ALiCS

Autonomous Litter Collection & Sorting

Group 6 Authors:

Andre Reveles, Computer Engineering
Luis Hernandez, Computer Engineering
Mario Puesan, Photonics Science Engineering
Mary Bartlinski, Computer Engineering
SeanWaddell, Electrical Engineering

Mentor:

Dr. Chung Yong Chan
Dr. Aravinda Kar

Committee:

Dr. Mustafa 'Mert' Bayer
Dr. John Aedo
Professor Mark Maddox

Table of Contents

1. Executive Summary
2. Project Description
    - 2.1. Project Background and Motivation
    - 2.2. Current Commercial Technologies
        - BeBot
        - iRobot Roomba
        - Beach Metal Detectors
        - Summary
    - 2.3. Goals and Objectives
        - Basic Goals
        - Advanced Goals
        - Objectives
        - Hardware
            - Rover–Objective
            - Robot Arm–Objective
            - Spectrometer–Objective
            - Light Source–Objective
            - Optical Fiber–Objective
            - Power Supply–Objective
            - PCB–Objective
            - Charging Station–Stretch Goal
        - Software
            - Rover–Objective
            - Robot Arm–Objective
            - Trash Detection–Objective
            - Unstuck Feature–Stretch Goal
    - 2.4. Product Specifications
3. Research and Part Selection
    - 3.1. Technology Research and Comparison
    - 3.2. Hardware Research
        - Navigation and Mapping
            - LiDAR
            - Camera-Vision
            - LiDAR vs Camera Comparison
        - Rover
        - Wheels
            - Choice for Ballon Wheels
            - Wheeleez Polyurethane Wheels
            - VEVOR Wheels
            - Extreme Max Wheels
            - Wheel Comparison
        - Motor
            - Brushed DC Motors and Brushless DC Motors
            - Stepper Motors
            - Motor Selection

    - 57BLF03 Brushless DC Motor
    - RM-ESMO-01F Brushless DC Motor
    - TEC 4260 Brushless DC Motor
    - Motor Comparison
- Sensor
    - Ultrasonic Sensor
    - Active and Passive Infrared Sensor
    - Choices for Ultrasonic Sensors
    - Ultrasonic HC-SR04 Sensor
    - MB1232 I2CXL Sensor
    - MA40S4S Sensor
    - Sensor Selection
- Power Supply
    - Battery
    - Lead-Acid Battery
    - Nickel Cadmium Batteries
    - Nickel MEtal Hydride Batteries
    - Lithium-Based Batteries
- Power Distribution Board (PDB)
    - Empty Board
    - Premade Board
    - PDB Selection
- Voltage Regulators
    - Linear Voltage Regulator
    - Switching Voltage Regulator
    - Voltage Regulator Selection
- PCB
    - MCU vs FPGA
    - Microcontroller Comparison
    - MSP-EXP430FR6989
    - ESP32 WROOM
    - ARDUINO MKR WAN 1310
    - Raspberry Pi 4 Model B
    - Raspberry Pi Zero 2 W
    - Rock Pi 4 SE
- Signal Filters
- Robotic Arm
    - Robot Arm Attachment
    - Robot Arm Dustpan
    - Robot Arm Comparison
        - Meca500
        - MechArm
        - UFactory xArm5
- Robot Arm Claw
    - Robot Arm Claw Comparison
    - Swaytail Professional Metal Robot Gripper

        - LewanSoul Mechanical Robot Claw
        - RobotIQ 2F-85
    - Spectrometer
        - Czerny-Turner Spectrometer
        - Echelle Spectrometer
        - Czerny-Turner vs Echelle Spectrometer
    - Near Infrared Spectroscopy
    - LiDAR Comparison
        - Slamtex RPLiDAR
        - Yahboom EAI X3
        - Inno Maker LD06
    - Camera Comparison
3.3. Software Research
    - Framework
        - Yarp
        - ZeroMQ
        - Robot Operating System (ROS)
    - Mapping and Object Detection
        - SLAM
        - Structure from Motion (SfM)
        - QGIS 3
        - YOLO
        - OpenCV
        - TensorFlow Object Detection API
    - Zemax
    - Meca500’s Web Interface: MecaPortal
    - Machine Learning
    - Pathing Algorithm
    - Global Pathing
        - Dijkstra’s Algorithm
        - Floyd-Warshall Algorithm
        - A* Algorithm
    - Local Pathing
        - Dynamic Window Approach
        - Time-Elastic Band
        - Active Scene Recognition

4. Standards and Constraints

4.1. Standards
    - ANSI
    - Robotics and Automation Standards
    - ANSI/UL 4600
    - Standard for Robotics and Automation: IEEE 1872-2015
    - Optical Standards–US Standard MIL-PRF-13830B
    - Laser Product Standard (LiDAR)
    - Communication Standard
    - Portable and Rechargeable Battery Standards

- ANSI for Portable Rechargeable Batteries
- Soldering Standards
- PCB Design Standards

4.2. Design Constraints
- Time Constraint
- Economic Constraint
- Ethical Constraint
- Environmental Constraint

5. Comparison of ChatGPT and Other LLMs

5.1. Introduction

5.2. Pros of AI Language Models for ALiCS
- Design and Development Support
  - Study Case 1
  - Study Case 2
  - Study Case 3
- Programming Assistance
  - Study Case 4

5.3. Cons of AI Language Models for ALiCS
- Accuracy
- Ethical Concerns

6. Hardware Design

6.1. Motor Controller

6.2. Ultrasonic Sensor

6.3. LiDAR

6.4. Camera

6.5. Robot Arm

6.6. Spectrometer–Optical Design
- Optical Design Parameters (Zemax)

6.7. Power Supply

7. Software Design

7.1. Software Architecture
- Software Use Case Diagram
- Software Data Flow
- Software State Diagram
- Class Hierarchy and Design
- Data Structure for ALiCS
  - Sensor Data
  - Integrated Data
  - Control Data
  - Data Storage
  - System State
  - User Stat
  - Main Control System
- User Interface–Spectrometer Software
- Raspberry Pi Software
- ESP Software

7.2. Machine Learning Implementation
- Roslaunch
- Rostest
- RViz
- ROSbag
- ROS Global and Local Pathing

7.3. Robot Arm Software Design
- Meca500 Command Reference Table

7.4. Algorithm Design
- Localization
- Rover
- Lidar
- Camera

8. System Fabrication/Prototype Construction

8.1. Fabrication Materials
8.2. Bottom Level
8.3. Middle Level
8.4. Top Level
8.5. PCB Layout

9. System Testing & Evaluation

9.1. Hardware Testing
- ESP32
- Meca500 Robot Arm
- Raspberry Pi 4
- Lidar Puck
- Spectrometer
- Rover
- Raspberry Pi Camera

9.2. Software Testing
- SLAM
- Camera Interface YOLO
- Meca500 Robot Arm
- Raspberry Pi 4
- Communication Software (ROS)
- Spectrometer Software

9.3. Performance Evaluation
9.4. Overall Integration
- Hardware Integration
- Software Integration

9.5. Plan for SD2

10. Administrative Content

10.1. Budget
10.2. Distribution of Worktable
10.3. Milestones

11. Conclusion
12. Appendix

12.1. A–References
12.2. B–Copyright
12.3. C–ChatGPT Case Studies
12.4. D–Data Sheet

Chapter 1: Executive Summary

With the advancement of technology, life for humans in general has continued to improve and advance by a huge margin. The technological devices that we use daily, weekly, or even monthly has helped make our lives easier than ever before, but what if, instead of using the technology to our benefit, we were to use it to help better the environment? There have been a multitude of creations that helped better the environment such as solar panels and wind farms that create a beneficial amount of renewable energy that helps reduce emissions. Another creation is electric cars that help reduce the use of gasoline that is harmful to our environment. Although these creations aim to help the environment, they also in some kind of way, benefit humans as well. That's why we wanted to create a design that 's sole purpose is to help its environment and that only. After some discussion, it was decided to create a device that helps beach areas.

The development of the Autonomous Litter Collection and Sorting (ALiCS) system marks a significant step toward maintaining clean and safe beach environments. ALiCS is designed to autonomously navigate beach areas, effectively identifying and removing debris that could potentially harm local wildlife and plant life. Utilizing advanced machine learning technologies, ALiCS continuously enhances its capabilities in real-time mapping and navigation. This ongoing improvement in its operational algorithms enables ALiCS to adeptly avoid obstacles, ensuring a thorough yet safe cleaning process. By deploying ALiCS, our aim is not only to preserve the natural beauty and safety of beach environments but also to protect the diverse species that call these areas home.

ALiCS has wheels that are capable of driving on sand, which is a loose and unstable surface that can constantly be altered at any given time. For that reason, the wheels are capable of decent off-roading, stability, as well as durability to keep our design moving around and not tipping over. In order to use these kinds of wheels, we need a motor with enough torque and power to keep those wheels constantly moving without expending all of our power. Speaking of power, ALiCS is running on rechargeable batteries, in order to keep ALiCS from needing as much human intervention. The batteries should also have a long life span so ALiCS can operate as long as possible. ALiCS is also equipped with a camera that can detect any kind of trash that is laying around at the beach. For this design, we currently aim to detect only aluminum cans, plastic bottles, and glass bottles, since we believe that those three things tend to be trash you would see the most at the beach. Once an object that fits the description of what we are aiming for, ALiCS will make its way towards the object and pick it up. The trash is picked up by a robot arm that is attached near the front of the body, and once the object is picked up, the object is scanned by the spectrometer in order to figure out what material the object is made of. The spectrometer is capable of using and measuring light in order to identify and or analyze materials. There are some long term plans for us to implement for ALiCS if we are able to implement them. One of them is for ALiCS to be able to pick itself up if it were to tip over. For that we would need a device that is strong enough to push the floor and lift ALiCS enough for it to land back on its four wheels. Another one is to create a self-docking station, so that when ALiCS is about to run out of power, it will head to the self-docking station and turn itself on. We may also be able to set it up so it can charge itself when connected to the station. With these things, we can create a strong prototype that one day we can improve our ecosystem and improve ALiCS.

Chapter 2: Project Description

2.1 Project Background and Motivation

In today's society, it has become more common for people to leave trash in places it does not belong, places such as parks, rivers, and more importantly beaches. Our group has decided to focus and provide a solution to combat marine debris. Marine debris; man-made objects that enter the marine environment through careless handling or disposal, intentional or unintentional release, or as a result of natural disasters and storms - is one of the ocean's most pervasive, yet potentially solvable, pollution problems. The presence of marine debris, coupled with its physical, ecological and socio-economic complexities, poses a severe threat to the sustainability of the world's natural resources.

Beachgoers find it convenient to leave trash behind, and they are unaware of the detrimental effects that these actions can cause to the environment, specifically sea-life. Litter and debris along the beach, including on sea turtle nesting beaches, soon makes its way to the sea where turtles and other marine life may consume or be trapped by these items. Clean, debris-free beaches will be beneficial to not only tourists and visitors but most importantly sea turtles and other marine life.

What if there was a way to reduce the amount of trash on these beaches? That's the question our group has decided to dedicate our senior design project to. We as a group have come together to solve one of the greatest problems of our generation by designing and developing a promising project. This project features a handful of subsystems containing both software and hardware that coincide with each other. The decision to choose this project is not only a demanding project that has pushed us to the limits of our capabilities, but also makes a positive impact on the environment and protects the 8,436 miles of shore line here in the State of Florida.

The motivation for this project is to take this challenge and use it to build something that can help benefit the environment. Using our design complete with convolutive hardware and software and using it to help marine wildlife is extremely beneficial. We hope that with our design, we are able to inspire companies, communities, or others to seek action on restoring and bettering the environment for the sake of those who are not able to, such as animals and plants.

In addition to protecting the environment with the creation of our project, our main motivation is to successfully complete the requirements of the senior design program. We wish to take this opportunity to show why we have made it this far and why we should be able to call ourselves engineers. We as a group have used this project as motivation and a stepping stone to help us become better at communicating, collaborating, and planning as a team. These skills will become extremely important once we graduate and are making our way into the industry.

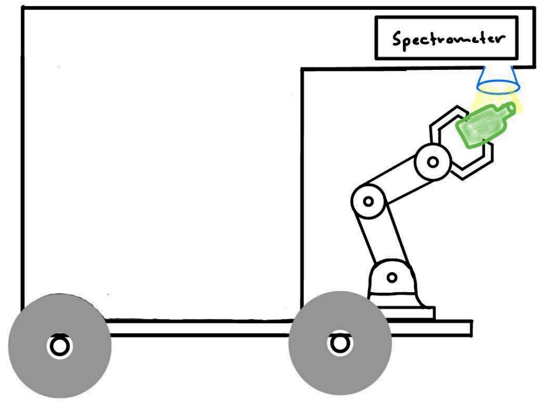


Figure 2.1.1: *Hand-sketch preliminary design*

2.2 Current Commercial Technologies and Existing Projects

BeBot

BeBot is a beach cleaning robot, essentially very similar to the project that we plan to create. BeBot utilizes a rake to pick up any waste from beaches, nature reserves, golf courses, etc. It contains a sifter that separates the trash from the sand. BeBot uses batteries and solar panels, which makes it 100% electric and does not use or produce any damaging gasses. It maneuvers using track-style propulsion and it is operated by a remote control that has a range of 150 meters. It contains a docking system to place the BeBot on before and after using it.

The BeBot should be stored in a temperature ranging from 41 to 86 degrees Fahrenheit. BeBot lasts up to 3 hours, and it takes up to 8 hours for it to charge. It has a maximum speed of 2.7 kilometers per hour and is able to carry weights up to 400 kilograms (about 881.85 pounds). The tray collection of BeBot has a capacity of 100 liters.

The BeBot has some similarities when compared to our design, ALiCS. The idea and goals of ALiCS are the same, in which we aim to clean the beach environment using an autonomous robot. However, the difference is that while BeBot utilizes a rake style system that scoops up trash, we are using a robotic arm that detects and picks up trash on the beach. This will help keep sea-turtle nesting areas safe from being accidentally scooped up. Also, they use remote control with a range limit while ALiCS is programmed to be autonomous and does not have a remote controlled limit or operator engagement. This allows an operator to monitor and manage multiple systems rather than a single unit. Lastly, ALiCS is equipped with a spectrometer to sort recyclables from trash and also leave seashells on the beach where they belong. ALiCS is 100% electric and has swappable battery capabilities.

iRobot Roomba

The iRobot Roomba has been a commercialized smart-home product that has had much success in the last decade. In September 2002, the company introduced the first Roomba model. Less than a year later, iRobot was selling a million units a year. Although early models were known for zigzagging randomly to achieve full room coverage, the semi-autonomous cleaning robot still had much success. In the last decade, the engineers behind the Roomba have fully transformed the mechanical maid to sweep, vacuum, and mop different types of flooring in homes and offices. iRobot's latest models feature a new visual simultaneous localization and mapping (vSLAM) navigation system, allowing it to clean more efficiently. Some of the other features Roomba include are cliff detect sensor, obstacle detection bumper, floor tracking sensor, acoustic and optical sensors (for detecting areas with excessive dirt and debris).

Our plan is to emulate this technology and apply it to a beach or waterfront environment. ALiCS will incorporate the same key technologies that are embedded in the Roomba design. While Roomba's are limited to indoor use only, ALiCS is an industrial grade robot that can withstand extreme everyday environmental conditions. Although our design is currently about 10x the size of a Roomba, it has the ability to pick up items ranging from cigarette butts to 2-Liter Soda bottles.

Beach Metal Detectors

Some hobbyists take to the beach to search for hidden treasures buried in the sand, and as luck would have it, prove to be pretty successful in their searching. Metal detectors are commonly used to locate belongings left behind such as jewelry, old-coins, and relics. The two common techniques used today are Very Low Frequency and Pulse Induction, both techniques similarly use coils to transmit and receive electromagnetic fields. Depending on the type of metal below the sand line, the electromagnetic field responds differently, which will trigger the indicator and the operator will then dig to find their reward. While this helps keep metals out of the sea, this is more of a hobby and the drive behind this is for personal gain. ALiCS is a conscious effort to help the environment with almost no tangible reward. As a potential stretch goal, this technology could be incorporated later on and our system can begin to collect metals just below the surface.

Summary

The ALiCS (Autonomous Litter Collection Sorting) aims to address the issue with litter on beaches through the design of an autonomous robot and its purpose to collect debris and trash. Moving right into the core components of ALiCS, the robot rover is equipped with sensors, a robotic arm, a power supply system, spectrometer, and a PCB design that will act as a control center. This rover is designed to navigate on the beach in different environments while avoiding obstacles. The robotic arm is used to collect the debris that the camera vision detected and is placed in an onboard recycle or trash pan.

ALiCS is outfitted with a VIS-NIR Spectrometer used to analyze items recovered from the ground and determine whether the item is recyclable or not. The power system of ALiCS is built on sustainability and will include an efficient battery that can supply power to the entire system. The enclosure will protect the system from the earth's elements and allow cooling to prevent any failures.

The motivation of ALiCS is mainly inspiring other communities and organizations to provide innovative results for a cleaner environment. This project explores current ideas and technologies we learn from and relate our project to. Learning about these existing solutions helps with understanding requirements and needs for the project as well as gain inspiration. We can use these ideas to creatively construct ALiCS.

2.3 Goals and Objectives

The function of ALiCS is for it to be able to successfully navigate its way through the sand while avoiding the water and large obstacles. It should also be able to successfully navigate over gravel as well since gravel could be found on some beaches. While ALiCS is making its way through the sand, it is scanning the floor in front of it using a camera to look for potential trash. When something has been found that is not sand, ALiCS will make its way towards the object so that it is close enough for the robotic arm to pick it up.

Basic Goals:

- Collection - Collect at least 8 aluminum cans, 5 plastic bottles, or 5 glass bottles at the beach and place them into three different bins based on the material or a mix match of different types of waste.
- Navigation - effectively navigate and localize the area of the beach tasked with cleaning.
- Obstacle Avoidance - avoid large obstacles and bodies of water that may damage the rover.
- Identify - accurately identify recyclables, trash, and seashells in real time.
- Ease - deliver a system that will take minimal training and operate independently.
- Power Efficiency - the system will be configured with a “stand-by” to preserve power while instruments are not being used, i.e, light source, spectrometer, and robot arm

Advanced Goals:

- Implement machine learning for the navigation of ALiCS
- Be able to distinguish aluminum cans, plastic bottles, and glass bottles from all other trash noticed at the beach
- Able to determine the material of the trash and sort them by placing them into different bins
- Robotic arm should be able to pick up trash in less than 5 seconds

- Identify and avoid sea turtle crawlings, nesting, or hatchlings.
- Implement metal detection technology to recover metals below the sand line.
- Design swappable battery system, to change batteries on the go.
- Onsite docking station, with self-emptying capabilities.
- Add solar panels to help keep the system running while in the field.
- Remote viewer where operators can remote-in a troubleshoot system if necessary.
- Implement a self-recovery mechanism for ALiCS.

Objectives:

- Camera Vision and Machine learning to navigate around obstacles and approach debris on the beach.
- Measure across visible and near-infrared wavelengths to accurately identify items recovered with fast readout capabilities.
- Measure items ranging in size i.e. aluminum cans, soda bottles.
- Be able to use the camera and lidar puck to correctly determine obstacles that should be avoided by the rover
- Determine what is the most efficient and effective way for the robotic arm to pick up trash and avoid clashing with other components.
- Implement redundant safety systems like emergency stops and robot arm force limits.
- Design an enclosure that allows ventilation and protects the system from the elements such as sun, heat, rain, and sand.

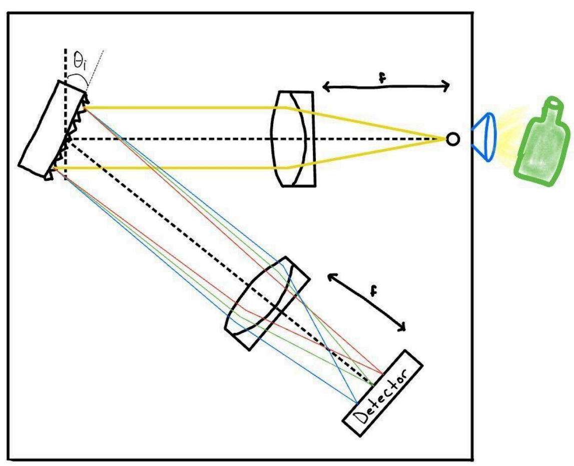


Figure 2.3.1: *Hand-sketch Spectrometer with TCD1304 Detector*

Hardware

Rover - Objective

The Rover is the subsystem responsible for traversing the beach for debris collection. It is a fully automated self-driving platform outfitted with sensors, motors, and durable beach style wheels. The rover includes an enclosure that will protect PCB, power supply, and electronics from the beach environment. The rover provides stability to successfully transport the robot arm without falling over or getting stuck on one side while it is picking up trash or traveling in heavy winds. The rover has a basic motor to move and is connected to our microcontroller. To move around on the beach and keep stability we chose the beach balloon wheels on the rover.

Robot Arm - Objective

The Robot arm is the biggest and boldest part of our project. We have access to an industrial grade robot arm and plan to utilize it in our design. The star of the show will be fitted with a claw like gripper that will be able to pick up objects ranging in size.

Spectrometer - Objective

The spectrometer determines whether the item collected is recyclable, non-recyclable, or possibly a seashell. The spectrometer measures across the Near Infrared spectrum (NIR) for analysis of the material. Measuring in this spectrum, 900-1600 nm, will give accurate results in real time. The basic function of a spectrometer is to receive light, break it into its spectral components, digitize the signal as a function of wavelength, and read out its analysis. We have designed and developed the spectrometer so that it is capable of measuring items ranging in size while maintaining accuracy and repeatability of what type of material is being measured.

Light Source - Objective

The light source is responsible for emitting light on the collected materials so that reflected light measurements can be recorded. The reflected light is currently the definitive portion of information the system will require to determine differences between materials. We can look at it as the unique fingerprint that helps determine what is recyclable vs non-recyclable. Therefore, the light source must be consistent and produce the same output measurement after measurement.

Optical Fiber - Objective

The optical fiber delivers the reflected light collected from the sample to the spectrometer. This is an important piece of equipment because it helps collect light remotely, without risking damage to an expensive spectrometer or its components. The fiber is also necessary to keep light as uniform as possible so accurate measurements can be recorded.

Power Supply - Objective

The power supply is protected by the rover enclosure and powers our motors, spectrometer, the PCB, and the robot arm. We aim to have an efficiency on the power supply of at least 80% minimum but will shoot for more around ~90% for the final product. The power supply will also either be able to be plugged straight into the wall to recharge or easily accessible to be taken out and recharged then put back in between charges. We would also like to shoot for a minimum of an hour battery life.

PCB - Objective

The printed circuit board is currently the brains of the operation. The PCB has all the programming to control ALiCS to the fullest extent. The PCB is able to interface with the motor of the rover to control driving. It uses the sensors to detect where trash is and drive to the trash to pick it up. In addition, the PCB controls the robot arm to pick up the trash. The PCB will have a decent microcontroller on it with enough memory to be able to run all the specified goals. It will also be able to easily interface with all other parts of the project.

Charging Station - Stretch goal

Our robot strives to be almost fully self-sufficient. The goal is to provide a turnkey solution where an operator can power up the system after it's charged and watch it go. Implementing an automated charging station will allow the system to truly be fully autonomous. That's where we design a landing pad for our rover to come home to. These charging stations would either work through wireless charging or precise programming to connect it to a charge port. The charging station would need to be mounted to a flat area on the beach such as the bottom of a boardwalk to make sure the bot can get there with no issues and without any changes to the charging station.

Software

Rover - Objective

The first objective is for the rover part of the project to be able to drive on its own while also making sure it is still on the sand. For this to be achieved, ALiCS must be able to communicate with the equipment that allows the rover to move and also the partial objective so that it can make sure it stays on sand. The partial objective is making sure that the camera and the lidar puck are able to contribute info to both the robotic arm as well as the rover. The reason it is needed for the rover is to make sure that it stays on the sand and if there are any obstacles in the way it should avoid it. The rover should also be able to drive close enough to trash so that the robotic arm can pick it up

Robot Arm - Objective

The second main objective of ALiCS's software is the robotic arm. The arm is capable of receiving commands from the software, and then position itself to where the trash is, and then pick it up without an issue. Once it picks up the trash, the spectrometer will then determine what type of trash it is and then the robot arm is programmed to place it in the correct bin. The arm is limited from rotating 360 degrees because the arm is placed in front and it could potentially collide with the enclosure behind it causing damage to both the enclosure and robot arm.

Trash Detection - Objective

The third objective of ALiCS's software is to be able to detect trash. The software detects trash from the sand and is able to tell the rover to go and pick it up. The software needs to be able to make use of the camera to detect what is trash and how big is the trash. We want to make sure it doesn't pick up anything like shells or rocks that would waste trash space. We started out with a goal of being able to pick up anything to the size of an aluminum can. We can then hope to grow this to be able to pick up something a little bigger and heavy like a beer bottle later on in the project. After picking up the trash, the spectrometer should also be able to scan the trash to figure out what type of trash it was.

Unstuck feature - Stretch Goal

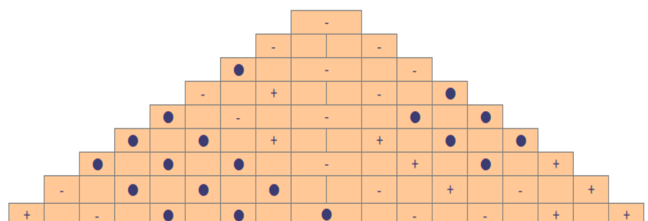
One idea that we thought would be very challenging, but useful, is to implement a self-recovery mechanism for ALiCS. The arm should be able to reach the ground and exert enough force to push or pull the robot. This requires lots of mobility and strength from the arm, and a control algorithm that controls the arm's recovery mode. To determine whether ALiCS is stuck or not, we can use wheel encoders to detect if the wheels are still spinning while simultaneously, the robot is not moving.

2.4 Product Specifications

Table 2.4.1: *Product Specification Table*

Component(s)	Parameter	Specification
Meca500 Industrial Robot Arm	Payload, Reach Speed	0.5kg 0.33m 150 deg/sec
Power Supply	Discharge Time, Efficiency	TBD >80%
Spectrometer	Readout Rate Sensitivity Optical Bandwidth	60Hz 1000 counts/(ms μW) 2.0-5.0μm
Raspberry Pi Camera	Resolution Depth of Field FOV Framerate	8 Megapixels 10cm to ∞ 62.2 deg 24 fps
Lidar Puck	Range Field of View Power Scan Rate	12m 360° 5v 2-10Hz

Motor (Drive)	Current Torque	6.6A 0.33 newtons-meter
Motor (Steering)	Torque	0.33 newtons-meter
Raspberry Pi 4	Pinout Decoding capability Peripherals	40 pin GPIO header 1080p60 decode, 1080p30 encode 2 USB ports 2 micro-HDMI ports Micro-SD card slot 5V DC via USB-C 5V DC via GPIO
Proximity Sensors	Range	4 - 30cm



	Column number		1	2	3	4	5	6	7	8	9	10
	Functional Requirements	Direction Of Desireability	Size	Weight	Battery Life	Accuracy of Robot Arm	Accuracy of Spectrometer	Accuracy of Camera and Sensors	Power Utilization Efficiency	Speed of Trash Collection	Motors	Wheels
	Direction Of Desireability		∇	∇	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
Customer Requirement	Cost	∇	↓↓	↓↓	↓↓	↓	↓	↓	↓↓	↓	↓↓	↓
	Energy Effiecncy	Δ	○	○	↑↑	○	○	○	↑	○	↓↓	○
	Power Delivery	Δ	○	○	○	○	○	○	↑↑	○	○	○
	Ease of use	Δ	○	○	↑	↑	↑	↑	○	↑	○	○
	Ease of Maintance	Δ	↓↓	↓↓	↑↑	○	○	○	○	○	○	○
	Product Efficiency	Δ	↑	↑	↑↑	↑↑	↑↑	↑↑	↑	↑	↑	↑
	Reliablility	Δ	↑	↑	○	↑	↑	↑	↑↑	○	↑↑	↑↑
	Time Saving	Δ	○	○	↑↑	↑	↑	↑	↑↑	↑↑	○	○
	Portability	Δ	↑↑	↑↑	○	○	○	○	○	○	○	↑

Correlation	
Positive	+
Negitive	-
No Correlation	●
Relationship	
Strongly Positive	↑↑
Weakly Positive	↑
No Correlation	○
Weakly Negative	↓
Strongly Negative	↓↓
Direction of Desirablity	
High	Δ
Low	∇

Figure 2.4.1: *House of Quality*

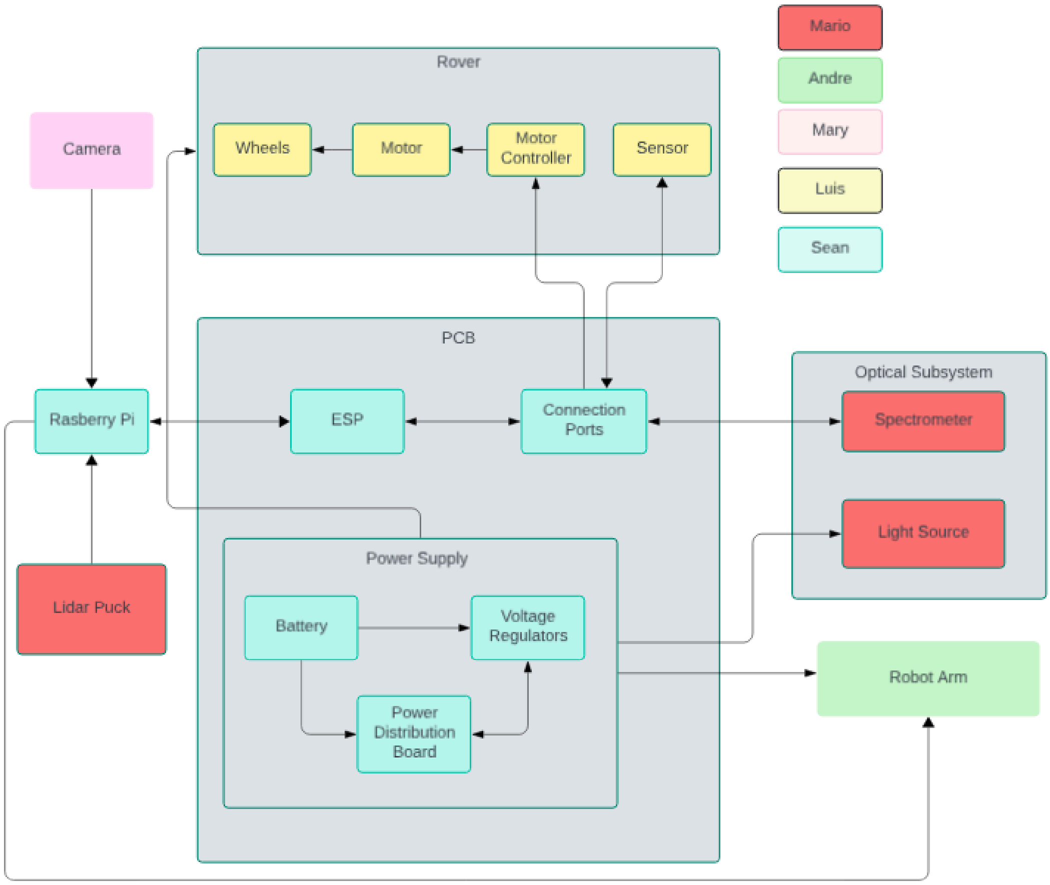


Figure 2.4.2: *Hardware Diagram*

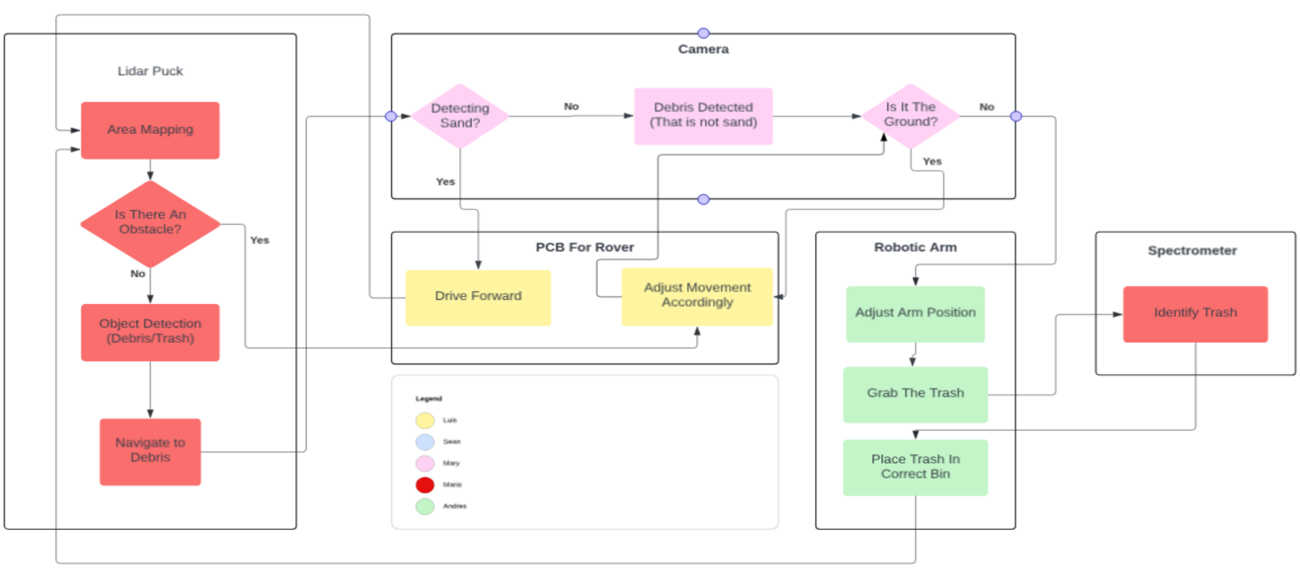


Figure 2.4.3: *Software Diagram*

Chapter 3: Research and Part Selection

3.1 Technology Research and Comparison

The first topic, and one of the most important ones, under the management of technology is generally research and development (R&D); examining new product strategies, organization for research, and the sequential process of winnowing the many ideas for product research and development to an affordable level, according to technical, market, and organizational considerations. In engineering design, thorough research is crucial to ensure optimal outcomes. The research process in engineering design involves meticulous exploration and evaluation to inform critical decisions throughout a project. We begin by identifying project requirements and specifications, then conduct extensive research to gather information on available technologies, materials, and methodologies. This phase includes literature reviews, previous senior design projects, and some experimental testing to validate assumptions. Our group has compared multiple options, considering factors such as cost, performance, reliability, and feasibility within project constraints. Throughout this process, we must document and refine design ideas to ensure that the final design meets or exceeds expectations while addressing any challenges encountered along the way. During this research, different technologies should be evaluated and compared to one another for each of the subsystems of the overall design. The comparison of technologies should be within similar scope and depth to truly gauge the technical approaches available to our design team. Once the executive decision is made to choose a particular technology, there must be some flexibility during the design to allow changes. The initial idea chosen may require adjustment or abandonment based on issues uncovered or obstacles encountered during the design phase. The process of developing a system or product is iterative, with discoveries often made during design, testing, or production. These findings can complicate or alter the product, requiring a return to the drawing board to redesign certain subsystems. Insufficient research beforehand can lead to technical setbacks or delays, as additional time may be needed to explore alternative technologies or solutions. Therefore, frontloading the research process with a thorough exploration of potential technologies for each subsystem or product segment is more efficient, reducing or eliminating research bottlenecks in the product design phase.

3.2 Hardware Research

Our design encompasses multiple sub-assemblies to complete its task of maintaining a clean and safe beach or waterfront environment. From autonomous navigation to sorting plastics and metals, this design required thorough research to find the best technological approach to launch a successful product. Our robot will employ a range of sensors, peripherals, algorithms, and other components to meet our hardware requirements and specifications. Each component and algorithm underwent thorough research and comparison with at least two other alternatives in its category.

Navigation and Mapping (LiDAR vs Camera-Based Vision)

In the last decade, there has been an increase in development of sophisticated technology that goes into creating safe and successful self-driving vehicles. One of the most crucial innovations is the technology that powers vision and navigation. There are several approaches we could've taken for autonomy, including Light Detection and Ranging (LiDAR) sensors and camera-based vision. We explored each of these two technologies and their unique benefits. Ultimately, for the autonomous portion of our design, we decided to focus on implementing LiDAR as the main source of navigation information. Our robot will still utilize a variety of sensors and peripherals as precautionary redundant safety measures.

The two technologies that are at the forefront of enabling autonomous vehicles to understand and interact with their surroundings: Camera-Based Vision and LiDAR. With camera-based vision, the system uses cameras to capture images of the environment. The data collected is processed using computer vision algorithms to identify objects, read road signs, and understand traffic signals, providing essential information for navigation. With LiDAR, sensors are equipped with pulsed laser beams, usually in the infrared spectrum, and receivers to produce a 3-dimensional point cloud.

LiDAR

Light Detection and Ranging (LiDAR) is a remote sensing technique that utilizes laser light pulses to measure distances and create a detailed, three-dimensional map of its surroundings. An active system means that the system itself generates energy – in this case light – to make measurements of the environment. LiDAR calculates the distance between the sensor and the target by emitting an electromagnetic signal as a laser beam towards a given direction. The elapsed time between emitting and receiving the laser beam is employed to calculate the target's distance. The system utilizes specially designed optoelectronics, mechanical moving parts, and high-speed data computing to develop a point cloud of its environment. A typical LiDAR data product is an extensive collection of point cloud data (millions to billions) of accurate 3-dimensional (X, Y, Z) points with other attributes like intensity, reflectivity, and distance. LiDAR technology has emerged as a revolutionary tool in various fields, including geospatial mapping, autonomous vehicles, environmental monitoring, and robotics.

Ranging utilizing Time of Flight (ToF) is a method employed in various technologies, including LiDAR, to measure distances accurately. The fundamental principle behind this technique is to calculate the time it takes for a signal to travel from a source to a target and back, using the speed of the signal as a reference to determine the distance (Equation 1). The LiDAR system begins with a laser diode or LED which is directed to emit infrared light. Direct ToF uses short pulses of light and measures the time until each pulse returns to the sensor to measure the distance to an object. Indirect ToF sensors emit a continuous wave of modulated light. A photoreceptor senses any reflections, and the timing and phase of the reflected light are used to calculate the distance to the object which produced the reflection. For direct ToF, the pulses are typically several nanoseconds wide. Shorter pulses give higher resolution but have less energy and thus the received reflection has inferior sign/noise ratio (SNR); that's one of the many tradeoffs.

$$d = \frac{\Delta t \times c}{2} \tag{1}$$

In Equation 1, *d* is distance from the sensor, *Δt* is the TOF, and *c* is the speed of light ($3x10^8$ m/s). The laser pulses impinge on the rotating and scanning mirrors, interact with the environment, reflect to the sensor and a collection of different distances are measured and produced in a point cloud. This technology creates accurate 3D maps of the environment, offering detailed insights into object shapes, sizes, and distances, even under varying light conditions.

Camera-Vision

Camera-based vision systems are integral to autonomous vehicles (AVs) and robotics, serving as the eyes of the system. Usually, these systems utilize one or more cameras mounted on the vehicle to capture visual data from the objects surroundings. Camera-based vision systems work by continuously capturing video footage of the environment. This footage provides a rich dataset that includes road geometry, traffic participants (vehicles, cyclists, pedestrians), traffic signs, and signals. Advanced image processing algorithms then analyze these images to identify and classify objects, interpret road signs, and understand the vehicle's position relative to road markings and other landmarks. This data is then processed and analyzed to inform navigation decisions, making camera-based vision a critical component of AV sensor suites. Computer vision and AI are essential for analyzing visual data from camera-based systems. Computer vision enables the detection of patterns and shapes, while AI algorithms, especially those using machine learning and deep learning, are taught to identify objects and situations. These technologies empower autonomous vehicles to interpret visual information, facilitating informed decision-making regarding navigation, obstacle avoidance, and compliance with traffic regulations.

The benefits of incorporating camera vision may include but not limited to, cost-effectiveness, high resolution imaging, and color/texture information. While there are great benefits in certain applications within autonomy, we believe this technology does not meet our requirements. Mainly, because camera vision can be directly affected by lighting conditions. Whether it be low light, direct sunlight or reflections from the water; these scenarios can impair image quality and limit the systems ability to accurately interpret visual data. When evaluating camera vision technology, we must take into consideration the beach or waterfront environment. If our system depends on camera vision for navigation, there must be sunlight so the system can image. If the customer is considering running the system overnight or once beachgoers have left the area, the lack of natural light may affect image quality. We also considered mounting a light source to the system to supplement the camera vision; however, coastal communities are limited to the amount of artificial light on/near the beach because artificial light may discourage female sea turtles from nesting and may cause hatchlings to become disorientated and wander inland. Below are some performance specifications taken into consideration while comparing the technologies available to us.

LiDAR vs Camera Comparison

The integration of these two technologies for our application is not a matter of deciding whether one is better than the other but rather understanding how each contributes to the success of the design. We have examined their strengths and weaknesses, and how they may enhance or inhibit our perception system, focusing on technical aspects of sensor integration and their role in developing fully autonomous navigation systems. LiDAR's key contributing factor is its ability to generate accurate 3D maps of the environment. These maps are comprehensive, including not only the position and size of nearby objects but also the topology and layout of the terrain. The spatial data collected by the system is essential for path planning, obstacle avoidance, and ensuring the rovers' safe interaction with the environment. Unlike camera-based vision systems, LiDAR does not rely on natural or artificial light sources, allowing it to work effectively in low-light or nighttime conditions without sacrificing performance. In camera-based vision, the system excels in object identification and visual features due to their high-resolution imaging capabilities. While this could be beneficial for our application in other aspects, i.e. collecting items from the environment, it does not support full autonomous navigation along the beach environment.

Table 3.2.1: *RPLiDAR A1M8 vs USB Camera 1080p Performance Specification Comparison*

LiDAR		Camera-Vision	
Performance Parameters	**Numerical Value**	**Performance Parameters**	**Numerical Value**
Sampling Frequency	8000 samples/sec	*Frame Rate*	Minimum: 60fps @ 1080P Maximum: 260fps @ 640x360P
Scanning Frequency (Hz)	Max: 10 Min: 5	*Pixel Size (µm)*	2.0 x 2.0 µm
Range Radius (m)	Max: 6 Min: 0.15	*Image Area (µm)*	5440 x 3072 µm
Scan Angle (°)	0-360°	*Resolution (megapixels)*	2.30 megapixels

Angular Resolution	≤ 1°	*Dynamic Range (dB)*	64.6 dB
Distance Resolution	≤ 0.5 mm	*Sensitivity (mV/lux-sec)*	1900 mV/lux-sec
Voltage (v)	Max: 5.2v Min: 4.8v	*Minimum Illumination (lux)*	0.1 lux
Supply Current (mA)	Min: 1000	*Voltage (V)*	5 volts
Working Current (mA)	Average: 350 Max: 500	*Price*	~$50-$130
Price	Average: ~$100		

Rover

After talking about the navigation and mapping of ALiCS, it is now time to talk about what part of ALiCS is going to benefit from the navigation and mapping, which is the rover. The rover is essentially the most important part of the whole project. The reason as to why is because the rover is the connecting piece to mostly every component. It uses the lidar puck's mapping of the area to drive around making sure that it avoids any kinds of objects that could harm it or vice versa. It uses the camera to find objects, and once the object has been detected as the trash we are collecting, the rover will then drive its way towards the trash. Once the rover has made its way to the trash, ALiCS will then use the robotic arm and the spectrometer. The rover has to make sure that there is enough distance from ALiCS and the trash so that the robotic arm is able to collect said trash. Once the trash is collected the spectrometer will analyze it to determine what type of material it is made out of. When it comes to the components that make up the rover, there are three main components and they are the wheels, motor, and the sensor.

Wheels

The wheels are what allows our rover to drive around the sand. These wheels should have excellent off-roading capabilities because our rover would be driving on sand and gravel which is a terrain that most wheels do not accommodate for. There are some things that we are not taking into consideration like durability. All wheels should realistically last as long as the time we have for this senior design project, so for that reason we are not considering how durable the wheels should be. If we decide to continue on with ALiCS after senior design, we can then possibly look into better durable wheels. Besides durability, there are other factors that we looked into that we deemed that are important, factors like size, cost, material, and weight.

The size of the wheel is important because we figured that we would want there to be some good distance from the ground to the body of the rover. The reason for that is when ALiCS is moving, there is a decent chance that wheels could kick up sand. We do not want the sand to kick up and land on the rover so that it will not fall in any of the components and possibly damage our components. The main component that we are concerned about is the robotic arm since it will be one of the only components that is not boxed in or is unreachable. The robotic arm is also borrowed so we would most definitely like to return it back undamaged. When it comes to the price of the wheels, we want to make sure that it is at an affordable price range for us considering that we, ourselves, do not have any sponsor.

We also want it to be at a price range where we can afford to buy it more than once in that chance that something unfortunate happens to our wheels. Things like, for example, hidden glass buried in the sand penetrates our wheel as our rover is moving. There are not a lot of materials you can use for tires. The main material that is used by mostly every car is rubber. There is, however, a material that is not talked about, that is very good for the tasks we need and that is polyurethane. Polyurethane is a type of plastic material that is believed to be more durable as well as having a longer lifespan than rubber wheels. Polyurethane is mainly used to navigate off road terrains like sand and gravel, which is perfect for what ALiCS is driving on.

The final factor that we reviewed was weight. There are two parts of the weight that is being thought about and that is; can the wheels carry the weight of everything on it, and also is the motor able to move the wheels, considering the weight of said wheels. When it comes to the movement of the wheels with the motor, we have to make sure that the motor is able to constantly rotate the wheels so that the car can move forward. Another thing is that if the wheel is too heavy, it would end up putting a lot more strain on the motor, which would mean that it would consume more power. So we have to make sure that the weight of the wheels is not too heavy where it would negatively affect the motor. The other part of the weight was making sure that the wheels are able to carry the weight of everything that is built on top of it. This would include the robotic arm, which weighs at around 4.3 kilograms or 9.5 pounds, the camera, spectrometer, lidar puck, the pcb, as well as the structure of the car which in total weighs probably in the range of 15-25 pounds. If the wheels are not able to carry said weight, ALiCS might be able to move forward, but it will have a difficult time in steering itself. So the wheels have to be strong and heavy enough to support the structure, but also not too heavy in order to keep the motor functioning properly. Listed in the table below are the comparisons we made in order to choose the correct wheel.

Table 3.2.2: *Comparison of Three Different Types of Wheels*

Wheels:	Balloon Wheels	Standard Wheels	Omni Wheels
Movability	Moves Forwards and backwards	Moves Forwards and backwards	Moves All Directions
Size (height):	Typically Tall Wheels (Around 9-12 inches)	Can Range From Small to Tall Wheels	Typically Small Wheels (Around 4-7 inches)
Material:	Polyurethane, rubber, PVC	Rubber	Plastic and rubber
Cost (for all four wheels):	High	Moderate	Low

Choice For Ballon Wheels

With all the specifications and requirements we need from the wheels, we were able to come to the conclusion that balloon wheels seem to be the best choice. The reasoning being that these wheels are well designed for off roading capabilities and that is something we need considering the rover will be driving around the beach. It's usually tall height is also a huge benefit to keep our design off the ground and keep the sand off our components to prevent any kind of damage. Using our choice of balloon wheels, we have three specific wheels that we believe would be best suited to be compatible with ALiCS. These three each have their own positives as well as things that hold them back which will now be discussed.

Wheeleez Polyurethane Wheels

For one of our choices of balloon wheels we believe that it would be the best choice for us to use Wheeleez's polyurethane wheel. The reason why is because in their website the wheels are described as beach balloon wheels that are designed for a specific reason and that reason is to navigate difficult terrains. Terrains like sand, gravel, mud, and rock. This type of wheel is perfect for the fact that it has the off-roading capabilities that is needed for ALiCS since it is intended to be driven around the beach. It has a diameter of 24 centimeters or 9.4 inches which is a very good height that could keep ALiCS off the ground and prevent any sand from entering ALiCS. Also they have more options that are higher than 24 centimeters if that is not enough. The main downside of these wheels is since it only travels in two directions, it would be difficult to position itself when it is time to pick up the trash. The main detriment about the wheeleez wheel is the price with it being at seventy dollars per wheel, meaning it would cost us two-hundred and eighty dollars just to get four wheels and if any of them pop or flatten we would need to spend another seventy for it.

VEVOR Wheels

For the second choice of balloon wheels we have the VEVOR beach balloon wheels. Another choice of balloon wheels that is intended and designed for beach use. The material that the VEVOR beach wheels are made of is polyvinyl chloride (PVC). These are decently heavy wheels at least compared to the other wheel options we have explored. They weigh at six pounds each and if we were to proceed with this choice, the weight would be important to note later on for our motor choice so that we would need a pretty solid motor that is capable of rotating six pounds constantly. The price is around the same range as the first one with being $59.99, but it is sold as a two pack, meaning it will cost $120 for four wheels and it also comes with an air pump which could be useful if mistakes were to occur.

Extreme Max Wheels

The last wheel choice we have is the Extreme Max wheels. Same as the other two wheels we have, they are balloon wheels that are specifically created for traveling on sand. The material that this balloon wheel uses is not specified but we are assuming that the material should either be polyurethane or PVC like the other two tires we have, or it could possibly be rubber as well. The weight of the wheels fall in the middle between the other two wheels, weighing at 3.85 pounds. The cost is a little on the pricier side, being $89.95 for two wheels so it will come out to be $179.90 for four wheels.

Wheel Comparison

With all specifications taken into consideration, we believe that what we want is for the wheels to have no problem navigating over rough terrain like sand and gravel that will constantly have different landscapes, which is what all three of these wheels are capable of. The main deciding factor was the price. Since we do not have any kind of sponsor, any way for us to save some money for our budget is what is best for us. These three wheels did not have any kind of vast difference that would have made the other two options worth the price that they are. If there are some issues with the weight of the wheel, it would be way cheaper to purchase a better motor than purchasing new wheels.

Table 3.2.3: *Wheel Comparison*

Wheels:	Wheeleez Polyurethane Wheels	VEVOR Wheels	Extreme Max Wheels
Material:	Polyurethane	PVC	Not Listed
Size: (Diameter)	24 centimeters 9.4 inches	33.02 centimeters 13 inches	27.94 centimeters 11.3 inches
Load Capacity:	88 pounds 40 kilograms	77 pounds 34.9 kilograms	Not Listed
Weight:	1.5 pounds 0.7 kilograms	6 pounds 2.7 kilograms	3.85 pounds 1.74 kilograms
Cost (All Four Wheels):	$280	$120	$179.90

Motor

The motor is known for turning electrical energy into physical motion. It is heavily used for any mechanical projects whether it's big or small. They are used even in everyday household items such as blenders, fans, and even washing machines. When it comes to the rover, the motor has a very important job and it's to move the wheels. It is important to realize that there are more than one type of motor, even when it comes to small autonomous cars like ALiCS. Although there are a lot to choose from, we need to realize what requirements from the motors we should prioritize.

One of the requirements we looked into was energy consumption. What we want from the motor is to use the power we have from the battery efficiently because there are multiple things taking from the power and it could lead to our battery for ALiCS draining fast if we are not cautious with our choices. Another thing to consider is whether or not ALiCS would use one motor per wheel, or one motor for the entire vehicle, or using three motors in total, but adding the fact that we have a robotic arm that needs to pick up trash, it would be best to have either one motor per wheel or one motor for the front wheels and one motor for the back wheels so that way the rover could have a better performance as well as better control. Since we plan on using four motors, we would have to make sure that the energy consumption of all four motors will not be an issue for our choice of battery.

Another requirement that is considered to be important is the size and weight of the motors. The reasoning for that requirement is that the motor would have to be able to fit in the design of our rover as well as not be extremely heavy to the point where the wheels can not bear the weight of the rover. This is a requirement that is not a total worry, but it does not hurt to double check and make sure that it will not be a problem now then have it affect us later. Now let's take a look at the choices we have for motors:

Brushed DC Motors and Brushless DC Motors

The first motor we will talk about is the brushed DC motor. The brushed DC motor uses electrical current that passes through coils that are set up in a fixed magnetic field. What this ends up causing, is the coils start pushing away from either the south or north magnetic pole and are then pulled towards the other south or north magnetic pole. There are fixed conductive brushes that help contribute power to the coils. All of this repeats and what is created is of course a motor. The benefits of using a brushed DC motor is that they have high efficiency and outstanding controllability. The brushed DC motor is able to reach maximum torque at certain points in the rotation. They are also used in a wide amount of applications, meaning that there would be a lot of outside resources available if something were to go wrong with the motor.

The second motor is the brushless DC motor. The main difference between the brushless DC motor and the brushed DC motor is the use of brushes. The brushed DC motor uses fixed magnetic poles and then uses brushes to make the coils move, but for brushless DC motors, it uses a rotor that has both the south and north magnetic poles attached to it while it is rotating. So instead of having the coils move, they are in a fixed position while the poles themselves are moving. Since the rotor has both of the magnet poles, there is essentially no need for current, which means that there is no need for the brushes. One downside of using a brushless DC motor is that the wiring would end up being a bit more complicated than other motors in which you just connect the power’s positive and negative terminals. Besides that there are some advantages that are also pretty similar to the brushed DC motor and one of them is efficiency. These motors are able to continuously be controlled at maximum torque as opposed to brushed DC motors which can only achieve maximum torque at certain points. Another advantage they have is that they are able to use feedback mechanisms that would allow them to have excellent controllability. These feedback mechanisms would allow the motor to precisely get the desired torque as well as the rotation speed. As a result of that, precisely controlling the motor would end up reducing energy consumption which will also help lengthen the life of our battery, which would be very beneficial for ALiCS. Since there are no brushes, there will be very little electrical noise being made, which can help prevent any harm to animals that are habitating the beach.

Stepper Motors

A stepper motor is also a DC motor just like the brushed and brushless DC motors. A stepper motor has a very similar structure compared to a brushless DC motor and a brushed DC motor in which it also has a rotor in the middle with coils surrounding it. Like a brushed DC motor, the rotor can be a magnet, but it also can be a variable reluctance iron core. In this case we will be talking about a stepper motor with an iron core as the rotor. Each coil gets energized when the rotor is aligned with the magnetic field it produces. This continues causing the motor to continue spinning. Using a stepper motor allows us to reach a higher speed, but it would end up with a lower torque. The big difference between this motor and the other two motors that were mentioned before is that the stepper motor has more precise movement which is something that we would need in order to have good positioning for our robotic arm to pick up trash.

There are a good amount of downsides for a stepper motor that should be addressed. One thing is that the motors, even when it is not moving, will always drain the maximum current. This causes the efficiency to be worse and it can also lead to overheating. It also can affect our battery life, which is not ideal for ALiCS. Another thing is that stepper motors tend to become very noisy only when it is at high speeds, so it would be best for us to avoid making ALiCS go as fast as possible when using a stepper motor. With all that we know, we determined that the best choice is a brushless DC motor mainly because of its high torque.

Table 3.2.4: *Comparison of Three Different Types of Motors*

Motors:	**Brushed DC Motor**	**Brushless DC Motor**	**Stepper Motor**
Torque:	Reaches its maximum torque at certain positions	Constantly reaching its maximum torque	Low torque, does not reach its maximum torque
Efficiency:	High	High	Low
Power Usage:	Moderate	Moderate	High
Weight:	Low	Low	Low
Cost:	Low	Moderate	Low

Motor Selection

After discussing the three types of motors that we planned on using and their pros and cons, we will now talk about three specific choices of the motor that we think are good choices for our design. All the motors have some benefits that make them better compared to the others but they also have some downsides that make it hard to justify using them.

57BLF03 Brushless DC Motor

The first brushless DC motor option we have is the 57BLF03 Brushless DC Motor. The motor has a total of 4 poles in the motor. This motor performs at a voltage of 24 and has a rated speed of 3000 revolutions per minute (rpm). Although, twenty-four volts is a lot when you consider that we may either use three of these motors or five of these motors depending on whether we want to do two wheel drive or four wheel drive. What's good about the voltage usage is that it means that because of the high power requirements it provides as a result a higher output, which is good in order to make sure the wheels have no issue moving the entirety of ALiCS. Another thing about us choosing two wheel drive or four wheel drive is that we would need to consider the cost of this motor. It also has a good torque of 0.33 newton-meters.

RM-ESMO-01F Brushless DC Motor

This motor is a brushless DC motor that aligns pretty well with the requirements we need. IT also has the same amount of voltage as the brushed motor we picked, but there is a big difference between the revolutions per minute (rpm). The no load speed of the brushed dc motor is at 326 rpm, while the no-load speed for the brushless DC motor is at 8000 rpm. Although the rpm is good, that is not what we consider important because we mainly value torque. The torque of this motor is 700 grams-centimeter which is vastly different compared to the brushed DC motor

TEC 4260 Brushless DC Motor

This motor is advertised as a high torque Brushless DC Motor, which would mean that it is considered to have a higher torque than most. With a torque of 500 grams-centimeter, it has the lowest torque out of all three options. This alone would make it hard for us to choose this motor for our device. It does at least have some speed compared to the first one with 500 rpm, but speed is not our main requirement. For its voltage range of 12-24, it does have some very solid specifications, however, compared to the other options, this motor is not that promising. The main reason why we would pick this option is if during testing, the other motor we are using is having issues with moving the wheels, since this one has a higher torque than the other two.

Motor Comparison

With all the specifications discussed about these three specific motors, it is a good time to decide which one will be implemented into our design. With the main consideration of the motor being torque, there is a very obvious motor that has way more torque compared to the other two and that is the 57BLF03 Brushless DC Motor. Considering voltage, they all use around the same amount of voltage so it does not mean much into our decision. The no-load speed may be beneficial as well, but the priority of the torque outweighs the no-load speed. The reason why torque is so prioritized is so that we are able to traverse the rough terrain that is the sand and or gravel.

Table 3.2.5: *Motor Comparison*

Motor:	57BLF03 Brushless DC Motor	RM-ESMO-01F Brushless DC Motor	TEC4260 Brushless DC Motor
No-Load Speed:	3000 rpm	8000 rpm	5000 rpm
Torque:	0.33 newtons-meter	700 grams-centimeter	500 grams-centimeter
Voltage:	24 volts	24 volts	12-24
Cost (For one):	$36.00	$24.85	$27.99
Rated Current	6.6 amps	2.5 amps	2500 milliamps

Sensor

The sensor that is attached to the rover will act as a failsafe. We want to make sure that ALiCS is safe and will not get in the way of any living being that is currently occupying the beach and for that reason, we added a second sensor that was able to detect any object in close range of ALiCS and if it does detect something it will stop the movement of the rover. In order to do this we would need to have a sensor that has a small range, having a sensor with a large range will defeat the purpose of the lidar puck and will also keep causing our rover to shut down. Price will also be in consideration mainly because in order to detect objects on any side of the rover, it would be best to have 4 small sensors, one on each side, to make sure there are no objects near. Another thing that matters for the sensor is precision and accuracy. It would be very problematic for ALiCS if it is constantly stopping for objects that are not actually in range as well as objects that do not exist. Another problem would be if the sensor keeps detecting the floor because of the potential height difference that can occur with sand. So it is important for our sensors to be precise and accurate. The size should not be an issue as long as it is able to be attached to the 4 sides of the rover which will most likely have a lot of space. The weight itself should not be too much of an issue as well because sensors seem to be very light. Now the two sensors we think that are usable for these tasks are infrared sensors and ultrasonic sensors.

Ultrasonic Sensor

The way ultrasonic sensors work is that it sends out ultrasonic sound waves, and if there is an object or an obstacle, the ultrasonic sound waves will bounce back towards the sensor which will then be able to measure the distance between the sensor and the object. The way it is able to measure the distance is by using time of flight, which has been mentioned and explained in our paper before **(3.1 LiDAR)**, but using the time of flight we are able to calculate the distance between the object and the sensor using this formula:

$$d = (\Delta t \times s)/2$$

Where d is distance, Δt is time elapsed, and s is speed of sound. If we use microseconds for the time of flight, this will give us the distance in centimeters. A problem with the ultrasonic sensor that we need to make note of is the fact that any kind of obstruction, such as dirt, snow, and in our case, sand, will disrupt the sound waves from reaching the sensor. This mainly depends on the height of the sensor and if the wheels will kick up sand towards the sensor. The benefits of using an ultrasonic sensor is that they are able to function properly regardless of light, smoke, dust, and color. This is important in the beach where it can be sunny or dark depending on time and there could be a multitude of colors in a populated beach.

Active and Passive Infrared Sensor

The infrared sensor is another type of proximity sensor, the same as the ultrasonic sensor, which is able to detect objects without having to have physical contact with said object. How the infrared sensor works is that it is able to detect any kind of change in light (that is not visible to the naked eye). It does this by either emitting or detecting infrared radiation from any kind of object/obstacle within their range and if there is any change then there would be an object detected. There are two types of Infrared sensors, active and passive. Passive is only able to detect any infrared radiation from objects but it would not be able to determine the distance from the rover to the object. The use of a passive infrared sensor would be a bit challenging compared to the other options, but it would end up being doable, it would just take a lot more effort. The other sensor is the active infrared sensor, which sends infrared light out and it also detects the infrared radiation from objects which would allow the distance to be determined from the sensor to the object. There are some issues when it comes to using infrared sensors, the one that is important to realize is that fact that infrared sensors are able to be interfered with by other sources of infrared radiation, such as the sun. This is crucial as ALiCS will be operating on the beach in which there will be nothing but the sun during the first half of the day. If we were to choose the infrared sensor, we would have to come up with a way to prevent the sunlight from interfering with the sensor.

Table 3.2.6: *Sensor Comparison*

Sensors:	**Ultrasonic**	**Active Infrared**	**Passive Infrared**
Wave Direction:	Travel To and From Sensor	Travel To and From Sensor	From Sensor
Can Determine Distance:	Yes	Yes	No
Range:	Typically Far Range	Typically Small Range	Typically Large Range
Power:	3.3 - 5 Volts	3.0-5.3 Volts	3.3- 5 Volts
Size:	Medium	Small	Small
Price:	Low	Low	Very Low

Choices For Ultrasonic Sensors

With everything discussed and researched about what we desire from sensors, we were able to pick three sensors that we believe to be beneficial to our design. One that is an ultrasonic sensor, one that is an active infrared sensor, and one that is a passive infrared sensor. These sensors have some positives that really add on to ALiCS and make it thrive as well as some negatives that would hinder ALiCS and create more obstacles for us to conquer.

Ultrasonic HC-SR04 Sensor

We believe that the HC-SR04 sensor is a solid choice for the ultrasonic sensor because of its capabilities. The sensor is able to have measurement functionality in the range of 2 centimeters to 400 centimeters which is a good range for what we are looking for. ALiCS only needs this sensor as a failsafe so it just needs to detect objects close to ALiCS which the HC-SR04 is perfect for. It also has an easy use and setup with just four pins and they are the VCC, Trig, Echo, and GND pins. The price is also very low which is good because we would need one on each side. The benefit of this sensor is the fact that it is able to detect mostly anything without physical contact. It has no issues detecting things regardless of light, dust, and color which is very useful considering that ALiCS will be operating at the beach.

MB1232 I2CXL Sensor

The MB1232 I2CXL sensor is another type of ultrasonic sensor that we could possibly use for this project. It has a very solid detection range from 20 centimeters to 765 centimeters, which has a higher max detection range than both of the sensors. As good as that is, it is not really necessary for the needs of this ultrasonic sensor. What we need is a sensor that is able to detect things up close because we already have the lidar puck that can detect things from afar. It has 7 pins, GND, V+, SCL, SDA, Status, as well as a temporary default pin that is internally pulled high. One of the 7 pins that were mentioned is not used but it does exist on the component, so we can possibly use it ourselves if it is needed. The main problem of the MB1232 I2CXL sensor is that the price of this sensor is $39.95 for just one.

MA40S4S Sensor

The MA40S4S sensor is the last and final ultrasonic sensor we considered for our design. This sensor is a simple sensor with no extra details to it besides it just functioning as an ultrasonic sensor. It does not have any pins. It also does not mention anything about its detection range which is concerning because that is definitely one of the most important factors to look for when using a sensor. The main good thing about it is that it has a directivity of 80 degrees which is higher compared to the other two sensors. The price of the sensor is also crucial because it cost $5.44 for one which is important considering our budget

Sensor Selection

With all the specifications and requirements for the sensor discussed, it is now time to compare and decide which sensor was the best choice. The biggest deciding factor is the accuracy of the sensors and anything that can be harmful to the accuracy is very relevant. The reason why is because this sensor acts as a fail safe and what good is a fail safe that has a huge chance of not working. With that being said, the problem with the maxsonar sensor is that when there is an object that is in front of the sensor that is below 20 centimeters, the sensor will display it as 20 centimeters. This is crucial because these sensors are only in this design to stop when an object is too close to ALiCS, and if the sensor can not detect it if it's too close, then there is no point in using that sensor. A similar problem with the MA40S4S sensor is that the detection range of the sensor is not displayed, even when looking at the datasheet of the sensor. With all the information considered, we determined that the HC-SR04 is the best choice.

Table 3.2.6: *Sensor Selection*

Sensors:	HC-SR04 Sensor	MB1232 I2CXL Sensor	MA40S4S Sensor
Detection Range:	2 - 400 centimeters 0.8 - 157.5 inches	20 - 765 centimeters 7.9 - 301.2 inches	Not Listed
Voltage:	5 volts	3.0- 5.5 volts	1.7 - 3.6 volts
Measure Angle:	30°	Not Listed	80°
Operating Temperature:	Not Listed	0°- +65° celsius	-40° - +85° celsius
Price (for one):	$4.50	$39.95	$5.44

Power Supply

The power supply for ALiCS is the life force of our robot. The job of the power supply just like in any other object took our robot to last as long as it needs to be. This will decide how effective our robot will be. The less human interaction ALiCS has to receive the better. This makes it to where the people working with or around ALiCS have less to worry about and leaving ALiCS to do their job cleaning up pollution off the beach. ALiCS needs to be a self-sufficient robot that can last for a long time while performing its duties. ALiCS's power supply needs to to be an efficient power supply that can draw power quickly and adequately through the system to efficiently run ALiCS. This power supply needs to be robust enough to take care of any leaks if any sand were to get into the ALiCS's main body so the power supply system is.

The first requirement is that the power supply must be extremely power efficient. The reason for this is obvious to keep ALiCS on the job longer without needing help from anyone. We would also want this to be a quick charging factor where it can recharge quickly and dispease energy slowly to keep our robot working for longer. This is why we need to make use of good batteries and voltage regulators to keep our efficiency high on the power supply. We need to make sure we are not draining the battery quickly and wasting energy on wasted resources and quickly turn on and off systems to keep the battery from draining quicker than it should be. This will be decided by the software but will ultimately make a massive difference on the total power drain of our system.

The second requirement is size and weight of the power supply system. The total power supply should be an appropriate size seeing how our design will perform better the longer it can run. However, we would still try to make it as small as possible to leave room for other systems on board ALiCS and the bins for trash and recycling. The next part is the weight of the power system should be as light as possible while still being efficient. If the power system is too heavy it will put more work on our motors and ultimately cost us more power than needed to keep the ALiCS running.

Battery

Batteries are the way to power the device while it's out on the field. The battery we decide to use will need to be capable of keeping the device out in the field as long as possible while still being efficient as possible. Another important thing about batteries is that there are two different kinds of batteries. The first type of batteries are primary batteries that are intended for one time use only. Secondary batteries on the other hand are meant to be recharged and maintained over long periods of time. For this project we will be using exclusive secondary batteries to make sure the only maintenance to ALiCS is that the battery is charged to use ALiCS once again.

Lead-Acid Batteries

Lead-acid batteries are the most widely used rechargeable batteries used today by demand from the automotive industry. There are two basic types of lead acid batteries which are the flooded and sealed type. Flooded batteries are used for starting, lighting, and ignition on cars. Then there are sealed lead-acid(SLA) batteries which are the type we would if we were to use them in our project. The discharge voltage or a SLA typically remains constant while discharging and drops quickly when reaching the end. With normal operating conditions the battery’s capacity will be stable for most of its useful life and then begin to degrade slowly with permanent loss slowly after enough charges. These batteries also happen to be low cost and have various size formats.

Nickel Cadmium Batteries

Nickel cadmium batteries use nickel oxy-hydrate for the active material of the positive electrode and the negative electrode uses the active material of cadmium. The discharge voltage remains constant throughout the discharge of the battery to a sharp drop off near the end. The batteries are also easily charged by applying controlled current rates. These batteries also suffer from the voltage depression effect also known as the “memory effect.” This means if the battery doesn’t work if in the wrong condition such as having a lot of partial cycles or overcharging at higher temperature then it could result in the cycle life ending sooner than it should. However, this condition can be fixed just by completely discharging and recharging the batteries. Cadmium is also poisonous resulting in this battery being more controversial.

Nickel Metal Hydride Batteries

Nickel metal hydride batteries have a 170% increase in energy density compared to NiCds and have the cadmium reflected by an alloy in the cells. This was the early choice for electric vehicles until 2005. They have similar discharge patterns to NiCds but uses a different material for the negative electrode. They also have an estimate of a 50% longer service life with the greater capacity.

Lithium-Based Batteries

Li-ion batteries deliver lightweight battery packs more suitable for smaller devices. These have three times the energy density as nickel based batteries and have three times the thermal voltage. Lithium based batteries have a lower discharge rate at 80% than 95% of the rated capacity for NiCd. Li-ion batteries also have greater fragility due to the nature of lithium. Lithium can not tolerate overcharge or over-discharge and are more extensively damaged by these actions. That is why failsafes are built into most battery packs to prevent this. Lithium batteries also require constant voltage and constant current to charge. Even with all these down sides we are still going to use these batteries for our project because of the size, weight, and power output.

Table 3.2.7: *Battery Comparison Part 1*

Parameters	Lead-Acid	NiCd	NiMH
Average Cell Voltage (V)	2	1.2	1.2
Internal Resistance	Low	Very Low	Moderate
Self Discharge (%/month)	2%-4%	15%-25%	20-25%
Cycle life (# of charge-discharge cycles until 80%)	500-2000	500-1000	500-800
Overcharge	High	Medium	Low

tolerance			
Energy by volume (Watt hr/liter)	70-110	100-150	200-350
Energy by weight (Watt hr/kg)	30-45	40-60	60-80
Operating temperatures	-20 to 50 C	0 to 45C	0 to 45C
Maintenance Requirements	3-6 Months topping charge	Full discharge every 90 days	Full discharge every 90 days
Safety Requirements	Thermally Stable	Thermal Stable, Fuse protection	Thermal Stable, Fuse protection
Cost	Low	Medium	Medium

Table 3.2.8: *Battery Comparison Part 2*

Parameters	Li-ion	Li-Polymer	LiFePO4
Average Cell Voltage (V)	3.6	1.8-3.0	3.2-3.3
Internal Resistance	High	High	High
Self Discharge (%/month)	6%-10%	18%-20%	1–3%
Cycle life (The number of charge-discharge cycles until 80%)	1000-1200	300 to 500	1500-2000
Overcharge	Very low	Very low	Very low

tolerance			
Energy by volume (Watt hr/liter)	200-330	230-410	200
Energy by weight(Watt hr/kg)	120-160	120-210	100
Operating temperatures	0 to 45C	0 to 45C	0 to 45C
Maintenance Requirements	Maintenance Free	Maintenance Free	Maintenance Free
Safety Requirements	Protection Circuit Mandatory	Protection Circuit Mandatory	Protection Circuit Mandatory
Cost	High	High	High

Table 3.2.9: *Battery Comparison Part 3*

Parameters	ExpertPower 12V Rechargeable SLA Battery	Mighty Max 12 Volt 18 Ah ML18-12	Universal Power Group 12 Volt 20 Ah UB12220
Voltage (V)	12V	12V	12V
Internal Resistance	≤ 15 mΩ	≤ 18 mΩ	≤ 14 mΩ
Capacity	20Ah	18Ah	20Ah
Self Discharge (%/month)	3%	3%	3%
Cycle life (The number of charge-discharge cycles until 80%)	200-300	200-300	200-300

Overcharge tolerance	Moderate	Moderate	Moderate
Energy by volume (Watt hr/liter)	90-100	85-95	90-100
Energy by weight(Watt hr/kg)	35-40	34-39	35-40
Operating temperatures	-20 to 60°C	-20 to 60°C	-20 to 60°C
Maintenance Requirements	Maintenance Free	Maintenance Free	Maintenance Free
Safety Requirements	Protection Circuit Mandatory	Protection Circuit Mandatory	Protection Circuit Mandatory
Weight	5.35kg	5.26kg	5.76
Cost	$40 - 60	$35 - 55	$45 - 65

As we can see, our choice between the three sealed lead acid battery choices gives us a well-balanced cost, performance and weight.

Power Distribution Board (PDB)

A power distribution board is a board that connects from the battery or power supply to the part that needs the required voltage. To do this the PDB has voltage regulators on the board to control the amount of voltage going through by either stepping down the voltage or stepping up the voltage. In most cases I would say you are stepping down the voltage to either 3V, 5V, or 12V. These boards make it possible to deliver the correct amount of voltage while not taking much space or wasting much energy from the power supply. These are an essential piece to our project and needed to divide the voltage correctly to each part of our system for ALiCS.

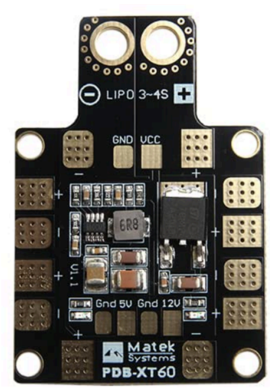


Figure 3.2.1: *Power distribution Board*

The figure above shows an example of how small these power distribution boards are and how they provide easy access to adding your own voltage regulators while still using the components provided inside the middle to help provide clear signals and transfer power to ground or other components.

Empty board

An empty power distribution board is just how it sounds. It is a power distribution board without anything on the printed circuit board with a few wire connections on the board and empty slots to add whatever pieces. These empty boards usually just have polarity symbols on them and holes for them to attach to other pieces. They also have no voltage limit on the board. This makes these boards a lot more customizable to needs of the project instead of being more general use boards with different voltages ranges. Instead the owner gets to decide which regulators and pieces go on the board and which do not. However, one must have basic knowledge of soldering on pieces to a PCB. They must also have a plan of what pieces they want to add beforehand to make these work. The boards also are generally a lot cheaper compared to premade boards because you are doing all the work to add on the parts needed.

Premade board

The premade boards are also just how they sound, the premade boards you can pick and choose what you need. They are like MCU where there are thousands of options you can pick from that would work for your project and someone has to narrow down the things that are needed or would help improve the project by making it easier or run better. These items that come on these boards are but not limited to on screen display, lc filter, current sensor, 5v and 12v regulator, adjustable voltage regulator usually. The lc filter helps a lot with display outputs and making them much clearer. Once one narrows down their board selection then they just have to solder the connections needed for their project to the board and attach the board to their device and it's as simple as that. The big downside to these boards are there aren't really slots to add more parts if needed and they usually cost marginally more for already having everything put together for the consumer.

Table 3.2.10: *PDB Comparison*

Parameters	Empty board	Premade board
Ease of use	Low	High
Customizability	High	Low
Flexibility	Low	Medium
Efficiency	High	Medium
Cost	Low	Medium

PDB Selection

The power distribution boards we decide to make use of are pre-made boards that leave space for us to solder on the needed components such as resistors, capacitors, inductors, filters, and voltage regulators. While still having components ready added in the middle for ease of use. This board will be extremely useful for sorting the amount of voltage and current needed for each part. Especially for getting the exact needed values to run these components and stay energy efficient to keep our system running for longer without needing a charge.

Voltage Regulators

Voltage regulators are essential due to having varied voltage requirements with all the other systems involved working together. The voltage regulators connect to parts to handle certain loads that require specific line voltage. This ensures that the voltage regulator can constantly support the power requirements of all connected systems. These DC/DC converters need to be able to respond quickly to power changes and demands on the fly. Like for motor controllers that require high currents suddenly. The two main types of voltage regulators are linear voltage regulators and switching voltage regulators.

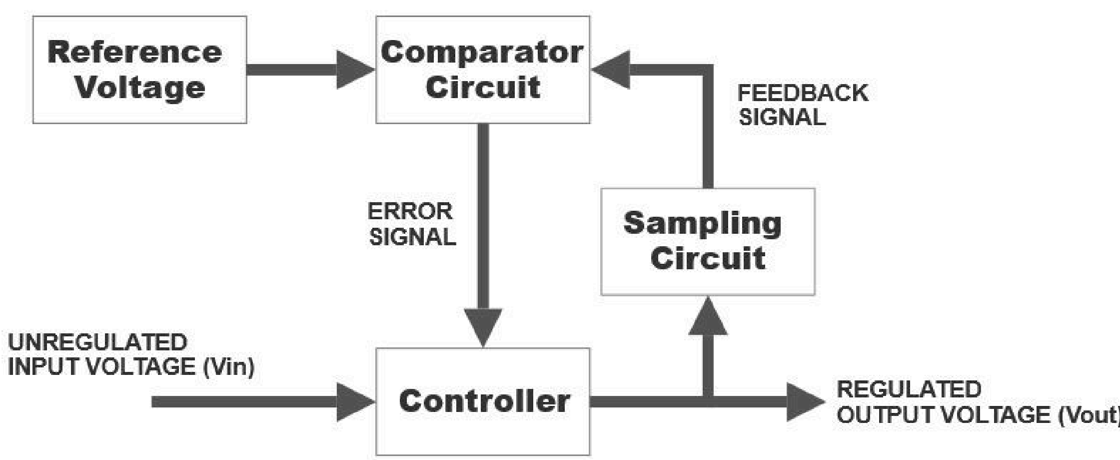


Figure 3.2.2: *Voltage Regulator Block Diagram*

Above in the figure shows a simple block diagram on how a voltage regulator works to correctly adjust the voltage to the correct amount needed by the circuit to correctly function. It shows the loop for how the voltage regulators bring down the voltage to the correct amount.

Linear Voltage Regulator

Linear voltage regulators are the less efficient, less flexible, and larger switching regulators of the two. However, it’s also more dependable, easier to design, and more cost effective. Linear voltage regulators can also only be step down voltage regulators. All excess voltage will be dissipated as heat making it a less efficient regulator. There are two forms of linear regulators which are series and shunt regulators. A series regulator works by putting lines in series or in parallel to the lines they are intended to regulate. A shunt regulator works by shorting the PV module and will have a constant short-circuit current across the shunt element. This causes shunt regulators to be less common and limited to systems with less than 20 amps. Both types of regulators use low voltage disconnects to disconnect the load when the batter is low on power.

Switching Voltage Regulator

Switching voltage regulators allow constantly switching the input voltage on and off using a PWM or pulse-width modulation to reach a specific output voltage. Pulse width modulation controllers constantly connect and disconnect the PV module to the battery enabling complete control on the rate of discharge from the battery. Doing this to batteries reduces the risk of overheating the battery and makes better utilization of the batteries discharge requirements. A PWM is also relatively cheap and easy to use on smaller scale systems. Using this PWM on switching voltage regulators we make a modulated signal that when pushed through an inductor-capacitor circuit can power a load continuously. There are multiple different types of switching regulators that allow the output to be a step-down, step-up or both. The common switching voltage regulators are Buck, Boost, and Buck-boost. One big thing to remember is that the ripple voltage in equivalent series resistance of the output capacitors increases while the temperature is lowered.

Table 3.1.11: *Voltage Regulator Comparison*

Parameters	**Linear Voltage Regulator**	**Switching Voltage Regulator**		
Type	N/A	Buck	Boost	Buck-Boost
Ease of use	High	Medium	Medium	Medium
Efficiency	Medium	High	High	High
Voltage output (compared to input)	Step-Down	Step-Down	Step-Up	Step-Up or Step-Down
Heat	Medium	Low	Low	Low

Output Ripple	Low	Low	High	Low
Cost	Low	Medium	Medium	High

Voltage Regulator Selection

For voltage regulators we have decided to make use of mostly adjustable voltage regulators. For their flexibility combined with their high efficiency. It makes them the better choice for most cases in our project. They are easily able to make the voltage we need and can be put on either the designed PCB we did not make or the power distribution board. However, there could be some cases where we could make use of a linear voltage regulator so in those cases where we will not lose much out of having medium efficiency and just makes the process simpler we made use of a linear regulator instead of an adjustable one.

PCB

The printed circuit board will be the brains of the operation. This will lay the groundwork for all the connections for all of ALiCS. The printed circuit will need to have the correct peripherals to connect to everything needed to make ALiCS work. This includes making connections to the sensors, camera, robot arm, motors, power supply, and spectrometer. These connections need to be secure and fast.

MCU vs FPGA

In this project we have an important decision between two great pieces of technology. The two best choices to use for the brains of our project are a microcontroller unit (MCU) or a field-programmable gate array (FPGA). There are pros and cons to both options but we need to decide which one is better for our project. Factors to choose from are processing speed, specialization, cost, and power draw.

FPGAs are more powerful and precise with little to no margin of error and specifically made for the task they are programmed to complete. This has its drawbacks consisting of higher cost, more power consumption, and almost no flexibility. FPGAs are also not really user friendly and have more complex programmable hardware made for industry applications not made for small projects. FPGAs use hardware level languages such as Verilog and VHDL making them harder to learn and use for a causal level. In addition FPGAs need to integrate external parts to make use of simple parts that come pretty commonly on MCUs like digital converters or communication interfaces like UART and I2C. One advantage to FPGAs is that they are scalable and wouldn't be limited like a MCU in needing more processing power for future upgrades. A disadvantage of FPGAs is that they are less robust than MCUs. This is because they are more sensitive to environmental conditions that can make the system unstable like temperature fluctuation and electromagnetic interference. This means in addition to making room for the FPGA in ALiCS we would also make another enclosure to ensure the stability and proper working conditions for ALiCS. So, overall for FPGA it seems not very straightforward for new users and a very deep knowledge set is required because of the non-standardized nature of the hardware. This would make it extremely hard when designing hardware for our project. In addition, our members would be forced to learn a hardware level programming language they have barely worked in before and would need to use in a way that is nothing close compared to what they have worked on before in it. It would also cause a lot of trouble handling errors with all new skills that would need to be learned while trying to code in hardware level languages.

MCUs usually come with most of their features already on the chip. MCUs also are more oftenly made to be low power devices that could let ALiCS run longer. They are relatively easy to use compared to FPGAs and have a lot of documentation with online resources available from the manufacturers. MCU's have standardized architectures with loads of field data making them proven good tools time and time again. MCUs also come standard with communication interfaces like UART and I2C that we made the most of on our project and do not need to be added like for FPGAs. Most MCUs also come with dedicated IDE's that receive constant support and updates. These IDEs also make use of common high level programming languages like C, Python, and Java. This makes programming errors easier to solve for people who have worked with these programming languages before. Our team all having worked with higher level programming languages makes this a much safer choice for programming letting us use previous knowledge and tools to debug problems with the software.

With all this information in thought our group decides to go with MCUs over FPGAs. It really an easy decision considering all the extra learning that would need to be put in to learn FPGAs. Compared to working with MCUs where the barrier of entry is significantly lower and already being used to working MCUs. In addition we have already worked with high level programming languages throughout our time in the field. MCUs also have built in error handling and are usually more stable products with more robust features that provide more safety that everything will work correctly on our project. Overall, the choice of picking the software and programming language we use to code is too nice an advantage to have plus when selecting hardware just needs to be compatible without adding external parts.

We have also decided that we needed to make use of two MCUs, one that controls the program data for just about everything but the camera and one that controls the camera and sends the minimal amount of data to the main MCU to tell the rest of the system how to respond based on the camera data. So for the MCU controlling the data it will need a lot of storage for holding video data in a good resolution like 1080p. Then the second MCU will be the main brains of the operation taking the data from the robot arm, sensors, spectrometer, and the motors.

Table 3.1.12: MCU vs FPGA Comparison

Parameters	MCU	FPGA
Power Consumption	Low	High
Ease of use	High	Low
Processing power	High	High (could be higher that MCU)
Environmental Robustness	High	Low
Scalability	Low	High
Peripherals	High	Low
Standardization/ Documentation	High	Low
Cost	Medium	High

Microcontroller (MCU) Comparison

The Microcontroller, or MCU, for our project is single handedly the most important part of our project. The MCU needs to be a safe and stable part of our project that can control ALiCS without worry that it will break due to environmental conditions. Like stated also earlier in our hardware research section we made use of two MCUs one for the camera and the another for the rest of the project. Now we will discuss between the five options we decided on which MCUs we have used and why.

MSP-EXP430FR6989

MSP-EXP430FR6989 or MSP430 for short is the first option we discussed is a board made by texas instruments. It comes with a 16-bit RISC architecture and up to a 16MHz clock speed. The supply voltage for the MSP430 is from 1.8V to 3.6V and has multiple low power modes depending on what is needed to operate ALiCS with best power efficiency. It comes with a RTC and a crystal oscillating clock. It has a 16-channel 12-bit ADC. It comes with a comparator and five timers. It has direct memory access allowing you to grab memory straight from the registers. It has a 320-segment LCD controller that allows you to display information on the LCD display with useful things such as the battery level. It also comes with 128KB of nonvolatile FRAM for storage. It has a 5V bus from USB and a 3.3V rail from the VBUS in the eZ-FET domain. MSP430 also comes with 83GPIOs pins. Overall is a good choice, the only downside is a little low on processing power and doesn't have much memory.

ESP32 WROOM

The ESP32 WROOM or just ESP32 board is made for RESSIF and is made for network applications and comes with Wi-fi and bluetooth features. It has 4 MB of nonvolatile flash memory installed on the device. ESP32 uses less voltage than most MCUs with a minimum of 3V. The minimum current draw is 28mA and peaks at 379mA. Making it a very power efficient option to keep ALiCS running longer. It comes with multiple connection peripherals already on it such as UART, SPI, I2C, PWM, ADC, and DAC. These boards also come with 26 GPIOs pins. This board also comes in at a really low cost compared to every other board. Overall, this is a very good board being power efficient and being powerful enough to still have run all the programs we would need with more memory space.

ARDUINO MKR WAN 1310

The Arduino MKR WAN 1310 runs an ARM cortex-M0 processor running at 48 MHz. The board is powered by 5V by USB. It also has a circuit Operating voltage of 3.3V so it can run at a low power of about the same as an MSP430. This chip also comes with a UART, SPI, and I2C. This chip also came with an internal memory of 256 KB Flash Memory and 2 MB external Flash Memory. It also has 32 KB of SRAM. The clock speed is 32.768 kHz RTC. Overall this MCU doesn't come with much flexibility but has a lot of good functionality in a small package that is very power efficient.

Table 3.2.13: *MCU Comparison 1*

Microcontroller (For parts beside camera)	MSP430	ESP32 WROOM	Ardunio MKR WAN 1310
Processor	16-bit RISC architecture	two low-power Xtensa® 32-bit LX6 microprocessors	SAMD21 Cortex®-M0+ 32-bit low power ARM

Clock Speed	16 MHz	80 MHz to 240 MHz	48 MHz
Voltage (V)	1.8 - 3.6	3.6	3.3
Current Draw (A)	0.2	0.379	0.154
Power Consumption (W)	1	1.3644	0.5082
Memory	128 KB	4 MB	32 KB
RAM	2 KB	520 KB	32 KB
Wi-Fi Features	No	Yes	No
Bluetooth	No	Yes	No
UART	Yes	Yes	Yes
I2C	Yes	Yes	Yes
SPI	Yes	Yes	Yes
ADC	Yes	Yes	Yes
DAC	Yes	Yes	Yes
Operating Temperature (C)	-40 to 85	–40 to 85	-40 to 85
GPIO Pins	83 Pins	26 Pins	28 Pins

Cost	Low (around $20)	Low (around $9)	Medium (around $40)

Raspberry Pi 4 Model B

The Raspberry Pi 4 is probably the most powerful option on the list while also providing the most flexibility. The Raspberry Pi comes with a quad core 64-bit A72 processor running at 1.5GHz. It comes with 4GB of LPDDR4 RAM. It also comes with a built-in hardware decoder called H.256 or HEVC that can do 4k 60fps. Then it has a second one with H.264 that can do 1080p 60fps. It comes with a built- in video chip that supports 3D graphics and has dual hdmi outputs that can display up to 4K 60fps. Then it has multiple upgraded interfaces from the last choices that include 802.11 b/g/n/ac Wireless LAN and Bluetooth 5.0 with BLE. It has 1 SD card for video storage from the camera port that it has. Its direct port is connected to a camera with 2-lane MIPI CSI which makes it a very good choice for the second MCU that we need for our project. It comes with two USB2 ports and two USB3 ports. Then there are 28 GPIOs interfacing pins with multiple interfacing options with, up to 6x UART, up to 6x I2C. Up to 5x SPI. 1x SDIO interface, 1x DPI (Parallel RGB Display), 1x PCM, Up to 2x PWM channels, Up to 3x GPCLK outputs. Overall this makes it a great choice for the second MCU being flexible and being able to take and receive camera data easily will work well and transmit the data to another MCU.

Raspberry Pi Zero 2 W

The Raspberry Pi Zero 2 W is a smaller more cost effective version of the Raspberry Pi 4. However it comes at a lot of costs of the flexibility of the MCU and a less powerful chip. The processor for the Raspberry Pi Zero 2 is a 64-bit Arm Cortex-A53 clocked at 1 GHz. The MCU comes with a microSD card slot and VSI-2 camera connector. Making this another viable option for connecting to the camera with an microSD card to hold all the camera data we need for our program. It also comes with a HAT-compatible 40-pin GPIO header. It has video output through a mini HDMI port. It is also powered through a micro usb port. The operating temperature of the MCU is -20C to 70C.

Rock Pi 4 SE

The Rock Pi 4 SE is an alternative for the Raspberry Pi 4 with very comparable parts. It uses the Arm big.LITTLE technology that has a dual Cortex‑A72 with a clock frequency of 1.5 GHz and a quad Cortex®‑A53 with a clock frequency of 1.0 GHz. It hosts the Arm Mali T860MP4 GPU for a display chip letting it make use of a hdmi port to display up to

4k 60fps. It uses the same two hardware decoder as the Raspberry Pi 4 H.265/VP9 (HEVC) hardware decoder that can up to do videos up to 4K 60fps and H.264 hardware decoder that can up to do videos up to 1080p 60fps. It also uses a power input of USB C. It comes with a 3.5mm audio jack with a mic. It has two USB2 ports and one USB3 port. It has the same type of camera port as both Raspberry Pi with 2-lane MIPI CSI. It also has 40 interfacing pins that consist of 2 x UART, 2 x SPI bus, 2 x I2C bus – 1 x PCM/I2S, 1 x SPDIF, 1 x PWM, 1 x ADC, 6 x GPIO, 2 x 5V DC power in, 2 x 3.3V power pin.

Table 3.2.14: *MCU Comparison 2*

Microcontroller (camera)	**Raspberry Pi 4 Model B**	**Raspberry Pi Zero 2 W**	**Rock Pi 4 SE**
Processor	Quad core 64-bit A72 Processor	64-bit Arm Cortex-A53	Dual cortex-A72 and quad Cortex-A53
Clock speed	1.5 GHz	1 GHz	1.5 GHz (A72) and 1.0 GHz (A53)
Voltage (V)	5	5	9 or 12
Current Draw (A)	3	2.5	2
Power Consumption (W)	3 to 5	1	2.7 to 5
Memory	4 GB	512 MB	4 GB
RAM	4 GB	512 MB	4 GB
Wi-Fi Features	Yes	Yes	Yes
Bluetooth	Yes	Yes	Yes

UART	Yes	Yes	Yes
I2C	Yes	Yes	Yes
SPI	Yes	Yes	Yes
ADC	Yes	Yes	Yes
DAC	Yes	Yes	Yes
Operating Temperature (C)	0 to 50	-20 to 70	0 to 50
GPIO Pins	40	40	40
Storage	1 SD card, 2 USB 2.0, and 2 USB 3.0	microSD	USB-C, 2 USB 2.0, and 1 USB 3.0
Cost	Moderate (around $30-$60)	Low (around $15)	High (around $50-$70)

Signal Filters

Signal filters are used to receive clear data void of noise or other interface with a data signal from an analog input signal. We need to make use of these when designing our PCB to make sure we receive clear data from our ultrasonic sensors. To make sure our data from these sensors is accurate and correct.

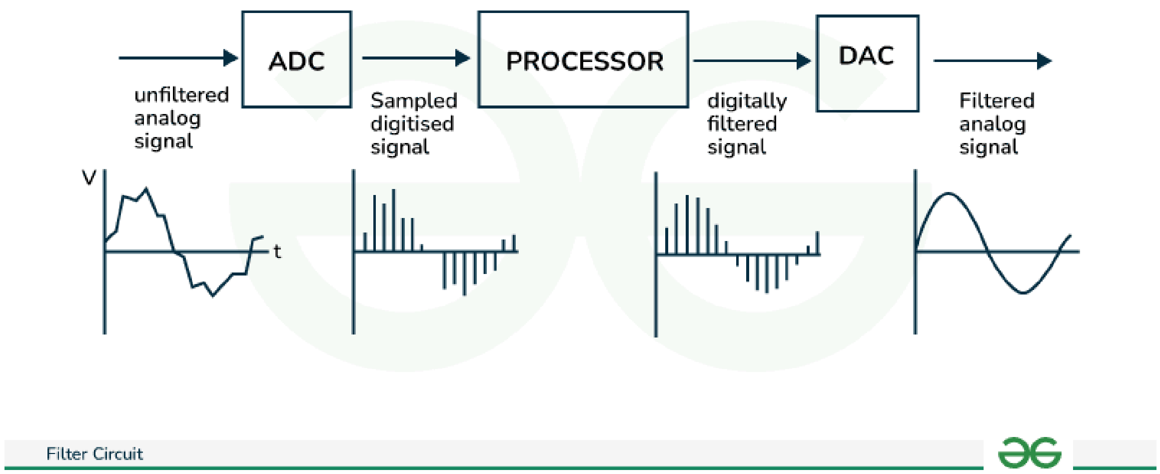


Figure 3.2.3: *Digital filter process*

The figure above shows every step simply defined for using a digital filter. This first takes the analog signal from the sensors, which in our case would be our ultrasonic sensors convert the data to a digitized signal through an ADC. This sends this data through our processor to filter the data for the correct information to reproduce the signal through a digitally filtered signal. This signal then goes through a DAC or digital to analog converter to convert the signal back with the correct information without noise or interference. Then we can use this correct information in our software to correctly position ALiCS.

Robotic Arm

Our method of choice for picking up the trash and debris, is to incorporate a precision, industrial robotic arm from Mecademic Industrial Robotics. The Meca500 is not the most cost efficient arm for the job however, we chose to implement this equipment as we have been granted access from one of the member's employers. The robot connects via ethernet cable, and acts as a slave component, making it unique in its industry, "And unlike most other industrial robots—which are usually complex stand-alone systems—the Meca500 is a plug-and-work automation component. Our robot is a slave component rather than a master, which makes it very easy to integrate via any computer or PLC" (Mecademic, 2023). The Meca500 is an extremely precise and compact device that will really maximize the performance and potential of ALiCS. Its high precision, with a position repeatability of 0.005 mm, ensures that the robotic arm can accurately and reliably pick up even the smallest pieces of trash. This level of precision is essential for the effective operation of ALiCS, as it needs to handle various types of debris with varying shapes and sizes found on the beaches.

We compared using the robotic arm versus sifting technique used in the commercial product available today. We determined our approach went a step above in terms of collecting and sorting the items collected from the environment. Also, because we avoid sifting through the sand, we are actively avoiding accidentally collecting sea-turtle nestings. Which are highly protected due to their endangered species status.

Robot Arm Attachment

The robot will need an attachment with it to connect to the top of the arm that can pick up the trash for sorting. This means we are looking for a means that can pick up the trash by grabbing minimal amounts of sand or other debris. This attachment also can either be made by the same company that makes whichever arm we choose or could be made to fit by 3d printing parts.

Robot Arm dust pan

A dust pan for the robot attachment would mean the robot arm would have to scoop up the trash and would most likely be limited by the range of motion of the size of the dustpan to go above the ram and back to the pan. This dust pan would also have holes to let the sand fall through while still being able to keep the trash in the pan. The only problem with this is we can be precise with where the trash will end up in the pan giving opportunity for the pan to either scan sand left in the pan or the actual dust pan itself leading to errors in the sorting process that would hurt the precision of our product over time.

Robot Arm Comparison

Meca500

The compact size of the Meca500, with a reach of 330 mm and a weight of 4.3 kg, allows it to be easily mounted and maneuvered within the confined spaces of the beach cleanup robot. It is mountable in any direction, and is energy efficient, as it consumes around 30W. The robotic arm’s compactness does not compromise its performance; instead, it enables the robot to operate efficiently in the dynamic and often uneven beach environment. The creators of the Meca500, Mecademic, have their own API for programming the robotic arm which is a nice touch in simplifying its implementation. The Meca500 is very expensive and is usually used for lab automation which requires very precise and delicate operations, thus for ALiCS, it is slightly overkill and the other options might be more viable in a cost efficient situation. We are ecstatic to work with such high quality technology and expect great results.

MechArm

An alternative solution, which is considerably pricey, ~$1200, is the MechArm compact robot arm. This arm has similar payload and movement, however it is slightly smaller and limited in range.

So while we might be killing mosquitoes with a gun using the Meca500, the MechArm from Elephant Robotics, is another suitable choice for the robotic arm on ALiCS. This arm would be our most affordable as well but comes with a few drawbacks. The MechArm is designed to be lightweight and versatile, making it easy to integrate into the robot's framework. Its reach of 270 mm provides a slightly larger working range than the Meca500, which can be beneficial in accessing debris spread over a wider area. The MechArm has a payload capacity of 0.25 kg, which is lower than the other two counterparts we looked at. However for lighter loads this could be sufficient as beach litter is usually under 1kg. Two huge benefits with the MechArm is that it has a built-in RaspberryPi which can ease development, and is much more lightweight which can lower its burden on the rover, improving mobility and maneuverability.

UFactory xArm5

The third option which could be used for a much larger rover is Ufactory's xArm5. With a much larger reach at 700mm, this arm can grab from much farther to reach places and before the rover gets too close to the litter. It also boasts a payload of 3kg which is much larger than the other two robots, but finding debris this heavy would be very rare and basically overkill. These specs also come with a very heavy robotic arm at 11.2kg, almost triple the weight of the Meca500. The xArm5 only has 5 degrees of freedom unlike its counterparts. A part this heavy would cause a lot of problems if our rover is not also much larger than we anticipate. Therefore while this arm seems great and also has raspberry pi compatibility, it is too large for our project.

Below describes all the relevant specifications we can take into consideration, to determine the best suitable choice for ALiCS. The relevant specs we want to focus on are the reach, weight, and position repeatability. They all have compatibility with RaspberryPi so that is not an issue when choosing either option. For control software, the Meca500 has an advantage as it has its own unique API which can simplify programming and won't be as ambiguous when bugs arise as when open-source is used. This is yet another attractive feature the Meca500 has over the others. Although still heavier than the MechArm, the Meca500 used less power which can ease the strict requirements of our power supply and save us money in that aspect. It has the perfect middle ground on reach and payload, not too small and not too large. The xArm 5 is eliminated due to its larger size as our rover will not be able to support that much weight on one component. 11 kilograms is too much and while the reach and payload will be nice, it is not ideal. Another main factor that is taken into consideration is of course, price. Realistically, if we were choosing to purchase any of these three robot arms, they would end up being too pricey to be within our budget. Thankfully, we were able to get the Meca500 loaned to us. While all great options, Mecademic's Meca500 ease of use, reach, compact size and state of the art accuracy and positioning, deem it the winner.

Table 3.2.15: *Comparison of Robotic Arms*

Specifications	Meca500	MechArm	UFactory xArm 5
Payload (kg)	0.5 (1 max.)	0.25	3
Reach (mm)	330	270	700
Position Repeatability (mm)	0.005	0.5	0.1
Weight (kg)	4.3	1	11.2
Degrees of Freedom (DOF)	6	6	5
Max Speed (mm/s)	1000	500	1000
Power Requirements (V)	24	12	24
Power Consumption (W)	30	60	200
Temperature Range (°C)	0 ~ 40	0 ~ 50	0 ~ 50
Material	Aluminum Alloy	Plastic	Carbon Fiber
Control Software	Mecademic API	Open-source	UFactory Studio

Communication Interface	Ethernet	USB,UART	Ethernet, USB
Raspberry Pi Compatibility	Yes	Yes	Yes
Cost ($)	$$$	$	$$

Robot Arm Claw

A claw for the robot attachment would make it to where the robot arm would need precision accuracy to acquire the trash. It would also need really high friction grips on the claw to stop slipping of slippy materials from the claw. However, since the claw is so precise there would be little to no chance of mishaps with the spectrometer meaning our sorting would correct more than 99% of the time. The code for the claw would also be another layer making it harder to make use. This also has an add on of power consumption even if little it still makes a difference over time to our battery.

Table 3.2.16: *Robotic Arm Comparison*

Parameters	Claw	Dust Pan
Precision	High	Medium
Ease of use	Low	High
Power consumption	Medium	None
Stability (chance of dropping trash)	Low	Medium
Cost	Medium to High	Low

Robot Arm Claw Comparison

For our claw we need to be secure and as close to frictionless as possible. The claw will need to be big enough to hold coke cans and glass bottles. This claw will not worry about whether it fits to the slot on our robot arm or not because we can always 3D print a connection piece later. The claw importance is not to be underestimated because we need it to keep our sorting on the spectrometer accurate and to be able to easily part waste or trash in the correct bin without having to worry about hitting other parts in our design.

Swaytail Professional Metal Robot Gripper

One of the first options I found was the Swaytail metal robot gripper. This was a very cost effective choice. It requires a very low input voltage of at most 7.2V. It weighs very little for a robot claw at 95g and can hold up to 500g. It can open up to 6.7 inches which is way over what we need for the trash we should be picking up on the beach.

LewanSoul Mechanical Robot Claw

The LewanSoul mechanical robot claw is another good option that happens to cost a good amount while still not being too expensive. It also has a lower input voltage at 8.4V maximum. It can open its claws to a size of 7.62 inches. It weighs 207g still keeping on the lighter side of claws. It is able to hold up to 700g of weight and would easily work for all the tasks we need from it.

RobotIQ 2F-85

The RobotIQ 2F-85 is a good option that happens to cost a lot more than every other option. It weighs 900g making it a very heavy option compared to others. This option has an extremely high input voltage at 24V. However with these costs comes great performance from the claw. This claw is able to hold items in two different ways. One way is the form-fit grip where it surrounds the object and fully encloses so it doesn't move and the other way is a friction grasp where it holds both sides with extreme pressure coming front the two pads on the end. We will likely make use of the form-fit grasp if we picked it because it has a way more secure hold on the trash. This also has a max payload hold of 5000g meaning it crushes the other options by far. One downside is that the open range is only 3.35 inches long which means it is a much smaller range than the other options. However, in our case this is fine because nothing bigger than that range is small enough to fit into ALiCS.

Table 3.2.17: *Claw Selection*

Specifications	RobotIQ 2F-85	LewanSoul Mechanical Robot Claw	Swaytail Professional Metal Robot Gripper
Payload (g)	5000	700	500

Open range (inches)	3.35	7.62	6.7
Weight (g)	900	207	95
Power Requirements (V)	24	6-8.4	4.8-7.2
Cost ($)	$$$	$$	$

Spectrometer

An optical spectrometer is a scientific instrument initially designed to separate a beam of light into its component colors, forming what is known as a spectrum. Historically, spectrometers were large devices primarily used in chemical laboratories for analyzing large samples based on their color and chemical composition. Over time, diffraction gratings have largely replaced glass prisms in spectrometers due to their superior resolving power.

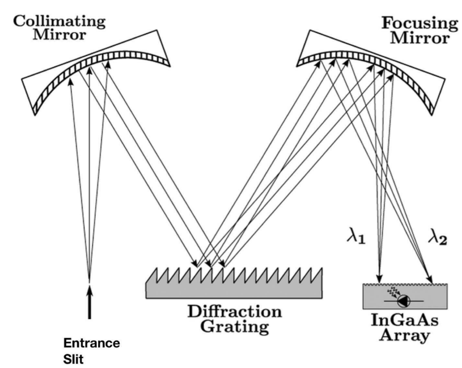


Figure 3.2.4: *Spectrometer Drawing*

A modern spectrometer typically consists of an optical system that includes a grating and an image sensor. In a typical monochromator, an exit slit is positioned perpendicular to the axis of the focusing lens. Mini spectrometers, on the other hand, are compact polychromators, where the image sensor is oriented perpendicular to the axis of the focusing mirror. Optical spectrometers measure light intensity in relation to frequency or wavelength. Spectroscopy, meanwhile, explores the electromagnetic spectrum, studying how light wavelengths are influenced by atomic and molecular interactions.

Over the past two decades, optical fiber spectrometers have evolved significantly, becoming essential tools for today's spectroscopists. These spectrometers function by taking in light, dispersing it into its spectral components, digitizing these colors into wavelengths, and then analyzing and displaying the data on a computer. An optical fiber cable serves as the input to the spectrometer, known as the input slit. In most spectrometers, the light is collimated onto the grating surface, which disperses the light into different wavelengths at specific diffraction angles. The dispersed light is then focused by a focusing mirror onto the image sensor for further analysis.

The spectrometer will determine whether the item collected is recyclable, trash, or possibly a seashell. The spectrometer will measure across the Near Infrared (NIR) spectrum, 900-1600 nm, for analysis of the material. Measuring in this spectrum will give accurate results in real time.

Researching spectrometers involves understanding various types, applications, and specifications relevant to scientific or industrial needs. Spectrometers are instruments that analyze the spectral content of light or other electromagnetic radiation. There are a few approaches or types of spectroscopy like absorption, emission, fluorescence, raman, and laser-induced breakdown spectroscopy. There are various methods through which a spectrometer can collect light.

- In absorption spectroscopy, light passes through a sample, and the reduction in intensity as it is absorbed reveals specific wavelengths absorbed by the material, aiding in identifying its unique composition.
- Fluorescence spectroscopy, a type of emission spectroscopy, involves illuminating a sample with high-energy light to induce fluorescence, wherein the emitted light is analyzed by a spectrometer to identify specific compounds.
- Reflectance spectroscopy is the study of light as a function of wavelength that has been reflected from a solid, liquid, or gas. Reflectance is light incident on the surface of a material that is reflected at an interface. Smooth surfaces have high specular reflectance, where incident light reflects in the same direction. Rough or matte surfaces have a diffuse reflectance, where incident light is scattered in all directions.
- Emission spectroscopy encompasses the collection of light emitted from a sample after excitation by an external source such as a laser or electrical discharge. For instance, atomic emission spectroscopy and laser-induced breakdown spectroscopy (LIBS) fall under this category.
- Raman spectroscopy, meanwhile, involves collecting light scattered after interacting with a sample through inelastic scattering, useful in chemistry for studying molecular and intramolecular bonds.

Czerny-Turner Spectrometer

The most common spectrometer found in the market is based on Czerny-Turner design. A conventional Czerny-Turner spectrometer achieves high light throughput by dispersing light using a robust first-order diffraction grating. We have opted to design a Czerny-Turner spectrometer due to its widespread use and cost-effectiveness in applications. Such spectrometers can be customized by selecting components that align with project requirements and budgetary considerations.

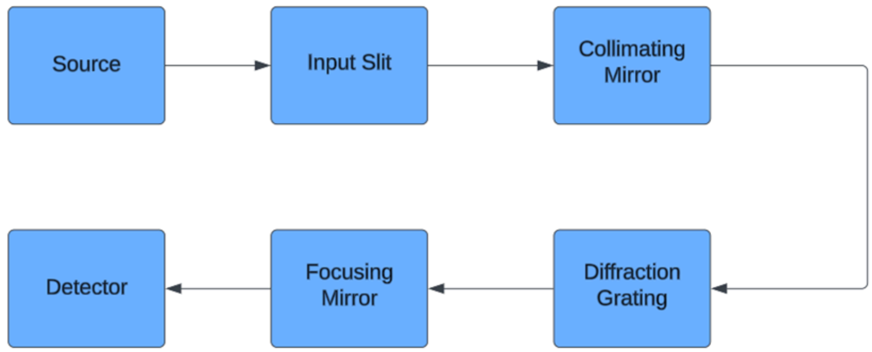


Figure 3.2.5: *Block Diagram of Czerny Turner Spectrometer*

The main advantage of the spectrometer is to provide high sensitivity and low stray light. This kind of spectrometer uses a detector array instead of a single detector and therefore only needs fixed components. Figure 3.1.2 Shows the general block diagram of a Czerny-Turner spectrometer and is explained in detail. Functions of major components utilized in spectrometers are mentioned below.

Entrance Slit: Entrance slit is also known as input slit. This slit is an aperture through which the beam to be measured is guided. Aperture Size has relevant effects on optical uniqueness such as throughput and spectral resolution. There are two light input methods which are spatial light and optical fiber cable.

Collimating mirror: The word collimate means parallel, here the input light which comes from laser or input slit with a particular aperture size, makes that beam parallel when it falls on collimating mirror.

Grating: The grating part of the incident light splits into different wavelengths through collimating lens and allows light to pass through each wavelength or reflect away that light at different diffraction angles.

Focusing mirror: The focusing mirror forms an image of light dispersed into wavelengths by the grating, against the linearly arranged pixels of the image sensor according to its wavelength.

Image sensor: The image sensor converts electrical signals into optical signals, which are focused by focusing mirror and detached into different wavelengths by grating and finally the output is shown on a spectrometer.

Echelle Spectrometer

An Echelle spectrograph utilizes two dispersion stages to spread light in perpendicular directions referred to as the X and Y directions. The term "Echelle," meaning ladder in French, aptly describes how spectral orders resemble the rungs of a ladder due to dispersion along both vertical and horizontal axes. Consequently, spectral data forms a 2D pattern and is ideally captured using a 2D detector such as an imaging CCD camera.

Most Echelle spectrographs are designed without moving parts, enhancing robustness and facilitating integration into portable and rugged environments. These systems can be configured for easy calibration and rapid analysis, making them well-suited for applications requiring in-the-field operation. One of the biggest strengths of the Echelle design is that it can provide a high resolution and large bandpass simultaneously. The resolution in Echelle systems is defined by a CSR = Wavelength/(Resolution at Wavelength)

The bandpass provided by Echelle systems is mainly dependent on the size of the Echelle grating and also the camera onto which the Echelle lines are focused. Typically, the maximum bandwidth that these systems give ranges from 200 nm to 950 nm in a single shot. Depending on the design, there are other Echelle designs that provide a smaller coverage, i.e., from 522 to 1076 nm (Compact Echelle from Andor Technology designed for Raman spectroscopy) or from 200 nm to 600 nm etc. The CSRs too are available in different configurations depending on the model used, and it can vary from 2500 (low resolution) to 40000 (very high resolution). Echelle systems can be made very compact and are typically designed without any moving parts and that makes them very suitable for field-based applications. These systems are known to have been deployed on-line processes to identify alloy concentrations in metal furnace hearths and have also been used in flight to capture trace element information of exhaust gas from jet engines.

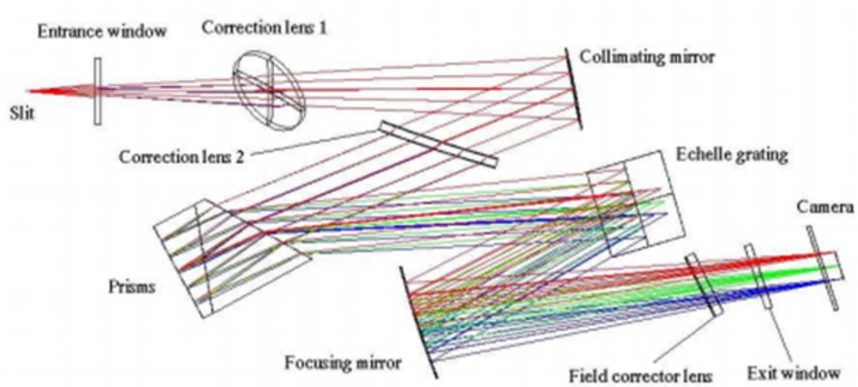


Figure 3.2.6: *Optical Layout of Echelle Spectrometer*

Czerny-Turner vs. Echelle Spectrometer

Echelle spectrographs are similar in some respects to Czerny Turner spectrographs, but they do have two dispersive components which can be either two gratings or two prisms or a combination. The two dispersive elements are arranged to disperse the light in two perpendicular or orthogonal directions. Consequently, echelle spectrographs can be designed to give simultaneously high spectral resolution and large spectral bandpass. Our group chose to implement and design the Czerny-Turner Spectrometer for our application. While the Echelle spectrometer has high spectral resolution, the two prisms can be hard to acquire when considering cost and lead times. Also, when considering optical alignment, Echelle spectrometers can be difficult to align and could be susceptible to falling out of alignment when in the field. While the high spectral resolution would be beneficial, it doesn't outweigh the possibility of failing in the field or increasing lead times and cost.

Near Infrared Spectroscopy

Near Infrared, or NIR, spectroscopy is a method that utilizes the near-infrared region of the electromagnetic spectrum (from about 700 to 2500 nm). By measuring light scattered or reflected off of or through a sample. NIR reflectance can be used to quickly determine a material's properties without altering the sample. This highly flexible form of material analysis, which can be applied to a broad range of research and industrial process applications, has been a cost effective tool in remote sensing technology.

LiDAR Comparison

To develop a robust robotic system capable of operating in dynamic environments, integrating dynamic sensing is essential to map and transmit information effectively to the central logic unit. This processing requires rapid sampling and processing, which is why we have chosen an embedded sensor solution for this purpose. LiDARs are widely used in autonomous vehicle applications due to their ability to effectively process dynamic environments.

For this module, achieving a 360-degree omnidirectional scan capability with a high sampling rate is crucial. Additionally, it is essential that the scanning range extends up to 6 feet to accommodate environmental variability caused by user interaction or moving objects. After thorough research and consideration, a SLAM LiDAR solution is deemed optimal for our requirements.

Slamtec RPLiDAR

The Slamtec RPLiDAR A1M8 is a cost effective, 360 degree 2D LiDAR scanner, with plug and play capabilities and range radius up to 12m. This LiDAR scanner can be used for ROS educational robots, open source hardware, UAV mapping and obstacle avoidance, synchronous positioning and navigation. RPLiDAR is basically a laser triangulation measurement system that can work in indoor or outdoor applications. This scanner will be used mainly for navigation and obstacle avoidance.

Yahboom EAI X3

The Yahboom EAI X3 is another cost-effective solution, however from information gathered online, reviewers have said this system does not perform well in outdoor applications. The laser triangulation method mainly uses a laser beam to map its environment, the laser is reflected and scattered on the surface of the target, and the reflected laser is converged and imaged by a lens at another angle, and the spot is imaged on a Charge-Coupled Device or CCD, position sensor.

Inno Maker LD06

The Inno Maker LiDAR LD06 is a time-of-flight laser ranging device that can perform 360° scan, measuring visual point cloud information, which can be widely used in map construction (SLAM), robot positioning and navigation applications, as well as intelligent equipment obstacle avoidance. Its brushless motor, high-speed wireless data transmission, and service life can reach 20,000 hours. The long detection range, with 45mm range resolution, can sense the environmental information which ensures the large scanning and mapping ability of the robot.

Table 3.2.18: *LiDAR Comparison*

LiDAR Sensor	Slamtec RPLiDAR	Yahboom EAI X3	Inno Maker LD06
Scan Range	12m	8m	12m
Sampling Frequency	8000 Hz	3000 Hz	4500 Hz
Scan Rate	Up to 10 Hz	Up to 10 Hz	Up to 13 Hz
Distance Resolution	≤ 0.5mm	≤ 0.5mm	≤ 0.5mm
Wavelength	785nm	775nm - 800nm	780nm - 800nm
Cost	~$100	~$100	~$100

Camera Comparison

When choosing a camera for this project, we selected the EMEET Webcam C960 due to its exceptional compatibility and versatility with the Raspberry Pi board. The EMEET C960 integrates seamlessly with the USB, eliminating complex setup requirements and optimizing performance. It offers a full HD 1080p resolution, which is crucial for accurately detecting trash and debris on the beach. Additionally, its built-in light correction and wide-angle lens make it highly adaptable to varying lighting and distance conditions in outdoor environments. The camera's configuration capabilities using Python or other programming languages make it easy to align with ALiCS's software framework. Its compact and lightweight design minimizes the rover's bulk, and its affordability ensures cost-effective scalability if additional units are required.

We initially considered the Raspberry Pi Camera, which seemed like an ideal choice due to its direct compatibility with the Raspberry Pi's camera port and its compact design. The camera offers high resolution and could theoretically integrate well with our system. However, we encountered significant issues when connecting it via the serial port, which caused delays and limited functionality. These connectivity problems made it unsuitable for the real-time requirements of ALiCS.

Another option we explored was the Arducam 8MP IMX219, a popular choice for projects involving the Raspberry Pi. This camera provides high resolution and a compact form factor, making it suitable for lightweight designs. However, it comes with a fixed focus, which is less than ideal for dynamic outdoor conditions like a beach environment, where objects at varying distances need to be detected. Additionally, it is relatively expensive, making it less attractive from a cost perspective.

Ultimately, the EMEET Webcam C960 emerged as the best choice for ALiCS. Its plug-and-play setup bypasses the connectivity issues associated with the Raspberry Pi Camera, while its high resolution, light correction, and affordability provide a reliable and efficient solution for our autonomous beach-cleaning robot. It balances performance, compatibility, and cost, making it an excellent fit for our project's needs.

Table 3.2.19: *Camera comparison*

Specifications	Raspberry Pi Camera	EMEET Webcam c960	Arducam 8MP Camera
Resolution	8MP	1080p (2MP)	8MP
Focus Type	Fixed	Autofocus	Fixed
Field View	62.2 degrees	70 degrees	76 degrees
Connection Interface	CSI	USB	CSI/USB
Cost	$	$	$$$

Suitability with ALiCS	Cost effective, good for fixed-range detection, easy set up	Easy set up, good for versatility	High Resolution

3.3 Software Research

Framework

In order for the robot to function effectively, it's important for it to be able to handle multiple tasks. Several of these tasks must be able to work in real-time to ensure the accuracy and performance of the robot. A good majority of this code will use open-source libraries to manipulate sensor data because the idea of switching between tasks to achieve the responsiveness for the amount of code that needs to be used is simply not the most proficient way. We have used a programming framework to test and implement different components together.

Yarp

Yarp (Yet Another Robot Platform) is a software framework for developing robot applications. It's a set of tools that will help communicate and integrate between the software modules. It allows developers to build complex systems by connecting components together that can communicate and work seamlessly well together. YARP is free to use and is written in C++; it can be compatible with other languages as well but it requires a compatibility layer. YARP is open source, which would be helpful to figure out any kind of problems we would have with our implementations, the reason why is because we would be able to look at how the code works (or try to understand how it works) and use it to figure out what errors are occurring. The main overall issue with YARP, is that although it is open source, it is something that is not widely used, so if we were to end up with a problem that has not happened to any of the other users, there will be no available solution online to our problem. If there were issues to occur when using YARP, there is information on its website that is able to talk about how to set it up, the basics of using it, as well as dealing with devices. There is a forum where we are able to submit any issues, so that could be helpful later on. This kind of information would be very important to our group as all of us have little to none experience with YARP.

ZeroMQ

Another framework is ZeroMQ which is an asynchronous general-purpose messaging library. It supports multiple communicating patterns making it easy to implement messaging scenarios. It is essentially able to use patterns like push-pull and client-server to create multi-threaded applications. ZeroMQ is designed to be fast and lightweight with reliable performance. It can establish communication between processes, locally and the internet. ZeroMQ is written in C++ but supports other languages as well. While ZeroMQ is a general-purpose library, it would take some extra time to learn and implement it for the robot. The main issue of ZeroMQ is that it is obviously not designed for robotic use and that would mean that in order for use to be able to use ZeroMQ, we would essentially need to spend more time researching and possibly looking for other software to help with ZeroMQ.

ROS

ROS (Robot Operating System) is another framework that is designed for robotic applications. Like the previous software frameworks, ROS provides set libraries and tools to help build the robot. ROS is open-source, giving us the flexibility to decide how to use ROS as well as the freedom to cater and customize our robot. ROS works well with C, C++, and Python but can also integrate other softwares into the equation. It is compatible for multiple platforms such as macOS, Windows, and Linux. ROS is specifically designed for robots and is improved by and for developers who contribute to robotic software.

Comparing all three of these frameworks, each one has their pros and cons. All of these provide a library that can support different programming languages and is free to use. Although ZeroMQ is a popular framework, it wouldn't be an option due to the fact that it simply wasn't built for robots alone and would require extra time to accurately apply it. Due to a tight schedule, that would push the project back. YARP and ROS are similar in many ways, however, YARP seems to be more limited in the libraries, sources, and overhead. ROS is specifically designed to build a robot fast and its install is more robust compared to the other two frameworks. For this, ROS would be the best framework to work with.

Table 3.3.1: *Robotic Framework Comparison*

Feature	**ROS**	**YARP**	**ZeroMQ**
Primary Use	Robotic middleware strictly for robot software development	Robotics middleware–modality	High-performance messaging library–general use
Data Handling	Supports complex data through topics	Efficient with data streams and interfaces	Lightweight and fast for raw data transfer
Compatibility	Extensive support for various robot hardware	Good for when ROS is not supported	Works across different platforms
Real Time	Support is limited	Better real-time	Not real-time, but

Capability		capabilities	fast
Scalability	Highly scalable with support for large systems	Scalable with focus on modularity	Highly scalable
Learning Curve	Depends on complexity	Moderate	Relatively easy
Flexibility	Highly flexible with many tools and plugins	Highly modular–customization	Very flexible
Use Case in ALiCS	Ideal for complex robotic integration	Useful for modular, distributed systems	Best for lightweight, high-speed data

Mapping and Object Detection

SLAM

A huge part of the robot is knowing where it is in its own environment. ALiCS is specifically designed to map out its location and detect obstacles whether that is landmarks or trash. The robot would need to take in data from the camera and continuously update the map providing the robot with real-time data. Besides knowing its location, the robot must detect what is trash compared to what isn't trash. More specifically, the robot will identify whether the piece of trash is glass, aluminum, or plastic. We can use the camera sensors and LiDAR for mapping and object detection.
Slam is a method used for robots to map out locations to localize itself. With the improvements of computer processing speeds and lower-cost sensors, SLAM is being used in a more practical way. SLAM consists of two parts: the front-end processing and the graph optimization which includes the back-end optimization.

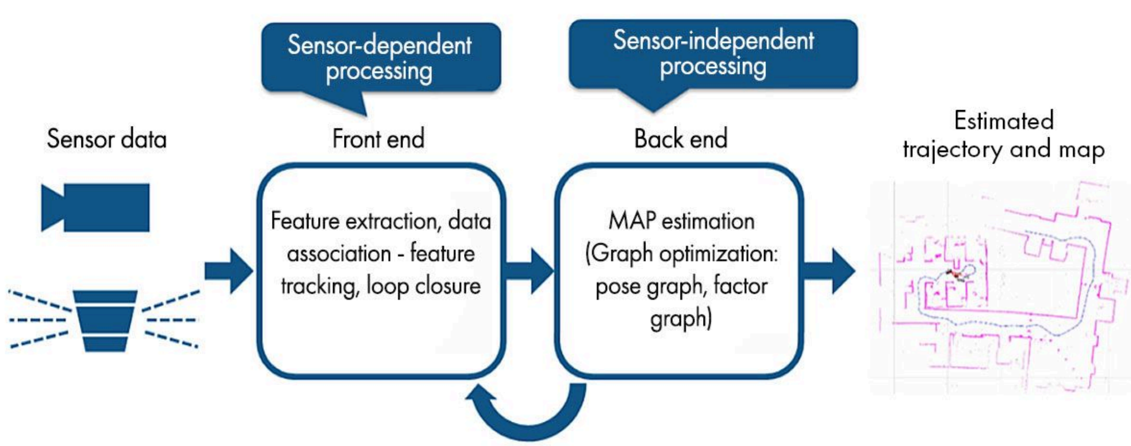


Figure 3.3.1: *SLAM Processing Flow*

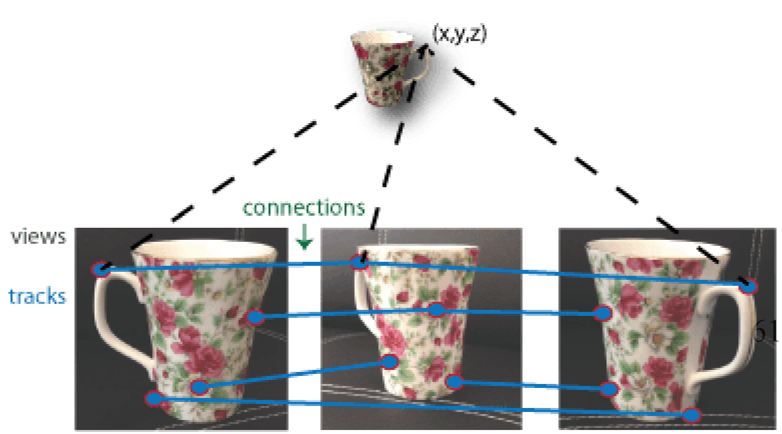
How SLAM works is using the sensors, in this case the camera, to scan the room and collect data about the environment. The data is then processed to identify objects or landmarks and the robot will then use the map to determine its own location. The robot will compare its current data to the map and estimate its own position. As the robot roams around, the robot is constantly taking in new data and updating the map.

There are different types of SLAM methods, but for this project, we implemented the lidar SLAM. The lasers provide precise distance measurements and work well in mapping with SLAM algorithms. We can use SLAM with either 2-D or 3-D LiDAR. There are a few different ways to estimate the relative transformation of point clouds such as iterative closest point (ICP) and normal distribution transform (NDT). Using lidar SLAM in ALiCS, the robot will be able to make a map to localize itself and detect objects and landmarks to avoid running into them. SLAM can also be used to remember where trash is located on the beach. While the system is continuously providing real-time updates, this prevents ALiCS from falling over or getting trapped. With SLAM, ALiCS can accurately position the robot arm and effectively pick up the trash.

Structure from Motion (SfM)

Structure from Motion is another technique we can use in creating 3D models. Unlike SLAM which is used for real-time mapping and localization, SfM is usually used for offline processing to generate structures of the surrounding environment. How Structure from Motion works, is that it starts capturing multiple overlapping images of a scene or object from various points of view. In our case, the camera on ALiCS can take continuous snapshots as the rover traverses the environment. During this process, key features are detected in each image using algorithms such as SIFT (Scale-Invariant Feature Transform). However, to find corresponding points we can also track points rather than match features, using feature tracking techniques such as the Kanade-Lucas-Tomasi algorithm, which works well when the cameras are close together, but falls apart as they move farther apart. These detected features are then matched across the image to find correlations. After the features are matched, SfM can estimate the camera's position and orientation for each image. This involves solving the perspective-n-point (PnP) problem to determine how the camera moved between the shots.

Once the camera poses are known, 3D coordinates of the matched features are triangulated to create a 3D point cloud defining the scene. This point cloud can be refined into a proper 3D model where the space in between the sparse points are interpolated and results in a detailed 3d structure. SfM can be used in ALiCS to create



detailed 3D models of the surrounding beach environment, aiding in planning and analysis. Thi can be useful for creating detailed maps that SLAM may not be able to generate in real-time.

Figure 3.3.2: *Structure from Motion Process*

QGIS 3

A third geographical mapping program we can look at is QGIS 3. QGIS is an open-source Geographical Information System allowing users to process and analyze data and create maps. QGIS is known for being one of the most popular free GIS software available today and has the newest update which offers a more modern and user-friendly user interface. This platform supports a variety of data which includes thematic, choropleth, and heat maps. It includes powerful tools for editing geospatial data in various formats and conducting spatial analysis. The functionality of QGIS is expandable using plugins and works on various operating platforms, including Windows, macOS, and Linux. The newest version of QGIS includes native support for 3D terrain and data visualization and outputs high-quality maps. QGIS has a strong and robust community of developers and users who actively contribute to the development of this platform. However there are a few cons to GCIS. Despite the interface being user-friendly, there are multiple tools and features that QGIS has which could take a little time learning. With a lot of documentation available, the quality could be inconsistent and for larger data or operations, QGIS could be slower and less efficient.

The table below shows the comparison between the three mapping interfaces. It is clear that for this specific project, SLAM would be the best choice to use for mapping. SLAM's capabilities of integrating multiple sensor data and its ability to operate in real time makes it a top contender for ALiCS. SfM requires static images to build a 3D model which is not ideal in this project. SLAM's real-data processing and adaptability support the core functionalities required for ALiCS–that includes obstacle avoidance, route planning, and area mapping. QGIS does not offer real-time processing which is crucial when ALiCS is moving autonomously across the beach. Choosing SLAM aligns with the project goals in high efficiency, safety, and reliability while being able to respond to different environments and conditions on the beach.

Table 3.3.2: *Mapping Comparison*

Feature	**SLAM**	**SfM**	**QGIS 3**
Primary Use	Real-time mapping and localization in Robotics	3D model reconstruction from images	Geographical Information System for mapping and spatial analysis

Key Features	Real-time operation, integrates multiple sensors, handle dynamic environments	High-resolution 3D models from overlapping images	Supports a wide range of GIS functions (data visualization, editing, and analysis tools)
Use Case	Autonomous vehicles, drones, robotics where GPS is unreliable	Topographic mapping where detailed 3D models needed	Urban planning, environmental management, resource management
Data Handling	Processes sensor data in real-time, using LiDAR, cameras, etc.	Uses image data, requiring significant computational resources	Handles spatial data in various formats, provides tools for data creation, editing, and analysis
Performance	High-performance in real-time applications	High computational demand, dependent on image quality	Generally good performance with large datasets, can be enhanced with plugins
Ease of Use	Requires knowledge in robotics and computer vision	Requires understanding of photogrammetry and 3D modeling	User-friendly with steep learning curve for advanced functions

YOLO

The camera is used to detect objects and act as the ‘eyes’ of the rover. You Only Look Once (YOLO) is a state-of-the-art real-time object detection system that is extremely fast and accurate. It can detect objects in pictures and videos in real time making it suitable for an autonomous robot like ALiCS. YOLO frames object detection as a single problem, straight from image pixels to bounding box coordinates and class probabilities. How it works is that the system divides the image into a S x S grid and each cell is responsible for detecting the objects within that cell. Each grid itself predicts the coordinates (x,y) and the width (w, h) of the object relative to the whole image. Each grid also predicts the class of the object such as person, car, building, dog etcetera.These predictions are combined to form final detections, filtering out low confidence scores and overlapping boxes using Non-Max Suppression (NMS).

Some key advantages of this software include incredibly fast speed, because of its prediction ability, being able to process quickly, making it suitable for real-time applications. YOLO also has the ability to train on full images and directly optimizes detection performance, making it a trainable system that can detect objects from scratch. It also boasts a very high recall rate, meaning it can detect objects with a high number of true positives, proving it does not miss many objects. However, YOLO comes with several limitations and challenges. The grid cell approach can limit its ability to detect objects smaller than the cell size, so this means YOLO will struggle with localizing small objects in images. YOLO's impressive performance also comes at the cost of a powerful GPU and lots of resources. For implementation in our project, YOLO can be used in ALiCS by detecting objects on the beach. It will use the camera to capture images in real time. YOLO can process the video feed from the camera to continuously detect objects, allowing the system to distinguish between trash and other objects such as rocks and other beach obstacles. It will also help in the navigation system; once the camera picks up an object, ALiCS can move towards the object, or if it picks up an obstacle, it can allow the rover to avoid collisions and navigate through the environment. To implement YOLO, our steps will consist of first gathering a dataset of images containing various trash items and background objects. We can train the software on the collected dataset, perfecting the model overtime to improve detection accuracy. Then YOLO when ready would be implemented with the camera system to move forward into real-time object detection. Overall, YOLO is fast and precise making it ideal for real time applications and its architecture is simple and unified, allowing for easy learning and training.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library that provides a comprehensive set of tools for image and video processing. It supports a wide variety of programming languages such as Python, Java, and C++. This software is a great option to implement in our design as OpenCV can be used in ALiCS for additional object detection tasks, such as distinguishing between different types of trash. It supports a wide range of algorithms for image processing, including object tracking, and image segmentation.

OpenCV is highly versatile and can be integrated with other libraries and frameworks, making it a valuable tool for enhancing the capabilities of ALiCS. Some of OpenCV's image processing functionalities include filtering, transformations, and color space conversions. Some other operations that are supported by the software include smoothing, edge detection, and sharpening. The software OpenCV provides pre-trained models for object detection using deep learning techniques like Convolutional Neural Networks (CNNs). It supports popular models such as YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), and Faster R-CNN. The software library also comes with video analysis. This includes capabilities for object tracking and motion detection.

For implementation in the ALiCS project, OpenCV can be useful in many ways. It can detect and classify trash found on the beach. By using image processing and its deep learning modules, it allows the robot to identify different types of trash such as plastics and glass, based on the features captured by the camera. As OpenCV is algorithm dependent, it also has the capability to track objects, and be used to ensure the robot arm precisely targets and picks up detected trash items. This can be very useful for dynamic environments, where trash or obstacles can move. OpenCV can also process and integrate data from the sensors involved in the project, such as the ultrasonic and LiDAR, to provide an understanding of the robot's surroundings. The software's video analysis tools can help the rover navigate the beach environment by analyzing the video feed to detect obstacles and map the field. This information can be sent to the SLAM module to update the robot's map in real-time.

Overall, OpenCV is a very powerful software library that can significantly contribute to the functionality of ALiCS

TensorFlow Object Detection API

TensorFlow is a powerful framework developed by Google used for creating and deploying object detection models. It provides a collection of detection models pre-trained which can be fine-tuned for our own custom applications, making it a very solid choice for many different detection tasks. Some notable features from TensorFlow include, high accuracy pre-trained models, high flexibility, which allows for the support of customization and training of these models, allowing us to detect specific objects relevant to ALiCS. TensorFlow is a very powerful and versatile tool for creating and using object detection models, it has very high flexibility and high accuracy, but it requires many resources and benefits from a powerful GPU.

In ALiCS, TensorFlow can be used for advanced object detection tasks. The API can be employed to create a model that detects and classifies trash or other items on the beach. The boot can then use that information to decide what to pick up and what to do with the objects, ensuring effective and correct operation. A possible workflow we can implement for ALiCS is shown below:

1. **Data Collection:**

- Collect images of different types of trash on the beach.
- Label the images with annotation tools to create a dataset for training the model.

2. **Model Training:**

- Use pre-trained models provided by TensorFlow Object Detection API as a base.
- Fine-tune the model on the custom dataset collected, and teach it to detect specific types of trash accurately.

3. **Integration:**

- Integrate the trained model with the ALiCS system using TensorFlow's APIs.
- Utilize the model in real-time object detection tasks, enabling the robot to identify and classify trash during its operations

The table below compares relevant features of each of the three object detection options showcased. Out of the three now side by side, it seems that TensorFlow and YOLO would be the best option. YOLO is known for its fast speeds, and capabilities for real-time object detection, which is vital for an autonomous device such as ALiCS. This is very powerful technology that also brings very high accuracy to the table, making it a suitable option for applications where quick decision-making is required. YOLO also can be easily integrated with various frameworks and platforms, and has very strong community support, just like OpenCV and Tensor. However, some cons of YOLO is that it does require more resources, usually a powerful GPU, and has limited customization, unlike the other options. The software is limited to object detection and is not as flexible as TensorFlow for custom tasks.
OpenCV shines in its versatility, customization, and resource efficiency. It is a comprehensive library for many computer vision tasks, such as object detection, image processing and feature detection. Being highly customizable as well is an advantage over YOLO and it also has much lower resource requirements than the other two, and can run solely on CPU, making it a great choice for systems with limited resources, and general purpose computer vision tasks. However, its performance and accuracy lacks, and is much more difficult to use than YOLO as it requires much more setup and niche algorithms. TensorFlow on paper is the best option, due to its high accuracy, flexibility, and big support system. TensorFlow allows for training on specific types of trash and objects. The software has the ability to handle a wide range of object detection tasks and integrate with other machine learning models, making it adaptable to future requirements and improvements in our project. Like YOLO, there is a significant abundance of tutorials and support, however it is much harder to use overall, due to its high complexity. Due to our very little experience working with these kinds of tools, YOLO provides a more beginner-friendly experience that will allow us to carry out the tasks necessary for ALiCS.

Table 3.3.3: *Object-Detection Software Comparison*

Feature	**YOLO**	**OpenCV**	**TensorFlow Object Detection API**
Speed	Very fast, with real time detection	Generally fast; depends on algorithms	Fast with optimized models
Accuracy	Supports complex data through topics	Varies with algorithm	High Accuracy
Ease of Use	Relatively easy with pre-trained models	Moderate, requires a decent amount of setup	Moderate to complex
Platform Support	Cross-platform	Cross-platform	Cross-platform
Pre-trained	Available	Limited, requires	Wide range of

Models		custom training	models available
Multiple-Object Support	Yes	Yes, algorithm dependent	Yes
Customization	Limited	High	High, with TensorFlow framework
Flexibility	Limited to object detection	Very flexible	Very flexible
Resource Requirements	Moderate to High	Low to moderate	High
Community & Support	Strong, extensive tutorials	Strong, extensive documentation	Strong, extensive documentation
Use Case in ALiCS	Used for real-time beach trash detection	Used for various image processing tasks	Potential for custom trash detection models

Zemax

Zemax is a software that would be used to design as well as analyze optical systems. It can be used in any area that involves optics such as, photonics, aerospace, and so many more. With Zemax we are able to indicate what is the purpose of the optical part of our design, like for imaging, illumination, etc.

We can then be able to talk about the specifications of the optics. The specifications of the optics can be the length of the focals, the tolerances, the range of the wavelength, as well as the field of view. How this software works is that we are able to select the type of lens that we would need, or we can choose the optical system of the lens instead.

Once all the setup of the specific lens that we choose is done, we were then able to begin simulating and analyzing the results of our lens. The types of simulation we could do is either ray tracing, to see the propagation of light, spot diagrams, to generate them so that we are able to evaluate it, and so much more.’

We ourselves are not too sure as to whether Zemax will be included in our design process, but if it were, we would use it purely to simulate and to test which lens do we think will be the most effective for our project.

Meca500’s Web Interface: MecaPortal

A very big attractor in considering which arm to use, is the Meca500's dedicated and integrated software tools. The Meca500 robotic arm comes equipped with its own interface: MecaPortal. This interface makes it easy to control, program, and monitor the robot's movement through a web browser. It is designed to be user-friendly for setting up and operating the robot without previous robot programming knowledge. It provides real-time monitoring so that the user can view the movements on the screen and simultaneously in person. The interface also provides diagnostic tools to troubleshoot any issues with the robot as well. The MecaPortal makes it really easy to connect with the robot and control the arm to move and pick up objects.

Machine Learning

Machine learning is a self-explanatory term in which the machine is able to learn on its own, gradually improving its accuracy, and it uses data and algorithms in order to improve and operate. A device that is made with machine learning should be able to operate on its own with very minimal human involvement. These are the main categories when it comes to machine learning.

The first one is called supervised machine learning, in which there are labeled datasets that are given to the device that is used to help train it by classifying data and allowing it to predict outcomes accurately. How this would be used in our case, there would be data in which there are obstacles and trash and we would have the difference between the obstacles and the trash already established so that way ALiCS would know what to avoid and what to navigate towards. Methods that tend to be used in supervised machine learning are neural networks and support vector machine, but there are a lot more than just those two

The second type of machine learning is called unsupervised machine learning. What this is is essentially just throwing the machine out into its environment on its own and letting it operate by itself. This would allow the machine to discover any kind of hidden patterns and or data groupings alone without any kind of human intervention. This kind of machine learning is interesting in the fact that we would get to see how the machine decided to learn about its environment instead of us deciding how it should learn about the environment. How this would work related to our device is that we would set ALiCS in an environment filled with obstacles and trash and see how it would classify and group both the trash and obstacles all by itself. Principal component analysis and singular value decomposition happen to be widely used for any kind of unsupervised machine learning.

The third and final type of machine learning is semi-supervised machine learning. This is the fine line between supervised and unsupervised machine learning. How this one works is that the machine is given a dataset just like the supervised machine learning, but this dataset is not labeled, meaning that the machine will have to take the dataset that is given and it should be able to make its own classifications and groupings. How this would work on our end is that we would set it in an environment with obstacles and trash, but not tell it which is which and see how it would figure it out on its own.

Given the three options we have discussed here, we believe that it would be best for us to use supervised machine learning. The reason for that is because machine learning tends to be very complicated, and in order to simplify it enough for it to still be considered machine learning, it would be best for us to use supervised machine learning. Not only that, but also our group has little to none experience with machine learning and we think that trying to go for the harder types of machine learning will be very difficult to pile on top of our already list of things we need to design and implement. Now that we have established which one we are using, we need to figure out which algorithms would be best for us.

Pathing Algorithm

When it comes to our device ALiCS and machine learning, probably the most important algorithm to implement would be the pathing algorithm. The reason why is because the main goal of our design is to be able to pick up trash using our robotic arm, but this will not be able to happen if the rover is not able to make its way to said trash. For this reason, we most definitely need a pathing algorithm. This algorithm should be able to use the information that is received from the LiDAR puck as well as the camera in order to create an optimal path either away from obstacles or towards trash. How path planning works is that it should allow the robot to be able to find the shortest path from a start to a goal without any issues. How that will work in our case will be that ALiCS should be expected to find the shortest path from its start, which is where it currently is, and the goal, which would be the trash, while avoiding any obstacles. When it comes to pathing, there are two types of paths that we are required to know about and they are global path and local path.

Global Pathing

Global pathing is, to put in simpler terms, seeing the bigger picture. It should be able to consider and use all information of its surroundings, meaning that the information of the environment should be completely mastered. To explain what it means for the information to be mastered, when looking at all the information given to the pathing algorithm from the LiDAR puck and camera, everything that is in range of the rover should be displayed. There should be no missing data so that the algorithm is able to correctly computate the best possible path that the information given has to offer. The main downside to this way of pathing, is that it is not something that will be constantly running. Global pathing takes a lot of information and computes it into an optimal path, constantly using global pathing would take a lot of time from receiving the information as well as computing it and the result it would give would be very similar to the previous result unless a massive change in the environment would occur. Since we are using ROS, ROS has built in global pathing algorithms that we will be using, but first we have to consider which global pathing algorithm would be best.

Dijkstra's Algorithm

Dijkstra’s Algorithm happens to be a very well known algorithm that is very good at searching for the goal. The reason why is because dijkstra’s algorithm is considered to be a greedy algorithm, which is an algorithm that is able to find the answer to a problem that is given, as quickly as possible. The way dijkstra works is that there is a graph, and the point of the algorithm is to be able to find the shortest path from a starting node to every single other node. The way it works is that it starts at the starting node, it then compares the length of every adjacent node, and chooses the node with the least amount of distance to then repeat the process with the low distance node as the new starting node. This is a very simplified explanation of Dijkstra’s algorithm, but the explanation should be enough to understand how it can relate to how we would plan to use it in our design. The starting point in our case would be the rover, and instead of looking for the distance of every single node, we would look for the shortest distance from the starting node to the goal node, which in our case, would be where the trash is. Also a main thing to consider, since there will be obstacles in our case, the nodes that contain obstacles will not be allowed to be considered in our computation so that way the rover is able to avoid said obstacles.

As seen in the image to the right, the starting point is the green dot and the goal is the red dot, everything that is not gray, green, or red, is the area that has been explored by Dijkstra's algorithm. The areas in gray are parts that have not been explored by the algorithm. This is important to note, because look at how much space that has been explored by the algorithm, in a situation where there are not many obstacles, it will take a while for the algorithm to be able to compute the best possible path to the goal because of how much space there is. This is crucial because in an area like the beach where it is either full of obstacles or no obstacles at all, there might be a lot of time spent computing because of the algorithm.

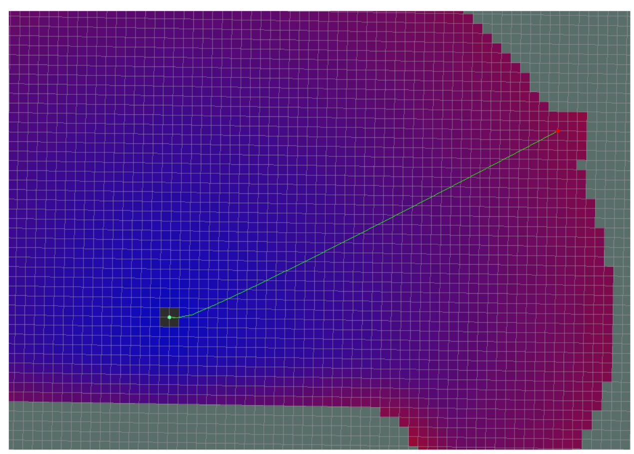


Figure 3.3.3: *ROS’s Implementation Of Dijkstra’s Algorithm, Referenced From [ROS Website]*

Floyd-Warshall Algorithm

The Floyd-Warshall algorithm happens to be pretty similar to dijkstra in which it is able to find the shortest path in a weighted graph. The vague difference between the two is that the Floyd-Warshall Algorithm is able to handle graphs that contain both positive and negative edges. This can give us some kind of creativity when it comes to deciding what parts of the map are positive edges and what are negative edges. For example, we can say that any position that has a ground height that is greater than or lower than the current height is a negative edge so that way we can avoid traveling around constant height changes. The main issue would have to be the complexity of coding said example. The reason why is because how would it translate from the lidar puck to the algorithm. One tremendous flaw of this algorithm is that it is not implemented within the ROS website, making harder to implement compared to the other two choices

A* Algorithm

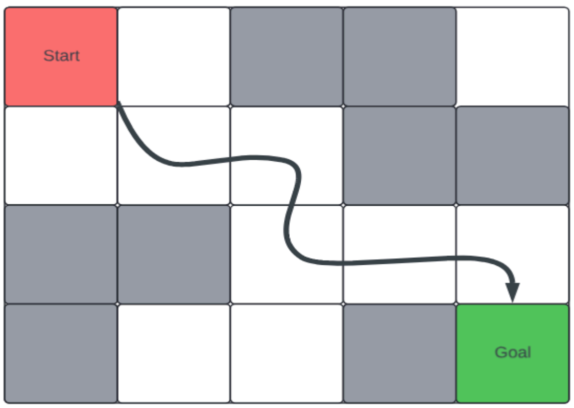


Figure 3.3.4: *A* Algorithm Model*

As seen in the image above, the A* Algorithm is able to create the mapping of the area into a two dimensional grid and is able to use said grid to find the most optimal path towards the goal. In the figure, the squares that are gray are considered to be obstacles that the robot can not travel in and the green square is the goal, or in our case, the trash that we would need to connect. The way this the algorithm works is very close to the first algorithm we mentioned, Dijkstra's Algorithm, it has two lists, the open list and the closed list. It starts with putting the square that says start on the open list, then checks if all adjacent squares are either open or an obstacle, then compares the distances from all the open squares to the goal and chooses the lowest one, then it starts on that square and repeats until it reaches the goal. When it comes to the idea of global pathing, We believe that this is a very promising algorithm to what we are looking for. It takes everything that we are trying to avoid, which is the obstacles detected, and it determines the best path to take to the goal (the trash we need to collect).

Taking into consideration the two algorithms we have for global pathing, we believe that the A* algorithm would be the best algorithm for our design. The main reason why is because how Dijkstra's algorithm would take a good amount of time to scan the entire area compared to how fast it takes the A* algorithm to find one optimal path. With our environment choice being the beach, there is a good chance that there will be little to none obstacles out on the beach (depending on the time of day), and since there is no obstacle, using dijkstra will take longer everytime we use it than using A* instead. There could be an argument on using dijkstra because it considers a lot more possibilities, but that is what local pathing is for.

Table 3.3.4: *Machine Learning Algorithm Comparison*

Algorithm	Dijkstra	A*	Floyd-Warshall

Type of Algorithm	Uniform cost search	Heuristic-based search	Uniform search
Way Of Search	Finds the optimal path	Finds the optimal path	Finds optimal path with both negative and positive edges
Speed	Slower in large graphs with little to none obstacles	Usually fast, even in large graphs	Slower because there is a lot to consider
Memory Usage	Usually high, but depends on size of graph	Usually high, but depends on size of graph	Usually high, but depends on size of graph
Area Exploration	Travels through all areas in the graph until the goal has been reached	Travels through the areas that would appear to be closer to the goal	Travels through all areas in the graph until the goal has been reached
Handles Negative Weights	No	No	Yes
Time Complexity	O(V^2)	O(V^2)	O(V^3)
Implemented on ROS website	Yes	Yes	No

Local Pathing

Local pathing is essentially the smaller version of global pathing. What local path planning requires is the information of the sensors that are available at real time. So to put it in simpler terms, global pathing is able to help find a general optimal route towards the goal, and local pathing is able to use the real time information to constantly update the optimal route in case that the obstacles have changed positions. Local pathing will still have to rely on LiDAR puck and the camera similar to global pathing, but the difference is how often local pathing will be used compared to global pathing. Local pathing will be able to take both the optimal path from the global pathing, as well as the global map in order to make small adjustments. It would make small adjustments if, and only if, there are obstacles nearby. Just like global pathing, ROS has some options for local pathing that we looked into and will talk about later in this paper, for now, let's look at specific algorithms of local pathing

Dynamic Window Approach

Dynamic window approach is a type of local pathing that is used by ROS that we considered using. The way dynamic window approach works is that it is able to sample a multitude of different sets of velocities, and accelerations, and then use those values in order to then comprehensively consider the constraints of the speed as well as the acceleration/deceleration performances of the rover. This will then calculate the trajectories of the velocities that were received, and then the trajectory will then be evaluated based on the evaluation index. The trajectory with the highest evaluation score will then be chosen, and the velocity and the acceleration that correspond to that trajectory, will be used as the driving parameters of the robot, or in our case, the rover. This is a very strong local path that would be very useful for our design, however, the most concerning problem of this local path is that there is a lot of important info that we would need to know about the rover in order to make good use of this local path. In order for us to implement this, we would need to know our maximum velocity as well as our minimum velocity, and the reason for that is because dynamic window approach would adjust our velocity either higher or lower depending on the situation, and if it were to choose a velocity that is higher than our maximum or lower than our minimum, that could bring issues with our rover. What could happen is that maybe the software would believe that we are able to avoid said obstacle if we traveled at a certain velocity, but, if our rover is not capable of reaching said velocity, then the rover may end up just crashing into the obstacle. So the important part we would need is to have our data of the rover be extremely precise to avoid any errors while using this software.

Timed-Elastic Band

Another local path is the timed-elastic band, which is a collision avoidance method that works for trajectory optimization. This algorithm is able to delete or add multiple constraints that have different needs in different settings. This would imply that the algorithm happens to have a high degree of flexibility. It essentially behaves the same way as the dynamic window approach, but the difference between them is that the timed-elastic band is way more concerned about time intervals. This algorithm is able to take the rover’s trajectory and it is able to optimize it with the use of time intervals. Another similarity with dynamic window approach, for us to implement a timed-elastic band, we would need to know about the parameters of our rover. Some of the parameters that would be needed to know about are maximum translational and angular acceleration as well as maximum translational velocity. Now it is not specifically shown or stated if these parameters are required in order to fully operate, but it is best to believe that it would be best for the functionality of our rover, for these parameters to be known and inputted into the software. As mentioned before, we would need to be extremely precise and accurate in our parameters so that way there will not be any issues when it comes to the path the rover is taking. If we input a maximum translational acceleration value that is higher than the rover's actual value, this could lead to miscalculation from the software that will end up creating collisions because the software used the wrong calculations that were received from us.

Considering the information that was discussed about dynamic window approach and timed-elastic band, there is one thing we can agree on when we have to choose between them and that is the fact that our rover's parameters are important. BOth algorithms require some kind of parameter from our rover, whether it's the rover's maximum translational acceleration or its maximum velocity. So when it comes to that factor, both algorithms have the same kind of level of importance. When it comes to the dynamic window approach, however, there is one factor about it that weighs it down, and that factor is the prediction of obstacles. Since the algorithm mentioned is mainly able to adjust the velocity and acceleration of the rover, the only obstacles the algorithm can predict are the ones that are within close vicinity of the rover. This would mean that the rover will have to be able to move out of the obstacle's trajectory very fast if the obstacle is moving at a very fast speed near the rover. Comparing it to a timed-elastic band, however, a timed-elastic band is able to predict nearby obstacle's movement, all while being able to continuously adjust the trajectory of the rover. With that being said, although it may have potentially more parameters to consider, a timed-elastic band will be the worthwhile choice to prevent any collisions which are important to our design.This could possibly change depending on the complexity of the timed-elastic band

Active Scene Recognition

Active scene recognition, or ASR is another type of local pathing algorithm. How it works is that it takes the optimal path that was received from the global path and then it follows it as accurately as possible while making very small adjustments. One thing about this planner that is different from the other two planners, is that this planner has a safety function. The way that the safety function works is that the robot itself does not plan on driving to any area that is near an obstacle. This works by the planner checking the global map and checking which area does not have any obstacles. If there are too many obstacles in every possible area, the rover will stop driving and it will only be able to drive again once it has detected any kind of change of obstacles. This would either work or not work in our case, it is not a bad idea to stop when there are too many objects around, however, it should be possible for our rover to drive around the obstacles in order to avoid them. We think that in order for this planner to be implemented, we would have to get rid of the safety function in order to keep this planner functioning around the same as the other two planners function. There are a multitude of resources provided by the ROS website we can use in order to have a better understanding of ASR. There are tutorials as well as videos.

A better example of how ASR works is a human leading a horse with a carrot on the stick. The horse will accurately follow the stick and it would make small adjustments if the human was to change the direction that the carrot is facing. It is able to make these adjustments using certain factors and they are the velocity and the rotation velocity.

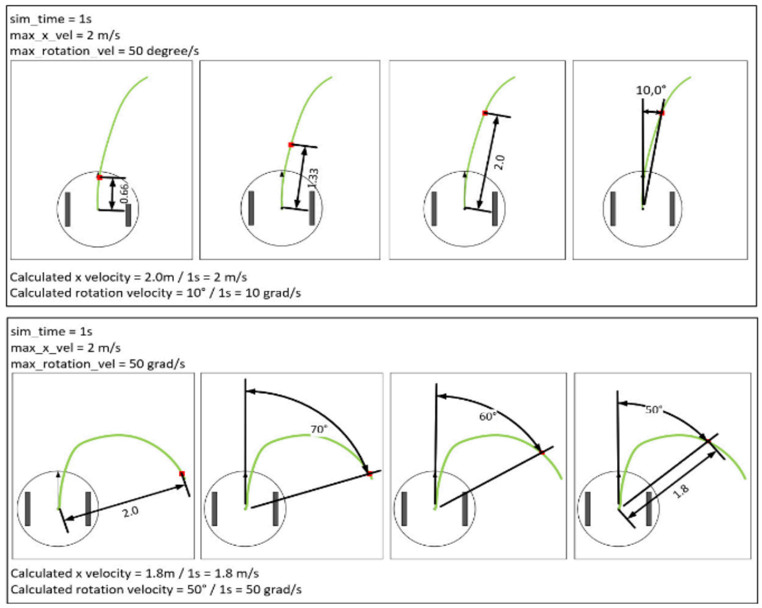


Figure 3.3.5: *ASR Calculations, Referenced From [ROS Website]*

The image shown above is an example of how the calculations would work. It is able to use the known values of the maximum x velocity as well as the maximum rotation velocity and is able to calculate how much distance it would cover and how much it will rotate. Another thing is that when an object is detected, it is able to adjust the path to avoid said object, which is something that is needed for our design.

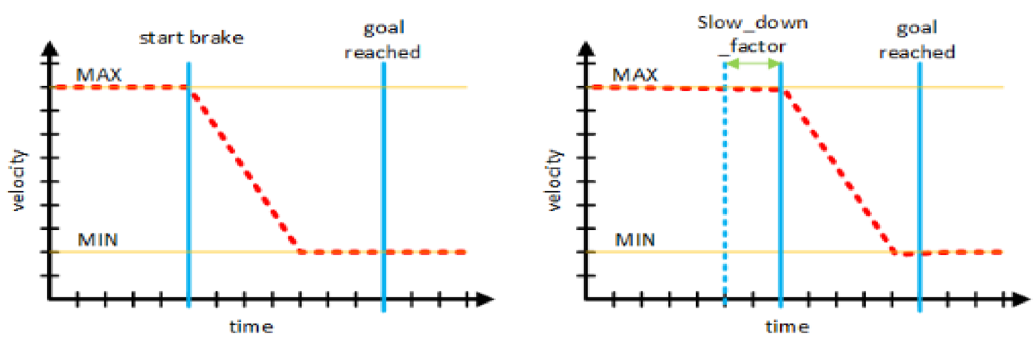


Figure 3.3.6: *ASR Calculations Graph, Referenced From [ROS Website]*

Another part you can see above is the calculations for the time it will take from the start brake and the slow down factor. This is important for us cause when we reach the goal (trash), we want to be able to slow down enough so that we do not end up passing by the trash because that will mean that we will not be able to pick it up and we would have to adjust the rover.

__Table 3.3.5:__ Local Pathing Comparison

Algorithm	Dynamic Window Approach	Timed-Elastic Band	Active Scene Recognition
Method	Uses rover’s velocity and acceleration to determine how to avoid objects	Uses time intervals to continuously change the rover’s trajectory	Faithfully follows optimal plan while making very slight adjustments to the optimal path it takes
Prediction Of Obstacles	Able to predict obstacle’s movements, but limited to close vicinity of rover	Able to predict obstacle’s movements	Able to recognize objects to alter path planning
Implementation	Moderate complexity	Harder to implement due to involvement of multiple parameters	Moderate Complexity
Requirements	Very accurate sensor data	Very accurate sensor data as well as environment mapping	Very accurate sensor data
Handling of Moving Obstacles	High	Moderate	High
Compatible With ROS	Yes	Yes	Yes

Chapter 4: Standards and Constraints

4.1 Standards

Standards in engineering documentation play a critical role in ensuring quality, reliability, safety, compliance, and efficiency throughout the lifecycle of engineering projects and products. Standards establish guidelines that ensure all engineering documentation follows a consistent format, structure, and level of detail. This consistency helps in understanding and using the documentation correctly. A standard is a formal technical document for generally accepted products, processes, procedures, and policies. Standards are not mandatory by law unless adopted as a code or regulation, however they impact various aspects of our life and are an integral part of every design.

Engineering standards provide a common language and format that engineers, designers, manufacturers, and users can understand universally. This facilitates effective communication across different teams and disciplines. Many engineering standards include safety guidelines and regulations that must be followed to ensure that products or systems are safe for use. Adhering to these standards reduces the risk of accidents or failures. Following established standards streamlines the design, manufacturing, and testing processes. Engineers can leverage existing knowledge and best practices, reducing the time and effort required to develop new products or systems.
Five key players in the development of standards include the American National Standards Institute (ANSI), the National Institute of Standards and Technology (NIST), standards developing organizations (SDOs), government bodies, and consortia. ANSI serves as the official coordinator for standards development organizations in the USA, publishing a wide range of standards such as basic guidelines, design specifications, management systems criteria, processing methods, product requirements, and testing procedures. These standards are crucial in the design and construction of our robot, ensuring compliance with various regulations essential for creating a functional product. Specific standards for robots provide standardized terminology and information, facilitating easier knowledge sharing among stakeholders.

ANSI

ANSI stands for the American National Standards Institute. In the context of engineering, ANSI plays a pivotal role in developing and promoting voluntary consensus standards for a wide range of industries and disciplines within the United States. These standards cover various aspects of engineering, including dimensions, performance criteria, safety guidelines, and testing methods for products and systems. ANSI standards are developed through a consensus-based process that involves stakeholders from industry, government, academia, and consumer groups. They aim to establish uniformity, reliability, and interoperability in engineering practices, ensuring that products are safe, compatible, and effective for their intended use. ANSI standards are widely adopted across industries to facilitate communication, improve efficiency, and meet regulatory requirements, both nationally and internationally.

Robotics and Automation Standards

ANSI/UL 4600

ANSI/UL 4600 is the standard that is used for the standard of safety for the evaluation of autonomous products. It is a standard that we should definitely be following considering that we aim to make ALiCS autonomous. This standard mentions certain item operation safety related issues which we'll go over and explain how it affects our project.

a) Operation of Autonomous items in potentially unstructured environments

In our situation, the beach is possibly considered an unstructured environment, however, the standard claims that we are able to operate in a potentially unstructured environment as long as it is disclaimed. How ALiCS tends to move its way through the rough and unleveled area of the beach is with the use of wheels that are different from normal ones. The wheels we tend to use allow us to navigate in off-road situations much easier than normal wheels. Also the way it would figure out how to navigate its way through an unstructured environment is with the help of our lidar puck scanning the area and giving back said info to our rover.

b) Operation with potentially inaccurate, incorrect, incomplete, or misleading data provided by sensors

With the use of a lidar puck as our sensor, the lidar puck should be more than capable of giving accurate data that will be provided to our rover. Even if the lidar puck fails to give accurate data, there will be proximity sensors attached to ALiCS that would work as a failsafe.

c) Potential defects and failures of hardware and/or software in the item, data collection functions, data processing functions, communications, engineering support systems, tools, and infrastructure support

Over the span of senior design I and senior design II, we worked on the hardware and the software for all that was listed. When our design of ALiCS was completed, we tested and checked for any and all types of potential defects and failures of hardware and/or software in order for our design to be considered completed. Not only did we test during the final stages of our design, we also tested throughout all stages of the design to make sure there is nothing wrong with our design.

d) Human contributions to potential risk, including occupants, pedestrians, other road users, non-road users, cargo handlers, maintainers and inspectors. This includes act of omission and commission; accidental and malicious physical acts; and human roles in creating as well as mitigating risk

Since the location of ALiCS would only be in a beach environment, there would not be a lot of road users, cargo handlers, maintainers and inspectors. The only human contributions towards risk would mainly be just pedestrians. ALiCS's way of avoiding colliding with humans is with the combined components of the lidar puck and the rover. The lidar puck will determine the position of the human and if it becomes too close it will notify the rover and the rover will then begin to make adjustments in order to avoid colliding with the human or any other kind of occupant.

e) Lifecycle considerations, including design data collection, engineering data management, tool qualification, design, implementation, testing, other validation, field data collection, operations, maintenance, updates, upgrades, and retirement.

Lifecycle considerations also encompass potential changes to the environment which may affect odds, changes in object types, changes in behaviors, etc.

One of the main considerations we have towards ALiCS's life cycle, was having all our hardware covered to prevent any damage from the weather such as sun, rain, and wind causing the sand to land on and damage our components. It should also protect the hardware from any of the wildlife that could potentially destroy or mess with our hardware. We also have gathered the info we need for all of our components to be able to verify how we should be able to keep all of our components safe and increase the lifecycle expectancy to as long as we can. Some changes to the environment we plan on figuring out how to deal with includes changing tides that could lead to the water reaching ALiCS or the opposite effect in which the distance the water covers decreases.

f) Ability to use a heterogeneous approach to arguments, including use of diverse standards to support safety (e.g., use of different but acceptable functional safety standards for different item subsystems).

One of our heterogeneous approaches is the approach we use when it comes to our object detection. We have the lidar puck, which is the main way of detecting objects that should be able to map out the area and let our rover know where it should go. Now if something were to happen and the lidar puck did not pick something up that is approaching the rover, we had proximity sensors installed as a failsafe that when an object is picked up by the sensors, the rover will stop automatically.

Standard for Robotics and Automation: IEEE 1872-2015

IEEE 1872-2015 or Standard Ontologies for Robotics and Automation is a core ontology describing general concepts, relations and axioms of robotics and automation. This standard is intended to be a reference for knowledge representation and reasoning in robots. Included are general concepts and their definitions, attributes, constraints, and relationships. IEEE 1872-2015 is composed of a core ontology called CORA, proposed by the working group (WG) Ontologies for Robotics and Automation (ORA) and sponsored by the IEEE Robotics and Automation Society. Published on April 10th, 2015, and ANSI approved on August 5th, 2016, the purpose of this standard is to provide both a unified way of representing knowledge and a common set of terms for sharing knowledge across humans, robots, and artificial systems. Some aims include
providing methodology, terminology, defining concepts in robots, knowledge representation, and facilitating the transfer of information among robotic systems. This standard is important for our design as we are aiming to build an autonomous robot which will interact with humans. The standard makes it easier for knowledge and information to be shared about robots not only amongst humans but for interactions between robots and humans. The standardized knowledge of robotics and automation allows ideas, designs, and data to be shared amongst large groups without information being lost or confused.

Optical Standards - U.S. Standard MIL-PRF-13830B

This standard describes surface quality by identifying a "scratch" number, any marking or tearing of the surface, followed by a “dig” number, a small rough pit or spot in the surface. This is based on calibration and quality assurance inspection standards widely used in the optical industry. The scratch number is an arbitrary number scale used to describe the brightness of a scratch on the lens, lower numbers being better. It also covers fractures and edge chips on a lens. The purpose is to define the allowable defects in a lens or coating or on the surface of an optical component. Dig numbers are the diameter of a pit or spot, specified by 1/100 mm. An example would be a lens that is rated 80-50, would have a surface dig somewhere on the lens of 0.5 mm in diameter and one with 80-20 would have a dig of 0.2 mm in diameter . The standard will only allow one “dig” of that maximum stated size for every 20 mm of lens diameter. The scratch number is measured by method of comparing to a calibrated sample standard by visual inspection under dark-field illumination conditions. This “by-eye” matching can lead to different evaluators scoring lens either higher or lower depending on what is observed. The evaluator is looking at each lens and matching the brightness of the two surfaces by eye. It is not a measure of the size of the scratch but the visibility of the scratch. The scratch standards as described in “Surface Quality Standards for Optical Elements (scratch and dig) Drawing C7641866” (Revision R as of 2010). The prototype calibration samples are kept at Picatinny Arsenal in New Jersey. They follow the order of (80, 60, 40, 20, 10), there is a pair for each number, one defining the maximum allowable visibility and the other defining the lowest allowable visibility to fit the criteria for each number. The military and its vendors can use these prototypes or working copies calibrated from them. The private sector relies on the working copies calibrated from the original prototypes costing several thousands of dollars, these can vary in quality from maker to maker but are close enough to be used as a standard. For this reason, it is important for a commercial supplier to include a surface quality inspection specification manufacturer name for the optical component being compared.

The ISO (International Organization for Standardization) is a worldwide federation of national standards bodies. They work to prepare international standards through technical committees. Each member represents a different technical interest on the committee, and they work with international organizations, governmental and non-governmental entities. Unlike MIL-PRF-13830B, ISO 10110-7 is a more quantitative method to characterize scratches and digs on an optical element. It specifies surface quality based on the physical sizes and frequency of surface defects. MIL-PRF-13830B, can be a more economical and faster method to characterize surface imperfections but ISO 10110-7 is a more precise method to identify these blemishes if it is required. This method is more time consuming because it requires the use of a microscope with strong enough magnification to identify all sized surface defects. Many measurements are required to image the entire sample and to visualize these requires a small field of view. This standard does not make a difference between scratches and digs, treating both as surface imperfections. It will indicate the allowed number of imperfections and equate a grade number to each optical element defined by the square root of the area of the maximum allowed imperfection. This measurement is known as the dimensionality measure, but ISO also can use the visibility method identical to MIL-PRF-13830B. The benefit is that they can both be used interchangeably and if the need for a high-precision application arises, the use of ISO is there but if application is not as strict, the more expedient and cost effective method of MIL-PRF-13830B can be used.

Laser Product Standards (LiDAR)

The industry guidelines related to this project include: General requirements (29 CFR 1910.132), along with the eye and face protection (29 CFR 1910.133) requirements. There are additional voluntary guidelines which are provided by the American National Standards Institute (ANSI). The ANSI guidelines related to the project are: Safe Use of Lasers (Z136.1), Recommended Practice for Laser Safety Measurements for Hazard Evaluation (Z136.4), Safe Use of Lasers in Educational Institutions (Z136.5), Safe Use of Lasers Outdoors (Z136.6). These are some of the safety standards required for operation of a class 4 laser which is the required class to create plasmas for the LIBS system project. These standards will be adhered to to the best ability of the group to ensure the safety of the group members and those who will be around the laser while in operation. Using optical lasers falls under the OSHA standards for laser hazards. The State of Florida does have some regulations and standards for the use and manufacturing of lasers as well. High power lasers (classes 3b and 4) fall under the jurisdiction of the Florida Department of Health in the Bureau of Radiation Control. The Bureau of Radiation Control oversees and registers many different sources of radiation emissions. These include Ionizing Radiation Machines (or X-ray machines), Radioactive Materials, Radiologic Technology, Environmental Radiation Programs, and Laser Radiation. The Florida Department of Health Bureau of Radiation Control has registered over 1700 lasers within the state. This laser will not be used on humans for any medical reason.

Lasers are classified according to their ability to cause harm. This factor, referred to as the accessible emission limits (AEL) is calculated using exposure time, laser wavelength, laser power, amongst other factors. Non-class 1 laser products may be dangerous, causing eye injuries, skin hazards, and fire hazards due to laser radiation. Classes of lasers include class 1, class 1M, class 1C, class 2, class 2M, class 3, class 3R, class 3B, and class 4. International standard IEC
60825-1 was published May 14th, 2014 and prepared by the IEC technical committee 76: optical radiation safety and laser equipment. IEC 60825-1:2014 is applicable to the safety of laser products emitting laser radiation in the wavelength range 180nm to 1mm. A laser product may consist of a single laser with or without a separate power supply or may incorporate one or more lasers in an optical, electrical, or mechanical system. The objective of IEC 60825-1:2014 is to introduce a classification system for laser and laser products within the wavelength range 180nm – 1mm to both aid in hazard evaluation and determine user control measures. This standard establishes requirements for manufacturers to supply information so the proper precautions can be taken. This information includes warnings of hazards associated with radiation from laser
products and how to reduce the possibility of injury. This edition includes updated accessible emission limits for class 1, 1m, 2, 2m, and 3R of pulsed sources.

Communication Standards

Communication standards for digital communication may include the RS-232, RS-422, and RS-485 standards. The LIBS system may use one of these communication standards to communicate with external devices or machines. These standards describe the voltage logic levels and speeds that are expected to be used by devices to communicate with one another. The voltage levels described in this standard indicate the voltages used to indicate a logic high and logic low signal. The speed at which data is transmitted is also listed by the standard. For the LIBS system this will likely be abstracted away via the digital controller, or communication protocol chip. However it is important to be aware of what communication standard is being used in the system. Without understanding the voltage logic levels and communication speeds used in the system, the ability to provide forward compatibility for new devices could be limited. The MCU specifically uses RS-485 for UART communication.

Portable and Rechargeable Battery Standards

When it comes to batteries, there are a lot of standards so it is important to know about which battery we plan on using and what standards apply to that battery. The first thing we should talk about is the standards for the general battery. The table below will talk about specific standards for the general battery that we believe is important to know about

Table 4.1.1: *General Battery Standards*

Standards	Description
IEC 60050	Vocabulary about electronics and technology relating to batteries
IEC 600086-1	Talks about Primary batteries in general
USNEC Article 480	How to store batteries
ANSI C18.2M	General and specifications about portable rechargeable cells and batteries
UL 2054	Testing for the safety of commercial as well as household battery packs

After talking about the general standards it is best to talk about the standards that relate to the batteries we plan on using. The three main batteries we are considering for now are nickel metal hydride, nickel cadmium, and lithium batteries, so it is important to know about the standards that relate to them. The reason why is that those standards talk about the specific ways of safety operation, electrical requirements, technical specifications, and many more. Written below is a table that talks about the standards for each one.

Table 4.1.2: *Battery Standards*

Batteries:	**Nickel Metal Hydride**	**Nickel Cadmium**	**Lithium**
Standards:	IEC 61436	BS EN 1175-1	BS 2G 239
	IEC 61809	BS EN 2570	BS EN 60086
	IEC61951	BS EN 60285	02/209100
			BS EN 60086

The table above only lists which standards apply to which type of batteries, so we will now talk about the important standard for each one. The three standards that were listed for nickel metal hydride mainly talks about secondary cells and batteries containing alkaline. For nickel cadmium, BS EN 2570 spoke about the technical specifications of the battery which is important to know if we plan on using said battery for our design. For lithium, the BS EN 60086 is very important as it discusses the safety standards for lithium batteries, which is valuable to know about if we plan on including it in our design.

ANSI for Portable Rechargeable Batteries

ANSI C18.2M is divided into two parts, part one covering general information and specifications, and part two, which covers safety standards. Both standards are developed and published by the National Electrical Manufacturers Association (NEMA) while processed and approved for submission to ANSI by the American National Standards Committee. ANSI C18.2M Part 1 is applicable to nickel cadmium, nickel metal hydride and lithium-ion portable rechargeable cells and batteries. This standard contains information addressing battery dimensions, terminals, standard charge methods, polarity, and standardized performance and

mechanical tests which the specifications in part two are based on. This standard also specifies the designation for secondary batteries, they must be designated by a number, followed by a letter, then by a sequential number. The first number represents the

nominal voltage, the letter represents the type (H: nickel metal hydride, I: lithium ion, or K: nickel cadmium), while the final number is assigned based on the battery's physical dimensions, terminal configurations, and
performance variables.

Soldering Standards

The J-STD-001 is a standard that is issued by the Institute for Printed Circuits (IPC). The point of the J-STD-001 standard is to talk about what is established as the best soldering practices. The reason why this standard is important is because it would help us guarantee that the product we design will have the highest quality as well as reliability in specific weather conditions. This is something that we should best follow considering that our design is going to operate in the beach in which it is susceptible to multiple weather conditions, good or bad. There are a few main important requirements for soldering and they are as follows:

1. It is important to keep the area clean to prevent contamination of materials, tools, and surfaces
2. Heating and cooling rates should be the same as the standards from the manufacturer
3. The wire stands should not have any kind of damage and the solder must wet the tinned area of the wire
4. Both the soldering and cleanliness aspects should be inspected before the applying of the coating and stacking
5. Soldering errors should be reworked or scrapped
6. Visual inspection should be performed using automated optical inspection as well as automated x-ray inspection

The picture to the right is an example of soldering and it would be best for us to make sure that ours follow the correct way in order to give us the best reliability and quality.

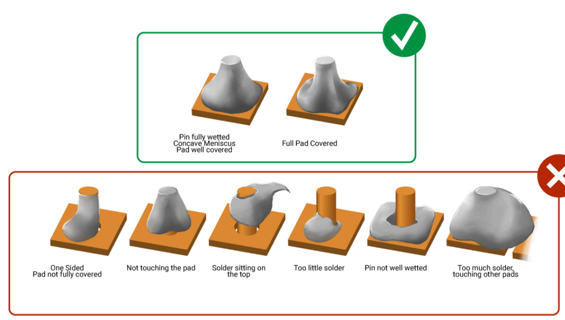


Figure 4.1.1: *Soldering Examples, Referenced from protoexpress website*

PCB Design Standards

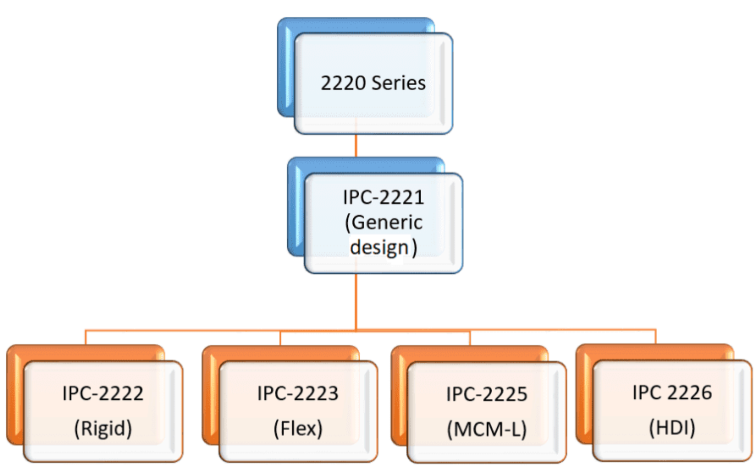


Figure 4.1.2: *PCB Standards*

The figure above shows how PCB standards are split up into four different types for a singular base type. The type of circuit board we will be focusing on for our project should be under the IPC-2221 stand for a generic design. This standard contains the general requirements for PCB design and aspects of electronic card assembly. However, depending on how specific we want the function of our board to be it could fall in different stands for how the board is made. These standards are made by ISO (International Organization for Standardization), ITU (International Telecommunication Union), IEC (International Electrotechnical Commission), and IPC (Association Connecting Electronics Industries). These leaders make up the IPC standards which stands for Institute for Printed Circuits. The IPC is made up of over 4,000 members companies that make advanced microelectronics, aerospace and military, automotive, computer, industrial equipment, medical equipment and devices, and telecommunications industry. These standards make it provide safe, high-performing, and reliable PCBs. These PCBs are divided up into 3 different classes for different types of products

- Class 1 - General Electronic Products:
  This is when a PCB product comes suitable for the primary function after assembly without any additional requirements

- Class 2 - Dedicated Service Electronic Products:
  This is where the PCB continued performance is required and downtime is not desired but not ultimately critical.

- Class 3 - High-Performance Electronic Products:
  This is where the performance of the PCB is needed constantly and is critical to the system. This means there can be no downtime and environmental conditions should be harsh to make sure this critical system never goes down.

Now for cases like ours we should be using IPC-2221 guidelines that are used for PCB like the ones we will be using. The key points of the IPC-2221 are:

- Design Guidelines:
  This offers guidelines on how the parameters for the PCB should be made such as spacing, layers, and trace width. This will make a PCB have better signal integrity, power distribution, and thermal management.

- Material Selection:
  This standard provides recommendations for the selection of material factors like dielectric properties, mechanical stench,and thermal conductivity.

- Component Placement:
  This standard also provides a guide for the placement of components of a PCB that addresses issues related to thermal management, accessibility, and signal interference. These recommended placements are important for overall functionality and reliability.

- Trace Routing:
  This standard provides a guide for routing tracing for factors like EMC, signal integrity, and impedance control. This gives the best practice for high-speed signals and minimizing signal interference or crosstalk.

- Quality and Reliability:
  This standard outlines the inspections and tests to employ on a finished PCB to ensure that they meet industry standards.

- Thermal considerations:
  This standard addresses the thermal management guide for a PCB. It explains how to make use of Thermal vias, heat dissipation, and the placement of heat generating parts. This is crucial to not overheat a PCB and keep the device from degrading over time.

- Design for Manufacturability:
  This standard explains the importance of designing a PCB with manufacturability in mind. This means how easy it is to assemble, fabricate, and test a given PCB.

4.2 Design Constraints

When it comes to creating a new project or some kind of new design, one of the most important factors that should be taken into consideration even during the starting process is design constraints. It is very relevant for those who are working on the project or design to figure out what constraints they should have, whether its economic, environmental, social, political, ethical, health and safety, manufacturability, etc. The more we are aware of their constraints, the easier it is for us to come up with a plan that meets the constraints we have. What is written below in the sections is the multiple design constraints that tie in to our project, and how we plan on addressing the constraint.

Time Constraint

The main constraint that we as a group realized was none other than the time constraint. In the beginning of this class we came up with a good amount of ideas but the main reason why most of our ideas were scrapped was because we believed it would not be realistic for us to achieve that idea in the amount of time we were given. Once we settled on an idea that we believed was achievable (ALiCS), the first thing we discussed as a group was what would be the best way for us to space out the time we have in order to make our design a reality. We came to the conclusion essentially very similar to how senior design I and senior design II are set up, in which we would spend most of our time in senior design I to research and come up with a plan on how to design and implement ALiCS and then during senior design II, we would then begin designing ALiCS, then start testing each individual hardware component to see if they are functional, combine all working hardware components to where we planned on having them, and then lastly, begin testing ALiCS and making adjustments based on said test. We also believe that it is best for us to get started on the design phase as soon as possible so if we have enough time, it would be in our best interest to start designing during senior design I to give us better odds of completing ALiCS.

There is one more time constraint that relates more to ALiCS than us as a group. The time constraint is none other than the battery life. There are a lot of components in play that require charge in order to function, so it would be best for us to have a battery that is capable of giving ALiCS a decent enough operation time for ALiCS to have an effect on the environment. It would not be wise to have a battery that is only capable of keeping ALiCS up for 30 minutes or less because the beach would barely end up getting clean if we have to keep charging the robot every 30 minutes.

Economic Constraint

During the course of senior design I, it was heavily suggested for groups to have a sponsor to help provide money, ideas, or both for the groups, unfortunately for us we were not able to secure a sponsor. What that means for us is that we are limited in our budget. The reason for that is because all parts and components that we would need for ALiCS, we ourselves would have to pay for it (unless we are borrowing or we already own it). This would mean that we would have to do more extensive research on all the parts and components we need before we actually purchase it. We can not afford the mistake of purchasing expensive items that we may not end up using or may not end up working out. The main thing we need to take into consideration is that we are 5 college students and most of us are only working part time jobs so we do not have excess money to be spending. Luckily for us we were able to borrow the robotic arm from mecademic robotics or else we would have to come up with a cheaper alternative.

Ethical Constraint

Since ALiCS's purpose is to help better the ecosystem of the beaches, it makes sense for our design to have some ethical constraints. One main constraint is the safety of all living things, primarily humans and wildlife. Ensuring that ALiCS does not pose any threat to the beachgoers and wildlife in its operational area is paramount. To achieve this, we have integrated a lidar puck that continuously monitors ALiCS's surroundings. The lidar puck sends real-time information to the rover, enabling ALiCS to navigate and move out of harm's way when an obstacle is detected. In the unlikely event of a lidar puck failure, we have additional proximity sensors installed on the robot. These sensors act as a secondary safety mechanism, detecting objects in ALiCS's path and automatically stopping the rover to prevent any collisions.

Beyond safety, another critical ethical constraint involves respecting the privacy of people on the beach. While our robot is equipped with a camera, we have taken measures to ensure that it does not record or capture images of individuals without their consent. The camera's sole purpose is to distinguish between trash and non-trash objects, aiding in ALiCS's primary function of beach cleanup. This approach ensures that the privacy of beachgoers is maintained while allowing ALiCS to perform its environmental duties effectively. Ethical considerations also encompass the data handling practices of ALiCS. Any data collected by the robot, whether it pertains to the types and locations of trash or the robot's operational parameters, is managed with strict adherence to data protection regulations. We ensure that no personal or sensitive information is gathered, stored, or processed by ALiCS, reinforcing our commitment to ethical data practices.

Additionally, our commitment to ethical design extends to minimizing the environmental impact of ALiCS itself. We are dedicated to using environmentally friendly materials and ensuring that ALiCS's operation does not contribute to pollution or ecological disruption. This includes designing the robot to be energy efficient and exploring the use of sustainable power sources, such as solar panels, to reduce its carbon footprint.

Environmental Constraint

Environmental constraint is an important constraint to use because the location our design desires to operate in is on the beach. Since we live in Florida in which it is always very sunny and rains very often, this constraint is something we need to be sure that we can have as much control on it as we possibly can.The sun and the rain as well as the wind can cause some problems for ALiCS. If it's too hot it can possibly melt some of the components as well as overheat the battery, causing it to operate in less time than usual. During the rain it can damage the components and it can also make the components short circuit. The wind is not that significant, but it can carry the sand and end up placing sand inside or components which can lead to our components possibly worsening over time. The solution we have for the weather is enclosing the pcb and the battery so that they can not be exposed to the weather. As for the robotic arm, since it needs to pick up the trash, the only cover we can provide for it is a roof so it can at least lessen the risk of the weather. The camera and the lidar puck will be on top of the roof so those two components would be at the highest risk of the weather, but as long as we are not constantly putting ALiCS out in bad conditions, those components should be fine.

Chapter 5: Comparison of ChatGPT and Other LLMs

5.1 Introduction

This chapter evaluates the pros and cons of ChatGPT in the context of our ALiCS (Autonomous Litter Collection & Sorting) project. We will examine how AI can contribute to the development of ALiCS, highlighting both its benefits and limitations. Additionally, we provide case studies to illustrate the practical applications and challenges of using AI for this project.

5.2 Pros of AI Language Models for ALiCS

Artificial Intelligent tools, such as ChatGPT and Gemini, are incredibly helpful in the design and development of ALiCS. AI helps lay out the skeleton of the project in which we can use and build upon. We can use AI in assisting us in ideas of improving ALiCS and using it to troubleshoot our problems.

There are a lot of things we can use ChatGPT for. ChatGPT can be used to generate multiple design concepts off of one prompt, and then we are able to take all the concepts, compare and contrast them, and then pick one of the concepts. If we are unsure about which component we think is best for the design, ChatGPT will tell us the parameters, performance, cost, as well as how it works for each and every single component. This will help lead us in the right direction of which part to pick. ChatGPT can also take the availability of all of our members and then provide us a calendar of good times for us to be able to meet together as a group. We can add on to that calendar ChatGPT made by giving it important dates, for example, the date of the presentations for senior design II, to add on to the calendar so we did not forget about it. When we are designing something that requires software that we know nothing about, ChatGPT can find and recommend several software that is close to what we are looking for, and it is up to us to find the perfect one that will be best for our situation.

Design and Development Support

AI language tools are incredibly valuable in supplementing the design and developmental phases of the project.We utilize them as idea generators to bridge the gaps in our design and identify enhancements for ALiCS. When provided with clear instructions, goals, and constraints, ChatGPT can outline the foundation for the robot, offering us a framework to build upon. Here, we evaluate the pros and cons of artificial intelligence, exploring their effectiveness and limitations in the realm of engineering.

It is important to be aware of the limitations of AI tools and know when to use it and when it is not as accurate. Since AI tools pull from everything on the internet, it is a random generated source that we cannot rely on. These AI tools are also simplified algorithms that cannot perform more complex mathematical equations which would need to be completed by hand.

Case Study 1: "What are good lenses to use for a spectrometer?"

