Appendix B: Equipment List

- ☐ Soldering iron
- ☐ Multimeter
- ☐ Power supply
- ☐ Computer (for programming and testing)
- ☐ mouse
- ☐ keyboard
- ☐ Various hand tools (screwdrivers, pliers, power drill, hammer,Rivet gun,sheet metal scissor, tap and die set etc.)
- ☐ rivets, screws and washers
- ☐ tap and die set
- ☐ Battery chargers
- ☐ Silicon grip Gorilla tape adhesive

Appendix C: Software List

All software used in the project is available on the team's GitHub repository:

https://github.com/maxwcurren/Unmanned-Surface-Vehicle/tree/main/USV

Appendix D: Special Resources

No special resources were required for this project.

Appendix E: User's Manual

The user's manual for operating the USV can be found here.

Appendix F: Vendors and Parts Information

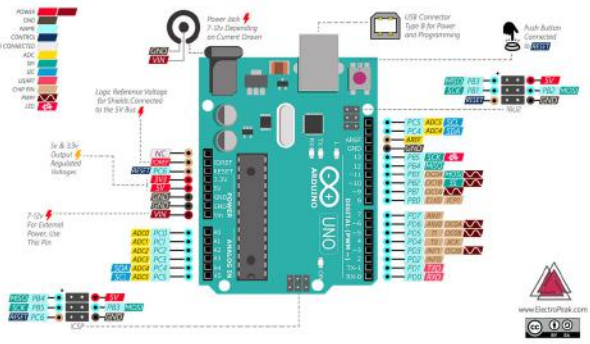
Amazon.com can be used to purchase to purchase all the components listed in Appendix A: Parts List

Information about the vendors used for purchasing parts and how to locate parts for similar projects can be found here.

Appendix G: Additional Documentation

Additional documentation, including diagrams, printouts, and detailed project logs, can be found here. This is information that supplements the design specification, including:

Arduino UNO Pinout:



Raspberry Pi 4 Pinout: