Reef Rover

Maddie Burroughs, Emma Hoffman, Abigael O'Donnell

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract* — Reef Rover is a minimally invasive robotic system designed for detailed exploration of shallow coral reefs through high quality live feed and image capturing. After each mission, the laptop-based interface processes video stills to generate a high-fidelity map of the traversed areas. A machine learning algorithm analyzes these images to detect coral bleaching so researchers can identify areas at risk. By enhancing reef monitoring efficiency and conservation efforts, Reef Rover offers a powerful tool for researchers and environmentalists, eliminating the risks and costs of human diving while providing more detailed data than satellite imaging.**

***Index Terms* — arduino, bleaching, classification algorithm, computer vision, conservation, coral reef, image recombination, machine learning, Raspberry Pi, robotics, WiFi**

I. Introduction

Coral reefs are some of the most important ecosystems on Earth. While only covering about 1% of ocean floor, they are home to over 25% of all marine life. Additionally, coral reefs protect shorelines from erosion, help form beaches, and provide homes for marine animals that provide critical economic and medicinal benefits. Many organisms found in coral reefs have been discovered to possess medicinal benefits needed for the development of drug compounds to treat cancer, arthritis, bacterial infections, and viral diseases.

Climate change, pollution, and overfishing has had a devastating impact on coral reefs. Over 50% of all shallow water coral reefs have been destroyed. When sea water temperature rises, algae that protect the coral from predators and disease are killed. The absence of this algae leaves coral susceptible to disease, causing them to lose their color in a process called bleaching. Bleaching is a key sign of a struggling coral reef that scientists monitor to assess reef health. 90% of the largest coral reef, the Great Barrier Reef, has been impacted by bleaching.

Scientists and conservationists are working hard to preserve the coral reefs, monitoring these reefs to track the amount of damage over time. Currently, scientists are using satellite images or human divers to detect bleaching. Satellite images can only detect large-scale bleaching, and detection with divers is a tedious process which can cause damage to the coral and poses risk to the divers themselves. We have built a robot, Reef Rover, that can detect coral bleaching on a smaller scale similar to what a diver could do, but with increased data collection, analysis, and reduced ecosystem disturbance. The robot is controlled remotely with a laptop interface that gives the scientists the ability to scan the coral reef, take photos, and then recombine these images to create a map of the area. A machine learning algorithm also analyzes the generated map and identifies four tiers of coral health: Healthy, Pale, Partially Bleached, and Bleached. The robot is also equipped with a temperature sensor enabling scientists to record data tracking these additional risk factors.

II. Use-Case Requirements

In order to ensure practicality and ease of use of Reef Rover, we've identified a number of use-case requirements which are fundamental to meeting user needs:

1. High-resolution Underwater Camera

Reef Rover must provide clear, high-resolution imaging to support scientists and other users in monitoring coral health. High-quality imaging is essential for effective device operation and ensures accurate processing by our image analysis algorithms.

2. Reliable Bleaching Detection Algorithm

After each operation, Reef Rover will be deploying a machine learning algorithm to assess the health of coral and determine risk through color analysis of the stills taken from livestream video. To ensure reliable monitoring, the algorithm must achieve at least 90% accuracy in classifying coral health. A high rate of incorrect classifications would compromise the system's effectiveness for long-term monitoring.

3. Map Generation in Less Than 10 Seconds per Square Meter

Reef Rover must generate maps and complete bleaching assessments in less than 10 seconds per square meter of surveyed area. While this performance metric is significant for large-scale applications, processing occurs in the background post-mission, requiring no user intervention and having no impact on real-time usability.

4. WiFi Connection to User's Laptop

To ensure ease of operation and local data storage, the device's control interface will run on the user's laptop, rather than a designated controller, communicating with the robot via a WiFi connection.

5. Boat Surveying Speed of 3 Inches per Second

The National Coral Reef Monitoring Program (NCRMP) defines a standard survey site as 60 square meters (~645 square feet). An in-depth survey of a standard site takes a human diver approximately 1.5-2 hours to complete. With a surveying speed of 3 inches per second, our device will be able to complete a detailed survey of the same area in approximately 1.43 hours.

6. Power Supply Life of at Least 2 Hours

At an average traversal speed of 3" per second, Reef Rover

must be able to operate for a minimum of 2 hours per charge. This ensures the device can complete a full standard survey without requiring a battery swap or recharge mid-mission.

III. Architecture and/or Principle of Operation

Reef Rover's architecture consists of a physical prototype, electrical hardware, and software. There are three main subsections - the above water component, submersible, and laptop interface.

A. Above Water Component

The above water component features two embedded devices: an Arduino Uno R4 WiFi and a Raspberry Pi 4. We added the Arduino Uno after the design report to provide more energy efficiency and to take advantage of Arduino Cloud, a pre-configured platform that enables remote interaction over Wifi. The Arduino controls two rear-mounted propeller motors for horizontal movement and a stepper motor that adjusts the depth of the submersible. It also reads temperature data from a water sensor.

The Raspberry Pi is dedicated to handling image capture via the Pi Camera Module 3 and transmitting frames to the laptop. Both devices communicate wirelessly with the user's laptop over a local WiFi network.

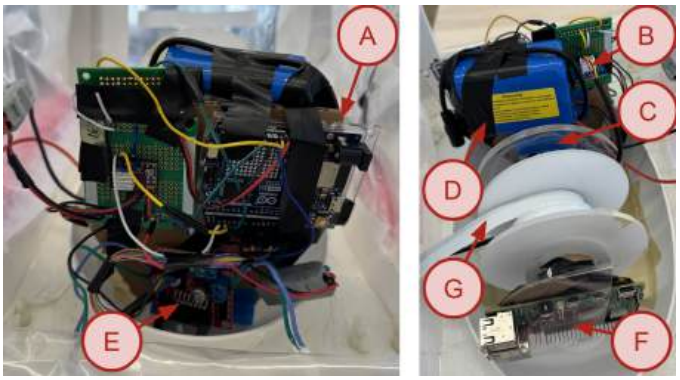


Fig. 1. **Electronic Hardware -** Arduino Uno R4 WiFi (A), Temperature sensor breakout board (B), Stepper motor (C), Rechargeable 12V battery (D), Propellor motor driver (E), Raspberry Pi 4th Generation (F), Raspberry Pi Camera Module 3 Cable (G).

B. Laptop Control

Onshore users interact with the system through two interfaces: a livestreaming window powered by the Raspberry Pi and a controller interface hosted on Arduino Cloud. The livestream provides a real-time video feed, photo capture capabilities, and, after operation, generates a map of the surveyed area. User-captured images are stored on the user's laptop in a pre-specified folder. The program also incrementally captures images automatically, which are then sent to our image stitching algorithm to produce the map of the surveyed area. Lastly, the resulting stitched image is then processed by the Classification module, producing a final high-resolution map with heat mapped identification of coral bleaching intensity.

C. Physical Prototype

Numerous materials make up the physical prototype of Reef Rover. We made the base with two parts; a plastic bowl and foam board. The plastic bowl holds the electronics and spool system to raise and lower the camera. It also acts as a moat, since the walls of the bowl go above the foam board base, protecting the electronics from any leaks into the boat. The foam board extends the base to increase the size and stability of the boat. We sealed the foam board with waterproof caulk to prevent waterlogging the boat and preventing mold or disintegration. The sides of the boat were constructed with four foam board corners and plastic walls. We added a rectangular foam board cut out to the top of the walls to give the boat a more defined structure. We sealed the sides of the boat to the foam board base with waterproof tape and waterproof caulk. Pool noodles on each side and underneath the boat keep it afloat.

The two thrustor motors on the back are screwed in underneath the foam board base. We made a cut out on the side of the boat for the camera cable and temperature sensor. The location of this cut out was different from the design report, where we initially were going to cut a hole through the foam board base so the camera went straight down. We realised that there would be no successful way to waterproof this hole even if we raised the boat up high enough so that there was an air gap between the base and the water. Additionally, raising the boat high enough out of water for an air gap would have reduced the stability of the boat and it could have tipped over easily. Also, we added a paper towel to the cable hole to dry off water as the cable came up above the water into the boat, further protecting our electronics. In order to maintain balance, we added a counter weight on the other side of the boat so the boat moved in a straighter line.

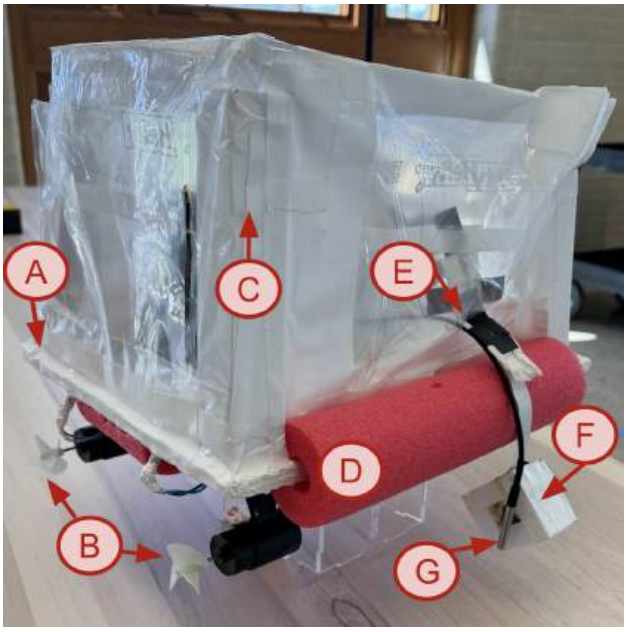


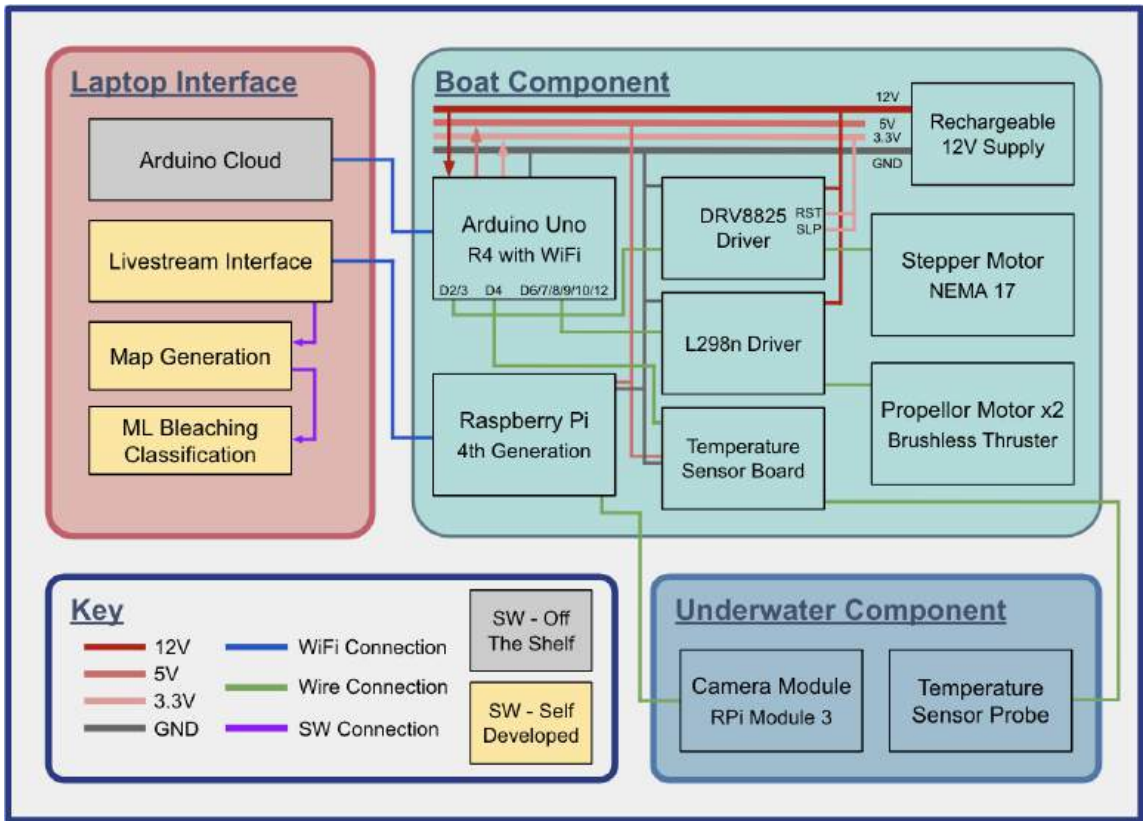
Fig. 2. (Above) Physical Prototype - Foam board base (A), Propeller Motors (B), Foam board corner and waterproof tape (C), Pool noodle (D), camera cable and temperature sensor hole, with paper towel (E), Submersible camera box (F), Water temperature sensor (G).

Fig. 3. (Below) Block diagram of the entire system, showcasing the different connections that were used, the three components, and what software was individually developed.

The underwater component has the camera and temperature sensor that are connected to our microcontrollers via waterproof cable. The Arduino connects to the motors and temperature sensors via wire connection. The Raspberry Pi and Arduino connect to their respective software interfaces through a WiFi connection, allowing the user to control Reef Rover at a distance. Images are then able to be post processed with the map generation algorithm and machine learning algorithm.

Engineering Principles:

One of the engineering principles we used to create Reef Rover was to test subsections before integration to avoid complicated errors. This was especially important because in our final product we had to integrate into a small waterproof area, so debugging after everything was together would not have been practical. We were able to each work on our separate software and hardware sections and test their functionality before we integrated everything, especially the hardware. Also, we had a large focus on our materials in this project to ensure we had waterproofness, balance, and floatation. The materials we used for waterproofing include rubberized caulk paste, polyethylene based tape, and liquid latex. To ensure the boat was balanced and stayed afloat, we added foam board, an already buoyant material and pool noodles such that more were in the back where the boat was heaviest. We had to iterate over the design of the spool so that the hole where the stepper motor was inserted wouldn't wear down. First, we used wood, but this was replaced with laser cut acrylic. We spent a lot of time designing the construction of the boat since we had all the materials, but had to customize it to our needs.



We also used engineering principles to design wireless communication. We were thinking about either bluetooth or WiFi for this application, but learned that bluetooth does not have the bandwidth to transfer images, so we decided to use WiFi.

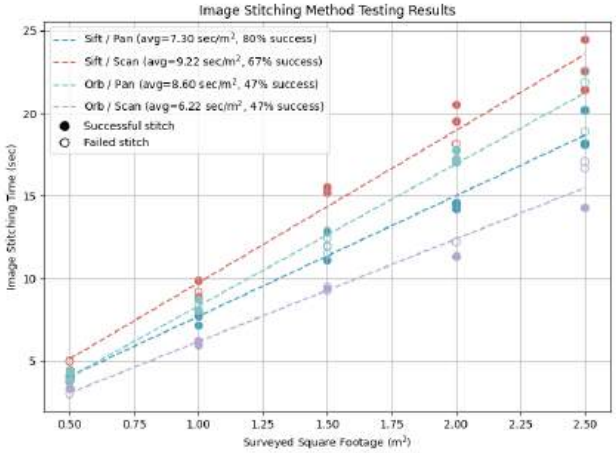
Scientific Principles:

The scientific reason we had to have the above water portion of Reef Rover was that water disrupts wireless data transmission. Radio waves which typically transfer data wirelessly are severely disrupted underwater because of high attenuation and absorption. Currently, communication across oceans occurs through tubes on the ocean floor called fiber optic cables that contain the radio waves. An emerging communication protocol called Li-Fi would overcome these obstacles posed by water, but this is out of scope for current industrial applications and our project [8].

Mathematical Principles:

We used mathematical principles when evaluating a number of our design requirements.

OpenCV offers two methods of key feature detection (SIFT and ORB) as well as two methods of image stitching (Panoramic and Scan). Our selection of which methods to use in our final post-processing algorithm was guided by mathematical evaluation of each method's performance. A series of controlled trials were conducted on the four permutations that can be created by pairing one feature detection method and one stitching method. For each permutation, we conducted 15 trials, recording the square footage surveyed, time taken to stitch the image, and whether or not a final image was created successfully. The overall results are depicted in the graph below. Ultimately, we selected the SIFT/Panorama algorithm, which demonstrated the second-fasted average time per square meter while significantly also outperforming the fastest method in reliability.



When implementing our ML algorithm, we tested several k values, including 3, 5, and 7. We decided that with the size of our training data, in order to prevent overfitting, that 5 was an appropriate number of neighbors to maximize our accuracy. Additionally, the program first averages the RGB value of the photograph in groups of 10x10 pixels, and then classifies each group individually. These individual classifications are represented in the output image, and then each output image is labelled in a .txt file with their corresponding overall classification.

We also applied mathematical reasoning and unit analysis when selecting a power supply. Reef Rover was designed to operate on a single charge for at least two hours. To meet this requirement, we calculated the estimated power draw of each electrical component, resulting in a total system power consumption of 29 watts. This value was converted to amp-hours, guiding our decision to use a 5.2Ah power supply – sufficient for two hours of operation with a reasonable margin. Exact calculations are provided in Section IV.8.

IV. Design Requirements

To meet the use-case requirements, the following design specifications outline the technical choices and constraints necessary for Reef Rover's functionality:

1. High-resolution Underwater Camera

To achieve high-quality imaging, the system must support high-resolution video capture. This will be accomplished using the Raspberry Pi Camera Module 3, which supports the industry standard of 330 PPI resolution, ensuring detailed and accurate image capture.

2. Low Control Latency

To enable real-time remote operation, the system must maintain a communication transmission latency below 300ms. Reef Rover will achieve this by rapidly transmitting the livestream video feed from the boat to the user and relaying user commands back to the boat via a WiFi connection. The latency can be controlled by adjusting the frames per second (FPS) to allow for smooth live video.

3. Bleaching Algorithm

The system must correctly identify healthy coral based on a predetermined range of pink and blue colors. This will be achieved by preprocessing the images taken by our robot to remove the background color of the water and then taking an average of the color value over a 10x10 pixel square. The algorithm will then identify both the color of the coral and the stage of health (healthy, pale, partially bleached, bleached). We will run a series of tests on both edited pictures with different color samples and images from the camera to measure the accuracy of the classification algorithm at the minimum threshold of 90%.

4. Map Generation Algorithm

The system must generate a complete survey map

post-operation, processing each square meter of surveyed area in less than 10 seconds. Map generation will be performed using the OpenCV library, leveraging the SIFT algorithm to detect keypoints within the images. These key points will be used in OpenCV's panorama stitching mode to align and merge images into a seamless map well within this time constraint.

5. Underwater Unit Size

To minimize risk of damage to the coral reef and reduce ecological disturbance, Reef Rover's underwater sensor unit must not exceed dimensions of 6" x 4" x 4".

6. Deployment Depth

Coral reefs are primarily found near the surface of the ocean, meaning that the underwater component of Reef Rover does not need to deploy to significant depths. For the purposes of this project, we have set a maximum depth of 2 feet. This depth is sufficient for testing and development, while allowing flexibility for future scaling in real-world deployment. Increasing this depth would add cost without introducing any new technical challenges.

7. Operable WiFi distance

For testing purposes, Reef Rover must maintain a reliable WiFi connection at distances up to 10 feet. While this distance can be easily increased for real-world deployment, doing so would introduce additional costs without adding significant technical complexity or functionality. For this reason, we've chosen to allocate our budget to other, more impactful aspects of the project.

8. Power Supply Capacity

To support a continuous 2-hour survey without recharging, our power supply must have a minimum capacity of 5Ah. This requirement is based on the following power consumption estimates per electrical component:

Raspberry Pi: 5W
Arduino Uno R4: 0.8W
Servo Motors: 5W per motor = total 10W
Stepper Motor: 12W
Camera: 1.5W
Temperature and pH sensor: 0.5W

With an estimated total power consumption of 29W over 2 hours, the required battery capacity is calculated as:

$$(29.8\,W \times 2\,hours) / 12\,V = 4.97\,Ah$$

To provide a margin of error, we've purchased a 5.2Ah rechargeable power supply.

V. Design Trade Studies

1. Processing Unit

The first processor considered was by Nordic Semiconductor, and it was a device that could transmit video over bluetooth and WiFi together. At this point, we were still debating between bluetooth and WiFi. However, it did not have good input and output pin connections for the motors. The next processor we looked at was the Raspberry Pi. It had WiFi capabilities and a simple connection to cameras and motors. We were debating next whether we wanted to use the Raspberry Pi 4 or 5. The fifth generation RPi had faster processing speeds with a newer CPU and GPU, but we had immediate access to a RPi 4 in the ECE inventory, so we wanted to use that one first to see if it worked for our use case.

Ultimately, the RPi 4 met our needs for live streaming, however, was having issues being able to control our motors. The live stream was operating with a high frame rate, which made it hard to send requests to control the motors at the same time. We tried to implement threading to control the motors and live stream separately, but we were unsuccessful. We opted to add an Arduino Uno R4 with WiFi to handle the motors and temperature sensor. The Arduino also is more energy efficient at controlling the GPIO pins, and we had access to a predefined wireless communication service called Arduino Cloud. We could easily set up motor control with this interface. This meant that users now interacted with two interfaces, however by putting the computer in split screen, we could easily operate both interfaces at the same time.

2. Communication Method

We were deciding between using Bluetooth or WiFi for our communication method. Ultimately, bluetooth was not going to have the bandwidth necessary to transmit video or photos, so we are using WiFi. WiFi also has longer range connection, and scientists already have WiFi on their research boats to communicate with people on land and run data analysis software. The WiFi they use is unique because instead of connecting to a cell tower it connects to a satellite.

3. Power Source

To determine the appropriate power supply, we first estimated the total power draw of our system and the required runtime per mission. Our system consumes approximately 29W in operation. Based on Reef Rover's estimated speed of 3 inches per second, we calculated that it would approximately 1.43 hours to traverse a standard survey site. Based on these metrics, we need a 12V supply with a capacity of 4.97 Ah.

We selected a 12V power supply with a 5.2Ah capacity, capable of delivering 29W continuously for 2 hours. While we considered larger power supplies, our chosen capacity minimizes cost and supports a full survey being conducted on a single charge.

4. Stepper Motor

To control the z-direction movement of the underwater system, we need a motor capable of precise, low-speed movement to deploy and retract the unit; a stepper motor is

ideal for this application. We selected a NEMA 17 stepper motor, based on its 59 Ncm holding torque, which exceeds our needs, calculated below:

Given that the underwater component has a maximum weight of 2 lb, the force required for deployment and retraction is:

$$2\ lbs \times 9.81\ m/s^2 = 0.907\ kg \times 9.81\ m/s^2 = 8.899\ N$$

Assuming a 2-inch spool radius, the minimum holding torque needed is:

$$8.899\ N \times 2\ in = 8.899\ N \times 0.0508\ m = 45.21\ Ncm$$

5. Stepper Motor Driver

We need a driver to control the motor. Three of the most common drivers for a NEMA 17 stepper motor are the A4988, DRV8825, and TMC2209. In choosing a motor driver, we prioritized cost and precision, while also considering ease of use and noise levels. Ultimately, we chose to use the DRV8825. While this driver is more expensive than the A4988, it offers quieter and more precise operation, allowing 1/32 microstepping, compared to the A4988 1/16. The TMC2209, on the other hand, offers 1/256 microstepping which is significantly more precise but exceeds our necessary level of accuracy. Additionally, the TMC2209's higher cost and more complicated setup made it less practical than the DRV8825.