"Choosing the right lenses for a spectrometer depends on several factors, including the specific application, wavelength range, resolution, and cost. Here are some general recommendations for good lenses to use in a spectrometer:

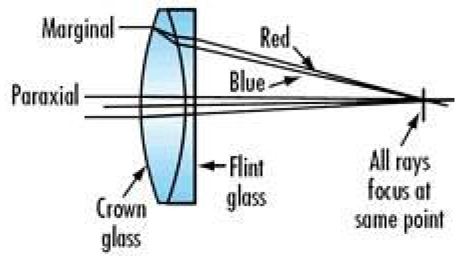
1. Achromatic Lenses

- **Description**: Achromatic lenses are designed to limit the effects of chromatic and spherical aberration.
- **Advantages**: They provide better focusing of different wavelengths (colors) of light, making them ideal for applications requiring high precision across a range of wavelengths.
- **Use Case**: Suitable for UV, visible, and near-infrared spectrometry." (OpenAI, 2024).

As referenced in the Appendix Case Study 1, ChatGPT provides multiple lens options that are general recommendations. It fails to target options that we could potentially use or consider in this project specifically. ChatGPT gives a brief overview of each lens and what it is used for, listing a small description, some advantages and use case (as seen above). What this would give us is a general overview of multiple lenses and in order for us to draw out a better answer, we would need to ask more specific questions in order to find out which option is truly best in our case. An example question would be something like "What are good lenses to use for a spectrometer that is cheap and suitable for high precision?" This will give us a smaller range of options that can make it easier to choose.

"Here are some examples of specific lenses that can be used in spectrometers:

- **Achromat lenses:** These are a good general-purpose lens for spectrometers that operate in the visible and NIR range. You can find achromat lenses from various retailers."



(Gemini, 2024)

Comparing Gemini's results with ChatGPT, Gemini focuses on two specific lenses providing more information including the description and how each lens would work with different wavelengths of the spectrometer. In addition, Gemini provides specific examples of lenses that can be used in spectrometers with pictures and diagrams. Gemini showcases general options so additional research would need to be completed to decide which lens to use. Gemini provides additional links through their detailed images. Although the pictures and figures shown are a beneficial touch towards the explanation, Gemini provides us with fewer options than what ChatGPT had. This could mean either two things, the first one is that Gemini was not able to process the other options that ChatGPT provided as lenses, or the other option is that Gemini had considered those lenses, but already decided that they were not optimal lenses to use for a spectrometer. Unfortunately, we are not able to know which of the cases it would be. Please reference Case Study 1 in Appendix B.

Case Study 2: "What microcontroller is more compatible with the raspberry pi 4?"

Just like our first case study, ChatGPT gave the most compatible microcontrollers from a general view. There are two possibilities we can gather from the response ChatGPT gave, and the first one is that ChatGPT lacks the capability to know fully about microcontrollers that are compatible with the raspberry pi 4 The second one, which is the more believable option, is that we are not asking deeper questions to get more in depth answers about the microcontrollers. ChatGPT gave useful tips and information as to include a "Key Considerations" section when choosing a microcontroller. This will allow us to seek more information about whether or not the microcontroller we choose is compatible with the raspberry pi, we can either continue asking ChatGPT to get to the answers we are looking for, or we can navigate to the websites that own each microcontroller to find out ourselves However, we still can't rely on ChatGPT to choose a microcontroller since it only provides a brief overview of each one.

Something that Gemini does that ChatGPT does not is include outer sources into the results. In the search results, Gemini provides suggestions to consider when choosing a microcontroller and two examples. For example, Gemini recommends the Raspberry Pi Pico and describes "it is also programmable in C++ and MicroPython" (Gemini, 2024) and includes the website link that the AI tool pulled it from, "Google Search found similar content like this:...” (Gemini, 2024). The link will take you to the direct page which is super useful to view the entire article to do additional research.

Case Study 3: Raspberry Pi 4

For this case study, we asked Gemini to provide a description of a Raspberry Pi 4. It gave a brief overview of the microcontroller. It also provided a couple links to purchase the Raspberry Pi which ChatGPT generally doesn’t provide. Here, I would need to ask Gemini more specific questions to get a more concrete understanding, as well as, doing more outside research. Below is ChatGPT’s response.

Like Gemini, ChatGPT gives an overview of the Raspberry Pi using similar phrases. However, unlike Gemini, ChatGPT asks a question to the user if there is something more specific we would like to know. This would imply that ChatGPT has something significant that Gemini does not have and that is humanistic qualities. It is more of a human quality than a robot quality to ask a follow up question after giving a response. This could imply that ChatGPT is technology more advanced than Gemini, but we do not know for sure. Here, we would need to ask ChatGPT more questions as well as search on Google to get a strong idea of what we are working with.

Programming Assistance

ChatGPT is an excellent resource for writing skeleton code and debugging the navigation system and robotic arm movements in our project. By leveraging the AI's advanced programming capabilities, we can significantly reduce development time and receive valuable troubleshooting assistance. These AI tools not only help us optimize time complexity but also suggest efficient algorithms that are best suited to execute the tasks at hand. Additionally, ChatGPT can provide insights into best practices and coding techniques, further enhancing the overall quality and robustness of our project.

Case Study 4: “How can I program the meca500 robot arm to pick up objects with a raspberry pi?”
“

```
import mecademicpy.robot as mecademic

# Define robot IP address
robot_ip = '192.168.0.100'  # Replace with the actual IP address
of your Meca500

# Initialize the robot
robot = mecademic.MecademicRobot(robot_ip)

# Connect to the robot
robot.Connect()
robot.ActivateRobot()
robot.Home()
```

```
# Function to pick up an object
def pick_object():
    # Move to the pick position (replace with your coordinates)
    # Close the gripper
    # Move to a safe position"
```
(Gemini, 2024)

We can see how the AI model can create a step-by-step guide on how to potentially set up the robotic arm and it can offer us any other advice on what we will need to move forward. Most of the steps that were suggested can be fact checked if needed, for example, in step two it tells us that python, PIP, and the Mecademic Robot Library are all needed software in order to create this application, so in order to fact check it, we would research each software and find out ourselves if those three softwares are actually needed for our application. (Check Appendix–B) It also provides a skeleton of code that we can follow and adjust to what works. This can expedite the programming phases tremendously and aid in potential excruciating debugging. Another thing to know is that if we were to ask ChatGPT to continue writing the code besides just the skeleton, it would do that, but the only way to guarantee that it is correctly coded would be to test it, either piece by piece or all at once. However, we must take this information with a grain of salt as it could be inaccurate and unhelpful, thus, when working with these components, it is imperative to read the respective manuals. We cannot simply rely on AI to code the robot, there must be outside knowledge to fully be able to code the specific tasks the robot needs to perform.

5.3 Cons of AI Language Models for ALiCS

Despite the benefits of using AI, there are some downsides when using tools like ChatGPT and Gemini. It's important to note that this is a machine generated platform and we cannot fully rely on the accuracy of the information that these tools present to us. Additionally, these AI languages lack the understanding and creativity that we as human beings inhabit which leads to the tool not fully understanding what we are asking for. In addition, with more of these AI tools available to us, a question of ethical standards arise. Even though we use these tools to help with the backbone of our project, it's important that we must cite the sources as well.

The biggest underlying con that no one really considers is how ChatGPT can negatively affect a person. Once someone starts using ChatGPT, they will start to think that it is easy to use and it gives a better and clearer answer than most, if not all search engines. This will cause the person to constantly use ChatGPT, whether it's for writing papers, homework, projects, or other personal activities. A side effect of that is this person will become lazy, always relying on ChatGPT for answers will not get them anywhere in life and thus, they will mentally be behind as compared to their classmates or peers. If we would want to use ChatGPT, it is important for us to not be reliant on it and only use it for partial work so that we can do the work ourselves so that we do not become over reliant on this one thing. Another con that is not currently a thing, but could possibly be, is how far will ChatGPT develop as an AI. Will it be able to one day surpass the capabilities of humans? For now ChatGPT is under our control but there is no guarantee as to what to come in the future.

Accuracy

A big limitation in using AI language models is that mathematical computations and complex calculations are usually incorrect and inconsistent. ChatGPT warns users at the bottom of the webpage, " ChatGPT can make mistakes. Check important info." This is a very accurate statement and we must be cautious when we are working with sensitive info that requires accuracy. This would mean that we should avoid strictly relying on only ChatGPT for our answers to our research, if we are researching, we should check it through multiple sources to make sure it is correct. It can also create false information about existing products or topics. For example, when asking about the specifications of certain parts suitable for our project, the data ChatGPT provides might not only be outdated, but wrong and fabricated as it copies the specs from another product. There are user manuals and datasheets dedicated to these parts on their respective sites that we must use. Usually when AI language models write code, there are many bugs associated with it. Another way to think about it, if a person is struggling to find any info on a component or whatever they are searching for, there is a good chance that when they search for it on ChatGPT that the information will most likely be misleading and incorrect. But, if it is a component that is more widely known, there is a good chance that the information could be correct, but it might be a combination of multiple sources. It's important to know that if any kind of code was received from an AI source, it should be thoroughly tested with many different variables, values, etc, to ensure that the code is actually performing as expected. However, trusting and using ChatGPT's code can actually cause more harm than good sometimes as it can lead programmers on the wrong path if they assume that ChatGPT or other AI is reliable.

Ethical Concerns

The use of AI language models such as ChatGPT in our project or any other project, raises some ethical concerns that should be considered and addressed. As AI technology improves, it becomes much easier to transition our workload to the language models. However, this reliance on ChatGPT can inadvertently reduce human expertise and judgment in the project, as over-reliance on AI might lead to undervaluing human skills and critical thinking needed for innovative and ethical engineering. There should be a healthy balance between AI influence and human input. Another issue that may arise when relying on AI is lack of accountability and transparency. When we use AI generated suggestions for our project, it can become difficult to trace those decisions and hold whoever is responsible for those important project details accountable. AI should only be a supplementary tool and is not a member of the project. Transparency should be required whenever the use of ChatGPT or other models is involved, and maintaining clear documentation is essential in mitigating these ethical concerns. The reason why we should be transparent when we use ChatGPT, is because if we use it without citing it as a source, it is considered plagiarizing and that is not something we should be doing.

Chapter 6 Hardware Design

6.1 Motor Controller

Considering the fact that for our design we plan to use motors, it would be best for us to add motor controllers along with our esp. The reason for that is because the motors we choose to add, (Robotdigg brushless DC motor) requires 24 volts to function and the ESP can only output around 3 volts, which would mean that it is insufficient to power the motor. The controller is also needed so that we are able to regulate the motor's speed, so we can make sure it is able to slow down or speed up depending on the situation. It can also change the motor's direction if we need to go forwards and or backwards. Since we are using a brushless DC motor, there is one motor controller that we determined that would closely match our requirements. The motor controller we plan on using is the RioRand Brushless DC Motor Speed Controller with an input voltage range of 6 - 60 volts, this motor controller is enough to power the motor we have chosen. The main concern of this motor controller is that it can only power one motor at a time, for that reason we would need to have four of them. The reason why is because we have come to an agreement to use four motors to control the entirety of the rover. The reason why we made that decision is because it will be better for the control of the rover to have access to every side instead of just the front and the rear. What this means is that we had four motor controllers, one for each motor. This image below will show the overall architecture of mainly the rover part. Another thing to consider for the image is that the ultrasonic sensors will also be a part of the rover design. When they detect an object that is too close to the rover (probably less than 30 centimeters), it will send that info to the ESP32, which will instruct every single motor controller to stop and that will end up turning off the motor.

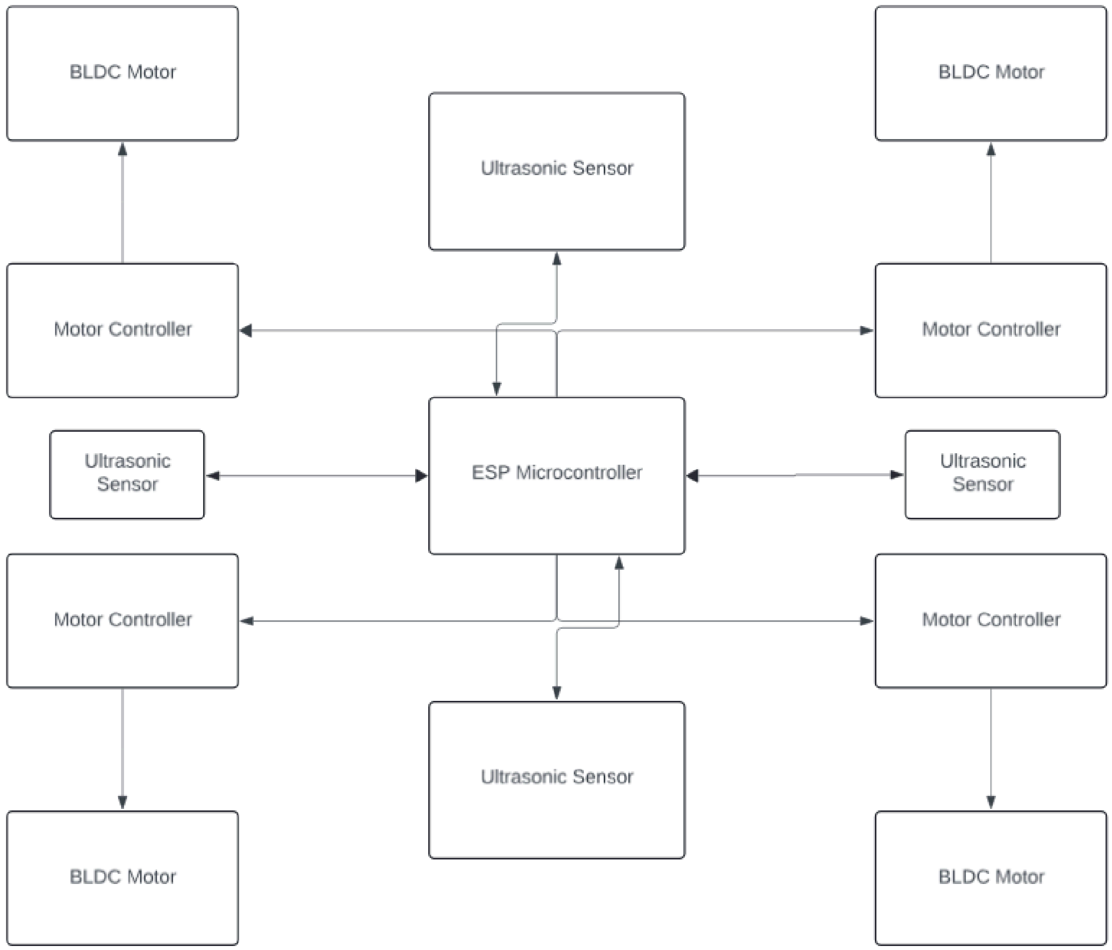


Figure 6.1.1: *Components of the Motor Part*

The motor controller we are using will use control input, direction control and stop inputs that will be used with our microcontroller, specifically the GPIO pins. The MA, MB, and MC are the pins for the phases of the motors that will connect to the motor. The Vcc and GND pins are the power output pins. There are also five more pins that connect to the hall sensor of the motor.

When it comes to how the motor controller makes the motor go backwards, or forwards, it uses something called the six-step driving method. A six-step driving method is common when using a brushless DC motor because it has three phases. How it works is that there are three phases, A, B, and C, or in our motor controllers case, it is labeled as MA, MB, and MC, and for the motor to turn 60 degrees, it would need to move by creating a rotating magnetic field by energizing the phases. A better way of explaining it is that for the first step phase A would be positively energized and phase B would be negatively energized, then the second step would be phase A is still positively energized and phase C is negatively energized. You repeat these two steps while replacing phase A with phase B, then repeat it again with replacing phase B with phase C, and what you have is 6 steps and in every step the motor is constantly rotating 60 electrical degrees. This is how you get the motor to drive forwards, but if you want it to drive the motor backwards, you will then need to put the steps in reverse. The truth table below shows how exactly it will work, assuming that the enable pins, (EN) are on.

Table 6.2.1: *Driving Logic For Driving Straight*

Step:	Phase A	Phase B	Phase C
1	PWM	LOW	HIGH
2	PWM	HIGH	LOW
3	HIGH	PWM	LOW
4	LOW	PWM	HIGH
5	LOW	HIGH	PWM
6	HIGH	LOW	PWM

Adding on to that, we should consider how we would be able to use this motor controller and our brushless DC motor to turn left or right. The important thing about our brushless DC motor is that since it has 3 phases, we would have to implement the six step driving method no matter what. Also, since we are using four motors and four motor controllers, we would need to write separate code for situations like these. For example, in order for the rover to turn to the right, the front left and back left motors would need to use more duty cycle (PWM), meaning more power, than the front right and back right motors in order to turn to the right. This is something that can be adjusted as we go along in order to pick the way we believe is best, but for now, we believe that in order to turn to the right, the front left and back left motors should use PWM and the front right and back right motors should use PWM divided by 2, so that way it ends up using less power than the two left motors, causing the rover to turn right. In order for the rover to turn left instead of right, it would be the same as turning to the right but with the PWM values switched, so front left and front back would use PWM divided by 2, and the front right and back right would use just PWM. Here is an example of how the logic should look and be implemented for turning to the right.

Table 6.2.2: *Driving Logic For Turning Right*

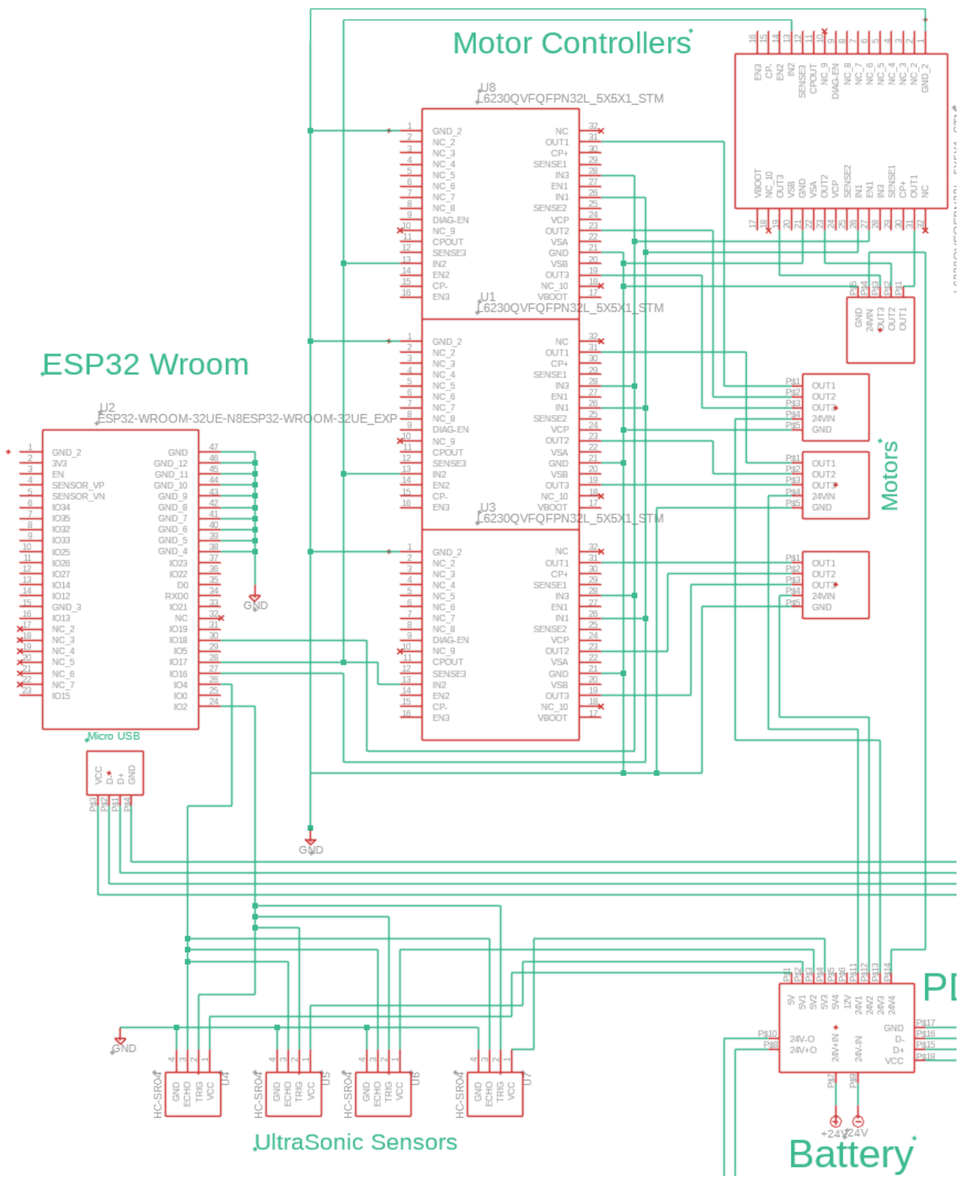
Steps	Phase A Left	Phase B Left	Phase C Left	Phase A Right	Phase B Right	Phase C Right
1	PWM	LOW	HIGH	PWM/2	LOW	HIGH

2	PWM	HIGH	LOW	PWM/2	HIGH	LOW
3	HIGH	PWM	LOW	HIGH	PWM/2	LOW
4	LOW	PWM	HIGH	LOW	PWM/2	HIGH
5	LOW	HIGH	PWM	LOW	HIGH	PWM/2
6	HIGH	LOW	PWM	HIGH	LOW	PWM/2

Another thing to note is that this motor controller has two different six step driving methods, the first one is the six step driving method with current control, and the second one is the six step method with Back Electromotive Force (BEMF) zero-crossing detection. What the current control one does is that it allows you to adjust the amplitude of the motor current, meaning that the torque of the motor can be regulated. This will be necessary if we are able to determine whether or not enough torque is being used for our device. How the BEMF zero-crossing detection works is that it is able to compare the voltage of the phase it is currently in to the center voltage in order to figure out what position the motor is in, so that way, there will not be any need for any sensors or encoders to figure out the position of the motor. With those two options in mind, we believe that our best option is the current control method so that we can have better control of the torque and speed so that way we can have better accuracy. Also if things are proven to become too difficult to implement the current control, we do not need to implement either, they are just an additional part of the motor controller.

6.2 Ultrasonic Sensor

Since currently, we are discussing what will be in the schematic of mainly the motor/rover aspect, we might as well talk about the inclusion of the ultrasonic sensor. There are four main pins that the HC-SR04 sensor (the sensor we decided on using), has and the first one is the VCC pin. The VCC pin is the pin that is used to supply power, it would usually look like a power supply connected to the VCC pin. The Trig pin is the second pin and it is used to trigger the ultrasonic sound pulses so that the waves can head towards all the objects and then bounce back. Once the waves bounce back, the Echo pin is able to produce a pulse when the wave that came back is received. Once it has been received, it can then find out the distance by calculating the time it took for the signal to be detected. The last pin is of course, the GND pin, and it should be connected to where all the other ground pins are located.



0

Figure 6.2.1: ESP32 Schematic

6.3 LiDAR

The Lidar for our project will be the first major use of our raspberry pi and will have to have a strong, stable connection to the raspberry pi to constantly update the data and delete the data to keep current information without issue of a memory leak. Therefore we made a simple block diagram to show the process of the connection points of our lidar and how it will functionally work in unison with our raspberry pi.

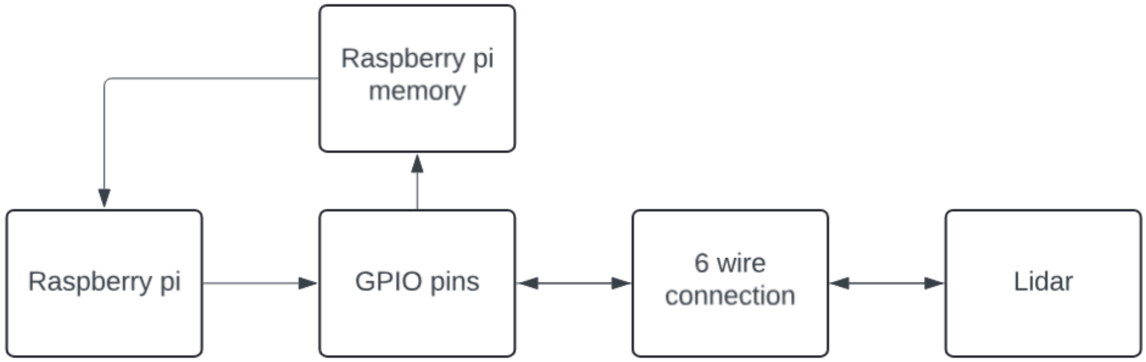


Figure 6.3.1: *Lidar Block Diagram*

The figure above shows the simple process of how our lidar will communicate with our raspberry pi. This figure represents that power will be sent from the raspberry pi to the GPIO pins to power the lidar. Then the data from the lidar will be quickly sent back to our raspberry pi's memory to analyze and update the information for our software to make the correct adjustments. The lidar for our project will use Slamtec RPLiDAR which is ideal for the quality of the product. This product will take a 5V input voltage from the raspberry pi that it will be connected to and needs to be able to to maintain that power while scanning constantly for obstacles. We need the lidar scanning constantine lit withs high scan rate of 10 Hz and 8000 Hz sampling frequency. In addition, it will be able to see the obstacles coming from 12 meters away making it one of the best indicators for our bot. The Lidar has two ways that it can connect to our raspberry pi either straight through the GPIO pins and then will connect to a small PCB on the bottom of the lidar. The other option is an attachment that connects from a USB port on the raspberry pi to a tiny PCB and from that PCB to the PCB on the bottom of the lidar. This option seems not optimal but is a backup if needed. Once the information from the lidar is sent to the raspberry pi, the raspberry pi will then be able to send the information that was received, to the other parts of the design. Most of the information that was received will be sent to the rover area of the design. It will take the information given, and convert it into a global map, if that was not done already, then it will use the global pathing algorithm to convert it into an optimal path that will be used by the local pathing algorithm that will essentially create a path that is free of any obstacles.

We connected the GPIO pins on the raspberry pi to the lidar. Looking at the datasheet in the appendix C you can see what pin connection we have used to power and run our lidar. We made use of pins 2, 4, 6, 8, 10, 12. 14. Pins 2 and 4 are both the 5V power pins for the lidar. Pins 6, 8, 10, 12, 14 are responsible for the TXD0, RXD0, and GPIO_GEN1. The TXD0 is used to send an output of the serial devices on a raspberry pi and sends a data byte that is serially shifted at the TX pin at a specified baud rate. It generally uses 8 data bits, no parity bits, and 1 stop bit. The RXD0 is almost the exact opposite; it receives the new data that is shifted onto the raspberry pi. Using these 6 pins it is able to fully communicate with our raspberry pi and transfer and receive the data we need to constantly make ALiCS not crash or run into objects or people. This figure above also shows the PCB of the Lidar which is a nice addition to show what is on the PCB and what parts we are working with to make a better understanding of our lidar and how to get the best use out of it through our raspberry pi. However, in the slight chance that we would have to make use of a USB port on ALiCS it comes with another connector that has all six ports that are needed to power and send data from the lidar that is connected to a specific port on the additional tiny PCB that comes with the lidar. Then we would just need to have a USB to micro usb cable connected to the raspberry pi to the tiny PCB. Then it would functully still work the same. The last but another important part is this lidar comes with standoffs as seen in the figure above to show how we secure the device to ALiCS and keep all our data coming from the lidar accurate.

6.4 Camera

The Camera was one of the main reasons why we chose the raspberry pi. The camera module port is one of the main reasons that we choose because of the ease of use of connecting a camera and the high level quality that the camera has the ability to take from this port. The camera port is a two lane MIPI CSI direct port that takes amazing video. The raspberry pi also comes with a lot of storage for our video feed and additional storage from SD cards that could save data if we needed to improve our algorithm.

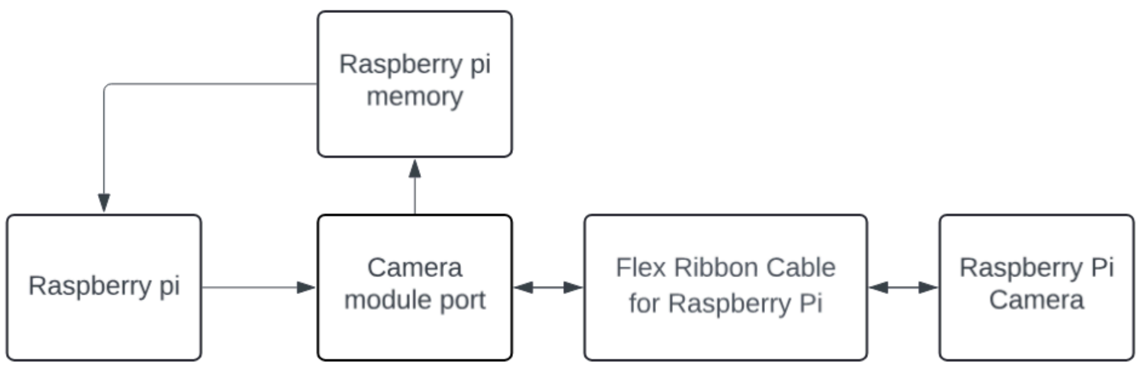


Figure 6.4.1: *Camera Block Diagram*

The figure above shows a brief understanding of how the board will communicate with the camera to send data and power to and from the camera. The camera port can be seen in figure 6.3.2 on the middle left side of the board to give a good understanding of how big the camera port is and how wide the flex cable will be. The camera we decided to use will be able to take up to 1080p with 30 frames per second video quality. This means we had high resolution pictures to feed to our algorithm to tell the robot arm where trash is and its precise location to pick up the trash to be sorted into one of our two bins. The raspberry pi camera module port is extremely impressive and still one of the major reasons why we chose the raspberry pi. The raspberry pi also is powering the camera through this port and the flex cable making life easier on us. The raspberry pi memory will quickly take the data given by the camera and feed the information straight to the raspberry pi processor to be analyzed and then after quickly deleted. Another piece is the flex ribbon cable that comes with the camera may be too short for our needs and we might need to buy a bigger one and cut it to our specific needs that just add the ends back on the both PCBS to make it function the same. In addition the small PCB the camera comes on will need to be mounted down onto the robot and secured tightly so that environmental factors like wind will not move it even slightly so that it doesn't mess with our calculations for the distance taken from the camera so that our robot arm stays accurate.

6.5 Robot Arm

The robot arm will need to communicate with the Raspberry pi for initial testing to make sure everything runs correctly. The robot arm connects to a computer through an ethernet cable directing into the robot arms connector port. Even though our raspberry pi has a direct ethernet port we are still planning to use the USB for the data transfer of our program to the robot cause of uncertainty of how well the ethernet port will work for transferring our program data cause the ethernet was made for internet connection not data transfer from the raspberry pi. The robot arm also required too much power from the raspberry pi to run so we needed another component in our system to power the robot arm. This component is the robot arm power brick. This brick also has a built-in safety that requires an override component on the brick.

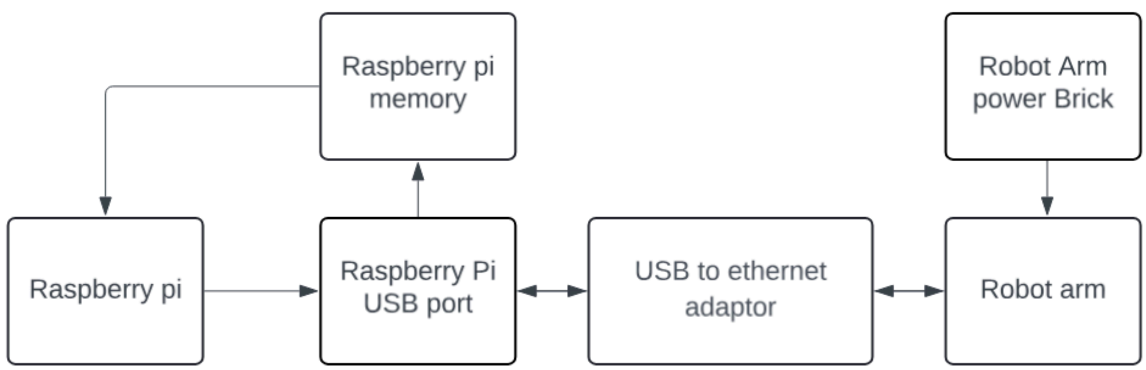


Figure 6.5.1: *Robot Arm Block Diagram*

The figure above shows a basic understanding of how we connected our raspberry pi to the robot arm. Notice compared to the others connected to the raspberry pi this will require a power brick that will need to be powered directly from the battery not from the raspberry pi because of the massive power draw required for the robot arm. The robot arm will receive instructions from the raspberry pi based on the information from the camera data. The robot will send data directly through the ethernet cable that is a M12 D-code to RJ45 cable. Then it will use coordinate data based on the calculation from the camera data to pick up the trash and brick the trash directly to the spectrometer to be scanned for sorting. The raspberry pi will save the robots constand coordinate data and constantly have to update the data based on position of the robot arm to ensure accuracy of the arms movements. The robot arm has holes on it built in so we can secure the robot arm down to ALiCS and keep the positional data from changing from environmental factors because it is not protected by the box and is left out in front of ALiCS. We also needed to make sure even while screwed into the frame that we didn't let the wind move any of the joints to mess with our positional data for the arm.

The smart power brick we used for the robot arm power brick on our robot arm requires 24V and 60 mA to power the power brick for our robot. We also needed to make use of the D-Sub 15-pin dongle to override the safety I/O port. The input for the actual power brick itself is a 4A and 90-250V AC power. This became a major power draw from our battery so we needed to make sure this power draw was worth it. The way this connects to the robot uses a DC power cable called a M12 circular male to M12 circular female power cable. This smart power brick will also need space for it specifically built into the design of ALiCS while staying close to the battery and able to connect directly to the robot arm.

With all this information for how we made use of the Raspberry Pi 4 we can make a Schematic with all the connection points needed from the raspberry pi. This will show port connections that are required to connect these parts and which parts of the boards we used and did not use.

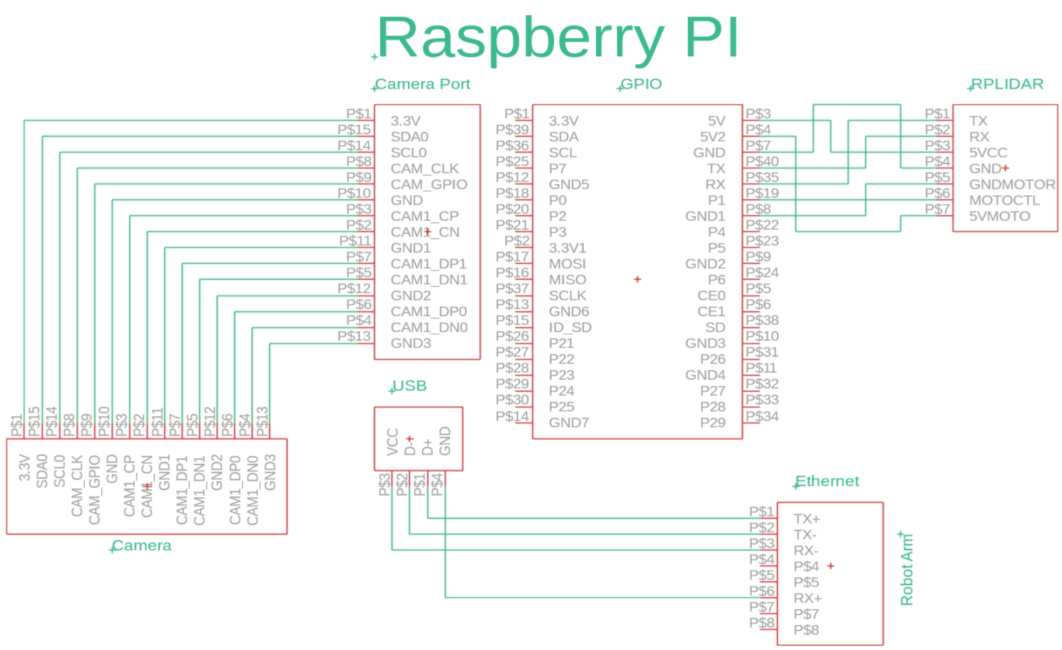


Figure 6.5.2: *Overall Schematic for Raspberry Pi Connections*

The figure above shows the finished overall schematic for our Raspberry pi for this project. In total we used the camera port, one usb port, and seven GPIO pins total. It also shows the power brick for the robot arm and the override required to make use of the robot arm. This means we don’t need to use a lot of the functions of the raspberry pi, but will be usable in case we need additional parts for our project. Also this makes a sort of problem for our raspberry pi with its location will need to be centralized so that it can access all required components without issue. However, overall it makes the raspberry pi and essential piece for a project and we are sure to make great use out of it.

6.6 Spectrometer - Optical Design

A fundamental portion of our design will incorporate a spectrometer to measure materials recovered from the beach and determine whether these items are recyclable or plain trash. Spectrometry is a general term used for analyzing the specific spectrum and employed for the analysis of materials. The Czerny-Turner imaging spectrometer consists of one reflective diffraction grating and two concave mirrors. The first mirror is a collimating mirror which collimates the beam and makes it equivalent to the grating surface of a spectrometer. The next mirror is known as a focusing mirror which focuses the input light from source to the image sensor. Czerny-Turner configuration is compact and flexible spectrograph design. Asymmetrical geometry, Czerny-Turner configuration is designed to produce a compressed spectral field and good quality coma correction at one wavelength. It contains spherical aberration and astigmatism at all wavelengths.

