Autonomous Power Cable Robot for Deterring Cormorants

A Major Qualifying Project Report:

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

in Robotics Engineering

By

RUSHAB PATIL

Project Advisor

PROF. GREGORY LEWIN

Additional Contributors

KOHMEI KADOYA

LUCAS BUERMEYER

TRANG PHAM

JAY YEN

ARTHUR AMES

SIDHARTH SHARMA

January 26th, 2022

Abstract

Large numbers of cormorants perch on powerlines to dry their wings, as a result most cormorants defecate in the pond. This defecation alters the water's chemistry such that it kills fish and causes other environmental issues. Additionally, cormorants are a loud bird species and disturb the residents in the vicinity of cormorants roosting locations. The goal of this project is to create an inexpensive, automated solution to deter cormorants from landing on the powerlines. This project aimed to design and test a prototype solution that can be field-tested. The robot developed during the project can traverse the power cable, detect the bird using ultrasonic sensor, and switch on sound and light deterrent.

Acknowledgements

I would like to thank my primary advisor, Professor Gregory Lewin for helping me throughout this project. Especially for being patient and pushing me to produce better results.

I would also like to thank Kohmei Kadoya, Arthur Ames, Trang Pham, Lucas Buermeyer, and Jay Yen for all the help, support, and advice during the project.

Table of Contents

Abstract 2

Acknowledgements 3

Table of Figures 7

Table of Tables 9

Chapter 1. Introduction and Background 10

1.1. Current Situation 11

1.2. Benefits of Solving the Problem 13

1.3. Past Solutions and Their Limitations 14

Chapter 2: Problem Statement and Objectives 16

Chapter 3: Project Requirements and Constraints 17

3.1. Requirements 17

3.2. Constraints 17

Chapter 4: System Overview 18

4.1. Hardware Architecture 18

4.2. Control System 18

4.3. Drive Train 18

4.4. Bird Detection 19

4.5. Bird Deterrents 19

4.6. Wireless Communications 19

4.7. Docking Station 19

4.8. Robot Function Overview 19

Chapter 5: Subsystems 21

5.1. Control System 21

5.1.1. Hardware 21

5.1.2. Software 23

5.1.2.2. Finite State Machine 24

5.2. Drive Train 24

5.2.1. Gears 26

5.2.2. Wheel Configuration – Front view - Cross Section 26

5.2.3. Wheel Configuration – Side View 27

5.2.4. Roller/Wheel Design 28
5.2.5. Spring Mechanism 29
5.2.6. Hinge Mechanism 29
5.2.7. Bumpers 29
5.3. Bird Detection 30
5.3.1. Hardware 30
5.3.2. Software 31
5.3.3. Mechanical Mount 32
5.3.4. Unit Testing 33
5.4. Bird Deterrents 35
5.4.1. LED Strobe Lights 35
5.4.2. Ultrasonic Directional Sound 36
5.5. Wireless Communications 37
5.5.1. Cellular 38
5.5.2. Radio Control 38
5.5.3. Wi-fi 39
5.6. Docking Station 40
Chapter 6: System Integration 43
6.1. System Changes 43
6.2. Hardware Wiring 43
6.3. Software Design 46
6.3.1. State Machine 47
6.3.2. Web Control 51
Chapter 7: System Testing and Validation 54
7.1. Subsystem Testing Criteria 54
7.1.1. Power Supply 54
7.1.2. Drive Train 54
7.1.3. Control System 54
7.1.4. Bird Detection 54
7.1.5. Bird Deterrents 54
7.1.6. Wireless Communication 54
7.2. Lab Tests 55
7.2.1. Test 1 - Drive Train and Radio Communication 55

7.2.2. Test 2 - Drive Train and Web Control ... 55
7.2.3. Test 3 - Drive Train, Deterrents, Power Supply, and Autonomy ... 56
Chapter 8: Future Work ... 57
8.1. Control System ... 57
8.2. Bird Detection ... 57
8.3. Bird Deterrents ... 57
8.4. Wireless Communications ... 57
8.5. Power Supply ... 57
8.6. Docking Station ... 58
8.7. Weather Proofing ... 58
Chapter 9: Discussion ... 59
Chapter 10: Conclusion ... 61
References ... 62
Appendices ... 65
Appendix 1.1. Battery Box ... 65
Appendix 1.2. Speaker Mount ... 66
Appendix 1.3. Docking Station ... 67
Appendix 1.4. TPU Roller ... 68
Appendix 1.5. Electronics Box ... 69
Appendix 1.6. Gimbal Top ... 70
Appendix 2.1. State Machine Code – boaw_sm.py ... 71
Appendix 2.2. Arduino Code – main.cpp ... 78
Appendix 3. Lab Test Video Links ... 82
Appendix 4. Turning Sound into a Laser ... 83

Table of Figures

Figure 1: Double Crested Cormorant [10] 10
Figure 2: Cedar Pond Location - Zoomed Out 11
Figure 3: Cedar Pond Location - Zoomed In 12
Figure 4: Average Number of Cormorants on Cedar Pond Power Lines (2011, 2012) [14] 12
Figure 5: Mass of Total Nitrogen and Total Phosphorous in Cedar Pond during Summer 2012 [14] 14
Figure 6: Hardware Architecture 18
Figure 7: Robot's Function Overview 20
Figure 8: Pixhawk 2.4.8 [2] 21
Figure 9: Arduino Mega 2560 [16] 22
Figure 10: Jetson Nano [1] 23
Figure 11: Drive Train - Side View 25
Figure 12: Drive Train - CAD Rendering 1 25
Figure 13: Drive Train - CAD Rendering 2 26
Figure 14: Wheel Configuration - Side View 27
Figure 15: Legend for Figure 14 27
Figure 16:Roller Wheel Design 28
Figure 17: Roller Wheel with three contact points 29
Figure 18: Bumper and Hinge Mechanism 30
Figure 19: Raspberry Pi Cam Module 2 31
Figure 20: MaxBotix Ultrasonic Rangefinder [19] 31
Figure 21: Camera Mount 33
Figure 22: Ultrasonic Sensors positions 33
Figure 23: Cormorant Detection – 1 34
Figure 24: Cormorant Detection – 2 34
Figure 25: Cormorant Detection - 3 34
Figure 26: Cormorant Detection – 4 35
Figure 27:LED Disc 36
Figure 28: Array of Strobe Light on a Mount 36
Figure 29:Ultrasonic Transducers 37
Figure 30: Array of Ultrasonic Transducers on a Mount 37
Figure 31: Cellular Modular attached to Jetson Nano 38
Figure 32: RadioLink Radio Controller [5] 39
Figure 33: RadioLink Receiver [24] 39
Figure 34: Wi-fi M.2 Key 40
Figure 35: Docking Station - Prototype 1 41
Figure 36: Docking Station - Prototype 2 41
Figure 37: Prototype 2 mounted on a power cable 42
Figure 38:Old System Architecture with Pixhawk 43
Figure 39: Electronics Wiring 45
Figure 40: Software Architecture 46

Figure 41: State Machine - States Hierarchy.......... 47
Figure 42: State Machine: Startup Code.......... 48
Figure 43: State Machine - Mode Selection State.......... 49
Figure 44: State Machine - Teleoperation Mode.......... 49
Figure 45: State Machine - Autonomous Mode.......... 50
Figure 46:Web Control Interface - Window 1.......... 52
Figure 47: Web Control Interface - Window 2.......... 52

Table of Tables

Table 1: Detection Accuracy (%) of different image classifier and Training Time (hours) ……………………….32

Chapter 1. Introduction and Background

Certain species of birds, such as geese, pigeons, flamingos, and cormorants, are known for flocking together in large groups [12]. Whenever they flock together on human properties, they cause considerable damage to it. They can be a significant bother for businesses, especially those handling food, such as food warehouses, food processing factories, and bakeries. Birds are infamous for causing problems like fouling from defecation, damaging the roosting sites and spreading disease through droppings [11]. Bird droppings can also corrode stone, metal, paintwork, and other building materials [11]. These droppings contain many types of bacteria, viruses, and parasites that can contaminate food, surfaces, and goods. The physical damage and disease spread can cause economic losses to the communities. The community living near the Cedar Pond in Orleans, MA faced similar problems due to flocking of cormorants on the power distribution line going over the pond.

The Double-Crested Cormorant (Phalacrocorax *auritus*) is a migratory coastal bird that resides in Cape Cod, Massachusetts from April to October, with the greatest number of cormorants around from August to September [6]. They feed on a variety of fish, which requires diving into water bodies. As such, they have less preened oil than other bird species, allowing their feathers to become wetter than those of other bird species [6]. Cormorants prefer to dry their wings in the sun usually near fishing sites [7]. These perching locations usually include trees, overhead power lines, and buildings nearby.

Figure 1: Double Crested Cormorant [10]

The power distribution lines over Cedar Pond are popular spots for hundreds of cormorants to gather and dry their wings during the day in late summer and early fall. These birds have caused water quality and noise problems for the community living in the vicinity of Cedar Pond.

1.1. Current Situation

Starting 25 years ago, cormorants began roosting on the 25kV power distribution line spanning Cedar Pond in Orleans, MA [14]. The distribution line over the pond is right above the fishing site and provides an ideal spot for drying wings. As of 2015, at least 400 cormorants perched on the line at a time [14]. While on the line, the birds habitually defecate into the pond below. Cormorant droppings are acidic and contain chemicals such as nitrogen and phosphorus, and various bacteria, altering water and soil chemistry, resulting in the death of fish and loss of vegetation, respectively.

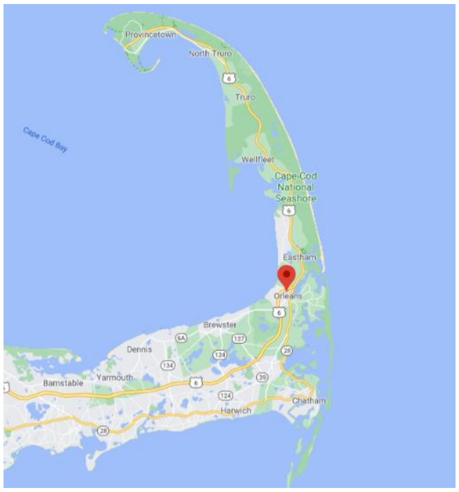


Figure 2: Cedar Pond Location - Zoomed Out

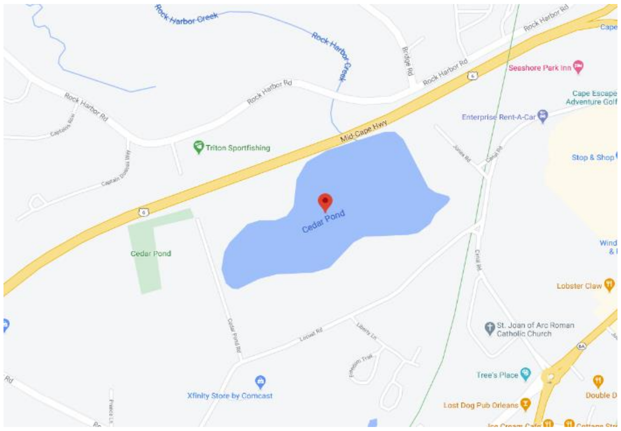


Figure 3: Cedar Pond Location - Zoomed In

In addition to being an environmental hazard, cormorants are a nuisance to residents. Their droppings have a strong odor that reduces property values. The substantial number of cormorants congregating on the line also creates a considerable racket that disturbs nearby residents.

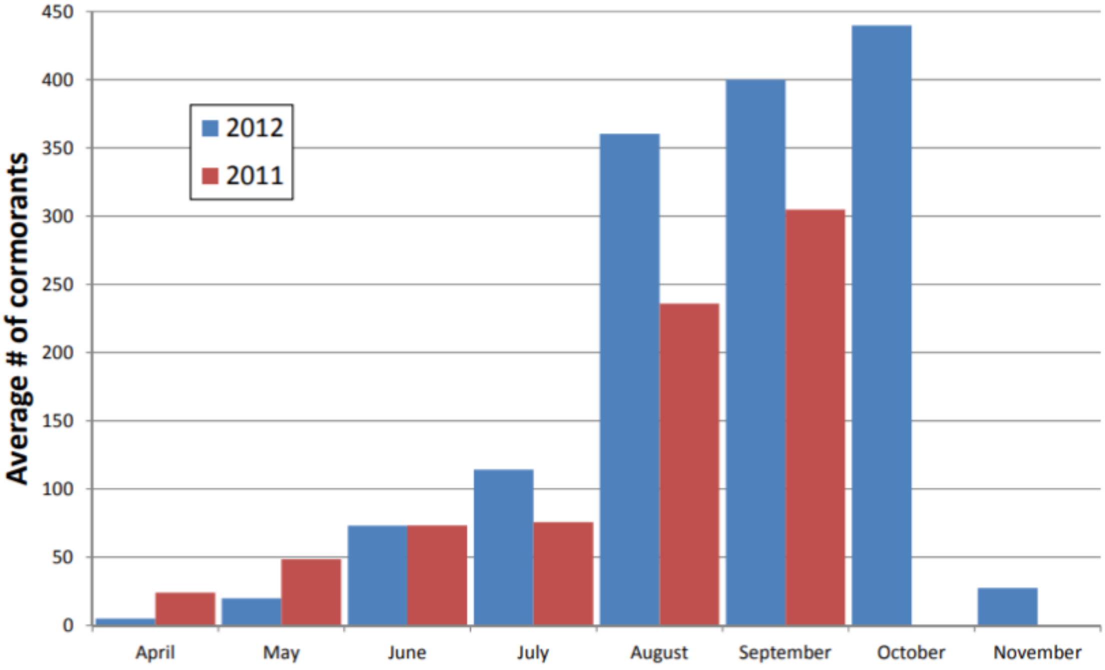


Figure 4: Average Number of Cormorants on Cedar Pond Power Lines (2011, 2012) [14]

In recent years, the town of Orleans has tried to restore the pond, which suffers from dissolved oxygen impairment and fish kills, among other issues. To restore the pond, the town would first have to resolve the cormorant issue. Residents have tried scaring the birds away, but the solutions only worked temporarily. A more permanent solution could include moving the lines. Eversource is an electric provider that was responsible for putting these lines over the Cedar Pond. The town and Eversource negotiated for several years before reaching an agreement. In 2019, Eversource moved the 25kV distribution lines from over the pond to nearby Locus Road, over land. Though this solution was successful in drastically reducing the number of cormorants over the pond, it was expensive and labor-intensive, and a few birds still roost on the 150kV transmission lines still above the pond. Though Eversource removed the power distribution line with lower voltage, they do not have any intention of removing the transmission lines with higher voltage [8].

1.2. Benefits of Solving the Problem

When the number of cormorants roosting over the power lines increases in fall, the amount of total Nitrogen and Phosphorus content in the water also increases. A study in 2011 and 2012 showed the trends of the Nitrogen and Phosphorus content of the water in the fall season. It showed that the arrival of cormorants and increase in their defecation caused approximately 100% and 125% increase in the Nitrogen and Phosphorus present in Cedar pond's water. Resolving the issue of cormorants will not only improve the water quality but also the soil quality, which will result in improving the surrounding habitat. With a smaller number of cormorants on the wire, the neighborhood will be more peaceful because of reduced noise disturbance.

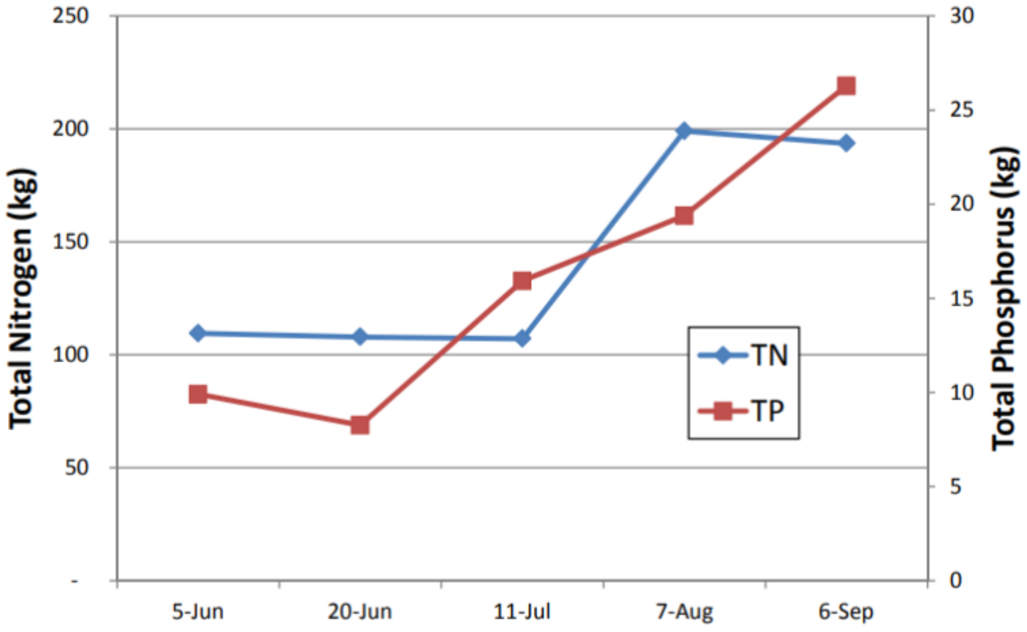


Figure 5: Mass of Total Nitrogen and Total Phosphorous in Cedar Pond during Summer 2012 [14]

1.3. Past Solutions and Their Limitations

Cormorants prefer to perch on the power lines because of their proximity to the feeding sites in Rock Harbor and Cape Cod Bay, in addition to Cedar Pond below. There are three options to address cormorant issues: scaring them away from the power lines, making the power lines difficult to perch on, and/or moving/removing the power lines.