Lastly, the precision afforded by the DRV8825 driver means that we no longer need to include an incremental encoder to the stepper motor. We originally planned on incorporating an encoder to ensure high levels of precision, however, our chosen motor and driver should allow for very little (< 10mm) positional error to be accumulated over two hours of operation.

6. Propellor Motor

In our design review, we discussed servo motors that were purchased to control the forward motion of the boat. However, these turned out to not be waterproof as advertised, so we went back to the drawing board to look for a new motor. We did not want to replace the motor driver, so we had to find a waterproof thruster motor that was brushless. After some research, we settled on a remote controlled boat thruster, that is energy efficient, can be powered with 12V, and does not have too much power so we can keep the speed of the boat at the desired 3 inches/second. Another option we had would have been suitable except it operated at 200 horsepower, too much for our application.

7. Propellor Motor Driver

The L298N was chosen because it can protect the Raspberry Pi from directionality fluctuations from the servo motor. Also, the L298N amplifies the small current from the GPIO pins on the RPi to be large enough to operate the servo motor. The L298N can run two servo motors at once, which is perfect for us because we will be using two servo motors. It comes with an enable pin in addition to the regular input to provide additional control. It can be configured to change the direction of rotation of the servo motor, which we can use to put the boat in "reverse gear" to go backwards if needed. We did not initially plan to implement backwards motion, but this component gives us the ability to if we end up needing it. The L298N can take in smaller amounts of power from the RPi to control the speed of the motor, which will allow us to adjust the speed of the boat or modify our turning mechanism if needed by having both motors running but one at a lower speed than the other.

8. Temperature Sensor

In order to measure the temperature of the water, we ordered the GAOHOU DS18B20 Waterproof Digital Temperature Sensor with Adapter Module. We chose this model because it is waterproof and the length of the cable would allow us to raise and lower the sensor with the camera unit. While we initially planned and ordered a sensor that would measure both pH and temperature, we could not accurately calibrate the unit and limited waterproof options for replacements resulted in the decision to prioritize one of these values. As temperature is more indicative of coral bleaching, we decided to go with this option.

9. Machine Learning Algorithm

As we discussed in our design review for the machine learning algorithm, we used the K-nearest neighbors classification algorithm in order to take advantage of supervised learning techniques. As our 'heat mapped' output image produces a 'clustered' color result, we found that this was a reliable way to process and represent data. The algorithm works by breaking the image up into smaller groups and then classifying each sub group.

10. Map Recombination Library

After completing initial research into existing libraries – primarily focusing on Python libraries, as this is the language that a majority of our codebase will be written in – we identified two promising options, each having distinct advantages and trade-offs.

The first library that we considered was COLMAP, which provides advanced capabilities beyond image stitching such as Structure-from-Motion (SfM), Multi-View Stereo (MVS), and 3D map generation from images. While these features offer interesting possibilities and additional insights for researchers, they require significantly more computational power. Achieving our target processing time of 10 seconds per square meter would be difficult using COLMAP's image stitching capabilities alone, and would likely be infeasible if we

implemented any of the additional features it supports.

The other library that we evaluated was OpenCV. While this library doesn't offer many of the advanced features that COLMAP does, it does support efficient image recombination. OpenCV's image stitching is designed for panoramic images, or images which connect in only one direction, meaning that we will need to develop an algorithm which adds support for image stitching both horizontally and vertically. However, even with this added implementation effort, OpenCV offers a clear computational advantage.

After considering both libraries, we decided to use OpenCV in our final implementation. Although COLMAP offers many features that could be valuable additions in the future, they exceed the needs of a standard coral reef survey. Since the NCRMP typically conducts 2D analysis, OpenCV aligns best with our project's objectives while also meeting our strict computational constraints.

VI. System Implementation

As mentioned in the system architecture section, we had three main subsystems in our design - the above water boat, the laptop interface, and the underwater component. We will go into detail about the implementation of each part of our design.

A. Raspberry Pi Live Stream Interface

The first microcontroller on our above water subsystem is the Raspberry Pi 4th Generation, which controlled image capture and transfer. We used the Raspberry Pi Camera Module 3 that connected to the onboard camera port to take the images. This camera has a 12 megapixel lens with autofocus, and is marketed as Raspberry Pi's highest quality camera. In order to reach our depth requirement, we ordered special camera cables that were 1 meter long. The cable was waterproofed with waterproof tape and then sealed with two layers of liquid latex.

The Raspberry Pi was configured on the SD card to connect to an iPhone hotspot network to wirelessly transfer images to the computer. We were able to SSH into the RPi and write a code script that ran on the Pi which turned on the camera, responded to requests sent from the computer, and sent still images to the computer that were captured on the camera. We used a socket server that was made with the Python library socket. The host computer script would send a request to the Pi to capture an image. Then, the script in the Pi would receive the command, use the library Picamera 2 to take the photo, and then send the bytes back to the host computer.

The current workflow is described below.

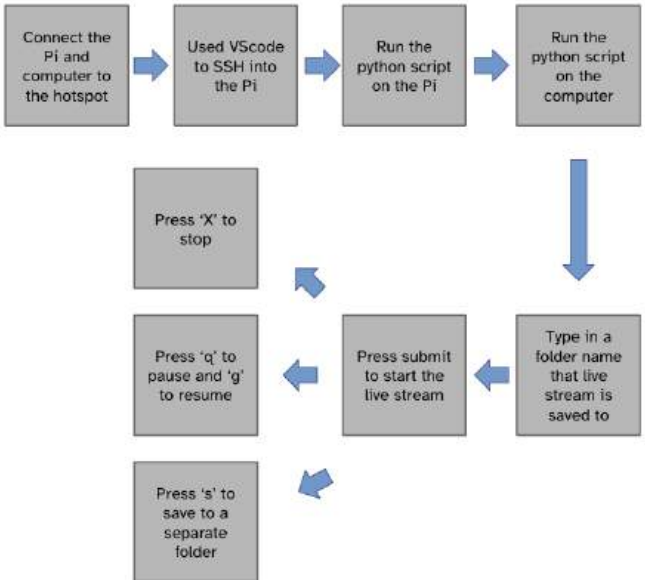


Fig. 4. Users flow through the Pi interface. Improvements can be made in the future to eliminate the SSH step and automatically run the script on the Pi.

We ran into issues with the color of the images coming into the computer, since the Pi was using a different pixel color representation. We were able to remedy this by using a built-in function to switch the pixel color representation before it was displayed and saved on the computer. Additionally, we found that the images were darker and more blurry than desired. To remedy this, we configured the camera settings on the python code in the Pi to increase the brightness and focus more on objects about 12 inches away from the camera, where we would expect coral to be most of the time.

B. Arduino and Arduino Cloud

The addition of the Arduino Uno R4 WiFi was made after the design review, due to complications with motor commands interfering with the live stream commands. We wanted the boat to be controlled without interrupting the live stream, and to accomplish this we added an Arduino. We took advantage of the Arduino Cloud, a predefined wireless control system, to control the I/O pins on the Arduino Uno R4.

When we first added the Arduino, we tried using the Arduino Nano ESP32, which is a smaller Arduino with WiFi capabilities. However, we ran into many bugs using this. First, one of the Arduino Nano's ports broke after only a few uses. Then, once we replaced the broken Nano, we tried to add our stepper motor, however with the same code and same wiring compared to using an Arduino Uno, the Nano ESP32 was not able to power the stepper motor correctly. We spent a lot of time trying to debug this particular section of integration, and ended up having to scrap the Nano entirely. We then ordered the Arduino Uno R4 with WiFi, since the Arduino Uno we had did not have WiFi capabilities.