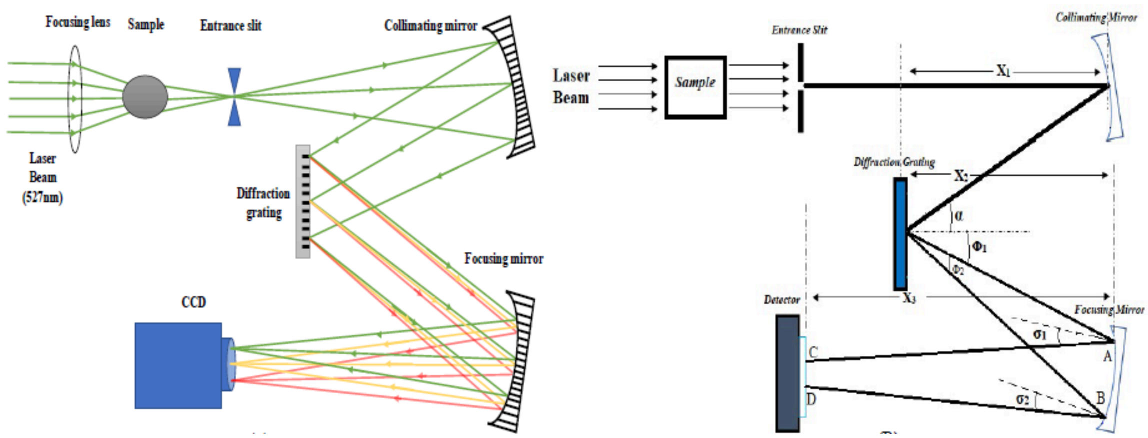


Figure 6.6.1: *Analysis of Spectrometer Imaging Plane*

In Figure 6.6.1, angles $\phi 1$ and $\phi 2$ are the diffraction angles of the two edge wavelengths after passing through the diffraction grating; they intersect the focusing mirror at *A* and *B*. $\sigma 1$, and $\sigma 2$ are the angles between the reflected light of the two edge wavelengths on the focusing mirror and the horizontal line. Finally, they gather on points *C* and *D* of the linear CCD. Using the diffraction grating formula,

$$d * \sin\theta = n\lambda \tag{6.6.1}$$

Where *d* is the grating constant, diffraction angle ϕ_1 and ϕ_2 can be calculated as:

$$d * (\sin(\alpha) + \sin(\phi_1)) = n\lambda_1 \tag{6.6.2}$$

$$d * (\sin(\alpha) + \sin(\phi_2)) = n\lambda_2 \tag{6.6.3}$$

Setting the diffraction order equal to one (n=1) and minimizing size (length) of the focusing mirror to reflect the two edge wavelengths according to the geometrical structure:

$$AB = x_2 * [tan(\alpha + \phi_1) - tan(\alpha + \phi_2)] \quad (6.6.4)$$

After reflection from the focusing mirror, the horizontal angle of the diffracted rays is changed by 2*θ as given below.

$$\sigma = \alpha + \phi - 2*\theta \quad (6.6.5)$$

The size of the image is calculated using the geometrical structure:

$$CD = AB - x_3 * [tan(\sigma_2) - tan(\sigma_1)] \quad (6.6.6)$$

Equation 6.6.6 shows the minimum length of the CCD detector surface. Based on the original configuration and using the equations (1- 6), all the necessary parameters to design the spectrometer can be determined.

Solving for Equations 6.6.1-6.6.6, when NA = 0.50, d = 0.100lines/µm, α = 16°, ϕ_1 = 7.2° and ϕ_2 = 25.2°, n = 1, from detector data sheet CD = 29.1mm :

$$d*(sin(\alpha) + sin(\phi_1)) = n\lambda_1 = (0.100lines/\mu m)*(sin(16°) + sin(7.2°)) = \lambda_1 = 400.9nm$$

$$d*(sin(\alpha) + sin(\phi_2)) = n\lambda_2 = (0.100lines/\mu m)*(sin(16°) + sin(25.2°)) = \lambda_1 = 701.4nm$$

$$AB = 96.1mm * [tan(16 + 7.2) - tan(16 + 25.2)] = \sim 125mm$$

$$\sigma_1 = \alpha + \phi_1 - 2*\theta = 16° + 7.2° - 2* 1.9° = 19.38°$$
$$\sigma_2 = \alpha + \phi_2 - 2*\theta = 16° + 25.2° = 37.38°$$

$$CD = AB - x_3 * [tan(\sigma_2) - tan(\sigma_1)] = \text{solve for } x_3 = (AB - CD)/([tan(\sigma_2) - tan(\sigma_1)])$$
$$x_3 = (125mm - 29mm)/(tan(19.38°)-tan(37.38°)) = 146.99mm$$

With these calculations, we were able to spec out some optical components from Thorlabs Inc, which are tabulated below.

Table 6.6.1: *Calculated Design Component Specs*

Component	Description	Manufacturer
100um	Ø1" Mounted Slit, 100 ± 4 µm Wide, 3 mm Long	Thorlabs PN: S100K
Collimating Mirror	Ø1" UV-Enhanced Al-Coated Concave Cylindrical Mirror, f=100.0 mm	Thorlabs PN: CM254-100-G01
Focusing Mirror	Ø1" UV-Enhanced Al-Coated Concave	Thorlabs PN: CM254-100-G01

	Cylindrical Mirror, f=100.0 mm	
Diffraction Grating	Ruled Reflective Diffraction Grating, 1200/mm, 500 nm Blaze, 25 mm x 25 mm x 6 mm	Thorlabs PN: GR25-1205

Optical Design Parameters (Zemax)

Table 6.6.2: *Zemax Design Parameters*

	Surface Type		Comment	Radius	Thickness	Material	Coating	Clear Semi-Dia		Chip Zone	Mech Semi-Dia	Conic	TCE x 1E-6	Par 1(unused)	Par 2(unused)	Par 3(unused)		Par 4(unused)
0	OBJECT	Standard		Infinity	5.000			0.000		0.000	0.000	0.000	0.000					
1		Coordinate Break			0.000	-		0.000		-	-		-	0.000	0.000	10.000		0.000
2	STOP (a	Standard	COLLIMATOR	-10.000	0.000	MIRROR		1.000	U	0.000	1.000	-1.000	0.000					
3		Coordinate Break			-3.500	-		0.000		-	-		-	0.000	0.000	10.000	P	0.000
4		Coordinate Break			0.000	-		0.000		-	-		-	0.000	0.000	-20.000	P	0.000
5	(aper)	Diffraction Grating	GRATING	Infinity	0.000	MIRROR		0.520	U	-	-	0.000	0.000	0.120	1.000			
6		Coordinate Break			3.500	-		0.000		-	-		-	0.000	0.000	-20.000	P	0.000
7		Coordinate Break			0.000	-		0.000		-	-		-	0.000	0.000	10.000	P	0.000
8	(aper)	Standard	FOCUSING	-10.000	0.000	MIRROR		1.000	U	0.000	1.000	-1.000	0.000					
9		Coordinate Break			-4.820	-		0.000		-	-		-	0.000	0.000	10.000	P	0.000
10	IMAGE	Standard		Infinity	-			0.444		0.000	0.444	0.000	0.000					

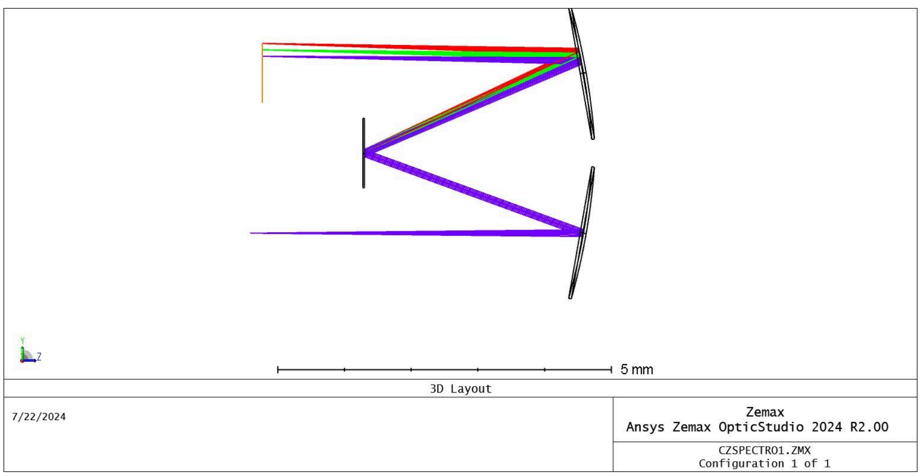


Figure 6.6.2: *Zemax 3D Layout Spectrometer*

The lens data editor window in Zemax, shown in Table 6.6.2, defines the different designing parameters, such as the surface type of each lens, the radius of curvature, thickness, and focal length. We adopted the physical optics propagation (POP) algorithm over geometrical ray-tracing because geometrical ray-tracing can only be used when diffraction limits are negligible. These parameters aren't exactly spec'd to our optical design, I kept getting an error while developing the optical design. I will continue to get this drawing spec'd to our exact parameters.

In conclusion, The spectrometer utilizing diffraction grating incorporates the standard Czerny-Turner system features with the flat-field grating spectrometer system. The spectrometer system's modeling approach is suggested by introducing the corresponding relationship between pixel and wavelength, linear CCD receiving surface length, and image surface dimension. The excitation light with a center wavelength of ~557 nm is used for simulation and optimization in the Zemax optical modeling program. Spot diagram, irradiance plot, MTF, and geometric encircled energy for Spectra of the sample, which lie in the visible range, can be optimized and simulated once I continue to develop in the optical design in Zemax.

6.7 Power Supply

The power supply is a critical component of our project, essential for ensuring that our robot operates effectively over extended periods. This system must not only sustain the robot for a substantial duration but also efficiently distribute power to various subsystems as needed. Moreover, it should optimize power consumption by deactivating certain parts when they are inactive.

The fundamental design of this system involves sourcing power from a battery to a power distribution board. This board is tasked with delivering the correct voltage to specific components and includes a USB port to power devices like the Raspberry Pi and ESP32. Additionally, the power distribution board is designed to accommodate two-wire connections to a wall outlet, ensuring the robot arm can operate continuously without interruption.

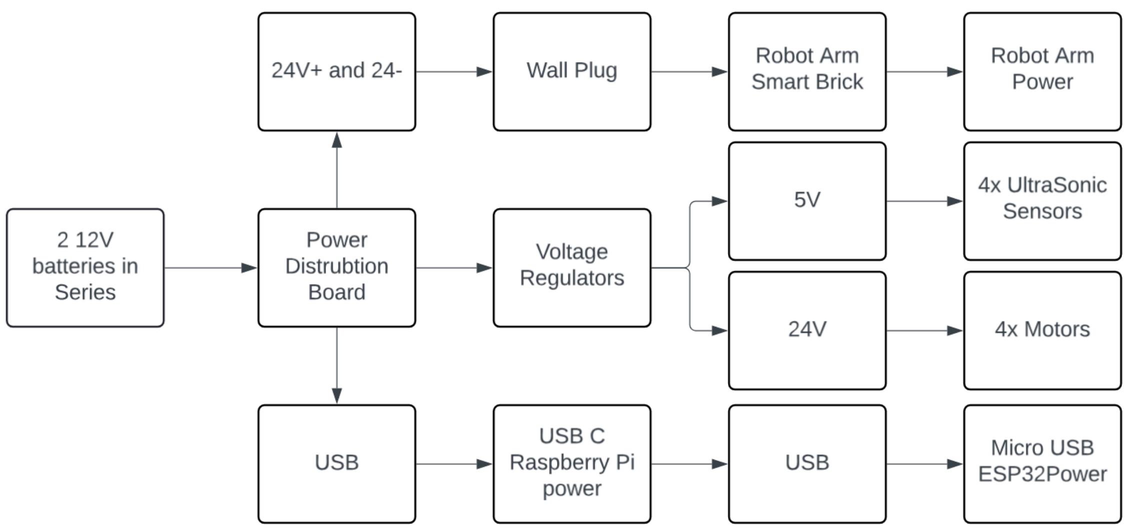


Figure 6.7.1: *Power Supply block diagram*

Looking at the figure above, the entire power supply system is shown to display where the power is distributed and how each part is connected with each other. With this we can easily make the correct power distribution board needed for our system. The power distribution board will use two 12V batteries in series to power our system. This board will need a 24V+ and 24V- for the robot arm. It needs four 5V and four 24V Voltage regulators for the ultrasonic sensors and a 5V usb to power our raspberry pi.

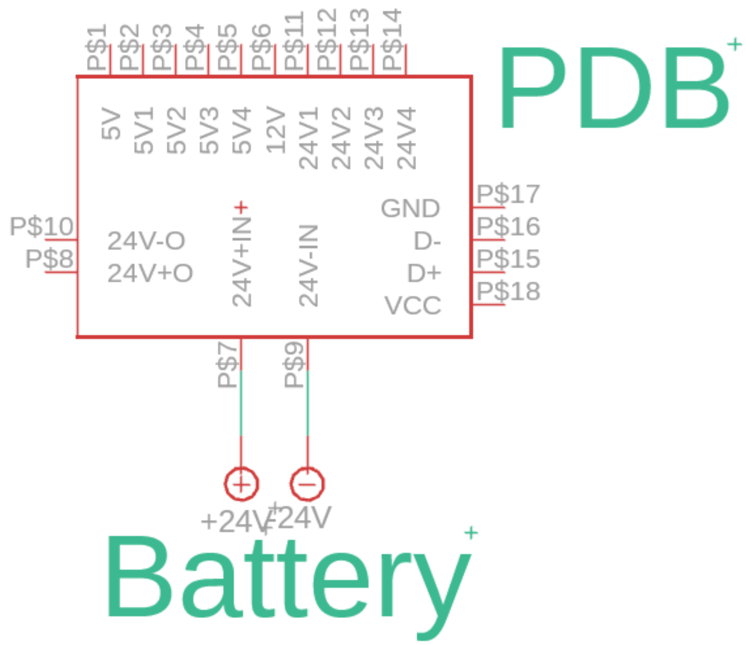


Figure 6.7.2: *Power Supply Schematic*

From the schematic above, we can see the completed design for the schematic of the Power Supply. This small system will have all the needed pieces we need for our system to work and run properly. The system has a USB port on the right side of the schematic to connect with the raspberry pi however it will need an adapter that is easily obtainable. The top of the power distribution board will consist of all the voltage regulators we need for our system. There may be additional voltage regulators added in the future if needed. These voltage regulators are needed to power the ultrasonics sensor and motors. On the left side is the 24V+ output and 24V- output needed for our wall plug to power the robot arm. Then on the bottom of the schematic it shows the battery connectors that will be used to take the power from the battery.

With this we have the final component needed to combine all the parts in our system to form an overall schematic for our system to complete the hardware design portion. This overall Schematic will show all the connections needed to make the complete system work using all of the designs made so far together into one massive schematic that will be the blueprint for how we make and prototype the project. This will show how the power from the power supply will be correctly distributed to all all the required parts to power our system. This will all show some middle men parts like the wall plug and power brick required for the robot arm. This also will show how the USB will connect the raspberry pi to the ESP32 board.

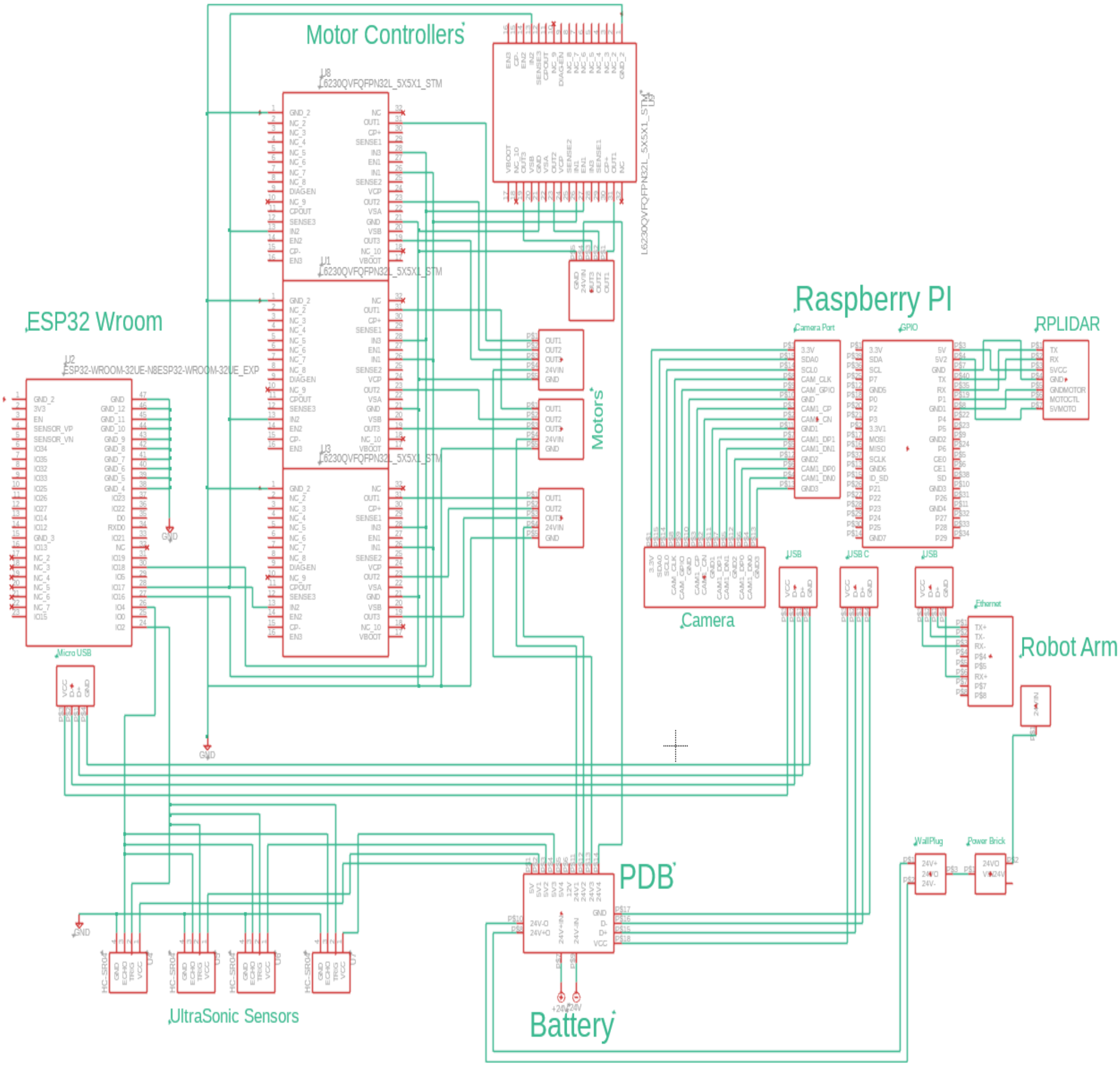


Figure 6.7.1: *Overall Schematic*

The Figure above shows the final design for our prototype layout that we constructed and tested for our project. With this layout we can make the hardware design complete and have the capability to make a fully functioning prototype.

Chapter 7 Software Design

The software involved in AliCS is just as prominent and important as the hardware design planned. Given the complexities of each process, the software is divided into many smaller components, the software responsible for the rover's movements, motor control, and steering, the robot arm's movements, path planning, and object detection. These processes work together through the Meca500, raspberry pi and the ESP32 microcontroller. The software is designed to be modular and extensible, allowing for easy updates and integration of other features.

7.1 Software Architecture

Software Use Case Diagram

Figure 7.1.1: *ALiCS Software Use Case Diagram*

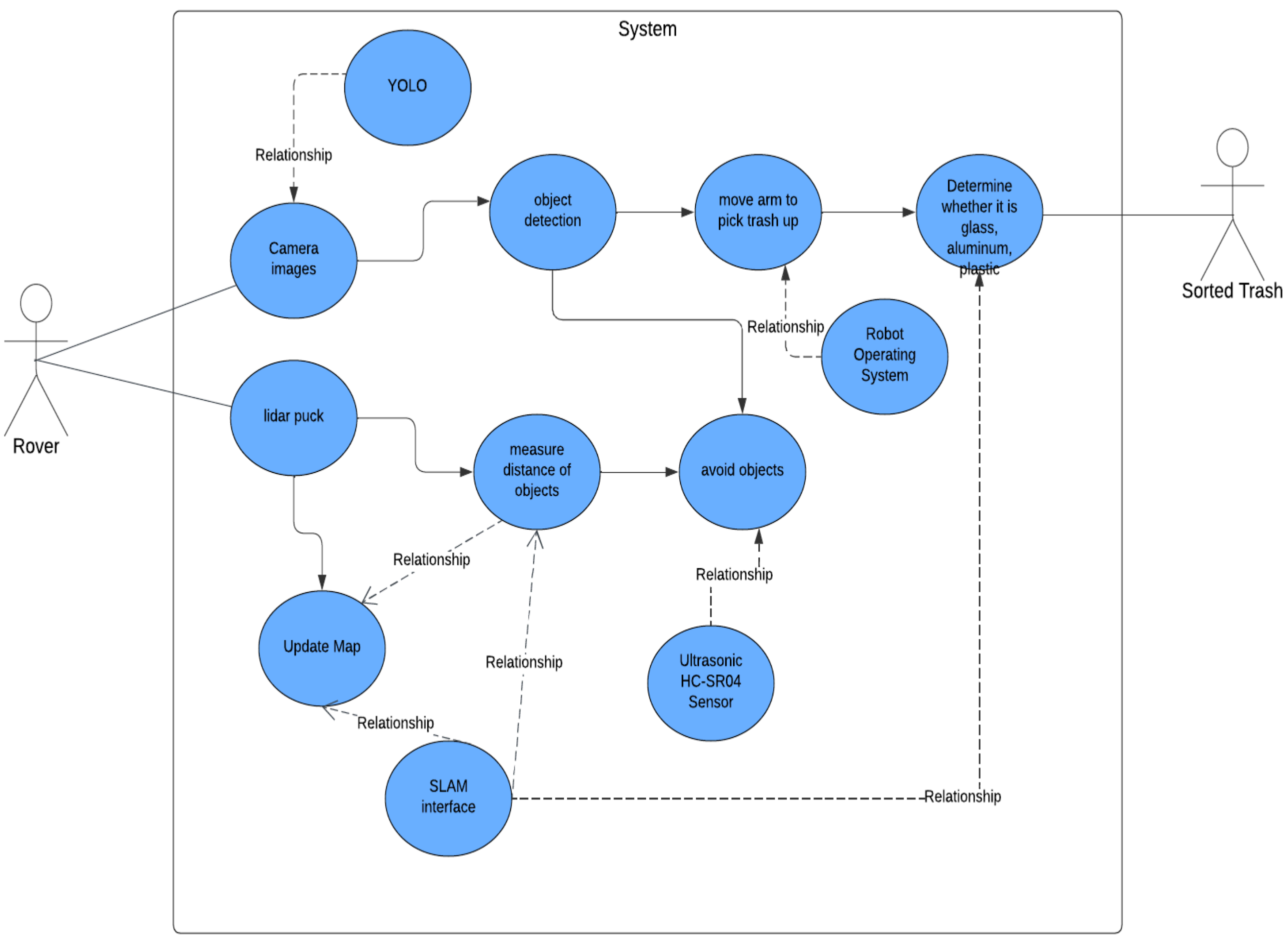


Figure 7.1.1: *ALiCS Software Use Case Diagram*

The software architecture for ALiCS is designed to ensure seamless integration and efficient operation of all its components. This section details the various software modules and their interactions. The use case diagram above illustrates different use cases for the software involved with ALiCS. The Ultrasonic sensor is dedicated to avoiding objects in the robot's path by providing real-time distance measurements. This data is used to adjust the rover's trajectory to prevent collisions. The rover uses its camera images while working with the YOLO software to detect objects. After object detection takes place, the arm's movements are next to position itself to pick up the unknown object. The robotic arm control involves precise control algorithms to ensure the arm can reach, grasp, and lift the trash accurately. All robotic arm handling is done in its user-friendly integrated software, "MecaPortal." On the other side we have the lidar puck measuring distance and updating the surrounding map of the rover. The SLAM interface then pieces these components together and meets at the end for the final trash sorting process. At the end of the process, after collecting objects, the system sorts the collected items to determine which are trash and which are not. This is done using the spectrometer data and predefined sorting algorithms. Further optimization of our software is done by implementing machine learning, through many tools designed to improve ALiCS' performance discussed later in this chapter.

Software Data Flow

The diagram below is a high-level overview description of how data flows throughout the various components and processes of ALiCS' software. At the top of the figure are the external entities and key components of ALiCS, which each transfer data to their respective process. The first of these components is the Ultrasonic sensor, which detects obstacles and provides obstacle data. This data is used for the Obstacle Avoidance process and then sends the necessary avoidance data to aid in proper navigation and path planning. Next external entities are the Camera and Spectrometer, which send image data and material data to the Object Detection phase. Object location data is sent out to the Navigation process, and spectrometer analysis is performed, to then carry over the material data to trash collection and sort the trash. Once Navigation and path planning are set, Rover Movement outputs the respective motor movement data, specifically the speed and direction the motors need. Lidar sends out distance measurements to the SLAM module to update and maintain the map, and that map data is used for Path Planning and Navigation. Last key component is the data flow from the Meca500 robotic arm, where its current position data is sent to Arm control for proper movements before trash is collected. Once trash is correctly collected or not, that data is shown as collection status. Other relevant data that is not shown in the diagram is the battery status data that is necessary to inform the robot to enter its low battery/charging state, and recharge accordingly before continuing operation.

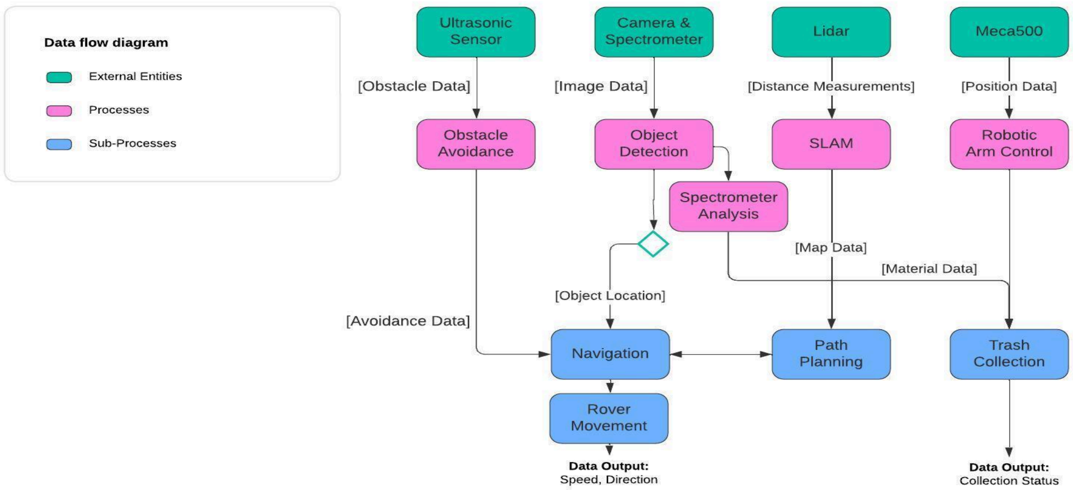


Figure 7.1.2: *ALiCS Data Flow Diagram*

Software State Diagram

The Software State diagram below shows the various states ALiCS can be in and activities ALiCS takes part in, such as Initialization, Path Planning, Collecting Trash, and Charging, and the transitions between these states based on different conditions and events. The Initialization state is the starting point for ALiCS. During this stage, the system can power up, perform self-diagnostics, and initialize all software and hardware components on board such as the LiDAR, spectrometer, actuators (Robotic arm and motors), the RaspberryPi and sensors. Once everything is confirmed working and initialized, ALiCS is able to transition into the next state. Next state would be sensor calibration where the sensors would be able to warm up and make sure they are operating effectively. Once again, once calibration is complete, the robot is ready for the next stage. Mapping and Localization is the stage where the robot begins mapping its surroundings, looking for obstacles, and prepares for the path it will soon take. Using LiDAR and the camera, ALiCS is able to detect objects in the map and use this information to make optimal decisions in determining its path plan. During Path Planning, the rover must calculate the most efficient path to navigate the beach, avoid obstacles, and look for trash.

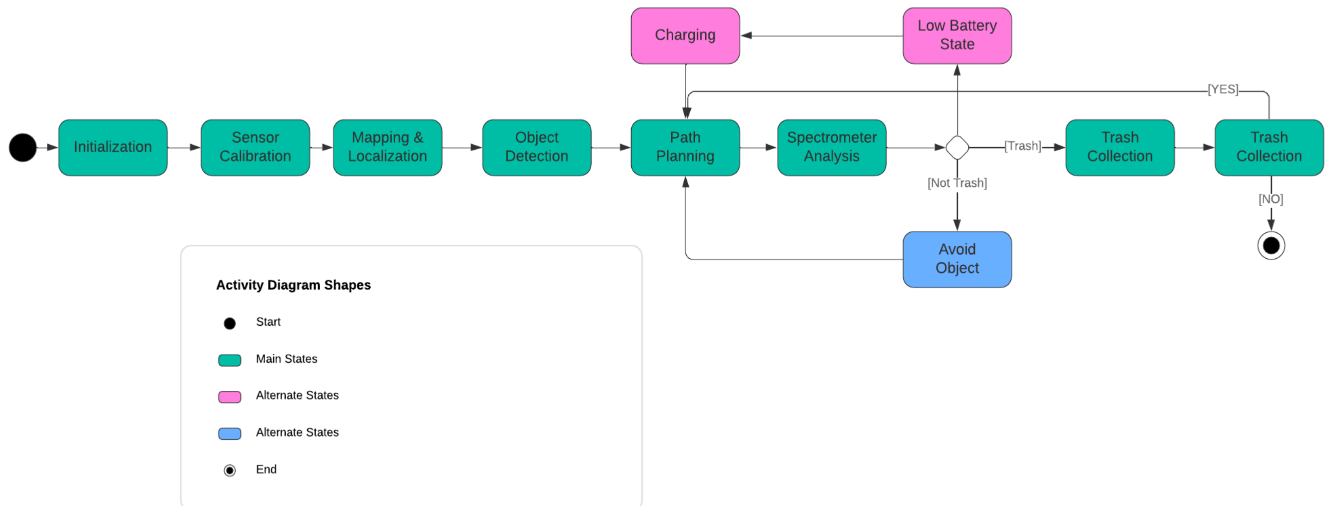


Figure 7.1.3: *ALiCS Software State Diagram*

It does this by combining data from the LiDAR, camera, and SLAM algorithms to be aware of its surroundings and determine an optimal route. Then ALiCS transitions to the Trash Collection state when it reaches a designated location identified as containing trash. If an obstacle is encountered, it enters the Avoid object state, and may return to Path Planning to recalculate the route. In the Trash Collection state, the robot maneuvers to position itself to prepare to pick up trash and place it in one of the storages on board by using the equipped robotic arm and vision systems to pick up the item. Once the item is picked up, the spectrometer must first identify the material of the item and help determine that the object is or is not trash and sort it correctly when picked up. When trash is correctly picked up and/or sorted, it can return back to Path Planning and begin the process anew. However, if all tasks have been completed but the rover is low on battery, it enters the charging state and returns to path planning. At any point in the process, if ALiCS has an error or technical difficulty, it will transition into the Fault state, which can be caused by many factors, such as environmental obstacles, software bugs, or even hardware failure. In this state, ALiCS will stop all operations and attempt to address the issue itself, by entering the Initialization state, but also signaling for human intervention in the case that it is not possible to resolve the errors present. Making the state diagram as simple as possible helps in streamlining and breaking down the software process into subsystems which enables us to debug and test step-by-step.

Class Hierarchy and Design

The class hierarchy begins with ALiCS as the parent class, or base class for all components involved. The Spectrometer class inherits from ALiCS and handles analysis of the objects to classify whether they are trash or not. This class can work hand in hand with the robotic arm class as it must detect if it's trash before collecting and storing the object. The Actuator class stems from the parent class as well and handles all actuators the robot contains, in this case the Meca500 arm and the motors. The robotic arm will perform trash collection as necessary, and the motors will control the movement of the rover as well as the speed. The Navigation class is responsible for the path planning and following, which stems into the SLAM class that is responsible for simultaneous localization and mapping, including map updates. The Sensor class contains all sensors used in the project, including the battery sensors which determine if ALiCS has low battery and needs a recharge, the Camera and Lidarclasses implement specific methods for their specific functionalities, and finally ObjectDetection, which handles the detection and classification of the objects located in the rover's environment. This diagram shows the inheritance relationships, key methods, and attributes of the classes involved in theALiCS software.

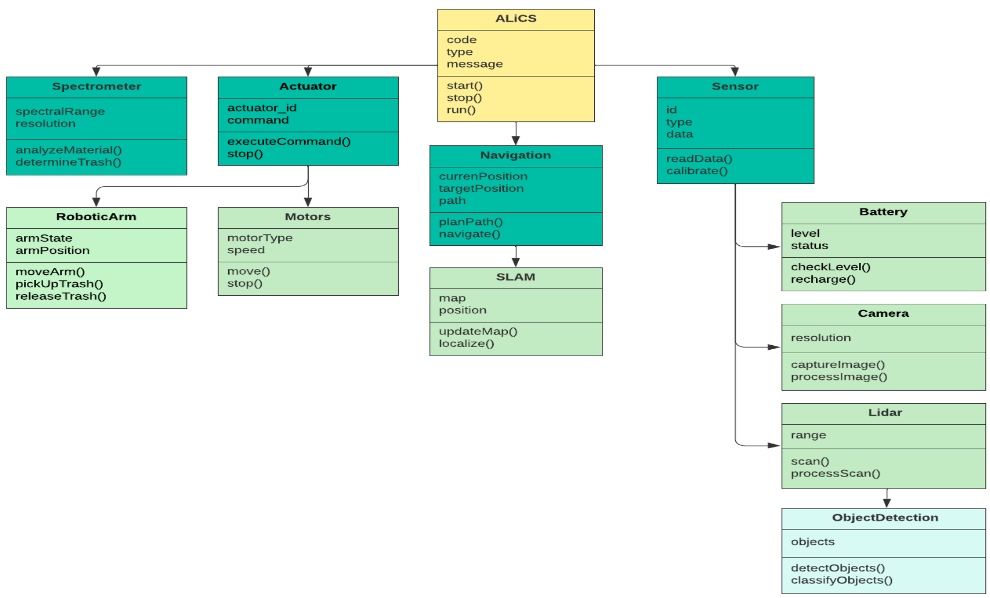


Figure 7.1.4: *ALiCS Class Diagram*

Data Structure for ALiCS

This data structure shows how various modules interact within the framework. The main control system is the central node that coordinates all communications across the entire software system. It manages and directs data flow between different modules and oversees the overall functioning of the system. The sensor data processing processes raw data from sensors and acts as a preliminary step before data is analyzed further or stored. This ensures that the data is clean and correctly formatted. SLAM is connected to both the sensor data processing module and the main control system. It uses the data from the sensor processing to construct a real-time updated map of the environment while tracking the robot's location. Besides SLAM, YOLO is another module connected to the sensor data processing, it uses the visual data from the sensors for object detection and classifying different trash using the YOLO algorithm. The Meca500 module handles every operation with the robotic arm receiving commands from the main control system and performs the tasks with precision and accuracy. All of the data is stored in the data storage for long-term retention. This can include detected objects found, mapping, sensor logs, and commands for the arm. This module feeds into the User Interface to allow users to access past data. The User Interface allows users to interact with the system. This includes programming commands, retrieving past data, visualizing data and more. This entire design ensures a seamless communication between the hardware and software components, enhancing the user's engagement.

Sensor Data

The Sensor Data module is in charge of collecting and processing raw data from the many sensors integrated into ALiCS, which include ultrasonic, battery, Lidar, the cameras, and the spectrometer. Each of these sensors will be continuously sending data into this module, where it can then be filtered to remove noise and irrelevant information. This clean, processed data will then be sent to the Integrated Data module for analysis. This module is very important as it handles the robot's ability to detect objects and its environment, measure distances and monitor its battery level.

Integrated Data

Integrated Data operates as the module where the data from the multiple sensors are combined and analyzed. It incorporates SLAM and combines the data from the LiDAR puck and camera to form the 3D map of the rover's surroundings. Integrated Data also makes sure all the data streams are synchronized. It matches up the sensor inputs with the state of the system. This integrated data is used by the Main Control System for decision making and by the Control Data module to perform actions.

Control Data

The Control Data module oversees the commands sent to the robot actuators, specifically the motors and the Meca500 robotic arm. This module receives the instructions from the Main Control System, and turns them into actions, such as moving the arm through sending motion commands based on joint angles and coordinates, rotating the motors, moving to a new position, navigation, and so on. This module will be responsible for making sure all control commands are executed correctly and in real-time, allowing the robot to quickly respond to environmental variables.

Data Storage

Data Storage is responsible for keeping logs, and storing all relevant data generated by the whole system. There will be a lot of data we managed, which include the raw sensor data, the processed integrated data, actuator control commands, and system state history. This is very important for us as it allows us to review and analyze the robot's performance post-operation, showing areas where improvement is needed. This module can also support the machine learning components of ALiCS by storing the trained data and results, allowing the systems to learn from previous experience and improve its navigation and object detection algorithms over time.