Removing the power lines over ponds involves burial of these power lines. These Costs of burial of existing lines range from $400,000 per circuit mile to $1.6 million per circuit mile depending on the details of the site used [14]. Typically, $1 million per circuit mile is used as a planning estimate [14]. Additional issues to the burial of power lines involve working through the burial details with MassHighway in case the wire goes below the highways. Resolving these issues will add overhead cost to the total cost. Though Eversource removed the power distribution line above Cedar Pond, they will not continue to remove lines because of the cost involved. Moving/removing the power lines is not a sustainable solution to the problem and was ditched to find other low-cost solutions.

Given that removing the local cormorant food sources is unrealistic, the remaining option to prevent cormorant contamination of Cedar Pond is to prevent or minimize their roosting on the power lines. Auditory scare tactics like alarm and distress calls played over loudspeakers, exploding shells, and automatic gas exploders have already been tried [14]. In addition, visual scare tactics, such as kites, balloons, reflectors, and lasers have also been tested [14]. Performance review of these methods showed that they succeed initially, but then the bird population becomes desensitized to their presence. Also, considering the feature of providing little or no disturbance of residents around Cedar Pond, visual scare tactics were preferred. Out of all the options, lasers showed the most promise. Based on USDA evaluation of use of lasers on double-breaster cormorants, the populations were reduced by at least 90% of pretreatment levels after 1-3 days of consistent use [15]. USDA evaluations have also shown that the lasers do not harm avian optics, and their usage is considered non-lethal [15]. More testing for this method is required with on a long-term monitoring to gauge its effectiveness on the cormorants. Testing lasers on the wire would require well defined safety procedures to protect travelers in case the power line in question is close to a highway, any aircraft flying over the area, and residents from direct exposure to the laser light [14].

To save the cost from moving/removing power lines and finding alternative solutions to the problem, Eversource presented the problem to National Science Foundation's center Robot and Sensors for Human Well-being (ROSE-HUB). As an active member of ROSE-HUB, WPI volunteered for the project to create an automated system that will use different deterrents to scare away the cormorants and actively keep them away from landing on the wire. In academic year 2020-21, a group of undergraduate students built a robot that went back and forth on the wire and switched on the deterrents when cormorants were detected. This prototype used flashing lights and directional sound speakers as bird deterrents. This system worked as an early proof of concept but needed to go through the redesign process to be able to function in real life scenarios.

Chapter 2: Problem Statement and Objectives

Eversource posed WPI with a problem statement of creating an automated system that will deter the cormorants that are perching on power cables. Based on this problem statement

The goal of this project is to develop an automated robot that can patrol a power line and deter birds from landing or resting on the line. The immediate focus of this Major Qualifying Project is to develop a prototype that can be used to evaluate the effectiveness of different bird deterrent methods and prove autonomous operation.

The objectives of the projects are as follows:

- Design and build a robot that will detect and deter cormorant by traversing over a power distribution cable.
- Test different deterrents to find the most effective deterring methods for scaring away the cormorants roosting over the power lines.
- Design and build a local charging station for the robot so that the robot can function for a long duration without any human interference.

Chapter 3: Project Requirements and Constraints

A list of project requirements was outlined to guide the development of the robot. The deployment of the robot affects the cormorants, birds, and residents of the nearby settlement. System constraints were also established to prevent any negative impacts from the robot.

3.1. Requirements

The list of project requirements is as follows:

- Patrol the power distribution cable.
- Climb a maximum inclination of 30 degrees without slipping.
- Detect birds that are landing on the power distribution cable.
- Perform wireless robot control through reliable long-range wireless communication methods.
- Withstand electromagnetic field due to the current flowing through the power distribution cable.
- Allow easy mounting and dismounting on the cable.
- Trigger failsafe systems for failure conditions.
- Withstand the varying extreme weather conditions like intense winds, heavy rain, and dust.

3.2. Constraints

The list of constraints are as follows:

- Do not harm cormorants while deterring them.
- Do not threaten any endangered bird species.
- Eliminate/minimize any disturbance to the residents in the surrounding areas.

Chapter 4: System Overview

4.1. Hardware Architecture

Hardware architecture, shown in Figure 6, shows the hardware components present in the robot, data being transmitted between them, and directions of the transmission.

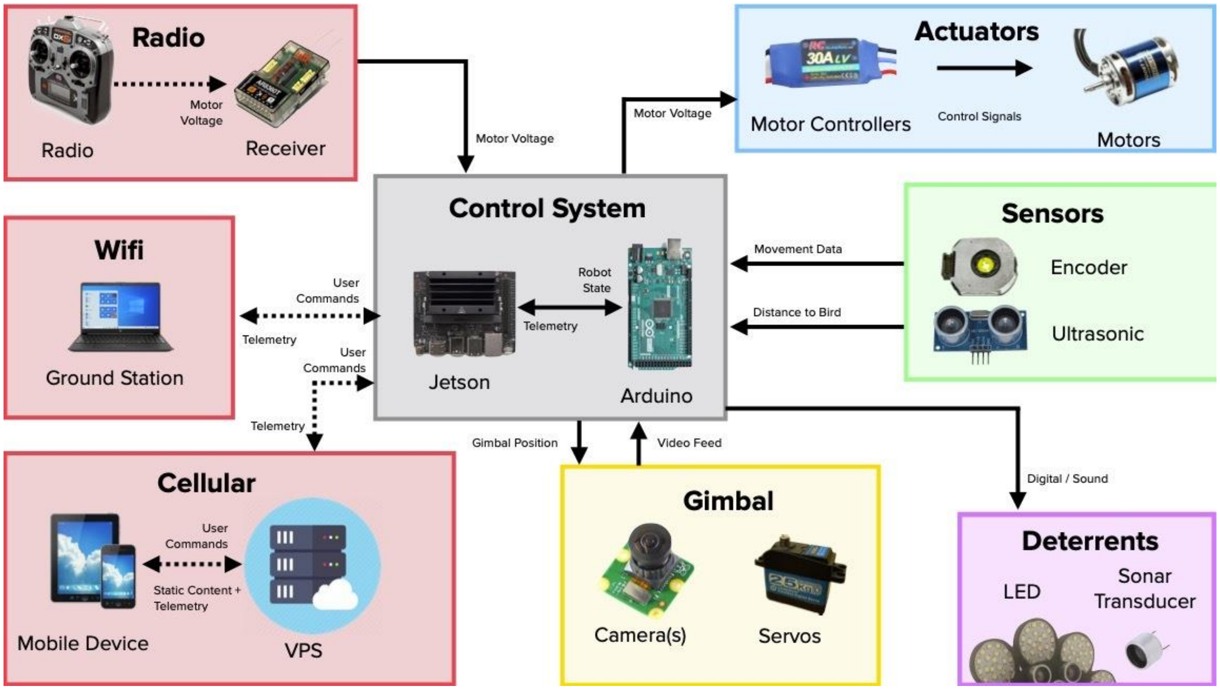


Figure 6: Hardware Architecture

4.2. Control System

The control system is responsible for getting the data from the sensors, processing it, and controlling the robot's motion on the power distribution cable. The control system consists of Arduino Mega and Jetson Nano. Arduino Mega is for controlling the motors of the robot; and Jetson Nano runs the bird detection algorithm and state machine to achieve autonomous operation.

4.3. Drive Train

The drive train provides the functionality of moving back and forth on the power. The drive train uses dual rollers to propel the robot along the wire, either patrolling and detecting birds or chasing birds within proximity. The drive train assembly also features the battery and electronics box, which can be found in Appendix 1.1. and 1.5. respectively.

4.4. Bird Detection

To be able to deter birds, they must be detected first. This is done with the help of onboard camera. This camera is mounted in line with the deterrent methods to aid aiming. Using an object detection model, we can detect birds as they are flying past or sitting on the wire. Two ultrasonic rangefinders, mounted in the front and rear of the robot, are also used as a mechanism to stop the robot when in proximity of the bird. These rangefinders also act as a failsafe system for preventing robot from harming any birds.

4.5. Bird Deterrents

Birds are known to have enough intelligence for pattern recognition, which means the solution must be able to deter the birds in multiple manners. Otherwise, the cormorants will be able to predict the robot's actions and adapt to them, rendering the solution ineffective. The first deterrent method is directional sound. The purpose of this method is to deter cormorants using focused ultrasonic sound. The second deterrent method is the strobe light array. The strobe light is intended to deter the birds with flashing bright light at a high frequency.

4.6. Wireless Communications

When the robot is actively deterring the birds when mounted on the cable, the user needs to be able to control the robot wirelessly in case of a subsystem failure or for doing regular maintenance. This robot uses two wireless communication methods: Cellular and Radio. Both are capable of long-range communication. Primary communication is done through cellular connection and Radio acts as a failsafe system. For testing purposes, Wi-Fi communication is also used.

4.7. Docking Station

The docking station is a pie shaped fixture at the end of the wire. When the robot is running low on power, it will drive over to the docking station and charge there. Additionally, the docking station is the home location to where the robot will return at the end of the day.

4.8. Robot Function Overview

As shown in Figure 7, the robot traverses between two poles while detecting the cormorants. As soon as the cormorants are detected, the robot moves toward them and switches on the light and sound bird deterrents to deter them. When robot is not working autonomously, it can be controlled wirelessly via radio control or cellular connection.

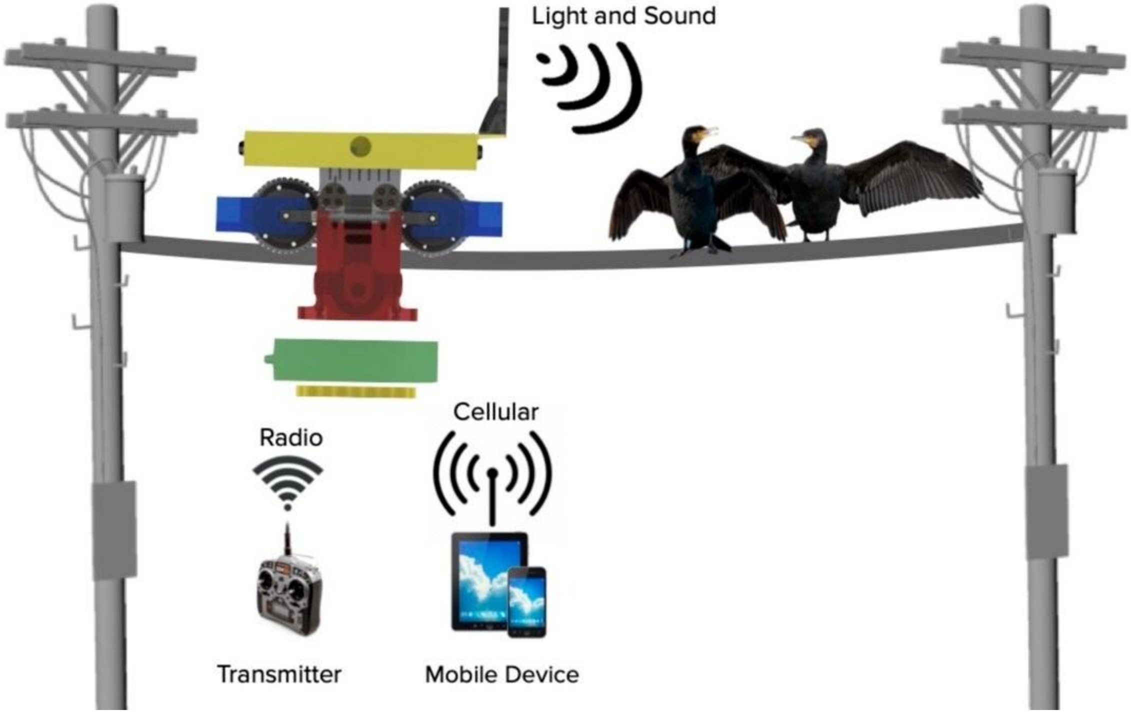


Figure 7: Robot's Function Overview

Chapter 5: Subsystems

5.1. Control System

The control system consists of two microcontrollers, Pixhawk 2.4.8 and Jetson Nano, to provide the capabilities required to achieve the project objectives. Pixhawk 2.4.8. acts as a motor controller that controls the drivetrain. Jetson Nano acts as a companion computer that makes, and relays logic decisions based on the data collected by the onboard sensors during the mission.

5.1.1. Hardware

5.1.1.1. Pixhawk 2.4.8

Pixhawk 2.4.8. is an autopilot board that allows the robot to perform teleoperated and autonomous missions. This autopilot board has a 32-bit ARM Cortex M4 processor with a floating-point unit. It comes with 256 kilobytes of RAM and 2 megabytes of flash memory. Along with the main processor, it also has a 32-bit failsafe co-processor [2].

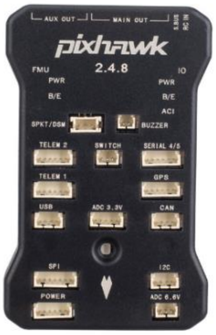


Figure 8: Pixhawk 2.4.8 [2]

The main functionality of Pixhawk is to control the robot’s forward, reverse, and stop motion on the power distribution cable, by executing commands received from Jetson Nano. Pixhawk was chosen over other microcontrollers that can perform the same functionality for several key reasons: support for open-source control software, support for interfacing with a wide range of peripheral devices, advanced data logging, support from the community, and industry-grade reliability. Pixhawk supports the two most popular and mature open-source software: PX4 and Ardupilot. On top of this, it acts as a hub for easily connecting sensors like wheel encoders, rangefinders, GPS, etc. This saves a lot of time during the development and allows users to focus on the custom functionality needed in the robot. The data logging feature allows the user to retrieve the telemetry data during the mission and analyze using the inbuilt graphing tools. Having a failsafe processor also adds redundancy to the system.

5.1.1.2. Arduino Mega

Arduino Mega 2560 is a microcontroller board with 54 digital input/output pins. As the Pixhawk was getting tested to control the drive train motors, Arduino Mega allowed parallel development and testing of ultrasonic rangefinders, directional sound speakers, and strobe lights.

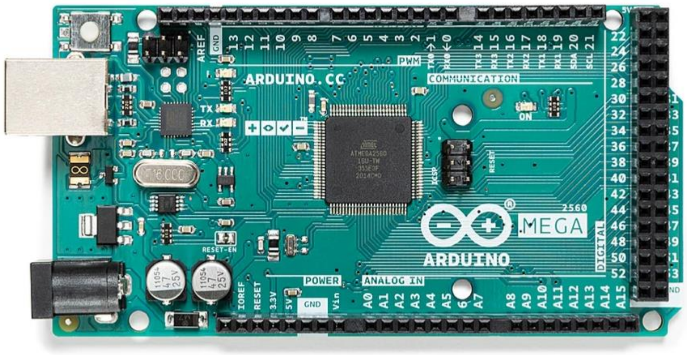


Figure 9: Arduino Mega 2560 [16]

5.1.1.3. Jetson Nano

Jetson Nano acts as a companion computer that provides additional processing power to Pixhawk and enables advanced functionalities like object detection. It is a single board computer that is designed for embedded applications and artificial intelligence. Jetson Nano has an ARM-based CPU with quad-core, each with a maximum clock speed of 1.43 GHz [1]. Its 4GB RAM also helps with multitasking [1]. To help with the artificial intelligence application, it comes with the NVIDIA 128-core Maxwell GPU [1].

Figure 10: Jetson Nano [1]

ROS is an open-source software framework that provides a collection of tools, libraries, and conventions that simplify the process of combining complex technologies to build robust robotics systems [3]. ROS was designed to be distributed and modular. ROS does this by using the concept of packages. Each package is a set of nodes that are interconnected using ROS communication protocols. Each node is an executable program that performs computations and makes decisions while the application is running.

Jetson Nano uses ROS's framework to run the State Machine and Object Detection during the autonomous operation of the robot.

5.1.2. Software

5.1.2.1. Ardupilot

Ardupilot is the autopilot firmware uploaded on Pixhawk board to control and customize its functionalities. Ardupilot is open-source software and is deeply tested and developed by a large community of professionals and enthusiasts [4]. It can control conventional and VTOL airplanes, gliders, multirotor, helicopters, sailboats, powered boats, submarines, ground vehicles, and even Balance-Bots [4]. In addition to the advanced control software, Ardupilot also provides tools for real-time data logging, graphing, analysis, and simulation.

Ardupilot has different versions based on the type of vehicle. In this case, the ArduRover 4.1.0 version of ArduPilot is being used for controlling the robot. ArduRover is specifically made for wheeled robots with different configurations. This robot uses skid steering control to move back and forth on the cable.

5.1.2.2. Finite State Machine

Finite State Machine is a computation model that can be implemented with software to simulate sequential logic [17]. A finite state machine with predetermined states is used to control this robot. This state machine software runs on Jetson Nano and takes in input from the peripheral sensors to makes decisions during an autonomous operation. For teleoperation, the state machine changes the states of the robot by listening to commands from the radio controller.

5.2. Drive Train

To achieve the task of being able to patrol the cable, the robot needed a robust drivetrain that can traverse over the cable effectively without slipping. After static analysis and five iterations of the drive train, the current drive train of the robot has two driven and one sprung wheel. The two wheels are driven by two brushless motors with a rating of 1300 kV running at 14.8 volts. Gear reduction of 1:3 to get the speed of 3 m/s. Each motor can pull the robot assuming that the robot's weight is equal to or below 2 kilograms. Two-motor configuration was selected to add motor redundancy to the robot.

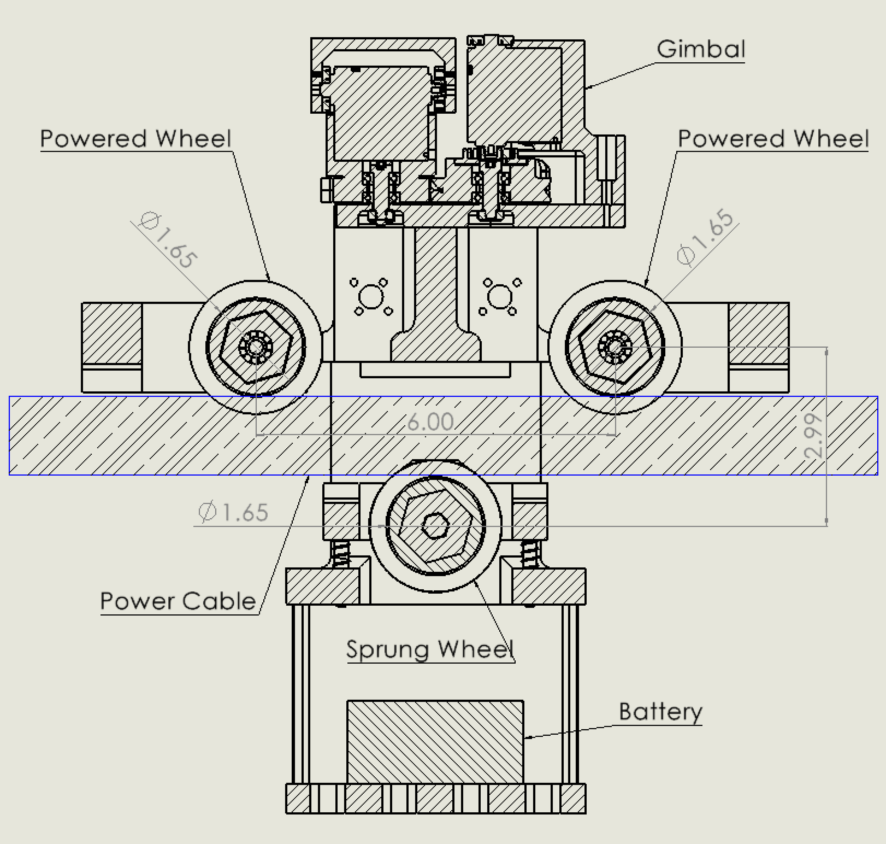


Figure 11: Drive Train - Side View

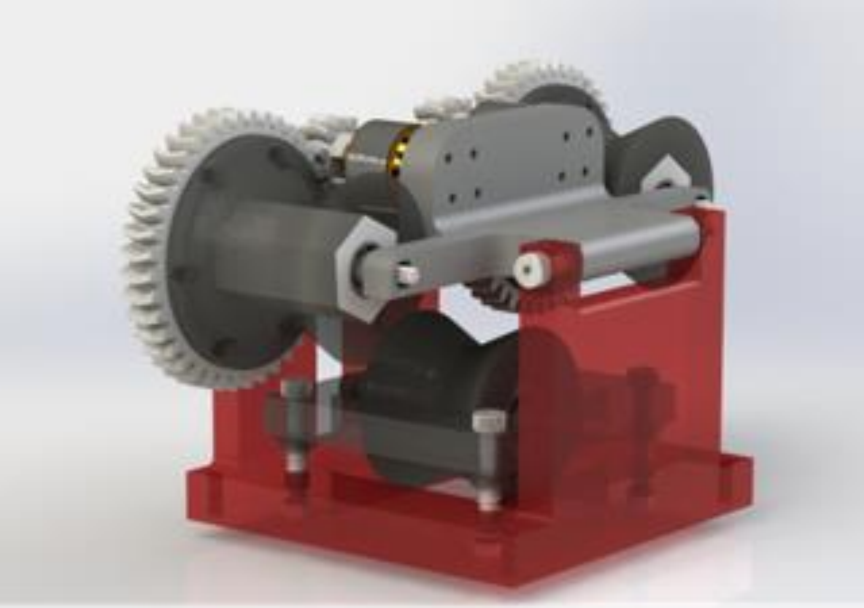


Figure 12: Drive Train - CAD Rendering 1

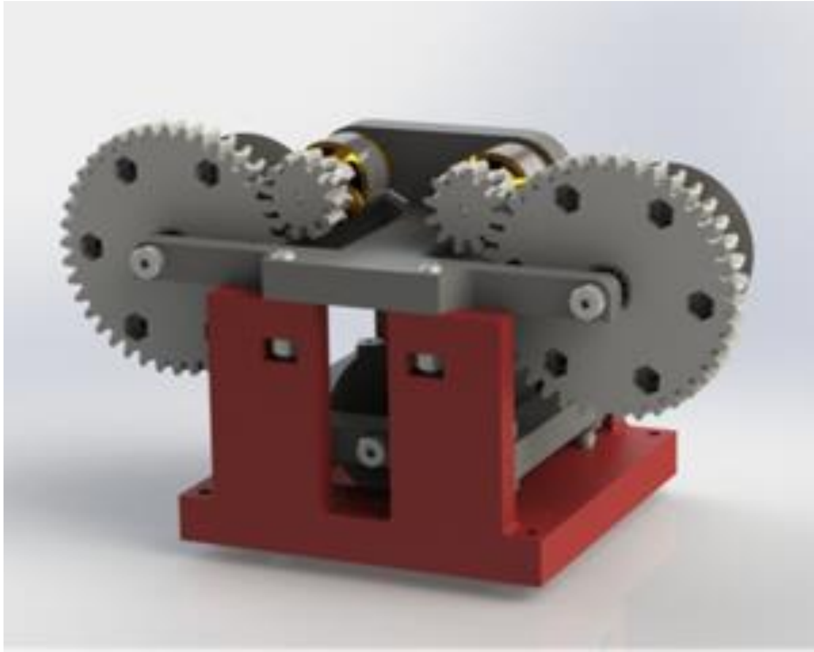


Figure 13: Drive Train - CAD Rendering 2

5.2.1. Gears

The drive train is geared with 3.33:1 ratio for torque with 2 sets of identical but mirrored gear transmissions. The Poly Lactic Acid (PLA) 3D printed gears have a self-aligning herringbone tooth profile. This minimized the amount of hardware needed to rigidly attach the smaller gear axially since there will not be a large axial force between the motor shaft and the gear. Physically, the gears have a 20-degree pressure angle, a 15-degree helix angle, and a diametrical pitch of 2mm. The smaller (driving) gear has 12 teeth with a 24mm diameter, and the larger (driven) gear has 40 teeth with an 80mm (about the length of the long edge of a credit card) effective diameter. With a theoretical max RPM (Rotations Per Minute) of over 21000 rpms on the smaller gear and 6552 rpms on the larger gear, heat caused from friction could melt off the teeth. Therefore, the smaller gear was printed with nylon with a melting point of 260 C compared to the larger gear which was printed in PLA (melting point 175C). Like the rest of the 3D prints, these parts were printed at 15% infill density, 0.15mm layer height, and triangular infill pattern.

5.2.2. Wheel Configuration – Front view - Cross Section

To make sure that the wheel configuration provides enough grip to the robot, different cross section wheel configurations were also evaluated to select the current design. Current design uses two concave wheels shown one on top and second on the bottom of the cable. Three different configurations were considered to replicate the action of gripping the wire. Different wheels were considered to increase the amount of the normal force between the power cable and the robot providing better grip through increased friction. During testing, it was found that each configuration provided the same effect. Thus, the final decision of choosing the two-wheel configuration was based on ease of manufacturing. Three-wheel configuration required manufacturing the drive train with extruding portions at an angle. Designing suspension for wheels at an angle was also time consuming and required machining complex parts. Four-wheel configuration required two wheels on either side of the cable, which made it difficult to design a mechanism for easily mountable and unmountable robot.

5.2.3. Wheel Configuration – Side View

The current drive train uses a three-wheel configuration, option 2 in the Figure 14, with sprung roller wheel at the bottom and the two driven wheels on the top with the cable going between them. While finalizing the wheel configuration, multiple options were considered. Option 1 had two wheels on the top and two sprung wheels in the bottom. Option 2 had two driven wheels and one sprung wheel on the bottom. Option 3, like option 2, was also a three-wheel configuration but in this configuration one wheel was driven supported with two sprung wheels.

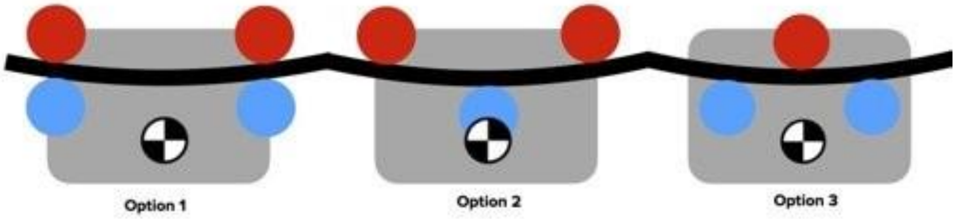


Figure 14: Wheel Configuration - Side View

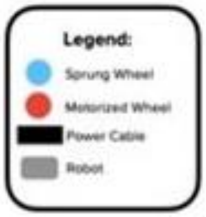


Figure 15: Legend for Figure 14

During the wheel configuration finalizing process, the selection was done based on the configuration's ability to accommodate and traverse seamlessly over a curved cable. Option 2 and 3 were best options based on the criteria defined. Fundamentally both the options had the same functionality, but Option 2 was selected because it had two driven wheels that allowed the redundancy of the motors. Another reason for selecting option 2 over option 3 was that it minimized the number of springs used in the system.

5.2.4. Roller/Wheel Design

Roller profile played a key role in gripping the cable and maximizing the traction available from the springs. Based on the application, two design requirements were listed for the roller profile: it should be able to accommodate wires with different thicknesses and it should maximize normal force from the springs. Current robot has the roller profile shown in Option 1. For fulfilling the requirements, three different options were considered. Option 1 has an angular profile with two contact points; Option 2 has a circular profile with one contact point; and Option 3 has an angular profile with a nub in the center that gave three contact points between wire and the roller.

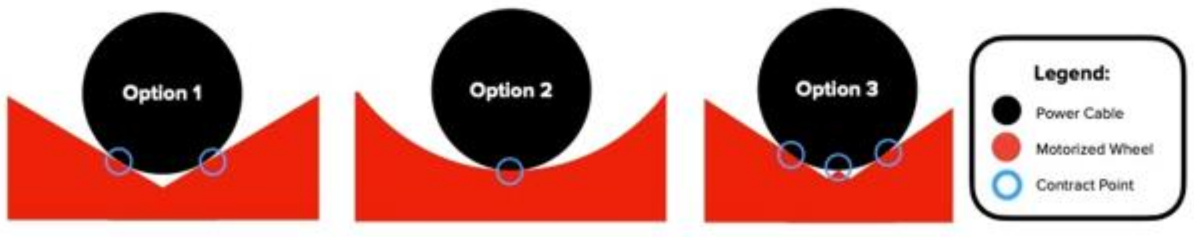


Figure 16:Roller Wheel Design

With three contact points, Option 3 was great option for maximizing the normal force transferred from the springs, but it failed to accommodate the wires with different diameters, which can be seen in Figure 17. Option 2 was the most efficient in terms of transferring the normal force, but it also failed in work with different cable diameters. The design of the roller can be found in Appendix 1.4.

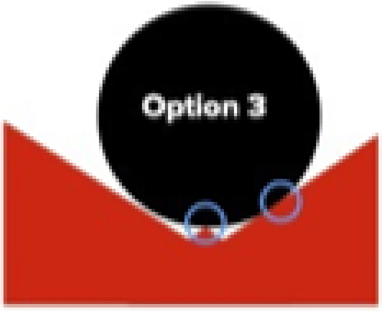


Figure 17: Roller Wheel with three contact points

5.2.5. Spring Mechanism

The traction force from the gravity of the robot was not enough to keep the robot from slipping on the cable with the inclination of 30 degrees. To prevent the slipping, the spring mechanism was used to provide the extra traction force. The spring mechanism has a non-driven wheel mounted on four springs. Static analysis was performed to approximate the extra traction force other than provided by gravity was required to keep the robot statically stable. This force was then used to approximate the spring constant of the springs used. For selecting the springs for prototypes the robot was assumed to have a weight of 2 kilograms.

5.2.6. Hinge Mechanism

Ease of use was one of the most important requirements for the robot. The two main physical interactions that the user would have with the robot are while mounting and unmounting the robot from the cable. A hinge mechanism was added to make this interaction convenient for users. The cross-section wheel configuration, in section 5.2.2., of the robot allowed us to divide the robot into two halves: the top half with driven wheels and the bottom half with the sprung wheel. This hinge mechanism, shown in Figure 18, connects the two halves, and allows the user to flip open the top half of the robot, wrap around the desired power cable, close the hinge, and mount the robot at the desired location.

5.2.7. Bumpers

While testing the robot on the wire, accidentally a wire got loose, and the robot accelerated to full speed and hit a pole on one side of the wire. This collision disintegrated all the subsystems on the robot. To protect the robot from such collisions, bumpers were added on both sides of the robot. These bumpers were 3d printed using Thermoplastic Urethane (TPU) filament. TPU filament was selected because of Its durability and flexible nature. As shown in Figure 18, the bumpers are placed at the front and the rear of the robot.

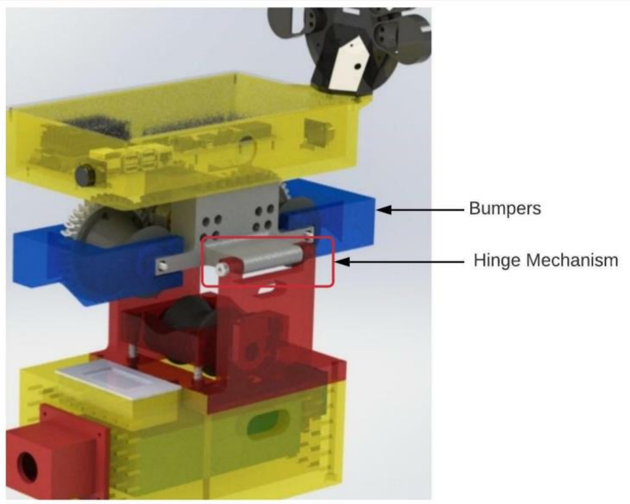


Figure 18: Bumper and Hinge Mechanism

5.3. Bird Detection

Detecting birds is a key functionality required for the robot to be able to keep the cormorants from landing on the power distribution cable. The detection of the cormorants also had to work in real time. To detect the bird, two kinds of sensors, a camera, and ultrasonic sensors, are being used. Combination of these two types of sensors is necessary for the application because they eliminate each other's limitations. For this application, the robot needed to detect birds but also prevent hitting the bird. Raspberry Pi cam 2 detects the Cormorants and MaxBotix Ultrasonic rangefinder keeps track of the distance between the robot and any other solid object. Raspberry Pi Cam 2 uses computer vision with deep learning to detect the bird, while the ultrasonic rangefinder detects the nearest object to the robot.

5.3.1. Hardware

5.3.1.1. Raspberry Pi Camera Module 2

Raspberry Pi camera module 2, in Figure 19, has a Sony IMX219 8-megapixel sensor and can take both high-definition video and still photographs [9]. This camera uses MIPI-CSI connector for connecting with Jetson Nano. There were two different interfacing options available for selecting cameras for bird detections. These two options were based on the camera interface available on the Jetson Nano: USB 3.0 and MIPI-CSI. Camera Serial Interface (CSI) is a high-speed protocol used for image and video transmission between cameras and host devices like mobile phones, tablets, laptops, etc. USB 3.0, Universal Standard Bus, is a standard interface used for connecting computers and electronics [18].

Figure 19: Raspberry Pi Cam Module 2

5.3.1.2. MaxBotix Ultrasonic Rangefinder

MaxBotix Ultrasonic Rangefinder, shown in Figure 20, can detect objects from 0 inches to 254 inches (6.45 meters) and gives sonar range information from 6 inches out to 254 inches with 1-inch resolution [19].

Figure 20: MaxBotix Ultrasonic Rangefinder [19]

5.3.2. Software

For detecting Cormorants through the camera using computer vision, EfficientDet image classifier is used. Object detection has a high computation expense, which makes it necessary to have an object detection model that can adapt to a wide range of resources/computational constraints. EfficientDet is a scalable and efficient object detection method that is suitable for embedded systems like Jetson Nano that has fewer computational resources.

EfficientDet is based on a family of scalable and efficient object detection networks (EfficientNet), where a novel Bi-directional feature network (BiFPN) has been incorporated [20]. This helps EfficientDet achieve state-of-the-art accuracy while being up to 9x smaller and using significantly less computational resources to prior state-of-the-art detectors [20]. Previous detectors relied on ResNets, ResNeXt as backbone networks which are either less powerful or have lower efficiency than EfficientNets. By combining EfficientNet backbone and BiFPN, a small size EfficientDet baseline is established [20]. When compared to results from ResNet and YOLO, the accuracy of object detection was improved by 4% while reducing the computation cost by 50%.

Model	Accuracy on CUB-2000 Dataset (%)	Training Time (hours)
SVM with SIFT	17.3	10
ResNet	83	21
PNASNet	85	22
EfficientDet	85	20

Table 1: Detection Accuracy (%) of different image classifier and Training Time (hours)

This image classifier was trained on two image datasets: COCO (Common Objects in Context) and NABirds (North American Birds). COCO dataset consists of more than 200K labeled images of different objects in a variety of contexts [21]. The second dataset, NABirds, is specifically made for detecting North American birds. This dataset consists of 48,562 annotated bird images of 404 distinct species [22]. Each species has more than 100 annotated images in the dataset [22].

5.3.3. Mechanical Mount

Camera is mounted statically on vertical plate with bird deterrents, shown in Figure 21. This was done to target deterrents in the direction of bird detection. Rangefinders are also mounted statically facing both sides of the robot. This acts as a failsafe system that prevented the robot from colliding with any bird landing on either side of the robot.

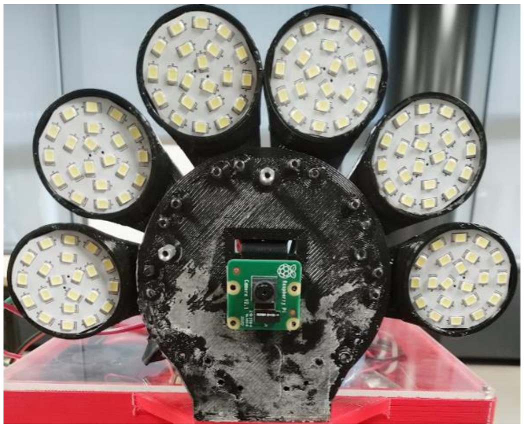


Figure 21: Camera Mount

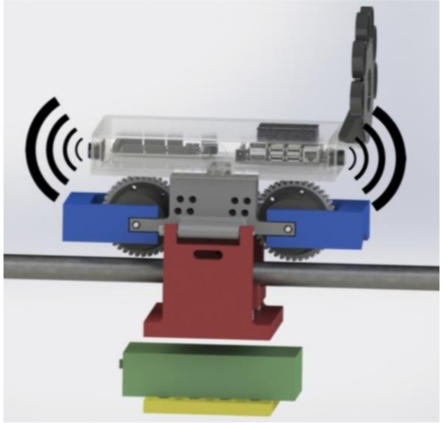


Figure 22: Ultrasonic Sensors positions

5.3.4. Unit Testing

For unit testing, the image classifier ran on the COCO dataset image with a cormorant in the image. Basic test on video samples was also ran to determine how fast the image classifier can process the image frames. After running the tests, the image classifier can detect Cormorants at 7 to 8 frames per second with an accuracy of 85%. The cormorant detections can be seen in the Figures 23, 24, 25, and 26.

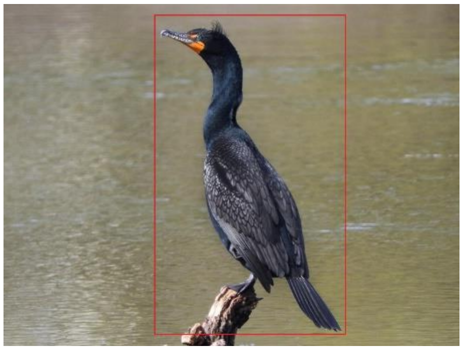


Figure 23: Cormorant Detection – 1

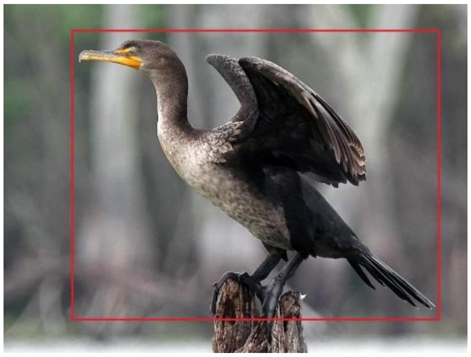


Figure 24: Cormorant Detection – 2

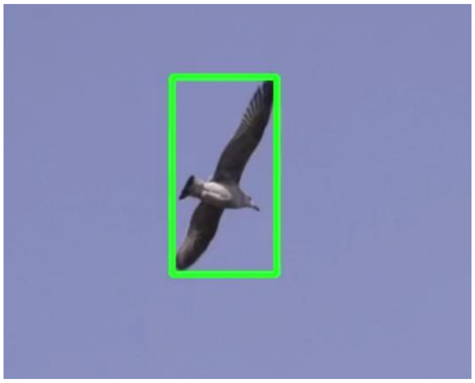


Figure 25: Cormorant Detection - 3

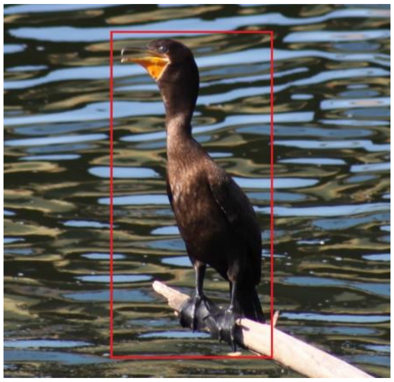


Figure 26: Cormorant Detection – 4

5.4. Bird Deterrents

Once the bird's location with respect to the robot is determined, the robot moves closer to the bird and switches on the deterrent modules to scare away the birds. While deterring the cormorants, these modules must minimize any disturbance caused to the residents in the area surrounding Cedar Pond. There has not been a lot of research on which bird deterrents are effective against cormorants, so this project provides a testbench for different traditional bird deterrents like LED strobe lights and ultrasonic speakers.

5.4.1. LED Strobe Lights

The strobe light is designed to work alongside the directional sound as a deterrent that works at both long and short distances. Seven LED discs, in Figure 27, are mounted around the directional sound, shown in Figure 28, such that the birds are deterred by both audio and visual irritants. To make the strobe light visible in bright daylight, we aimed for the brightness of at least 2000 lumens: each disc has 350 lumens of brightness for a combined total of 2450 lumens. Once activated, the lights flash once every 400 milliseconds, or at 2.5 Hz. This frequency is based on research suggesting flash frequencies ranging from 0.1 Hz to 3 Hz are the most effective in bothering birds [25].

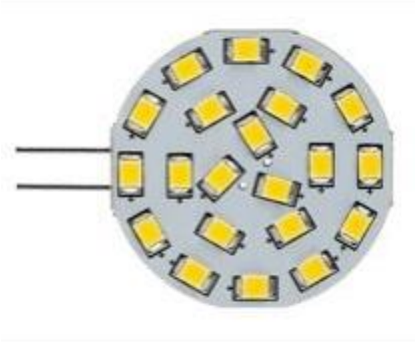


Figure 27:LED Disc

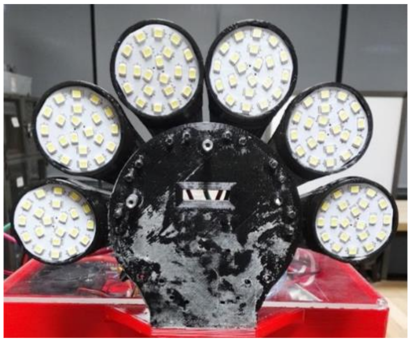


Figure 28: Array of Strobe Light on a Mount

5.4.2. Ultrasonic Directional Sound

Ultrasonic Directional Sound module, in Figure 30, uses a speaker that emits a sound wave modulated with the carrier wave of frequency 40 kHz. The directional sound device is a long-range deterrent that uses ultrasonic transducers, in Figure 29, to emit audible sounds in a highly directional manner. The inspiration behind the directional sound was a speaker that would provide a higher sound output at a distance without being a disturbance to the people around it. Initially, we weighed two options, a parabolic speaker, and an ultrasonic speaker. Because of concerns over the large footprint of a parabolic reflector, the ultrasonic speaker was chosen for the final product. The sound module's design is based on a video, link in Appendix 4, detailing the creation of an ultrasonic speaker using common parts.

Figure 29:Ultrasonic Transducers

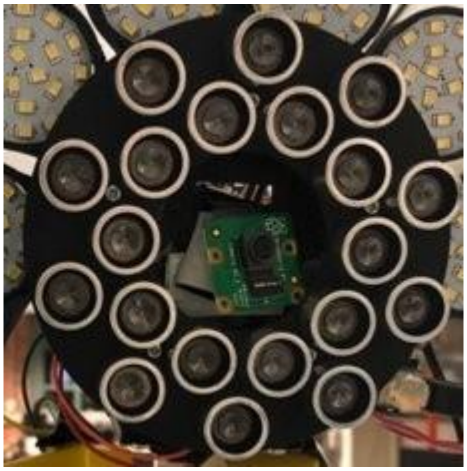


Figure 30: Array of Ultrasonic Transducers on a Mount

The directional speaker consists of twenty 40 kHz transducers wired in parallel arranged in a circular pattern. To drive the transducers, we used a MOSFET controlling a twelve-volt source; a modulated 40 kHz signal is then output from the Arduino to control the MOSFET. Through qualitative testing, we have seen the directional sound can be heard from distances greater than three meters away while also maintaining its focused nature. The sound wave is only audible when it hits a surface in the pointed direction. The sound wave produced is in the hearing range of the Great Cormorant (1 kHz – 4 kHz) [10].

5.5. Wireless Communications

Reliable wireless control of the robot is crucial to perform regular maintenance on the robot when it is deployed on the power distribution cable. If the robot is performing a mission is currently positioned in the middle of the wire's length, away from the reach of the maintenance personnel, wireless connection with robot allows the user to bring the robot to the accessible location. For this purpose, two long-range wireless communications are set up on the robot. Wi-fi wireless communication was added to help with the system tests during the development process. Since most of the testing was done on the WPI campus Wi-Fi communication was set up to take advantage of WPI wireless network.

5.5.1. Cellular

The final robot should be able to communicate over the web irrespective of the location of power distribution cable. To tackle this issue, cellular connection was selected as the main wireless communication method. The cellular connection will be setup with the help of a cellular modem, shown in Figure 31, that will be connected to the Jetson Nano through USB connection. The work for setting up the cellular connection will be done in the later parts of the project.

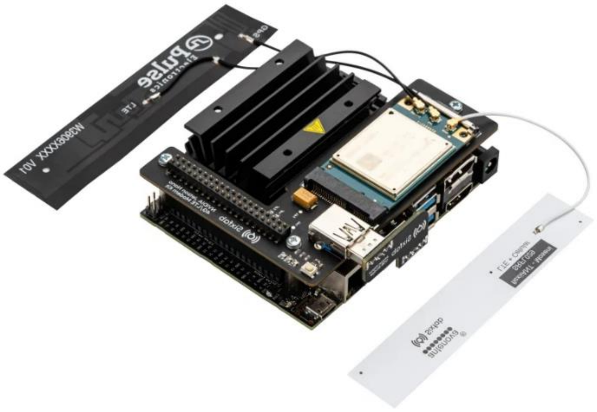


Figure 31: Cellular Modular attached to Jetson Nano

5.5.2. Radio Control

Radio Control (R/C or RC) is the use of control signals transmitted by radio to remotely control a device. The Radiolink transmitter AT9s pro, in Figure 32, operates at the frequency of 2.4 GHz with the R/C distance of 3.4 KM (2.11 Miles) [5]. The length of the power distribution cable is approximately 0.5KM. This transmitter will easily be able to control the robot from any of the ends of the cable. Similarly, the R9DS 8 channel receiver, in Figure 33, would be enough since only three channels are required to control the functioning of the robot.

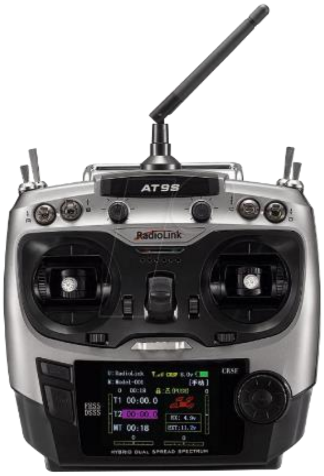


Figure 32: RadioLink Radio Controller [5]

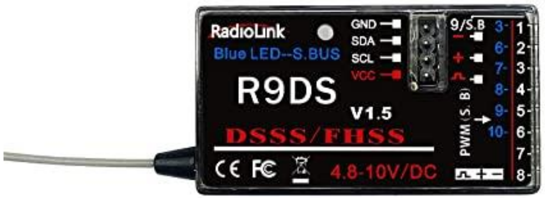


Figure 33: RadioLink Receiver [24]

5.5.3. Wi-fi

To communicate with the robot over Wi-Fi, a M.2 wireless key, shown in Figure 34, is installed on the Jetson Nano. This connects the robot's main computer to the WPI Wireless Network. The network provided IP of the Jetson Nano is used to establish remote Secure Shell (SSH) connection with the robot. This provides exact functionality as connecting with the robot through a wired connection.

Figure 34: Wi-fi M.2 Key

5.6. Docking Station

Originally, electromagnetic induction was considered for charging, among other renewable power sources. However, according to the calculations using Faraday's Law, the wire did not meet the specifications to produce the required power for the robot. Other solutions, such as a small wind turbine or solar panel, also did not meet the required power requirements while maintaining a small size. Thus, we decided to power the robot with a battery that would be recharged by a docking station at the end of the wire.

Presently, the docking station will be powered by a small solar panel attached to the top. Though a small solar panel is insufficient in powering the robot, it can power minimal electronics on the docking station in addition to charging batteries over time while the robot is patrolling. When the robot docks, it is charged from the batteries that have been charged by the docking station.

The station consists of three main parts: a clamp, a hinge, and the main body. The clamp piece rests on top of the wire and screws into the hinge piece with four screws through threaded inserts. The hinge piece is secured to the bottom of the wire by the clamp and holds the charging face such that it can rotate forwards and backward. The body initially was fixed to the clamp, but this interfered with the connection between the bot and docking station due to the curvature of the wire. The hinge accommodates this curvature. The main body of the station is where all the electronics of the station are fixed. These electronics include a power supply, a solenoid, a radio, and a microcontroller. The robot connects to the main body of the docking station using magnets.

The docking station, in Figure 36, is designed such that the robot can easily attach and detach from it. To facilitate attachment, the docking station and robot have two magnets faces that are aligned to connect. Each magnet face is made up of two magnets stacked on top of each other to strengthen the magnetic connection. The interfacing between the robot and the docking station can be seen in Figure 37. Sandwiched between the magnets on the docking station side are wires connecting to a power supply.

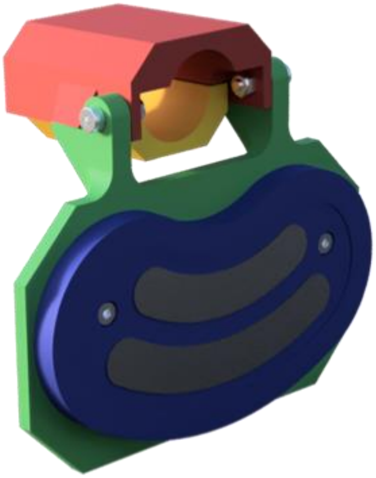


Figure 35: Docking Station - Prototype 1

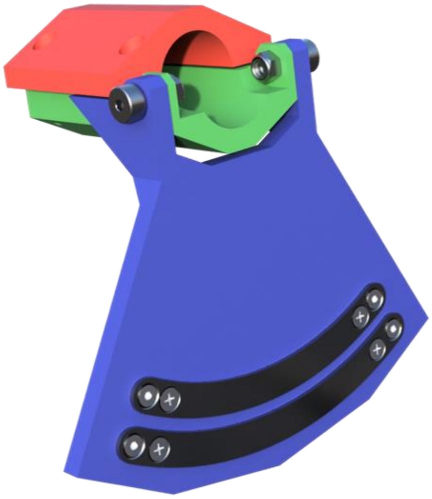


Figure 36: Docking Station - Prototype 2

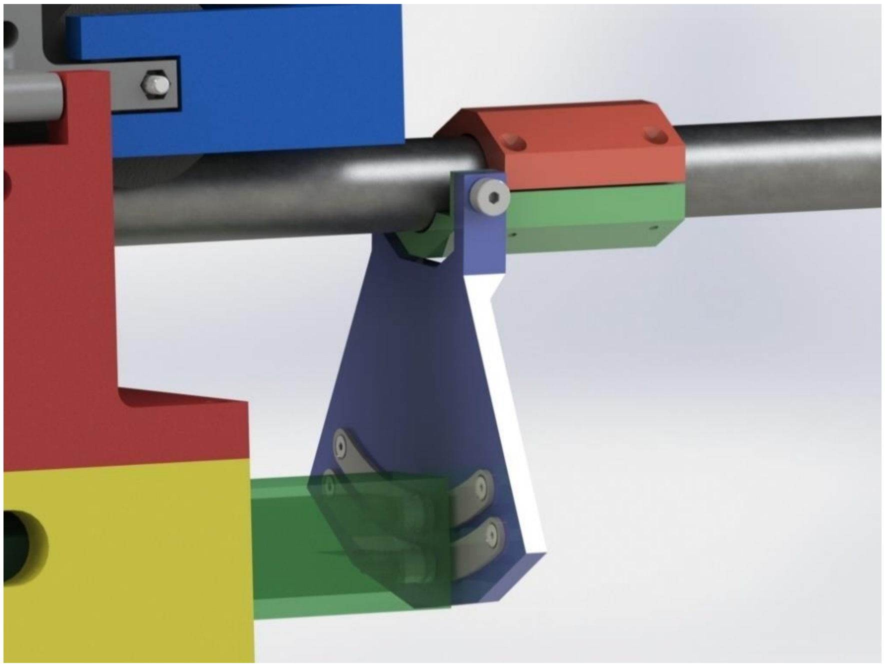


Figure 37: Prototype 2 mounted on a power cable

Chapter 6: System Integration

6.1. System Changes

During the integration, a key decision about the use of Pixhawk versus the Arduino Mega was made. For subsystem unit testing, interfacing of Pixhawk and Jetson Nano was done in parallel with the development of the bird detection and deterring subsystem on Arduino Mega 2560. Though the interfacing of the Pixhawk and Jetson Nano was successfully done, getting this setup ready for testing was taking longer than expected. By the time Pixhawk was ready to do more testing and integrate other subsystems, the progress of the robot with Arduino Mega was ahead. Getting Pixhawk to the same level of functioning as Arduino would have taken exceedingly long time. To save the time spent in making the Pixhawk work and make the robot development faster, Arduino Mega was chosen as the motor controller and hub for interfacing other electronics components. Figure 38 shows the old system architecture. The new hardware architecture can be seen in Figure 6.

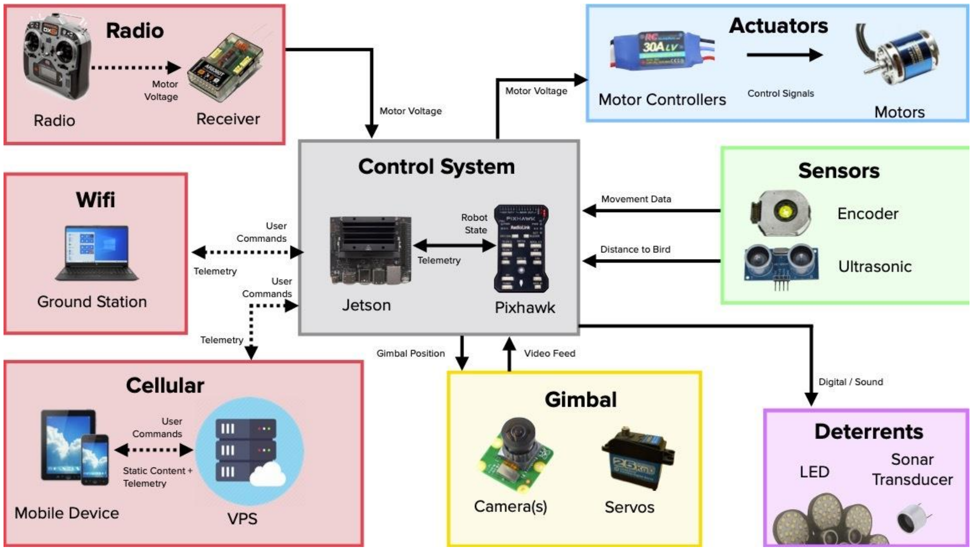


Figure 38:Old System Architecture with Pixhawk

6.2. Hardware Wiring

The following wiring diagram shows all the connections made during the subsystem integration process. Each box in the following diagram represents all the parts of an individual subsystem. While wiring the robot, the greatest number of additions were made to the Power Supply of the robot. During the unit testing of the subsystems, lab power supplies were used to get different input voltage levels, 12 volts for LED Strobe Light and 5 volts for everything else. Getting varying voltage outputs from the same battery needed addition of voltage converters and MOSFET. Additionally, enabling the battery monitoring, emergency stop, and charging the robot from the docking station required connecting more hardware. Battery monitoring was done using a voltage monitoring chip and was connected parallelly to the power supply. Charging the robot from the docking station required the inclusion of the Battery Management System (BMS) that enabled balanced charging of the LiPo cells. The robot also has two magnetic leads, power, and ground, for transferring power from the charging station to the onboard LiPo through BMS. These two leads were also connected parallelly to the BMS as seen in the figure. The emergency switch was added to switch on and off the robot during the lab testing. This emergency stop will be removed in the last version of the robot. The emergency stop is connected in between BMS and electronics to be able to cut off the power supply completely.

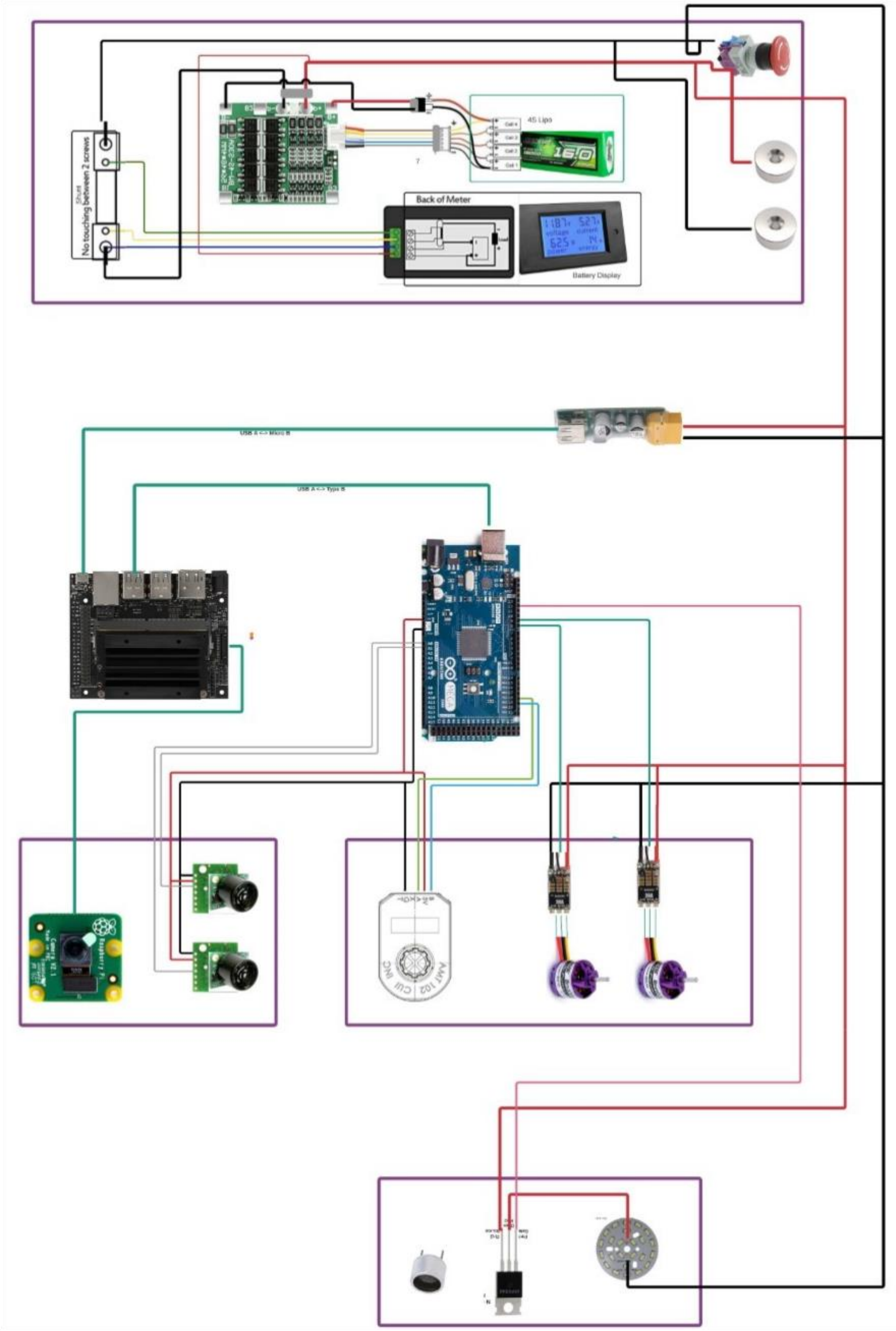


Figure 39: Electronics Wiring

6.3. Software Design

The ROS software package for this project spans three pieces of hardware: Jetson Nano, Arduino Mega, and any computer with internet connection. Jetson Nano runs a State Machine node that is responsible for making all the on-board decisions and relaying them to the respective subsystems. In the network of nodes, Arduino Mega acts as a node which allows for easier communication between Arduino Mega 2560 and Jetson Nano. This is made possible with the help of ROS Serial package available through the ROS open-source community. Figure 40 shows the most updated software architecture.

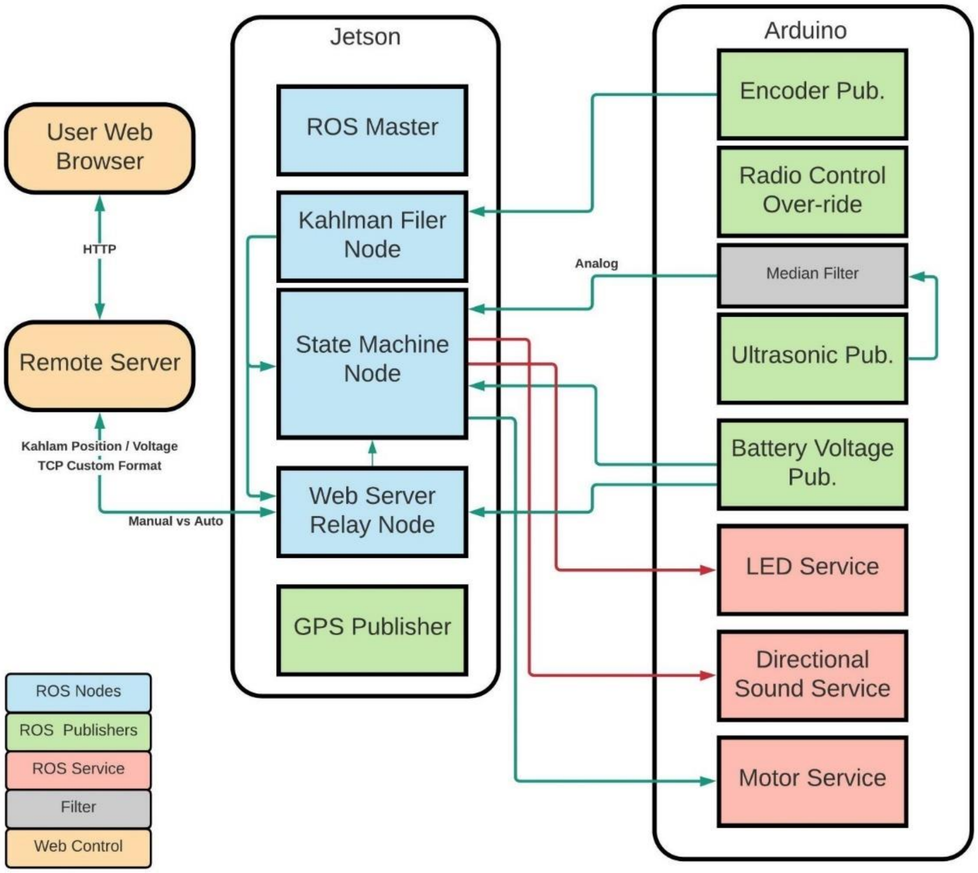


Figure 40: Software Architecture

6.3.1. State Machine

A Finite State Machine was designed to stitch together all the different subsections of the software to produce a cohesively functioning system. State Machine is a behavior diagram that is made up of states and transitions. Based on the current state and input given from the user or the peripheral sensors, the state machine performs the transition and produces a certain output. This state machine design uses a hierarchical state machine (HSM), in Figure 41, to perform manual and autonomous control. The system starts with Startup state that runs some essential start up processes. It also runs a Mode Selection state machine that selects between autonomous and manual mode based on users input from the user interface. Once the mode is selected, the state machine associated with that mode will run. These mode state machines consist of Macro States like Patrolling and Approaching. These macros states are also small state machines that are made up states like move, stop, moveForward, moveBackward, moveAtSpeed. Defining the various levels of states helps with the debugging of the code while running.

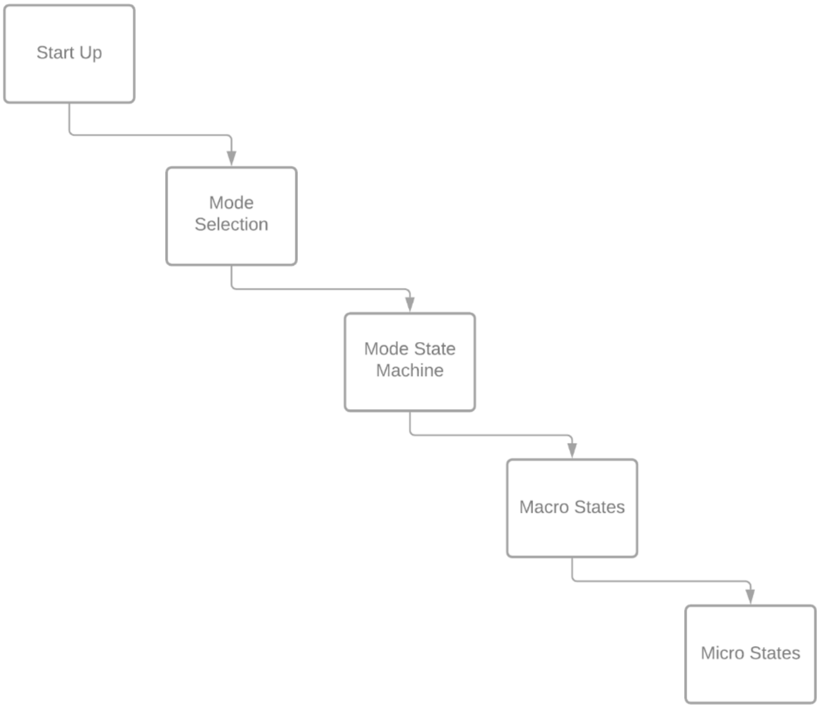


Figure 41: State Machine - States Hierarchy

6.3.1.1. System Start Up

Startup state machine, in Figure 42, starts BatteryLevelMonitor, BirdDetection, Timer, and Mode Selection state machine. BatteryLevelMonitor monitors the battery voltage, BirdDetection detects any cormorants in the vicinity of the robot, and Timer keeps track of the time elapsed since the robot started functioning. These processes are background processes that run while the robot is performing other functions. These processes publish the information that is used in the mode state machine to make transitions between states. After initializing processes, the robot enters the mode selection state machine that waits for the user input to run the rest of the code.

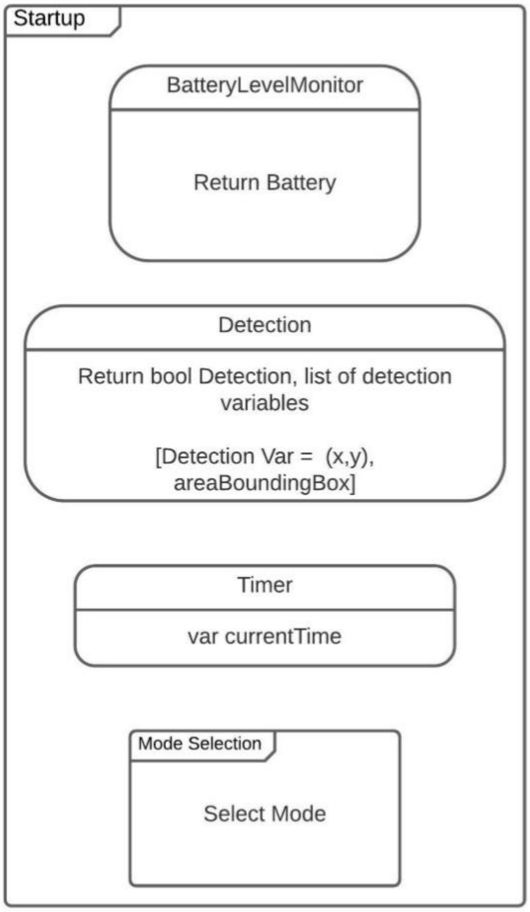


Figure 42: State Machine: Startup Code

6.3.1.2. Mode Selection

Mode Selection State Machine, in Figure 43, makes the robot wait for a user input from a Web User Interface that allows the user to select the robot mode. Two modes that a user can select are Teleoperation and Autonomous. Each of these modes has a state machine associated with them.

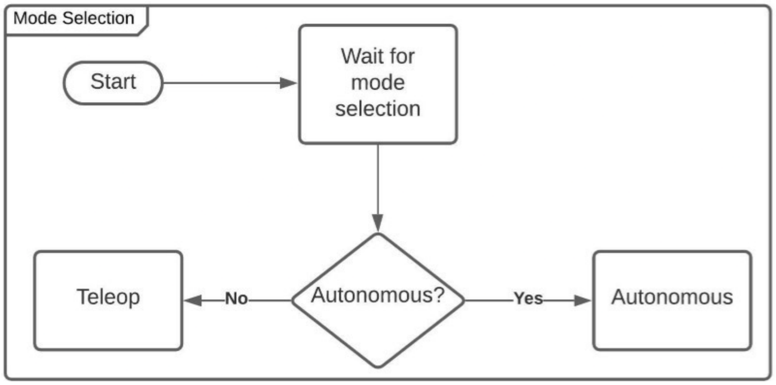


Figure 43: State Machine - Mode Selection State

6.3.1.3. Teleoperation Mode

In Teleoperation mode, in Figure 44, the user can move the robot forward and backward and stop it. The user can also move the robot at a desired speed. This mode also controls the different sensors and deterrents mounted on the robot. This mode is helpful for testing the individual systems without running every other function of the robot.

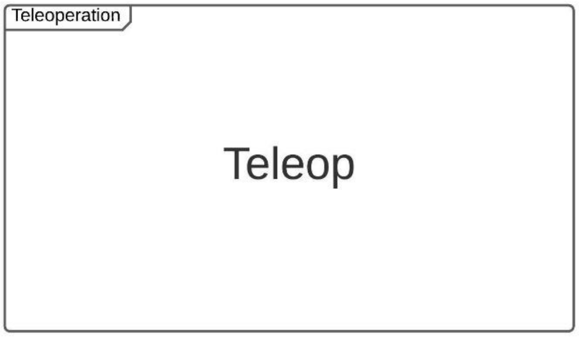


Figure 44: State Machine - Teleoperation Mode

6.3.1.4. Autonomous Mode

In the Autonomous mode, shown in Figure 45, robot starts making decisions based on the inputs from the startup processes at the beginning of the process.