Once we had all of the hardware wired to the Arduino Uno R4 with WiFi, we had to integrate the stepper motor with Arduino Cloud. We had to adjust the code so that the stepper motor activated based on a defined variable that changed when the user pressed a button widget on the dashboard. We spent a lot of time debugging this as well because the delay time we used was critical, and we were sending commands to the Arduino too quickly and the stepper motor was not making a full rotation. Once we added a longer delay between the commands to make a step, we were able to control the stepper motor with the Arduino Cloud.

We also added a temperature sensor to the Arduino so that the user could see the temperature in real time and correlate the water temperature with coral health. The temperature sensor was mounted onto a breadboard with a power source, ground, and data line. We connected the dataline to digital pin 4 on the Arduino and we read the temperature once every second. We used a code function from the sensor developers to calculate the temperature from the data received, and displayed this information on the Cloud dashboard. We also included a graph that displayed the temperature over time.

The propellers were also controlled by the Arduino Cloud. We created two variables on the cloud that controlled the boat's forward movement and turns. The first variable controlled straight forward movement. This was a scale bar with values of 0 to 3. When the user set this to be 0, the propellers would stop moving. The values 1 to 3 corresponded with different movement speeds (1 being the slowest and 3 being the fastest). On the back end, this one variable would control both propellers. When the user changed this variable, the Arduino would send a PWM signal (using the analogWrite command) to both motors. The value of this signal was determined by extensive pool testing to see which values made the boat move as straight as possible. Since the camera was coming down on the right side of the boat, the right motor propelled faster to combat the weight and resistance on the right side of the boat. Despite our PWM signal tuning efforts, we could not overcome the weight of the camera with motor speed alone, so we added a counter weight on the left side to improve our control. This component was made identically to the waterproof camera box that we constructed, substituting the internal camera component for weights. This ensured the counter weight was nearly identical to the component it was balancing against.

We had one more variable, which was called “make turn,” which turned on the right motor only, turning the boat around so the user could start surveying in the other direction.

The Arduino Cloud had unexpected drawbacks, including a limit of 5 variables per device on the free plan. In order to add more variables, we had to purchase a subscription. This limited our control of the motors. At first, we had two scale bars that controlled the right and left motor respectively, but this was not intuitive for users and it was impossible to turn the motors on at the same time, which made the boat never go in a straight line. Future work could be done to improve the control of the boat by integrating a remote control with more precise motor movement.

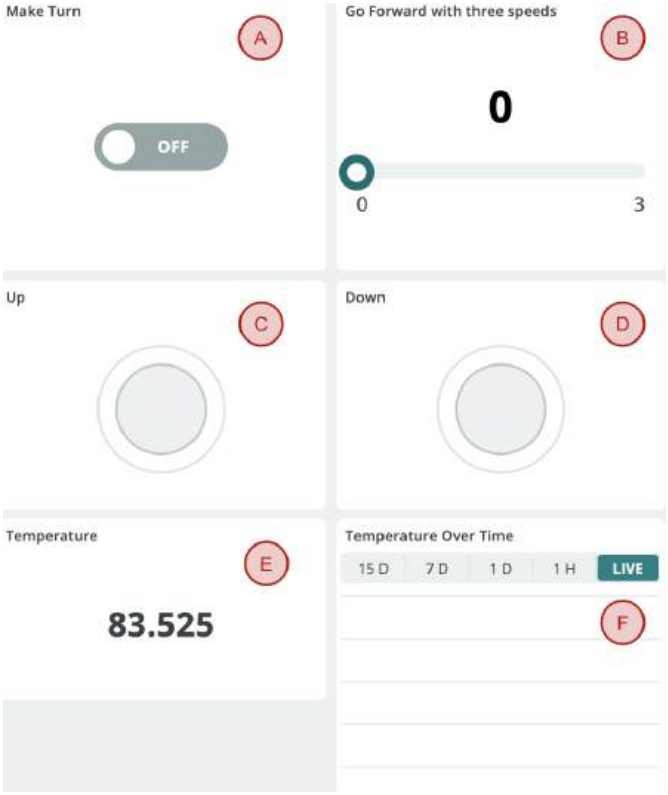


Fig. 5. The Arduino Cloud interface. The make turn variable (A), the forward motion scale bar (B), the up direction camera button (C), the downward camera button (D), the numerical temperature in fahrenheit (E), and the temperature over time (F).

The user will use these two interfaces, the live stream from the Raspberry Pi and Arduino Cloud at once to control Reef Rover. Below is an example of the two interfaces on a laptop in split screen. The split screen made the configuration of the widgets different from when it was full screen, which was another drawback to the Arduino Cloud interface that we tried to overcome but were limited by their software.

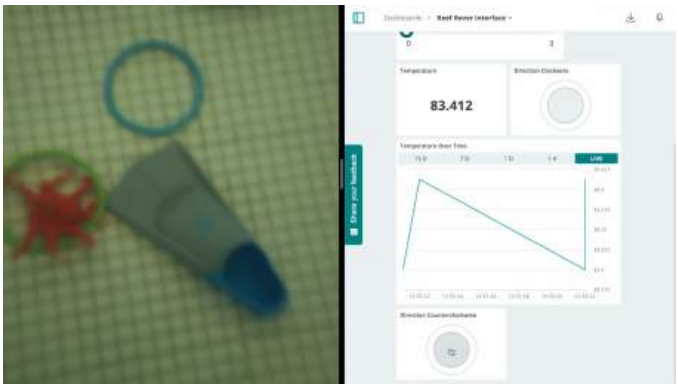


Fig. 6. The live stream image (left) with the Cloud interface (right)

C. Machine Learning

For the machine learning section, we used a K-Nearest

Neighbors algorithm in order to utilize supervised learning techniques. The program was designed to create an overall 'heat map' representation of coral bleaching by running on the output of the combined images, or to run on the entire folder of images collected from the livestream to help scientists pinpoint different areas of bleaching.

As part of the image processing, the program also removes the image background to prevent darker ocean colors from skewing the results. It works by first sectioning each input into 10x10 pixel groups, which was chosen in order to both retain the integrity of the initial image while maintaining a reasonable run time on potentially very large batches of images. The average RGB value of each section is then taken and used to classify that section.

Our program was designed to be most accurate on two types of coral, pink or blue. Once each section passes through the first KNN classifier, where it is assigned one of these labels, it then goes through a second layer fine tuned for either pink or blue coral where it is labeled as either healthy, partially bleached, pale, or bleached.

By splitting the images into smaller sections, we are still able to achieve the benefits of a clustering algorithm such as identifying healthy and unhealthy subsections of coral. We also utilized several libraries in this process, namely Scikit-learn for machine learning, numpy, PIL and Rembg for image processing, and os, io, and collections for data management from the user's computer.

D. Map generation

Our map generation algorithm takes in a folder of images automatically captured by the livestreaming interface during a survey and generates a high-resolution composite map of the surveyed area. This subsystem is implemented entirely in software and leverages multiple OpenCV functionalities. It supports variations in image overlap—caused by inconsistent survey speed—as well as changes in lighting and perspective.

The algorithm centers on the OpenCV *cv2.Stitcher_create()* interface, which supports stitching using either Panorama or Scans mode. Before stitching, each image is analyzed to detect key features using either the SIFT (Scale-Invariant Feature Transform) or ORB (Oriented FAST and Rotated BRIEF) methods.

Based on the analysis detailed in *Section III, Mathematical Principles*, we selected the SIFT method for feature detection and the Panorama mode for stitching. Panorama mode stitches images in a single linear direction, ideal for constructing wide, continuous image sequences. To accommodate this, our program first compares keypoint matches to organize the images into rows that represent individual survey passes. These rows are stitched horizontally using Panorama mode and then vertically combined to produce the final stitched map.