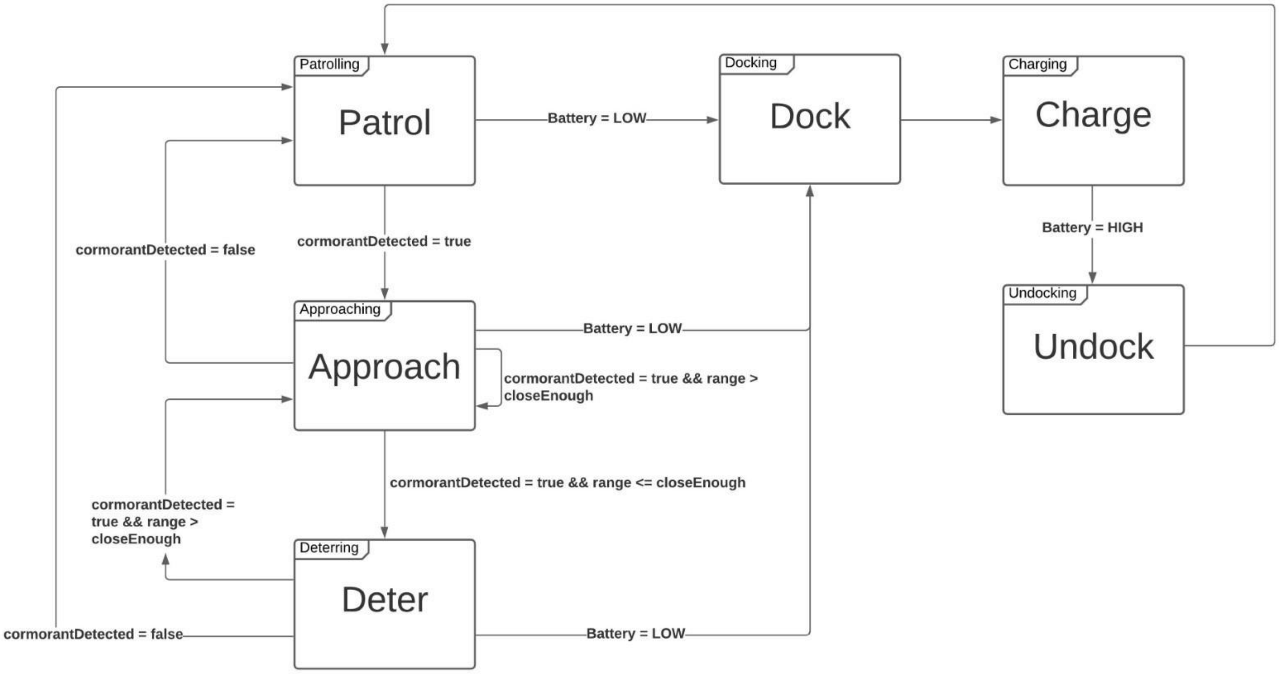

Patrolling
Patrol
Docking
Dock
Charging
Charge
Battery = LOW
Battery = HIGH
Undocking
Undock
cormorantDetected = true
cormorantDetected = false
Approaching
Approach
Battery = LOW
cormorantDetected = true && range > closeEnough
cormorantDetected = true && range <= closeEnough
cormorantDetected = true && range > closeEnough
Deterring
Deter
cormorantDetected = false
Battery = LOW

Figure 45: State Machine - Autonomous Mode

The Autonomous mode state machine is made up of siX macro level state machine and they are as follows:

- Patrol

In this state, the robot goes from one end of the power distribution line to the other. After the "Start Up" state, this is the default state of the robot. The robot switches out of the state only when the mounted camera detects a cormorant in its range.

- Approach

When the bird is detected, robot goes into "Approach" state. This state moves the robot closer to the bird. This state implements velocity control to get closer to the bird without harming. While the robot is approaching the bird, the detection is still running in the background. This is done to keep checking whether the robot needs to keep approaching the bird or not because if the robot stops detecting the bird it should go back to "Patrolling" state.

- Deter

While approaching the bird if the robot detects the bird continuously "Deterring" state will be triggered. "Deterring" state turns on all the deterrents mounted on the robot to scare away the bird.

- Dock

At any point during its operation, if the robot's battery level goes below a threshold value it will drop any assigned task and enter "Docking" state. In this state, the robot moves to the charging station while avoiding collision with any birds.

- Charge

In "Charging" state, the robot has docked to the charging station and is charging.

- Undock

As soon as the robot is fully charged, it will undock through this state and go back to the "Patrolling" state.

6.3.2. Web Control

Web Control software is also currently being developed to allow the end user to monitor and control the robot from the convenience of their work location. This software consists of a front-end and back-end. The front end of the software involves an interactive user interface, and the back end takes care of connecting with the robot and relaying all the information from Jetson Nano to the control website.

6.3.2.1. User Interface

The User Interface (UI) of the web control software allows the user to see the current state of the robot and control its motion. Figures 46 and 47 show the UI mockups for the final design of the Web Control application. These mockups are tentative and are subject to change based on further testing.

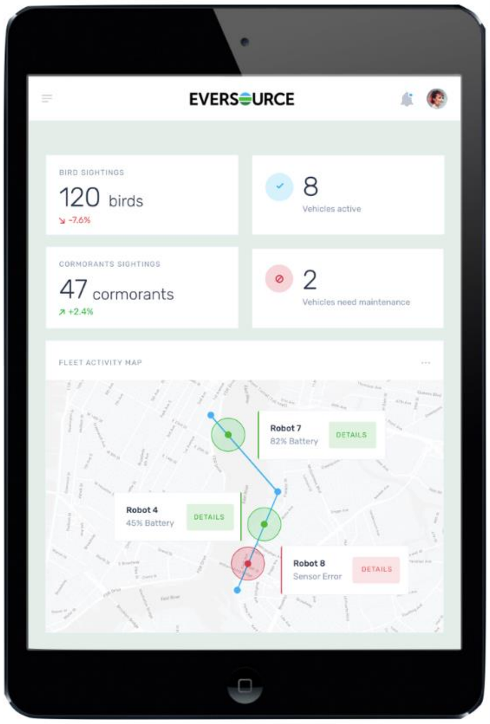


Figure 46:Web Control Interface - Window 1

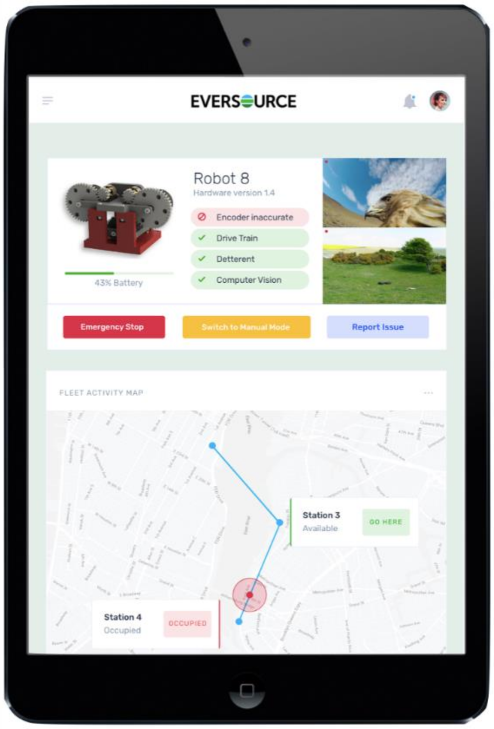


Figure 47: Web Control Interface - Window 2

6.3.2.2. Back End

The back end of the application consists of the webserver and Web Server Relay node running on Jetson Nano. The web server is responsible for hosting the website and data that Is being displayed on the user side. The Web Server Relay node has become part of the ROS node network. This node acts as translation layer between the web server and Jetson Nano. It subscribes to the robot status published on the different ROS topics and converts it into the data packets that can be transferred over TCP protocol. Similarly, any user input from the user interface is converted to ROS messages that can be published on ROS topics that control the different subsystem.

Chapter 7: System Testing and Validation

7.1. Subsystem Testing Criteria

Based on the system requirements and objectives, different success criteria were defined for each subsystem, and they are as follows:

7.1.1. Power Supply

- Supply required voltage to all the electronics hardware
- Provides operation of minimum 12 hours
- Does not short circuit while the robot is functioning
- Stops immediately after the emergency stop button is pressed
- Successfully charges the LiPo (Lithium Polymer) battery through BMS and magnetic pins

7.1.2. Drive Train

- Climbs a maximum inclination of 30 degrees without slipping
- Easily mounts and dismounts from the cable
- Grips the cable and does not rotate while moving back and forth on the cable

7.1.3. Control System

- Receives feedback data from the sensors connected to the system
- Reliably relays the control commands from web or radio control to the other subsystems

7.1.4. Bird Detection

- Detects the birds on the cable
- Distinguishes between Cormorant and other birds
- Relays information to the control system
- Triggers failsafe system that uses ultrasonic sensors to avoid hitting the birds

7.1.5. Bird Deterrents

- Listens to the commands from the control system
- Executes the settings determined for deterring cormorants

7.1.6. Wireless Communication

- Connects with the web control user interface without any connectivity issues

7.2. Lab Tests

7.2.1. Test 1 - Drive Train and Radio Communication

7.2.1.1. Test 1 Objectives

- Test the control of the drive train using radio control
- Test the sprung wheels for any slippage

7.2.1.2. Test 1 Results

The robot was able to move back and forth on the cable. The robot speed was set at a very high value because of which it slammed into the ends of the cable while going back and forth. This slamming damaged the drive train’s driven wheels. The radio communication and calibration of electronics speed controller worked reliably throughout the test. The test video can be found in Appendix 3.

7.2.1.3. Design Changes

The slamming of the robot prompted the addition of the bumpers in the robot design. TPU 3D printing filament was used for the bumpers because of its durability and shock absorbing properties.

7.2.2. Test 2 - Drive Train and Web Control

7.2.2.1. Test Objective

- Test the drive train control through a web interface and the power supply
- Test the bumpers added to the drive train design
- Test hinge mechanism
- Test sprung wheels for slippage

7.2.2.2. Test Results

The robot was able to move back and forth with Forward and Backward buttons. The stop button also worked reliably and stopped the robot within 10 centimeters. The test video can be seen at the link in the Appendix 3. The bumper added to the robot also kept the drive train from getting damaged from any accidental slamming. The hinge mechanism allowed the convenient mounting and dismounting of the robot.

7.2.2.3. Design Changes

The web interface was quite simple but functional. The new more friendly interface was designed and implemented after this test.

7.2.3. Test 3 - Drive Train, Deterrents, Power Supply, and Autonomy

7.2.3.1. Test Objective

- Autonomous working of the robot using state machine
- Test robot’s docking to the charging station

7.2.3.2. Test Results

The robot successfully stopped 15 centimeters before the plushie and switched on the mounted deterrents. The mounted deterrents include strobe light and speaker for making high pitch noises. The strobe light was flashing, and the speaker was creating an irritating noise. This can be seen in the video at the link in Appendix 3.

7.2.3.3. Design Changes

The bird detection was only done using ultrasonic sensors. Next tests of the robot should integrate the camera in the bird detection system.

Chapter 8: Future Work

8.1. Control System

Software package for the autonomous capability of the robot needs further development. In the upcoming terms, State Machines designed should be implemented and refined. During testing, incompatibility between state machines and the sensor data streams from ROS made it impossible to test them.

8.2. Bird Detection

To prevent harming any birds, having an extremely accurate bird detection system is vital. Currently, ultrasonic sensors work as a failsafe system to keep the robot from hitting the birds. If there are any malfunctions in the ultrasonic sensors, its wiring, or software controlling it, there would be no way for the robot to avoid hitting the bird. To prevent this, different options of rangefinders with higher accuracy and industrial quality standards will be explored. In addition to that the camera's detection accuracy will be improved by gathering more image data on cormorants and training the current image classifier on it.

8.3. Bird Deterrents

Currently, the only functional deterrent is the LED Strobe light. The directional sound module is functional but is not able to produce enough volume to scare away the birds. The primary problem with the electronic circuit is that the volume of the sound is uncontrollable. The electronic circuit of the directional sound will be modified to be able to produce the volume that will scare away the birds. Use of lasers as deterrents will also be considered in upcoming terms.

8.4. Wireless Communications

Cellular connection will be added to the system and tested with the help web application. The first step towards having the cellular connection will be communicating with Eversource to decide on the financing of the cellular service.

8.5. Power Supply

Power supply has been volatile during the testing. There were instances when the power supply would shut off and the exact reasons for these stoppages were undetermined. Power supply will be made more reliable and compatible with any new electronics component that will be added to the robot.

8.6. Docking Station

Power is a limiting factor for the robot to be able to stay on the power cable for a long duration. The robot should be able to function at least two weeks on the power cable without any human interruption. Currently, with 15000 mAh battery, the robot only lasts for more than 24 hours. This problem can only be solved by charging the robot regularly.

8.7. Weather Proofing

Weather proofing has been a big concern from the very beginning of the project, but there won't be any progress on this until the robot's subsystems are fully tested and there won't be any major changes in the robot's design.

Chapter 9: Discussion

The robot developed over the period of this project has a drive train that went through six prototyping iterations, a bird detection system with ultrasonic sensors, and a bird deterrent system with a flashing strobe light and speakers. Currently, the robot can move back and forth on the cable, detect the bird using ultrasonic sensors and stop without hitting it, and switch on the deterrents once in the close vicinity of a bird. The robot can be controlled wirelessly using the web interface and it can also perform the move – detect – deter routine autonomously. This can be seen in the video that can be found at the link in Appendix 3. The bird detection system did not have the camera integrated because of the delays in gathering the bird image data that would be required for training the bird detection algorithm. Rushab Patil's major contributions were in the redesign of the system architecture, design calculations required for the drive train, testing Pixhawk to determine If it can be integrated with the robot, software architecture and state machine design. He also contributed to the design decisions of other subsystems. Rushab worked with Kohmei Kadoya to conduct testing of robot's subsystems.

The decision to prioritize the redesign of the drive train gave a great start to the project. Development of a robust drive train has provided a concrete foundation for building and testing other subsystems. The continuous revision of the system architecture to accommodate new or redesigned subsystems has helped to have a big picture throughout the project. Setting up multiple wireless communication over WPI network also helped to test the robot and work on web control without any cellular connection. The rapid prototyping of the robot's subsystems, especially drive train and power supply, helped with reevaluating and updating the design decisions.

During the integration of Pixhawk with other subsystems, there were many instances when a custom solution had to be found to integrate Pixhawk. For example, the non-GPS functioning of Pixhawk had to be researched and implemented to be able to test the robot with Pixhawk. This was a non-standard feature of the Pixhawk and lacked proper documentation. This took a long time to function properly and made it difficult to test Pixhawk with other subsystems. With the limited time available, integration of Pixhawk with other subsystem was not a good decision. As a result, Pixhawk was rejected in the integration phase of the project and Arduino

Mega 2560 was selected instead. An important lesson to be learned from this mistake was that if there is a limited time availability, it Is better to stick to already known solutions, in this case Arduino Mega, instead of finding and implementing a custom solution. Another important lesson learned was that the sponsors can take a very long time to respond to queries that are necessary to move the project forward. Therefore, plan the project subtasks in a way that delays in sponsors response does not halt the progress of the project.

The robot developed during this project has the fundamental ability of traversing cables. There are many applications of this functionality in different industries globally. This robot with sensors to detect spills or cracks in the pipes can be used in the oil industry for regular inspection and maintenance of underwater pipelines. According to the sponsor of the project, there have been reports of drones tampering with the power cable lines. With the increasing availability of cheaper drones, these number of these reports are going to increase. This robot can also be used to detect these drones and jam the local radio communications to disable these drones.

Chapter 10: Conclusion

This project focused on developing a robot that can be used to evaluate the effectiveness of bird deterrents and prove autonomous operation. The final robot developed was able to autonomously perform the move-detect-deter routine. This project used agile methodology to rapidly prototype the robot and fulfill the objectives. This project emphasized the reliable functioning of the robot since the robot was going to be used to birds without harming them. Since the robot Is working with living animals, the future designs should follow the industry standards of reliability and robustness. The current robot provides a great starting point to build an industry grade autonomous bird deterring robot.

References

[1] "Jetson Nano Developer Kit," *NVIDIA Developer*, 14-Apr-2021. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed: 29-Oct-2021].

[2] "Pixhawk overview," *Pixhawk Overview - Copter documentation*. [Online]. Available: https://ardupilot.org/copter/docs/common-pixhawk-overview.html. [Accessed: 29-Oct-2021].

[3] "Robot operating system," *ROS*, 2021. [Online]. Available: https://www.ros.org/. [Accessed: 29-Oct-2021].

[4] "About," *ArduPilot*. [Online]. Available: https://ardupilot.org/index.php/about. [Accessed: 29-Oct-2021].

[5] Radiolink Electronic Limited. (2020, March 19). *F450 User Manual*. Radiolink Electronic Limited. Retrieved October 30, 2021, from https://radiolink.com/f450_manual.

[6] J. Hatch, "Find A bird," *Mass Audubon*. [Online]. Available: https://www.massaudubon.org/our-conservation-work/wildlife-research-conservation/bird-conservation-monitoring/breeding-bird-atlases/bba1/find-a-bird/(id)/30#:~:text=The%20Double%2Dcrested%20Cormorant%20is,greatly%2C%20especially%20since%20about%201970. [Accessed: 30-Oct-2021].

[7] "Double-crested cormorant life history, all about birds, Cornell Lab of Ornithology," *All About Birds, Cornell Lab of Ornithology*. [Online]. Available: https://www.allaboutbirds.org/guide/Double-crested_Cormorant/lifehistory. [Accessed: 30-Oct-2021].

[8] E. Maroney, "With power lines relocated, Will cedar pond be free of cormorants?", *Cape Cod Chronicle*, 06-Mar-2019. [Online]. Available: https://capecodchronicle.com/en/5410/orleans/4161/With-Power-Lines-Relocated-Will-Cedar-Pond-Be-Free-Of-Cormorants-Waterways.htm. [Accessed: 30-Oct-2021].

[9] Raspberry Pi, "Buy A raspberry pi camera module 2," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/camera-module-v2/. [Accessed: 26-Jan-2022].

[10] "Double-crested cormorant identification, all about birds, Cornell Lab of Ornithology," *All About Birds, Cornell Lab of Ornithology*. [Online]. Available: https://www.allaboutbirds.org/guide/Double-crested_Cormorant/id. [Accessed: 26-Jan-2022].

[11] Rentokil Pest Control Ireland, "Problems caused by nuisance birds on commercial properties," *deBugged - The Pest Control Blog*, 20-Jun-2019. [Online]. Available: https://www.rentokil.ie/blog/tag/bird-control/. [Accessed: 26-Jan-2022].

[12] A. Jerrett, "Types of birds that form large flocks together," *Sciencing*, 22-Nov-2019. [Online]. Available: https://sciencing.com/types-form-large-flocks-together-6790830.html. [Accessed: 26-Jan-2022].

[13] D. M. G. H. Audubon, "Grays Harbor Birds: Double-crested cormorant (Phalacrocorax auritus)," *The Daily World*, 26-Jan-2019. [Online]. Available: https://www.thedailyworld.com/sports/grays-harbor-birds-double-crested-cormorant-phalacrocorax-auritus/. [Accessed: 26-Jan-2022].

[14] Eichner, E., B. Howes, and D. Schlezinger, "Cedar Pond Water Quality Management Plan," Coastal Systems Program, School for Marine Science and Technology, University of Massachusetts Dartmouth. New Bedford, MA, Tech. Report. 2013.

[15] J. F. Glahn, G. Ellis, P. Fiornelli, and B. Dorr, "Evaluation of moderate- and low-power lasers for dispersing double-crested cormorants from their night roosts," In Proc. Eastern Wildlife Damage Management Conference, 2001, pp. 9:34- 45.

[16] "Arduino Mega 2560 REV3," *Arduino Official Store*. [Online]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3. [Accessed: 26-Jan-2022].

[17] K. Moore and D. Gupta, "Finite State Machines," *Brilliant Math & Science Wiki*. [Online]. Available: https://brilliant.org/wiki/finite-state-machines/. [Accessed: 26-Jan-2022].

[18] "Mipi Camera Serial interface 2 (MIPI CSI-2)," *MIPI*, 17-Dec-2021. [Online]. Available: https://www.mipi.org/specifications/csi-2. [Accessed: 26-Jan-2022].

[19] "MB1240 XL-maxsonar-EZ4," *Ultrasonic Sensors That Work*, 04-Mar-2021. [Online]. Available: https://www.maxbotix.com/ultrasonic_sensors/mb1240.htm. [Accessed: 26-Jan-2022].

[20] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and efficient object detection,” *arXiv.org*, 27-Jul-2020. [Online]. Available: https://arxiv.org/abs/1911.09070v7. [Accessed: 26-Jan-2022].

[21] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft Coco: Common Objects in Context,” *arXiv.org*, 21-Feb-2015. [Online]. Available: https://arxiv.org/abs/1405.0312. [Accessed: 26-Jan-2022].

[22] “CCUB NABirds 700 dataset competition,” *CCUB NABirds 700 Dataset Competition*. [Online]. Available: https://dl.allaboutbirds.org/nabirds. [Accessed: 26-Jan-2022].

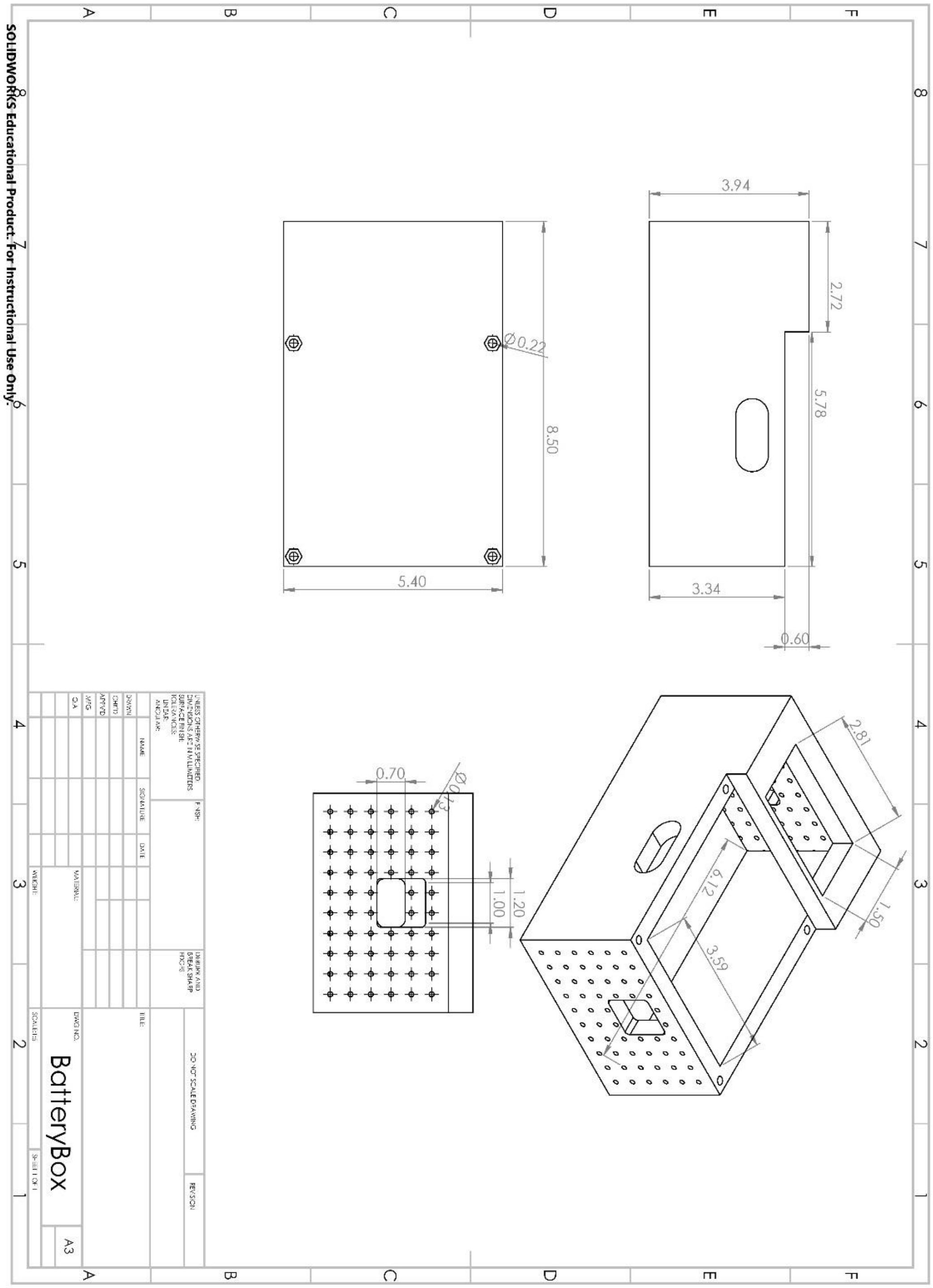
[23] “4G/LTE cellular modem kit for Nvidia Jetson Nano Dev Kit (pre-order),” *Sixfab*, 11-Jan-2022. [Online]. Available: https://sixfab.com/product/4g-lte-cellular-modem-for-nvidia-jetson-nano-developer-kit/. [Accessed: 26-Jan-2022].

[24] “R9DS Receiver,” *9/10 channels receiver R9DS*. [Online]. Available: https://www.radiolink.com/r9ds. [Accessed: 26-Jan-2022].

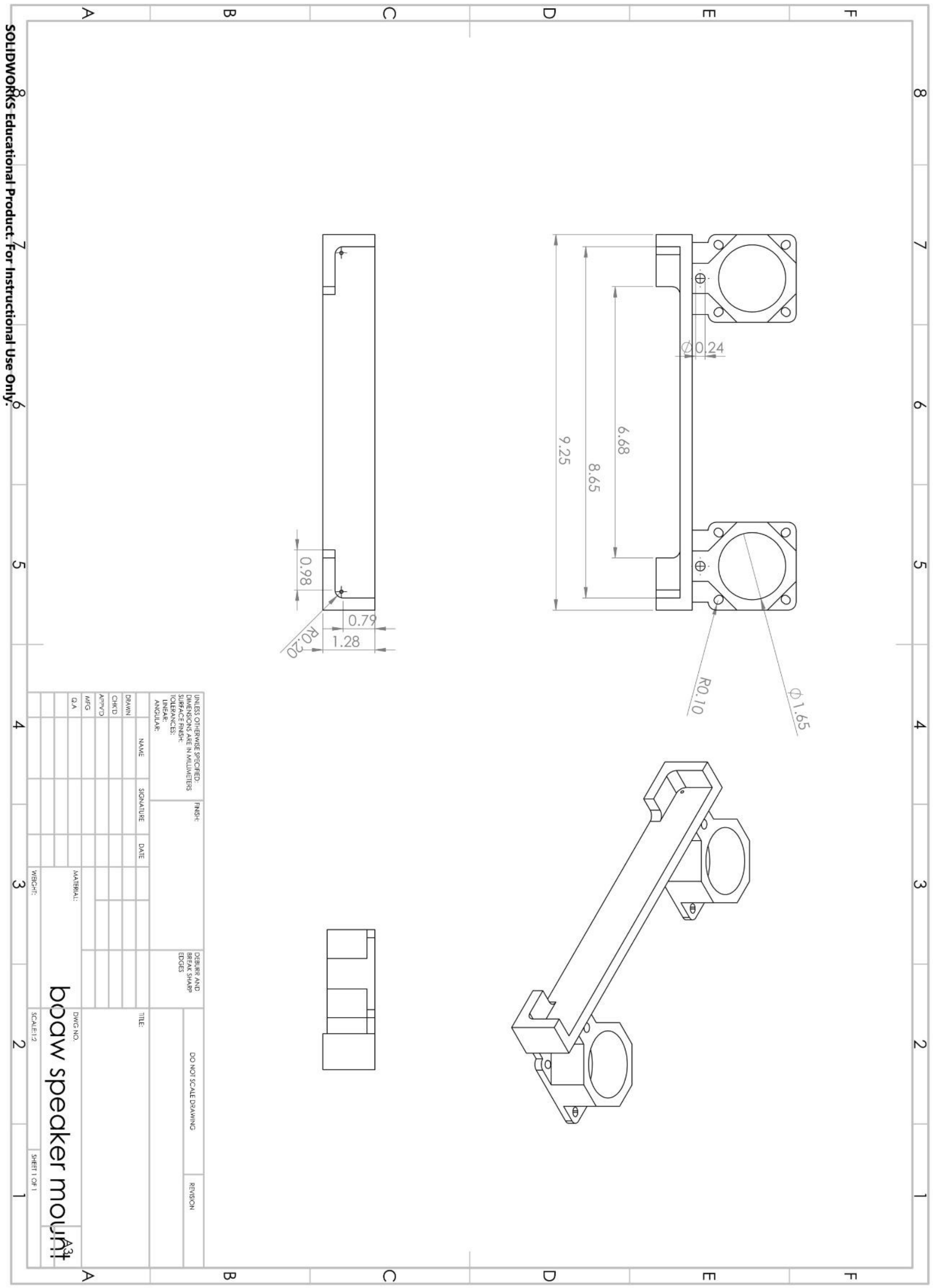
[25] “Strobe Lights,” *Bird Control With Strobe Lights - PCRC*. [Online]. Available: https://www.pigeoncontrolresourcecentre.org/html/reviews/strobe-lights-bird-control.html. [Accessed: 26-Jan-2022].