Through iterative testing, we identified two key optimizations. First, adjacent images often contained more overlap than necessary, allowing us to skip redundant frames. We introduced an image_step parameter to stitch only every nth image within a row, significantly improving performance without sacrificing accuracy.

Second, we observed that while Panorama mode was more reliable overall, it occasionally failed on certain image batches. Interestingly, these failures were often resolved by switching to Scans mode, which, while generally less successful, handled some edge cases better. To implement this, we added a try/except block that defaults to Panorama mode and falls back to Scans mode upon failure.

To minimize reliance on the less consistent Scans mode, we introduced a "batch stitching" method: each row of images is split into smaller batches (based on a `batch_size` parameter), which are stitched individually. This limits the number of images per stitch attempt, improving the chance of success and allowing recovery from partial failures. The final map is built by recursively stitching the resulting batch outputs.

The overall pipeline of this algorithm is as follows:

1. Accept command-line parameters to override default values (i.e. *image_step* and *batch_size*); --help provides usage guidance
2. Prompt the user to enter a valid image folder path
3. Load every *n*th image from the folder (where *n* = image_step), resizing them to a consistent width
4. Apply SIFT feature detection
5. Organize images into rows based on keypoint overlap
6. Divide each row into smaller batches of size batch_size
7. Attempt to stitch each batch using Panorama mode; if it fails, fall back to Scans mode
8. Recursively stitch all batch outputs to produce a single final image

E. Spool design

To allow for the user to adjust the depth of the underwater camera during operation, we implemented a custom spool system controlled by a stepper motor. This subsystem raises and lowers the camera via (un)winding the camera cable.

The spool was fabricated from laser cut acrylic and designed to maintain consistent tension on the camera cable. This allowed for deterministic depth based on spool position, and prevented damage to the cable. A stepper motor, controlled by the Arduino Uno, drives the rotation of the spool. This motor was chosen for its ability to precisely control rotational movement, allowing for fine-tuned adjustment of the camera's depth based on survey needs.

One key design challenge that we encountered during designing this subsystem was that, due to the length and rigidity of the RPi camera module's cable, the cable cannot

twist freely during spool rotation without risking damage or disconnection from the RPi. To resolve this problem, we mounted the RPi directly to the side of the spool such that the RPi spun with the spool. This ensured that the anchored end of the camera cable rotated in sync with the spool itself, avoiding torsion and maintaining a reliable data connection.

VII. Test, Verification and Validation

To test our implementation and verify that it satisfied the use case requirements, once we built Reef Rover we tested in the Cohon University Center pool. We used dive toys to act as our coral, since they are often vibrant colors. We tested the latency, operable distance, and deployment depth. We tested other design requirements out of the pool, including the live stream and map generation latency, the classification accuracy, and image resolution. Due to time constraints, we were not able to test the battery life at once, however we were operating the boat for a total of 4.5 hours of pool testing before the demonstration, and then another hour of demonstration before our battery died, which was more than enough to accomplish our theoretical scan of about 2 hours.

A. Machine Learning Model Accuracy

In order to test the performance of our Machine Learning model, we used a database of over 900 pre classified images of coral reefs. To develop the training data, which used RGB data points, we manually sorted through images of pink and blue hued coral and used a color picker to sample data points. Once these points were collected, we validated the performance of our model on 100 of the pre classified images, split evenly between healthy and unhealthy coral. These images were processed in ten tests using ten images each, and the overall classification of each image was written to a .txt file at the end of each test. As we were using four categories (healthy, partially bleached, pale, and bleached) and our sample dataset only had two categories (healthy and bleached), we considered results of healthy and pale to be healthy and results of partially bleached and bleached to be bleached for validation purposes.

We also ran the program on images we took during our in water training of Reef Rover. Here, we were limited based on the colors available to us in the objects we were using in the pool, as these were all fairly vibrant colors, most of our items were classified as healthy, however we were able to place more lightly colored objects which were classified in one of the bleached groups. Overall, we were able to reach an 84% success rate for our algorithm, which falls slightly below our goal of 90%. While we were under this testing threshold, issues such as differences of coral depth affected the lighting on the coral pictures, making it hard to find a standardized way to evaluate each image. Reef Rover was designed to operate in shallow water, where the coral will remain more ‘true to color’ underwater. With more time, we also could have tested a variety of more complicated models which may have resulted in an improved performance.

B. Live Stream Latency Design Requirement Results

Our design goal was for our live stream latency to be less than 300 ms. This would give users real time feedback on where Reef Rover was, making surveying easier and avoiding crashes into coral. We used the time library in Python to measure how long it took for the program to send a request to the camera until all the image bytes were received by the computer. We took the average latency for 25 frames and found this value to be 70 ms, much below our maximum requirement.

While we were testing in the pool, we found that the latency was fast enough to see where the boat was in the water. We found it challenging however to understand which direction the boat was pointing in by just the image of the bottom of the pool. To improve the users understanding of where Reef Rover was, we could add another camera to the front of the boat.

C. Image Stitching Latency

One of our design requirements was to reduce the post processing time by our map generation algorithm running with a latency of no more than 10 seconds per square meter surveyed. This was defined as the total stitching process time divided by the area surveyed.

This was tested by using the Python time library to record the duration of the stitching process for each image set or survey. This duration was then divided by the known approximate area of the survey (in square meters), which was measured during test surveying sessions. Multiple trials were conducted across different surveys, with several repetitions per image set to account for variability in processing time due to hardware performance and image content.

The results of these trials were averaged to compute a representative stitching latency. We observed some variation between trials depending on lighting conditions and image complexity, but the final average latency across all trials was 7.83 seconds per square meter.

Normally, images that scientists take must be individually inspected. Now, we are able to combine hundreds of images together for comprehensive review. This metric meets our design requirements, and confirms that our selected OpenCV methods provided the intended balance between accuracy and efficiency, suitable for the target use case of Reef Rover.

D. Deployment Depth Result

Our design requirement was to be able to deploy our camera at least 2 feet below the water. We tested this by fully deploying the camera and measuring the depth it can reach.

We had a depth of 25 inches, which allowed us to meet this design requirement.

We noticed in the water that the camera would sometimes float upwards in the water, but it never went farther than the cable would allow. This prevents the camera from accidentally knocking into things below it, but it could be higher than the user anticipates. This control is limited due to the fact that we needed to pull the camera up and down and the camera cable is very flexible.

E. Operable Distance

Our design requirement was to be able to operate the boat from at least 10 feet away, simulating how a researcher would be far from Reef Rover during surveying. To test this, we turned on all of our systems and connected them to our WiFi network, moving the source of the WiFi and the computer away from the boat. We then turned on the live stream and the motors, and began to operate as usual. We moved as far away as we could in the room we were in (27 feet) and the controls were working just as if we were next to Reef Rover.

This satisfies our use case that users need to be able to operate the boat from a long distance away. In the ocean, Researchers often cannot drive right alongside the robot, so this is an indicator that this could be scaled up well to even farther distances.

F. Image Resolution

Our design requirement for image resolution was 300 pixels per inch (PPI), which is a metric used to define an image as "high resolution." To test this, we would inspect a saved image from the underwater camera and see what the PPI was. When we inspected our images, no matter how clear the image was, the PPI was 72. We compared this to a saved image taken on a GoPro, and these crisp images still had a PPI of 72. We realized that this was not a useful metric in determining whether an image was "high resolution." Moreover, if we had images that were 300 PPI, this would slow down our map generation and machine learning algorithms significantly, likely making us unable to meet these requirements. A better metric for camera quality in our use case would be color accuracy, which images can be compared to predefined charts to ensure the colors are accurate. This would ensure that coral color is being accurately recorded. Another important camera quality would be image sharpness, which would allow the user to be able to see detail within the image enough to identify predators on the coral or other coral disease besides bleaching [9]. We were limited in our camera options because we had to be able to connect the camera to an Arduino or Raspberry Pi, because we cannot transfer images wirelessly directly from an underwater camera.