Appendices

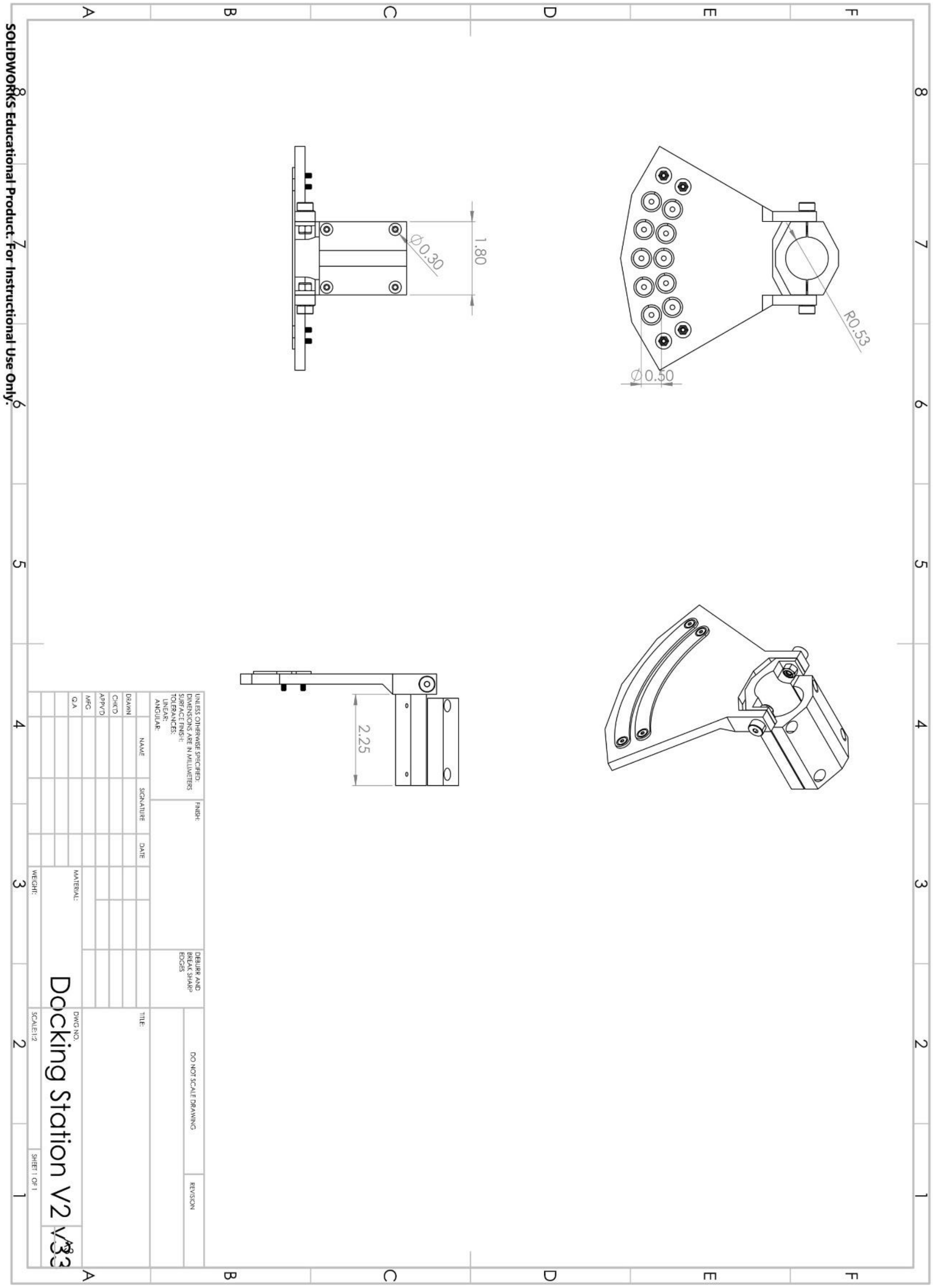
Appendix 1.1. Battery Box



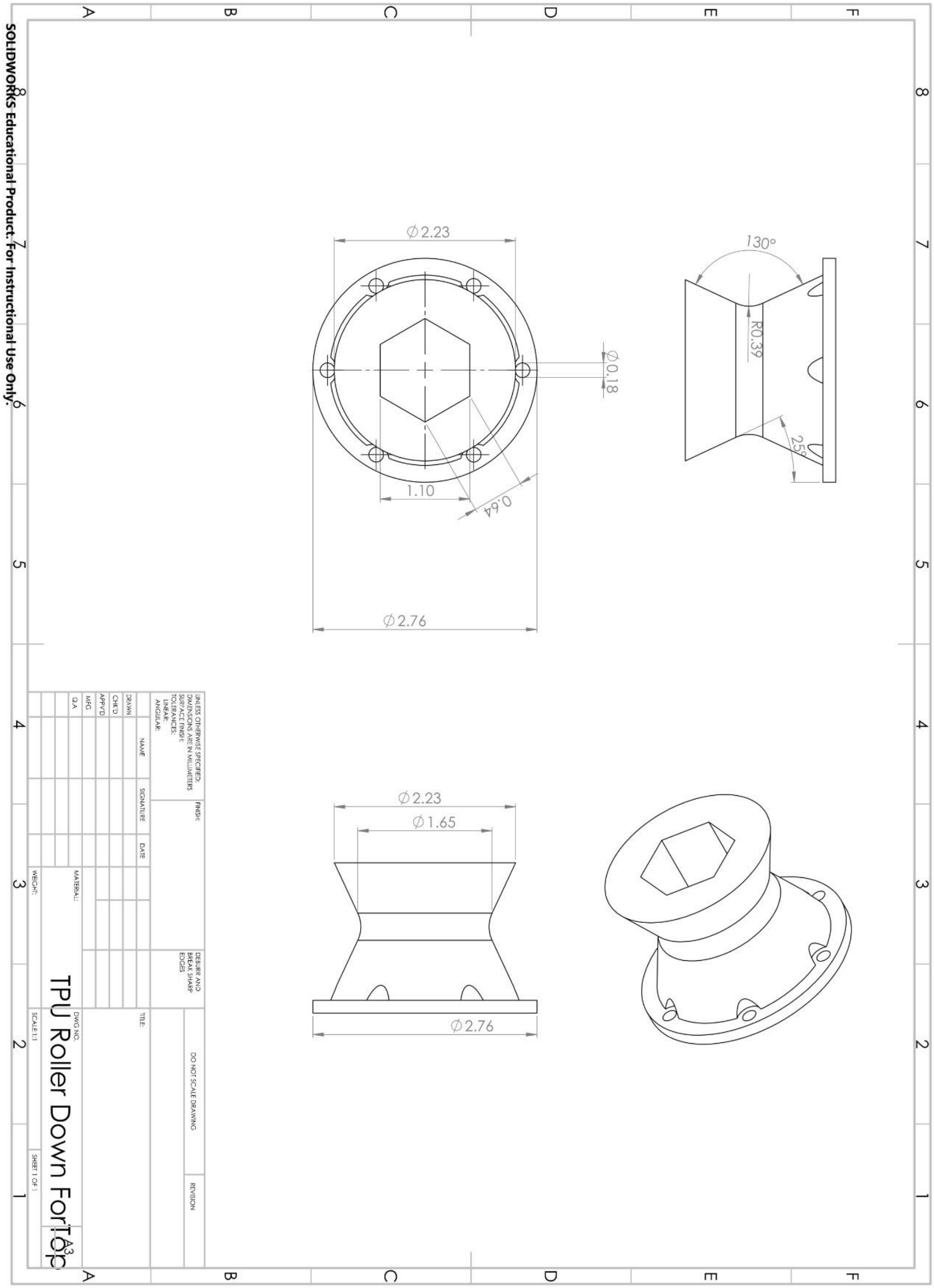
Appendix 1.2. Speaker Mount



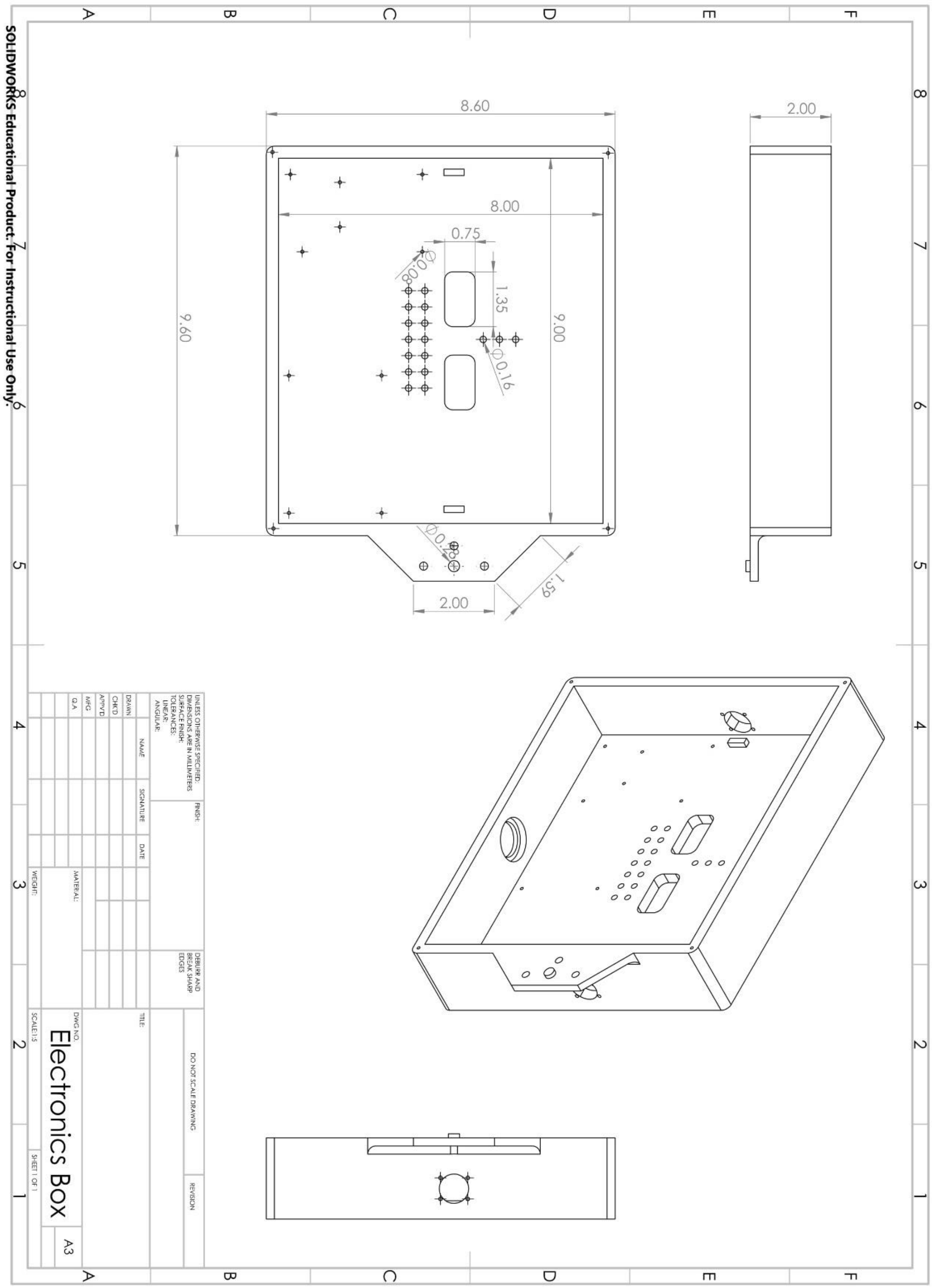
Appendix 1.3. Docking Station



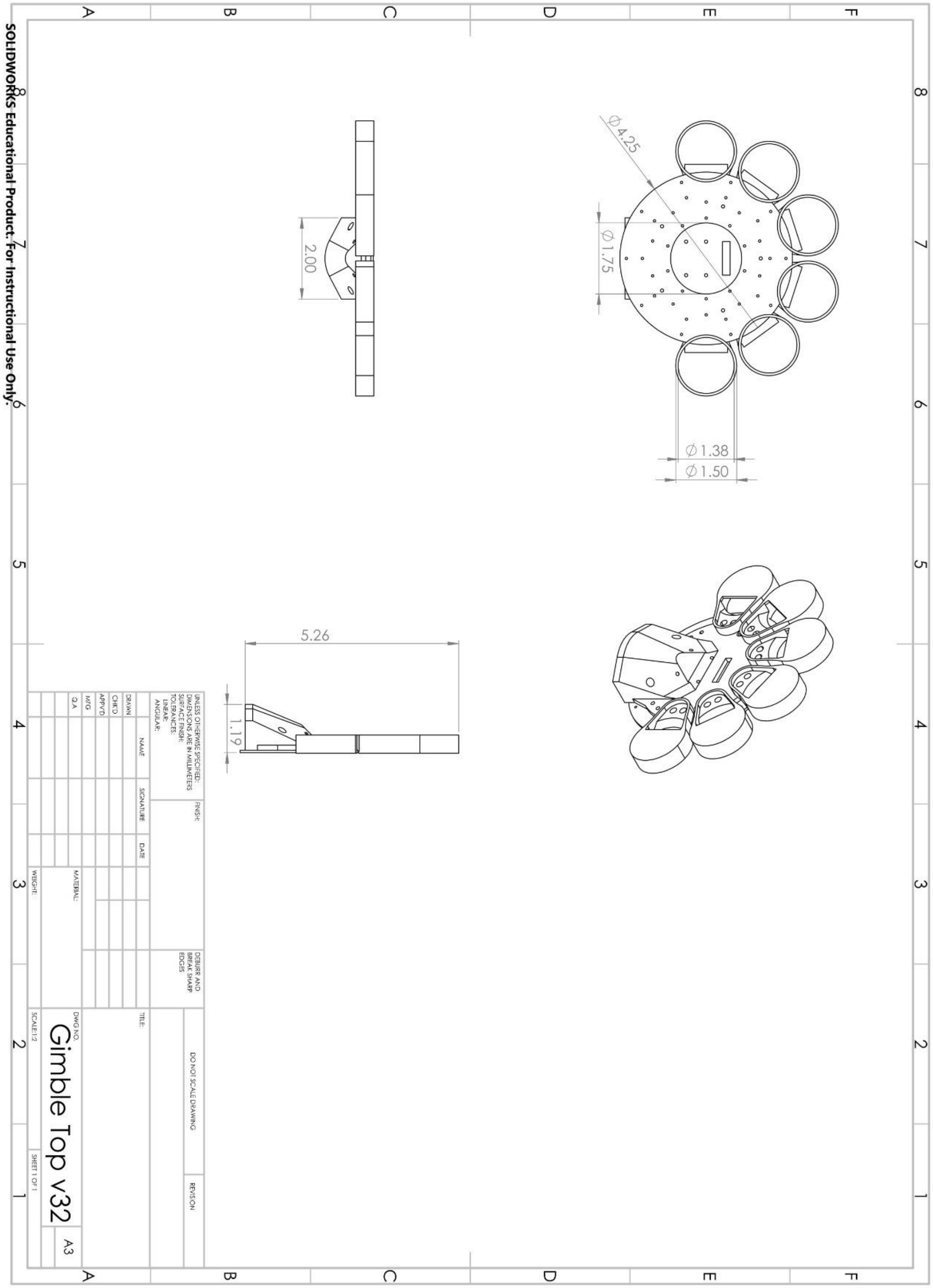
Appendix 1.4. TPU Roller



Appendix 1.5. Electronics Box



Appendix 1.6. Gimbal Top



Appendix 2.1. State Machine Code – boaw_sm.py

```
#!/usr/bin/env python
#Author: Rushab Patil
import roslib; roslib.load_manifest('smach_tutorials')
import rospy
import smach
import smach_ros

#Global Variables
rfReadingGlobal = 0
switchGlobal = 'OFF'

# define state Static
class Static(smach.State):
    def __init__(self):
        smach.State.__init__(self, outcomes=['ON', 'OFF'])
        self.switch = switchGlobal

    def execute(self, userdata):
        rospy.loginfo('Executing state STATIC')
        if self.switch == 'ON':
            return 'ON'   #switch to Move State
        else:
            robot_speed = 0
            robotSpeedPub.publish(robot_speed)
            return 'OFF'




# define state Move
class Move(smach.State):
    def __init__(self):
        smach.State.__init__(self, outcomes=[True, False])
        self.rfReading = rfReadingGlobal

    def execute(self, userdata):
        rospy.loginfo('Executing state MOVE')
        if self.rfReading <= 0.5    #0.5 meters
            robot_speed = 0        #stop the robot
            robotSpeedPub.publish(robot_speed)
            return True     #switch to Approach State
        else:
            robot_speed = 1        #move the robot forward
            robotSpeedPub.publish(robot_speed)
            return False




# define state Approach
class Approach(smach.State):
    def __init__(self):
        smach.State.__init__(self, outcomes=[True, False])
        self.rfReading = rfReadingGlobal

    def execute(self, userdata):
        rospy.loginfo('Executing state APPROACH')
        if self.rfReading <= 0.2    #0.2 meters
            return True
        else if ((self.rfReading > 0.2) and (self.rfReading < 0.5)):
            #velocity curve
            robot_speed = 0.2         #replace with actual values
            robotSpeedPub.publish(robot_speed)
            return False
```

```
# define state Deterrents_On
class Deterrents_On(smach.State):
   def __init__(self):
      smach.State.__init__(self, outcomes=[True, False])
      self.rfReading = rfReadingGlobal


   def execute(self, userdata):
      rospy.loginfo('Executing state DETERRENTS_ON')
      if self.rfReading <= 0.2:    #0.2 meters
         #turn on the the deterrents
         #flash the LEDS for 5 seconds
         flashLightStatus = True     #Turn on the leds
         flashLightPub(flashLightStatus)
         #create the sound for 5 seconds
         soundStatus = True         #Turn on the speaker
         soundPub(soundStatus)
         return False
      else if self.rfReading > 0.5:
         return True



# define state Deterrents_Off
```

```
class Deterrents_Off(smach.State):
    def __init__(self):
        smach.State.__init__(self, outcomes=[True, False])
        self.rfReading = rfReadingGlobal

    def execute(self, userdata):
        rospy.loginfo('Executing state DETERRENTS_ON')
        #turn on the the deterrents
        #flash the LEDS for 5 seconds
        #create the sound for 5 seconds
        flashLightStatus = False      #turn off the leds
        flashLightPub(flashLightStatus)
        #create the sound for 5 seconds
        soundStatus = False          #turn of the speakers
        globals()['switchGlobal'] = 'OFF'
        soundPub(soundStatus)
        return True


def RfCallback(data):
    # assign rangefinder reading
    globals()['rfReadingGlobal'] = data
```

```
def SwitchCallback(data):

    # assign rangefinder reading

    globals()['switchGlobal'] = data




def main():

    rospy.init_node('BOAW_SM')


    # Subscribers

    rospy.Subscriber("/rangefinder/front", Float32 , RfCallback)

    rospy.Subscriber("/switch", String , SwitchCallback)


    # Publishers

    robotSpeedPub = rospy.Publisher('/motor_speed', Float32, queue_size=10)

    flashLightPub = rospy.Publisher('/deterrents/led', Bool, queue_size=10)

    soundPub = rospy.Publisher('/deterrents/speaker', Bool, queue_size=10)


    # Create a SMACH state machine

    sm = smach.StateMachine(outcomes=[True, False])


    # Open the container
    with sm:

        # Add states to the container

        smach.StateMachine.add('STATIC', Static(),

                               transitions={'ON':'MOVE', 'OFF':'STATIC'})

        smach.StateMachine.add('MOVE', Move(),

                               transitions={True :'APPROACH', False:'MOVE'})

        smach.StateMachine.add('APPROACH', Approach(),

                               transitions={True :'DETERRENTS_ON', False:'APPROACH'})

        smach.StateMachine.add('DETERRENTS_ON', Deterrents_On(),

                               transitions={True :'DETERRENTS_OFF', False: 'DETERRENTS_ON'})

        smach.StateMachine.add('DETERRENTS_OFF', Deterrents_Off(),

                               transitions={True :'STATIC', False:'STATIC'})

    # Execute SMACH plan

    outcome = sm.execute()




if __name__ == '__main__':

    main()
```

Appendix 2.2. Arduino Code – main.cpp

```
#Author : Kohmei Kadoya

#include <Arduino.h>

#include <ros.h>

#include <std_msgs/Int32.h>

#include <std_msgs/Float32.h>

#include <std_msgs/Bool.h>

#include <ESC.h>

#include <Encoder.h>



#include "constants.h"



#define USE_USBCON



//For MaxSonar

#define USPin1 A0  //front

#define USPin2 A3    //back



ESC esc1(ESC1_PIN, MOTOR_FULLBACK, MOTOR_FULLFORWARD, MOTOR_STOP);

ESC esc2(ESC2_PIN, MOTOR_FULLBACK, MOTOR_FULLFORWARD, MOTOR_STOP);



Encoder encoder(ENCODER_PIN1, ENCODER_PIN2);
```

```
ros::NodeHandle nh;
std_msgs::Float32 enc_val, rf_front_val, rf_back_val;
ros::Publisher pub_enc("/encoder", &enc_val);
ros::Publisher pub_rf_front("/rangefinder/front", &rf_front_val);
ros::Publisher pub_rf_back("/rangefinder/back", &rf_back_val);

void cb_led(const std_msgs::Bool &msg) {
  int state = msg.data ? HIGH : LOW;

  digitalWrite(LED_PIN, state);
};

void cb_speaker(const std_msgs::Bool &msg) {

}

void cb_motor(const std_msgs::Float32 &msg) {
  int speed = map(msg.data, -1, 1, MOTOR_FULLBACK, MOTOR_FULLFORWARD);

  esc1.speed(speed);
  esc2.speed(speed);
}


ros::Subscriber<std_msgs::Bool> led_sub("/deterrents/led", &cb_led);

ros::Subscriber<std_msgs::Bool> speaker_sub("/deterrents/speaker", &cb_speaker);

ros::Subscriber<std_msgs::Float32> motor_sub("/motor_speed", &cb_motor);


void setup() {

  pinMode(LED_PIN, OUTPUT);


  esc1.arm();

  esc2.arm();


  delay(500);


  esc1.speed(MOTOR_STOP);

  esc2.speed(MOTOR_STOP);


  nh.initNode();

  nh.advertise(pub_enc);

  nh.advertise(pub_rf_back);

  nh.advertise(pub_rf_front);

}
```

```
void loop() {

  // Publish Encoder

  enc_val.data = encoder.read();

  pub_enc.publish(&enc_val);



  // Publish Rangefinders

  rf_front_val.data = (analogRead(USPin1) / 1024.0) * 512 * 2.54;

  pub_rf_front.publish(&rf_front_val);

  rf_back_val.data = (analogRead(USPin2) / 1024.0) * 512 * 2.54;

  pub_rf_back.publish(&rf_back_val);



  nh.spinOnce();

}
```

Appendix 3. Lab Test Video Links

Test 1. https://youtu.be/C4wvQekqup4

Test 2. https://youtu.be/t6RZ0PL-Zn4

Test 3. https://youtu.be/KnT2CKEwFAQ

Appendix 4. Turning Sound into a Laser

Youtube Video Link: https://www.youtube.com/watch?v=aBdVfUnS-pM