VIII. Project Management

A. Schedule

Approximately halfway through our implementation period, we created an updated Gantt chart, which is attached in the as Table II. While we initially followed the timeline outlined in our design report, several delays affected our overall project – most notably waterproofing.

The propeller motors that we originally purchased were marketed as waterproof however, immediately upon delivery we realized that they were not waterproof and would not meet our needs. The ordering and delivery of a suitable replacement immediately put us about 2.5 weeks behind schedule. Despite this setback, as well as multiple following, though smaller, setbacks that resulted from water damage to our underwater camera, we ultimately recovered enough time to complete all core functionality by the final project demo and poster session.

B. Team Member Responsibilities

Abigael was primarily responsible for the machine learning algorithm for classifying coral health, including both developing and validating the model. Additionally, Abigael worked on the integration of the temperature sensor, construction of waterproof camera cables, and was our go-to in the pool team member during testing sessions. Abigael took the lead on waterproofing our entire boat system, ensuring that no water leaked or splashed into the main, above-water section, damaging any electronics. Abigael took the lead on our final video as well.

Emma was responsible for the construction of the boat boat, as well as the processor as a whole. Processor responsibilities include the setup and testing of all motors, WiFi connection, and live stream video set up. This involved a last minute pivot from using an RPi to Arduino for motor control, which Emma researched and set-up a new interface via Arduino Cloud. She also created the user interface and backend connection to allow remote control of the hardware systems. Emma took the lead on the final report as well.

Maddie was responsible for the map generation algorithm portion of post-processing, which was modified throughout the implementation process to no longer require the capture of still images from the livestream feed, as the feed automatically saved as images rather than video. Additionally, Maddie designed and constructed the mounting system for all of the on-boat hardware, including the spool system. Additionally, Maddie was our team's soldering expert, transferring all electronics from protoboards to perfboards. Maddie took the lead on the final poster as well.

Fig. 7. Schedule example with milestones and team responsibilities

C. Bill of Materials and Budget

As our project developed, we went through several design iterations and pivots, mostly related to the need to waterproof our product. We used almost every item purchased, with the exception of the first set of motors we ordered which were not waterproof as marketed, and a cover for our underwater camera which was also not waterproof. Additionally, our first temperature and pH sensor ended up not working. Most of our unexpected costs came from different waterproofing methods. Please refer to Table I for a more detailed description of the bill of materials and the budget.

D. TechSpark Usage

Our team used Techspark, as well as IDeATe, as a resource for laser cutting acrylic components including the underwater camera box and the spool/hardware mounting. We also borrowed tools from TechSpark such as tape measures, hot glue guns, scrap metal, tape, and much more. No construction or testing took place in Techspark.

E. Risk Management (used to be Risk Mitigation Plans in Design Document)

The largest risk associated with our project was the nature of it happening in the water. This presented a high probability of us damaging various electronic components. In order to mitigate this, we did intensive research on waterproofing techniques, including various tapes, liquid latex, silicone caulk, and more. We also opted for pre-waterproof components when possible, otherwise using a variety of methods to waterproof components ourselves. Despite these efforts, both our camera and camera cable sustained damage a number of times throughout the process. We anticipated this, however, and preemptively reduced the strain this damage would cause to our project by opting for cheaper components, allowing us to purchase multiple initially so that we had replacements on hand. At no time in the process did we run out of a component due to it having been damaged by water.

The second risk that we associated with our project was our underwater images not being high-resolution enough for our post-processing to be successful. In order to mitigate this, we prioritized having scaled down versions of our post-processing code completed early in the process. This allowed us to continually test, and have a baseline expectation for the results, anytime a modification was made that impacted image capture quality. Luckily, we were able to successfully waterproof the camera in such a way that allowed for clear enough imaging.

IX. Ethical Issues

Reef rover attempts to improve coral ecosystem monitoring so that scientists have reliable and accurate data. Understanding when coral reefs are at risk and giving scientists more time to react has many positive effects. As mentioned earlier, coral reefs have attributes that are used in medicines to treat human diseases, improving public health. Coral reefs protect shorelines by taming waves, which increases public safety by protecting people from dangerous currents. As for public welfare, coral reefs protect marine ecosystems that are a major food source.

Coral reefs exist in oceans all over the world, and they are a major resource for countries who have reefs within their borders. For example, Mexico has many coral reefs, which protect the land from flooding. They estimate that for every 1m of coral lost, 15,000 people will be at risk for flooding with annual damages of 452 million USD. Worldwide, coral reefs save 94 million USD in flooding damages annually. This phenomenon occurs because coral reefs reduce wave power by about 97%.

Coral reefs are a cornerstone of many cultures, allowing fishing to become a prominent occupation and source of food. Research has shown that every square kilometer of coral reef can provide between 5 and 10 tons of fish annually. This is a source of food and income for local communities, which has an estimated local economic benefit of 6.8 billion USD a year. In addition, 97% of these fishermen live in developing countries, meaning coral reefs are a major source of independence and survival for many people around the world.

Socially, coral reefs attract many tourists to the tropical coastlines. 350 million people travel to see coral reefs annually. Their beauty is unmatched, and this tourism also brings in billions of dollars in revenue [10].

While Reef Rover aims to give scientists a resource to preserve these reefs for all of the ethical reasons listed above, we must ensure that we designed Reef Rover in a way that does not harm the reef or that Reef Rover can’t be easily used with ill intentions. We limited the size of the submersible so that it would have the smallest impact on the ocean life underwater. However, there are some improvements that could be made to limit the chance of accidental reef contact. For example, we could add sensors to the submersible to detect when it is within a certain distance of a coral and automatically stop the boat from moving towards it. Also, we could protect the software with passwords so that data cannot be tampered with.

Other future design considerations would be to use materials that are very durable so that parts do not contribute to ocean pollution. Additionally, more metrics should be determined to define what areas Reef Rover can be used for, and ensure that Reef Rover is not deployed in areas it is not intended for. If it

were to be deployed in an area where coral is too deep, for example, the data collected could be less accurate than desired, harming scientific research. Ultimately, misuse of Reef Rover could harm ocean life and, indirectly, humans by continuing to put coral reefs in jeopardy. Some ethical concepts that arise are accountability, if researchers deploy the device improperly and do not take responsibility for this misuse. Trust, if researchers misconstrue data collected by the device. Responsibility, if the device is not used in a moral or ethical manner. Bias, if certain reefs that are in a geographical region are not monitored because of political conflict or racial discrimination. Overall, Reef Rover's mission is to combat the effects of climate change on coral reef ecosystems, but there are critical ethical considerations that need to be taken into consideration for scaling up this device.

X. Related Work

There is a move to use robots to help scientists monitor and understand coral reef ecosystems. In the article "A systematic review of robotic efficacy in coral reef monitoring techniques, Marine Pollution Bulletin," many types of robots are discussed including underwater vehicles (UAVs), and aerial robots. Aerial robots take photos from satellites to detect bleaching. UAVs are newer technology, and the category that Reef Rover falls into. There are three categories of UAV: the ROV, AUV, and HUV. ROVs are remotely operated vehicles, which is what Reef Rover is. They are medium sized robots that are manually launched by scientists. AUVs are autonomous underwater vehicles that are deployed and collect data on their own, usually on the ocean floor. These robots have complex algorithms that change the robots motion and can even use sound in the environment to guide themselves.HUVs, or hybrid underwater vehicles, are a combination of the two. Reef rover will be a ROV. Some of our use case requirements have been inspired by a robot and algorithm that detected coral only, with an 89% accuracy, and a battery life of 2 hours.

XI. Summary

Reef Rover is a laptop controlled water robot that features an above water boat design with a submersible sensing and camera unit connected to the boat. Our design uses image recombination and machine learning to classify surveyed areas into different levels of coral health in order to help researchers monitor levels of coral bleaching over time, which is an important indicator of overall ocean health. Reef Rover provides a video livestream and real time temperature data, and map generation to the user with low latency. There were a few areas where Reef Rover could have improved, including implementing a higher resolution camera, a remote control with more movement precision, and a backend software connection between the live stream and map generation algorithm. Moreover, our classification accuracy was about 5% lower than desired. Due to limited image data and time constraints, we were unable to improve this to 90%, however with more data points and more time to evaluate the boundaries between classifications, we could increase our accuracy to 90%.

Additionally, Reef Rover is non invasive to the ocean environment and does not require training divers or potentially putting humans at risk due to the nature of underwater work. Through image recombination and classification, scientists will be able to extract consistent data using the same metrics across different areas, something that is key to this area of research. They will also be able to track changes in the area over time more easily. Overall, Reef Rover provides scientists with a conveniently sized and durable opportunity to monitor coral reefs without expending the resources needed to send human divers, making research more accessible.

Some things that we learned going through the design process was that it is important to get the end to end product working as fast as possible, while making sure subsystems worked well. We ran into difficulties with integrating our stepper motor into the Arduino cloud and moving all our components onto one Arduino board. We should have accounted for more integration time and left more time to assemble the boat and test in the water. The waterproofing was a continued challenge, and testing multiple ways of waterproofing early would have made testing in the water more successful.

There is a lot of future work to be done to scale this project up to a full ocean environment. For example, we would design and manufacture a more durable and waterproof boat body. Additionally, we could add sensors to the underwater camera to detect how far away from a coral it is and prevent the user from getting too close. Lastly, we could make Reef Rover more autonomous, like others on the market, so that it could survey an area defined by scientists on its own without human control. Overall, Reef Rover gives scientists a powerful tool for coral reef monitoring, but there is more to be done for the future ocean use.

Glossary of Acronyms

GPIO – General-Purpose Input/Output
KNN – K-Nearest Neighbors
ML – Machine Learning
RPi – Raspberry Pi
Pi - Raspberry Pi
Cloud - Arduino Cloud
ROV - Remotely Operated Vehicle
AUV - Autonomous Underwater Vehicle
HUV - Hybrid Underwater Vehicle

UAV - Underwater Vehicle
PPI - Pixels Per Inch

References

[1] Jennifer A. Cardenas, Zahra Samadikhoshkho, Ateeq Ur Rehman, Alexander U. Valle-Pérez, Elena Herrera-Ponce de León, Charlotte A.E. Hauser, Eric M. Feron, Rafiq Ahmad, A systematic review of robotic efficacy in coral reef monitoring techniques, Marine Pollution Bulletin, Volume 202, 2024, 116273, ISSN 0025-326X, https://doi.org/10.1016/j.marpolbul.2024.116273. (https://www.sciencedirect.com/science/article/pii/S0025326X24002509)

[2] Collvy. (2022, August 7). Controlling DC Motor with Raspberry Pi and L298N motor driver. YouTube. https://www.youtube.com/watch?v=OV0S_3KMj2A

[3] Camera software - Raspberry Pi Documentation. (2025). Raspberrypi.com. https://www.raspberrypi.com/documentation/computers/camera_software.html#libcamera-vid

[4] Electronic Wizard. (2024, August 14). How to Build a Remote-Controlled Boat from Scratch: DIY RC Boat Project. YouTube. https://www.youtube.com/watch?v=SYsk6XNfR4o

[5] Lazy Tech. (2023, July 20). How to Setup a Raspberry Pi and Access it Remotely! (Headless setup). YouTube. https://www.youtube.com/watch?v=m6aS9YF-0xo

[6] US Department of Commerce, N. O. and A. A. (n.d.). NOAA CoRIS - Regional Portal - Puerto Rico. Www.coris.noaa.gov. https://www.coris.noaa.gov/monitoring/biological.html

[7] Manderson, T., Li, J., Dudek, N., Meger, D., & Dudek, G. (2016). Robotic Coral Reef Health Assessment Using Automated Image Analysis. Journal of Field Robotics, 34(1), 170–187. https://doi.org/10.1002/rob.21698

[8] Ataa, M. S., & Sanad, E. E. (2025, April 23). *A fast secure and more reliable underwater communication system based on Light Fidelity Technology*. Nature News. https://www.nature.com/articles/s41598-025-96484-8

[9] Image quality factors (key performance indicators). Imatest. (n.d.). https://www.imatest.com/docs/iqfactors/

[10] Brajcich, K. (2025, April 21). *How coral reefs support local communities*. Sustainable Travel International. https://sustainabletravel.org/coral-reefs-local-communities/#:~:text=Physical%20Protection,face%20of%20increasing%20climate%20change.&text=In%20addition%20to%20protecting%20buildings,of%20coral%20reefs%20every%20year

[11] Raijin. “Healthy and Bleached Corals Image Classification.” Kaggle, https://www.kaggle.com/datasets/vencerlanz09/healthy-and-bleached-corals-image-classification.

TABLE I. BILL OF MATERIALS

Description	Model	Camera	Quantity	# Cost
Microcontroller	Raspberry Pi 4	ECE Inventory	x1	$0.00
Microcontroller	Arduino Uno R4 WiFi	Arduino	x1	$27.50
Camera	Module 3 Standard - 12MP Autofocus	Adafruit	x1	$25.00
Camera	Arducam for Raspberry Pi Camera Module 3, 12MP IMX708 75°(D)	Arducam	x1	$45.99
BNC connector	XMSJSIY BNC Female to Dupont Male Wire Test Leads Cable Connector	XMSJSIY	x1	$9.80
Temperature and pH sensor	Double-Junction Electrode with 3 Ft Cable	Gigicial	x1	$13.29
Temperature Sensor	GAOHOU DS18B20 Waterproof Digital Temperature Sensor with Adapter	GAOHOU	x1	$12.94
Flex Cable Sleeve 1m	1 meter Product ID: 2143	Adafruit	x3	$3.95
Camera Cable	Long Flex Cable for Raspberry Pi Camera - Black 1m/3.28ft (5 Pack)	A1 FCCs	x1	$14.99
Power Supply	KBT 12V 5200 m Ah Rechargeable Li-ion Battery	Keep Better Tech	x1	$32.99
Stepper Motor	STEPPERONLINE Nema 17 Stepper Motor Bipolar 2A 59Ncm (84oz. in)	StepperOnline	x1	$13.99
Driver	HiLetgo 5pcs DRV8825 Stepper Motor Driver Module	HiLetgo	x1	$14.49
RPI to Motor Adapter	L298N Motor Drive Controller Board DC	BOCOLL	x1	$6.90
Propeller Set	EUDAX 6 Set DC Motors Kit	EUDAX	x1	11.99
Underwater Motors	10-20V RC Boat Waterproof Underwater CW CCW Propeller	NUHFUFA	x2	$48.75
Heat Shrink	XHF 2 Inch (50mm) 3:1 Waterproof Heat Shrink Tubing	XHF	x2	$16.52
Patch and Seal Paste	Gorilla Waterproof Patch and Seal Paste White Sealant 1 Poun Can	Gorilla	x2	$21.93
Sealant	Gorilla Glue Clear Caulk and Seal, 10 oz	Gorilla	x1	$9.84
Foam Board	11x14 inch White Foam Boards, 1/8" Thick	Mat Board Center	x1	$14.69
Waterproof Tape	T-Rex Flexible Waterproof Tape	T-Rex	x1	$25.85
Liquid Sealant	17.5 Oz(500g) Liquid Waterproof Sealant	DSZHSNJ	x1	$9.99
Camera Cover	Arducam Camera Case	Arducam	x1	$11.44
Additional Shipping Charges				$70.07
Total Cost				$558.00

TABLE II. REVISED GANTT CHART