System State

The System State module tracks the current status of the entire ALiCS system. It is responsible for monitoring system variables such as battery level, sensor status, and the position of the robot and its components. An example of this monitoring can be seen when the battery level drops below a critical level, the system takes action and will enter the low-power state and order the robot to be charged. This module also provides key information to be sent to the Main Control System, to allow the system to make informed decisions based on its current state. In the diagram, Control Data and Main Control System interact directly with System State and system state sends to Data Storage.

User Interface

The User Interface module provides a platform for users to be able to interact with ALiCS. The interface will take in input user commands, allow users to check the robot's status, and monitor the data collected during any of the robot's operations. To specify, the user interface will be able to display real-time data coming from the onboard sensors, images and visualizations of the planned path and surrounding map, as well as notifications of the system's status or health. It has direct connection with the Main Control System and Data Storage.

Main Control System

This module is the brain and headquarters of ALiCS, communicating with all the different modules and ensuring they work together seamlessly. The main control system processes information gathered from the Sensor Data and Integrated Data modules, and makes decisions based on the pending objectives, such as moving according to the planned path, and picking up and sorting the trash found. The Main Control System can then send commands to Control Data which carries out the actions decided by the main module. It also communicates with the System State module to monitor the robot's conditions, and adapt its behavior and make real-time decisions. This centralized control makes it easier to manage all the components of ALiCS and help in smoothing its overall operation.

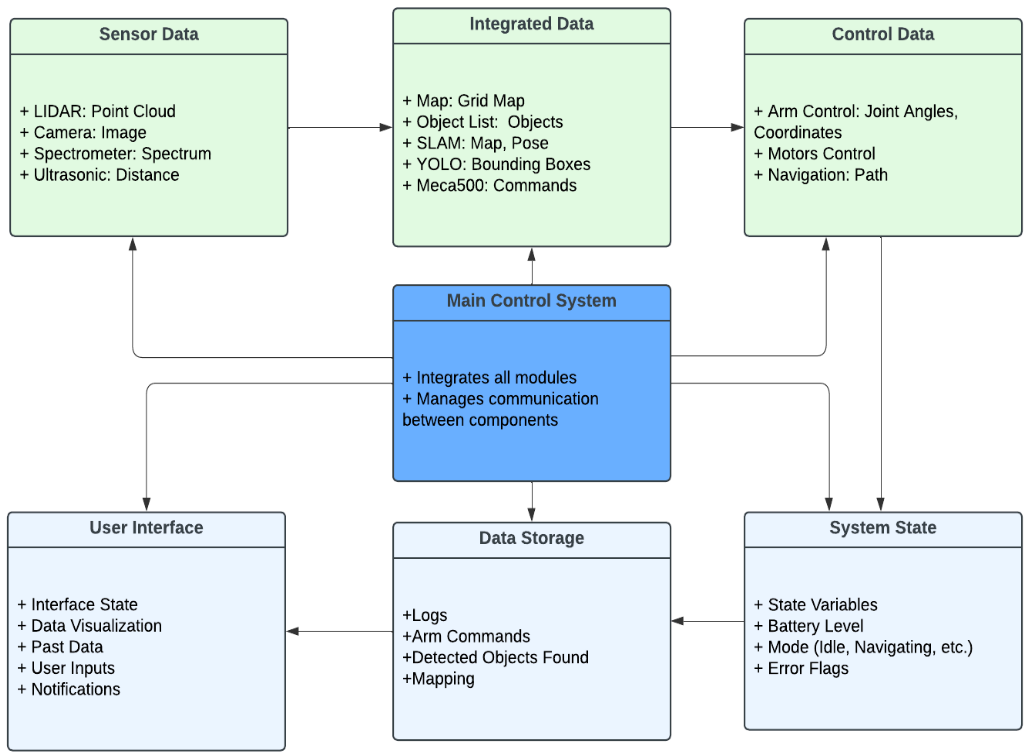


Figure 7.1.5: *ALiCS Data Structure Diagram*

User interface–Spectrometer Software

The user interface for the spectrometer software is constructed to be intuitive and user friendly with a central "Main Menu" serving as the central hub. Users can access different key functions such as: capture data process data, visualize data, settings, and help. To ensure the management of data acquisition, users can initiate and stop data collection in the Start and Stop capture features and then save the collected data for future analysis. The process data section provides tools for refining and analyzing captured data and can apply various filters with the data filtering. The visualize data section consists of different visualization tools such as the graphical view which displays the data in different formats (graphs and charts), spectrogram, and the zoom & pan feature which allows users to closely zoom in on specific data segments. The settings allow for customization to the device and software to fit the user's needs. The help section provides a User Manual, FAQs, and support contact information for the users when they need technical assistance.

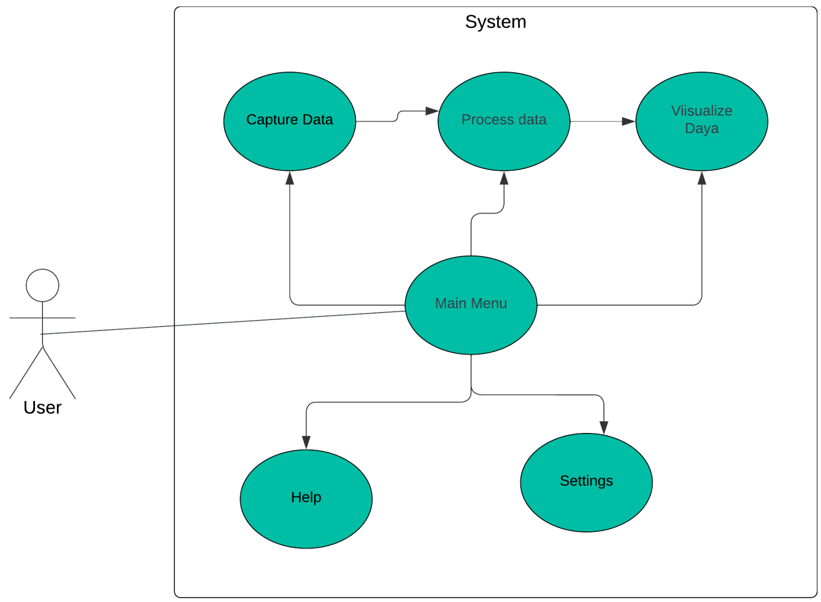


Figure 7.1.5: *User Interface Diagram–Spectrometer Software*

Raspberry Pi Software

The central processing unit for the robot is the Raspberry Pi, it is responsible for the high-level software which includes the Main Control System, SLAM module, and object detection systems and algorithms (YOLO). Python is the primary programming language used in the RaspberryPi, due to its extensive libraries which can help with computer vision, machine learning, and navigation. The Raspberry Pi is responsible for managing communication with all sensors, performing data processing, and making real-time decisions based on the information and data received from the various modules of ALiCS' system. Software involved with the RaspberryPi include ROS (Robot Operating System), which was used to communicate between different components in the system, SLAM libraries which provided the backbone for localization and mapping to create the navigation and path planning for ALiCS.

To use the device in our project, we installed all the necessary software required, starting with RaspberryPi OS, which comes already equipped with the programming language of choice, Python. Once that is setup, we configured remote access via SSH or VNC to program the device on our personal computer. ROS was also installed as we used it for inter-process communication, OpenCV was used for computer vision tasks, which then we also installed the YOLO framework and respective machine learning software. The development tool of choice was the powerful IDE Visual Studio Code which we were able to combine with a remote SSH extension which let us program and test the RaspberryPi directly from our computers. Since ROS was used, our code was structured into ROS nodes which each handle different tasks, including SLAM, motor control, and object detection. Python scripts were then used to test the RaspberryPi and ensure everything works as expected. Finally we were able to simulate the whole system using tools such as RViz, discussed in the next section.

ESP32 Software

The ESP32 microcontroller was able to handle the real-time control tasks involved in ALiCS. Some of these low-level tasks include manipulation of the motors, monitoring sensors and their data, and executing commands from the RaspberryPi for controlling the Meca500 robotic arm. The programming languages normally used for the board were C and C++, due to the precise control they have over hardware and efficiency when it comes to real-time systems. For the coding platform, Arduino IDE was perfect in developing these tasks as it is very simple to use and has large community support. FreeRTOS was a prominent software involved with the ESP32 because of its task scheduling and timing control. We basically had to develop some custom framework to be able to handle specific tasks like the motor control and communication with the RaspberryPi over serial and Wi-fi connection. The MCU software was the foundation of ALiCS' behavior, while the RaspberryPi was held responsible for the more complex tasks.

7.2 Machine Learning Implementation

As discussed in chapter three in our paper, in order for us to consider this design as machine learnable, it will need to be able to learn and improve on its own, capable of increasing its accuracy all while using data and algorithms to be able to improve and operate. There are a multitude of algorithms and other things to incorporate for any robot capable of using machine learning, but the main three that would be needed in our case of a self-driving rover would be testing, global pathing, and local pathing. Testing is needed in every case of machine learning, the machine would not be able to learn without it being constantly tested to make sure it is operating correctly. Since we planned on using supervised machine learning, we tested it with specific datasets and saw how it improves from there. In our case, datasets will essentially be specific situations in which the rover is placed in a spot and there are multiple obstacles and only one piece of trash nearby. Global pathing will then come into play in which it will be able to create an optimal path from the rover to the trash while avoiding every obstacle. It creates an optimal path using what visualization the LiDAR puck and SLAM creates. Once the optimal path is created, local pathing will then start in which it will follow the optimal path, but it will adjust to any and all obstacles that come into its path. This will be important for any obstacles that are moving around such as humans or animals. The image below shows how the testing for machine learning looks like.

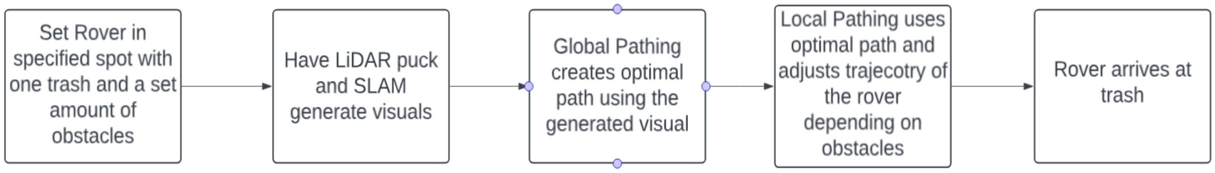


Figure 7.2.1: *Rover Machine Learning Testing*

The image above is a general overview of how the testing should be, there are, however, a few testing functions that ROS has that we believe would be of good use for us and our design.

Roslaunch

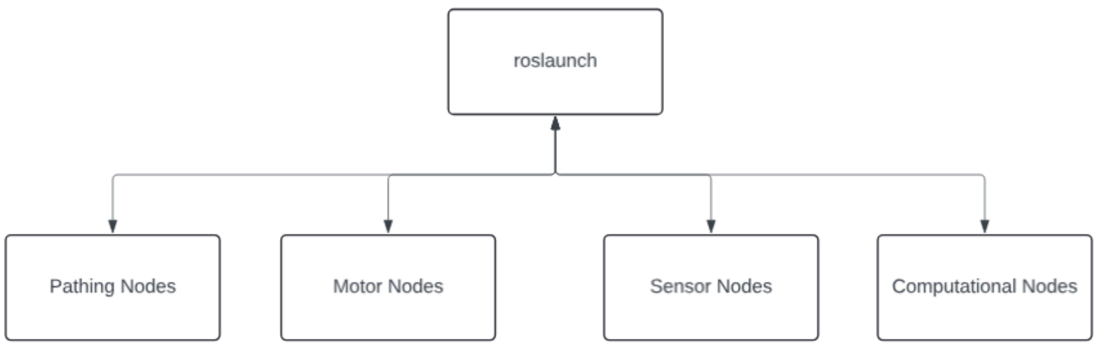


Figure 7.2.1B: *Rover Machine Learning Testing*

When it comes to the use of ROS and all its capabilities, roslaunch is the most important part. Roslaunch is essentially the main tool, as you can see in the figure 7.2.1B, It is the tool that starts up the entire ROS system. With roslaunch, we were able to select specific nodes that we would want to use, in our case, nodes that deal with pathing, testing, etc, and add them to what is essentially now this one, unified system. Not only can we do that, but, we are also able to set the parameters of our design in the roslaunch. Parameters like the rover height, rover width, and other things like rover weight can be added to the system to combine everything all together. The use and understanding of roslaunch is extremely important when it comes to our project because it is essentially a huge part of our software. Most of, if not all, the software we plan on using will be capable of having some kind of compatible ROS function that we can end up using for our design. It would've become a huge problem if we struggled to set up roslaunch correctly, because that would have led to potential errors that we would not know how to solve. Thankfully, when it comes to setting it up, there are a lot of resources on the ROS website such as tutorials, guides, as well as video tutorials that can help us if we are having any kind of trouble with setting it up ourselves.

Rostest

Rostest is a function that ROS has. Rostest is an extension of roslaunch, and how it works is that rostest is able to use roslaunch files as test fixtures. Using rostest, we are able to test any individual ROS node that is within the roslaunch system, but not only that, we are also able to do full integration testing across multiple nodes. Rostest is a pretty valuable function to have from both start and finish. We can use it in the beginning to make sure that every node we add to the system is performing as we want it to. Also, we are able to test multiple nodes together to see that each node is not only working, but working with each other node. It would be best to use rostest everytime we add a node to make sure everything is fully functional. It was also useful during the end of our design to make sure that everything is still intact and all nodes are working together.

RViz

RViz is a tool that ROS has that was used for testing of our rover. What RViz does is that it is able to create 3D visualization. It is able to visualize the data from cameras, LiDAR data, the trajectory of the robot, as well as global pathing and local pathing. This is a very effective tool for design, and we were able to use this tool from the beginning till the very end. The reason why RViz is so important is because with the LiDAR data, robot trajectory, global pathing, and local pathing, there was really no effective way to visualize it. We would have had to just assume that everything is working as intended, but with the help of RViz, it is able to visualize our data so that way we are able to see and understand what our data is doing. At the beginning of our design phase, it would be best for this to be set up so that we can see our data and make sure that it is being displayed correctly. If it is not being displayed correctly, then we would be able to determine that there is either an error with RViz or the data we are specifically testing. This could also be beneficial to see for ourselves if the path from the rover to the trash is really being optimized and if not, we would need to adjust the global pathing algorithm. Even towards the end of the completion of our project, it was still in our best interest to have RViz open on the side making sure that any and every piece of data is being displayed correctly. ROS also offers user guides, tutorials, as well as troubleshooting for RViz so even if we struggle to implement it on our own, there are valuable resources we can use to help implement it.

ROSbag

ROSbag is another function that ROS has that could potentially be useful when it comes to collecting and analyzing data. What it does is that ROSbag is able to record sensor data while the rover is operating. It is also able to record data such as the paths the rover took. What this means is that, we are able to let ALiCS operate on its own for a while, and when we come back, we are able to observe the data that is collected and analyze the data. An example could be that if we left about 7 pieces of trash around ALiCS and when we come back, the 7 pieces of trash is still there, we could be able to analyze the path ALiCS took in order to see what went wrong and we would end up realizing that perhaps

ALiCS was constantly navigating towards two different trashes and it got stuck. What this will give us is the ability to be able to let ALiCS operate as well as allowing us to receive data, without us needing to be present. The thing we would have to be wary of, is how much data can be stored, because we do not want to use up all of our memory from ALiCS.

ROS Global and Local Pathing

Considering the results we gathered from chapter three of our paper, the global pathing algorithm we used from ROS is the A* algorithm, and the local pathing algorithm we used from ROS is the timed-elastic band algorithm.

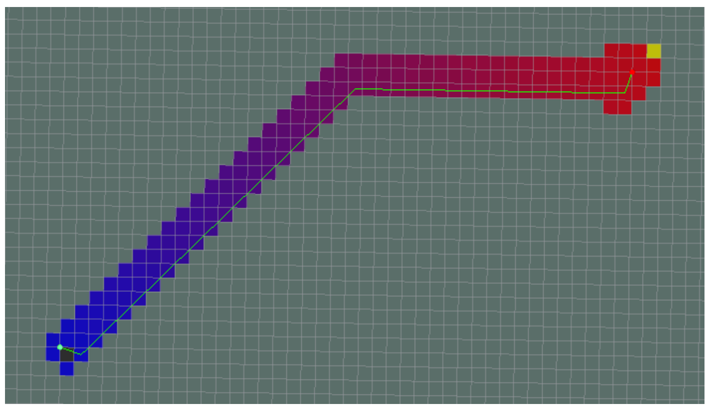


Figure 7.2.2: *Pathing of A* Algorithm, Referenced from* [ROS Website]

The image above is an example of how the pathing of the A* algorithm will work. It is able to find the optimal path from the start (the green dot) all the way to the goal (the red dot). Most of the global pathing options in the ROS website are all more than capable of giving us a decent optimal path, but the main reason why we choose the A* algorithm is because it is able to give us an optimal path while wasting very little time on exploring the outer area. The algorithm is focused on the optimal path and the optimal path only and for that reason we believe that it will be a good choice for us to use it. One important thing to note is that the A* algorithm is able to create this optimal path because it is using the information that was received from both the LiDAR puck as well as the camera. It may seem like a bad idea for using A* algorithm because it does not look at the surroundings which could be a problem because of moving obstacles, but it will not be a problem as long as we implement a local pathing algorithm

The local pathing algorithm is able to take the optimal path that was created and it is able to constantly adjust the path slightly if, and only if, there are obstacles that are in the way of the path to the goal. The local pathing algorithm we used is the timed-elastic band algorithm. How this algorithm works is that it is able to take in the parameters we give of our design, and it is able to take all information that was given to calculate an optimal trajectory to take. This operates on time based intervals, meaning that it is able to adjust and change trajectory if there is an obstacle in the way. This was of good use for our design because it is able to constantly adjust, and in an unpredictable environment such as a beach, constantly adjusting is a necessity in order to avoid any collisions. The image below shows the overall software implementation of machine learning

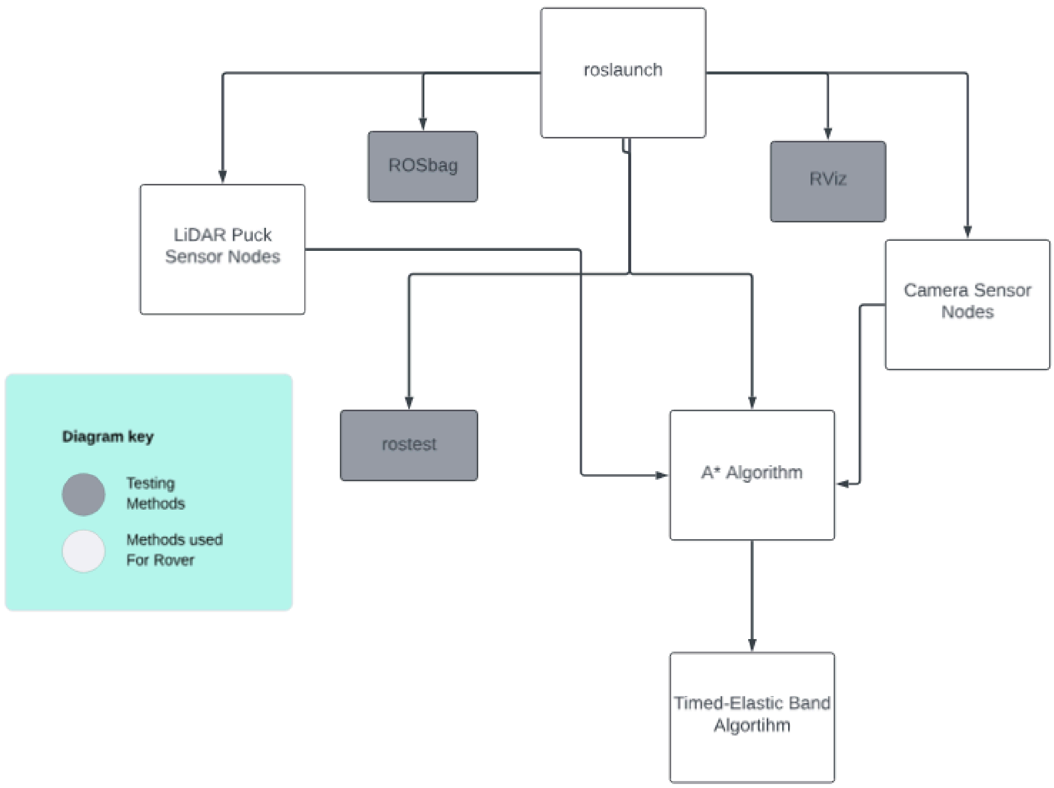


Figure 7.2.3: *Software Diagram of Rover*

7.3 Robot Arm Software Design

Meca500 Command Reference Table

This table provides an overview of how the Meca500 robot arm can be programmed and controlled, showcasing important commands and their respective descriptions that was in use in ALiCS. Most commands consist of motion commands and their respective speed commands. This information was pulled from the Programming Manual for Mecademic Robots. For the MoveLin command, the arguments x,y,z: the coordinates of the origin of the TRF with respect to the WRF, in mm; the arguments α, ß, γ: Euler angles defining the orientation of the TRF with the respect to the WRF, in degrees. The robot arm uses right-handed Cartesian coordinate systems or reference frames. The WRF is defined as the World reference frame, which is the main static reference frame and coincides with the BRF (Base reference frame), the frame fixed to the robots base. The TRF (Tool reference frame) is responsible for the robot's end-effector.

Table 7.3.1: *Robot Arm Commands*

Command	Description
ActivateRobot()	Activates all motors on the robot and disables brakes on the joints.
DeactivateRobot()	Disables all motors and enables brakes on the joints
Delay(t)	Used to add a time delay after a motion command.
MoveLin(x, y, z, α, ß, γ)	Moves the end-effector to a specified position and orientation in a straight line (linear motion).
MoveJoints(q1, q2, q3, q4, q5, q6)	Moves the robot arm to the specified joint angles, as fast as possible.
MoveJointsVel(q̇1, q̇2, q̇3, q̇4, q̇5, q̇6)	Makes the robot displace simultaneously its joints with the specified joint velocities. All joint displacements start and stop at the same time.
MovePose(x, y, z, α, ß, γ)	Makes the robot move its TRF to a specific pose with respect to the WRF.
Home()	Moves the robot to its home position, a predefined safe location.
GripperOpen()/GripperClose()	Used to open and close external grippers attached to the robot arm.

The first step in designing the control algorithm for Meca500 robot arm is defining its requirements. Specifically, define tasks such as picking up trash and sorting it, the range of motion, payload capacity, and precision. Any constraints should be considered as well, including speed, safety protocols, and any environmental factors. Some control strategies could involve kinematic control, which focuses on the arm's joint angles and their relation to the end-effector position without considering dynamics like forces and torques, dynamic control, which considers the forces and torques acting on the arm for more precise and smooth movements, or a hybrid control strategy that combines aspects of both.

Now we can develop the kinematic model which includes forward kinematics–this calculates the position and orientation on joint angles. It also includes inverse kinematics which determines the required joint angles to achieve a certain position and orientation. Some inverse kinematic equations could be:

$$x = l1\,cos(\theta 1) + l2\,cos(\theta 2)$$
$$y = l1\,sin(\theta 1) + l2\,sin(\theta 1 + \theta 2)$$

Where l_1 and l_2 are the lengths of the arm segments, and $\theta 1$ and $\theta 2$ are the angle of the joints. Besides the movement and orientation of the arm, motion planning is another crucial step, when designing, to ensure that the robotic arm can move from one spot to another in a collision-free path.

Sensors need to be integrated and there needs to be feedback to ensure accurate readings and adjust the arm movements. There is a safety protocol to ensure that when the robot is operating, it operates safely, and there is an emergency stop button for any dangerous situations.

The software is developed usually in languages such as Python or C++ and using the ROS for communication, sensor integration, and provides overall system management. It was important to test in a controlled environment to identify any defaults and fix any issues before we started testing in a real-world environment.

7.4 Algorithm Design

Localization

When designing a localization algorithm for ALiCS, it involves multiple steps to ensure the accurate position of the arm and rover and movement in the environment. We must define such requirements for ALiCS such as, desired accuracy, the environment (beach in our case), and type of trash (glass, aluminum, plastic). It's important to define these requirements so that we know and the robot knows its abilities and limitations. We can now choose an appropriate localization technique based on our requirements. For this project, we went ahead with SLAM which we integrated with LiDAR and the camera to build a map and localize itself.

Next, we implement the sensors for localization. This includes the LiDAR puck to measure the distance of objects and the camera for visual data and object detection. Data fusion now combines the information from these sensors improving the localization accuracy. We can implement some techniques to integrate noisy sensor data such as Kalman Filter, Extended Kalman Filter, and Particle Filter.

A huge part of localization is its mapping. Once the algorithm is developed, it needs to be implemented and tested in a simulation to ensure that it will work in a real-world situation and validate the algorithm's performance. Optimization and tuning are crucial to the algorithm's performance and accuracy. This includes adjusting its parameters based on the test results and refining the algorithm to improve robustness. SLAM integrated with ROS packages, like Gmapping or Cartographer, helps build and update the map using LiDAR data.

Rover

Designing the rover algorithm for ALiCS is similar to the robot arm and localization algorithms. First, we must clearly define the objectives and requirements like we did for the other algorithms. Recapping our goals, the rover needs to be able to navigate across different terrains in different environments. It also is able to interact with objects, such as when the robot gets notified of a piece of trash, the rover will move to the object with the correct distance so that the arm can pick up the trash.

For navigation, we implemented algorithms like A*, Floyd-Worshall, and Dijkstra's Algorithm to ensure that the rover takes the shortest, most convenient path. The rover will integrate the sensors to enhance its capabilities. LiDAR is used for obstacle detection and distance whereas the camera is for vision-based tasks. This will help localize the rover so that the rover won't collide with any obstacles.

It's key that the algorithms are tested and optimized to improve efficiency and correct any errors. Simulating the rover in a controlled environment allows us to see where our issues arise before taking it out onto the beach. It's critical that if the rover were to malfunction, there are safety and emergency procedures that will halt the rover to stop in its tracks. Integrating these components and testing for the robustness of the rover gave ALiCS a strong base for the arm and sensors to sit on.

Lidar

To design effective use of our lidar puck we need to make it use the localization algorithm that defines where the rover is and update this data constantly using the lidar and other systems. To do this we must define the goals and objectives of our lidar. From our goals our lidar will instruct the rover of incoming obstacles that could cause problems towards ALiCS then update the system to adjust properly. After relying on this information on our system we can quickly delete this data and adjust the values in realtime to constantly update the system with constantly new information.

When implementing this we will implement algorithms like before to ensure that we make our rover take the shortest while safest possible route avoiding obstacles like rocks, people, or the water.

To test this system we had to constantly try to improve the range limit from obstacles to be as safe as possible while still keeping up speed to have a high efficiency on ALiCS. We also had to implement tests to see how close to moving objects like humans can be before ALiCS stops moving to be sure that we don't run into any issues like driving into people who move in the way last second.

Camera

To design an effective algorithm for our camera detection software we made use of machine learning algorithms. To make this we must define the original goal we had for our camera. So that our camera can learn to identify trash and detect the coordinates to send this information to our PCB then our robot arm to bring the trash in for sorting.

The main function of the camera is to detect scattered trash along the beach and being able to distinguish it from other objects such as rocks, shells, and sand. After the trash has been identified, the camera must accurately calculate the coordinates within its field of view. The camera must be able to identify such items under different weather conditions and annotate the images to label the types of trash (plastic, aluminum, glass). When implementing machine learning algorithms, the camera can identify image tasks and localize trash items. The camera will be integrated with other sensors to enhance the "eyes" of ALiCS. Finally, it's important that the camera and algorithm is tested to ensure that it is operating effectively and efficiently.

Chapter 8 System Fabrication/Prototype Construction

8.1 Fabrication Materials

For the construction of our prototype for this project we have decided to make the frame out of 25 mm optical construction rails that will make our robot a strong frame while still staying relatively lightweight. The construction rails come in either raw aluminum or black anodized aluminum. These construction rails work easily together and are fitted to whatever size we need while attaching easily to one another almost like legos. These construction rails secured by this frame will be strong enough to support ALiSC with its heavy robot arm, collective trash and recycled materials it will hold in its two bins and hold our heavy battery and other parts. The 25 mm Optical Construction rails have a Mass per length of 0.811 kg/m and have a cross-sectional area of 3.13 cm^2. It has a 6.5 mm opening in between each corner and these openings accept t-nuts, rail joiners, and enclosure panels. The 25 mm construction rail also has a ¼ inch hole made for standard compatible optomechanics. These construction rails are perfect for housing electronic components safely and without worry of breaking or crumbing ALiSC.

Weatherproofing enclosures is essential to protect electronic and mechanical components from the elements and ensure reliable operation in outdoor environments. Effective weatherproofing involves using materials and techniques that shield against moisture, dust, extreme temperatures, and other environmental hazards. Enclosures often incorporate seals, gaskets, and specialized coatings to create a barrier against water ingress, while ventilation systems prevent heat buildup and condensation. Additionally, durable construction materials such as stainless steel or fiberglass reinforce the enclosure's resilience to impact and corrosion. Proper weatherproofing not only extends the lifespan of equipment but also maintains its functionality under varying weather conditions, making it a critical consideration in outdoor installations across industries ranging from telecommunications to industrial automation. We implemented ⅛” tinted plexi-glass to protect the components from the elements. The plexi-glass sheets will come in 4’x4’ pieces and we cut them down to fit the frame. To help protect from electrical shock, we installed a rubber gasket between the plexi-glass and aluminum rails. This gasket is rated for water-proofing in light conditions.

Then to support the different levels and make walls around our bot we plan to make use of aluminum sheets of metal to build a solid box for ALiCS to protect it from environmental conditions. These aluminum sheets should make it easy to make the needed holes for standoffs and screws to hold our components in place while keeping out sand and unwanted materials. This will also allow us to make custom 3d printed parts to be made for whatever need we might need them. We may also use these aluminum sheets to make secure areas to hold wire connections. Such a channel from the motor controls to the motor for making sure the wires don’t get stuck on anything underneath or so the wires don’t become exposed due to the environment stripping the wires slowly over time.

8.2 Bottom Level

The bottom level or the undercarriage of ALiCS is the foundation for a good robot. It holds the entire project and needs to be strong enough to hold everything at once without stress or strain on ALiCS. This means we made use of the most efficient part choices we selected earlier to make a great robot that shouldn’t budge or fall without extenuating circumstances. This bottom lover of ALiCS will have the motors, wheels, and a hole to connect the middle layer of ALiCS. We needed to make the robot have a good position on the wheels and motors to keep the center of mass in the middle to make a more stable robot.

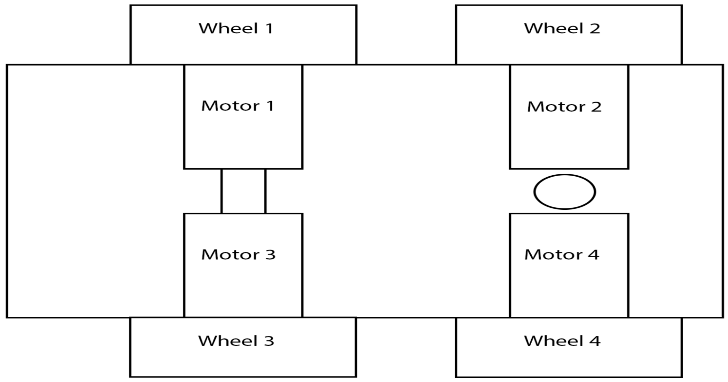


Figure 8.2.1: *Bottom layer of prototype*

Looking at the figure above we can see the locations we have decided for a great base for ALiCE. The wheel positions that were chosen were to make it as even as possible for the center mass to be as close to the middle as possible. This in theory should make our bot near impossible with all the parts and trash mass added to tip over the robot without someone or something physically pushing it over that is not a standard environmental hazard such as high winds or a bump on a rock. The nine inch wheels we chose should keep us well over the sand and the strong motors we chose should be able to move the entire cart at a relatively good pace to getting to the next piece of trash. The hole in the back will connect to the motor controllers to allow our prototype to actually move and be able to connect directly to our MCU.

8.3 Middle Level

The middle layer of the project is the most important part of our project. It holds most of our major components that are required to work and needs to be able to evenly disturbed the weight around the robot to keep a good center of gravity. This means all the heaviest or could be heaviest components need to be opposite of each other. This also is the most vertical section of the bot meaning we have access to the most space of all our sections. This section needs to be able to hold the trash bin, recycle bin, all PCB and circuit board components, the Robot arm, Robot arm power brick, spectrometer, four ultrasonic sensors, and our battery. Also needs to be able to fit any other company that we are unaware that we might need to wanna add in the future. It will also need a hole on the top and bottom to connect to components on both other levels of the project.

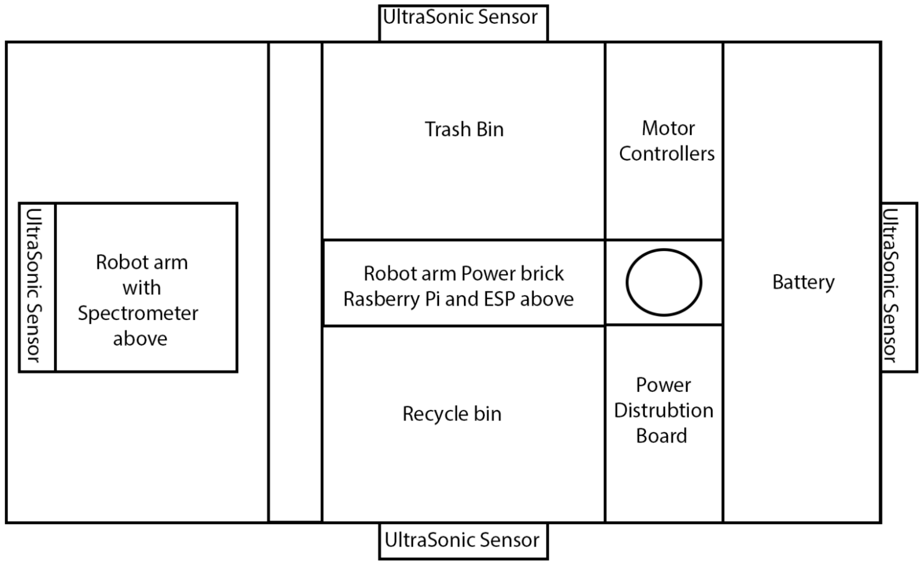


Figure 8.3.1: *Middle layer of prototype*

The figure above shows how the middle level will be sorted out for the most optimal use of the area. We first put 4 ultrasonic sensors on each side of ALiCS to tell ALiCS about their surroundings. The two on the sides with the wheels will be put above the wheel height to be able to tell without the wheels interfering with the data. The robot arm will of course be in the very front with another ultrasonic sensor in front of it. Then to counter balance that we have the next heaviest part on the other side, the battery. Then to keep the weight center we have both waste bins on the opposite side of each other from the top to bottom side of ALiCS. In between these bins we have the power brick needed to power the robot arm and above that we have a place to store both of the MCUs we are using that will be directly in the middle of ALiCS and have the best access to the top and the bottom of ALiCS without compromise. Then we have a little space in the back to fit both the Motor controllers and Power distribution boards that will be needed. Then there is a hole that connects the motor controllers to the motors on the bottom. Then we also have the spectrometer on top of the robot so it can correctly sort the trash. The last thing that is not on this figure but the next is that there will also be a small hole above the two MCUs to connect to the components on the top level of the prototype.

8.4 Top Level

The next and last level is the Top level of our robot. This level of ALiCS is more of just a sensor suite than anything. This part will hold our camera and spectrometer. It will also need a hole to connect to the middle part where the raspberry pi is.

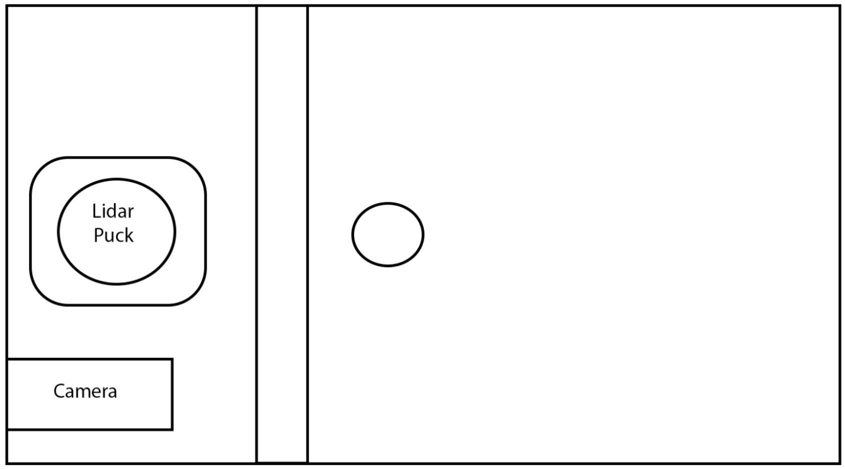


Figure 8.4.1: *Top layer of prototype*

The figure above shows where the lidar puck and camera will be on the top level of ALiCS. From this position the lidar puck should easily be able to identify any incoming obstacles that the ALiCS would need to avoid. The camera from this position should be able to detect where trash is and signal to the robot arm to pick up and show the spectrometer for sorting.

8.5 PCB Layout

This will be the first design for our PCB layout for the prototype. This will not be the final design because later on during senior design two we will remake the ESP32 board and add more connections peripherals to connect to the robot ARM and possibly Lidar directly to the board from the additions on the design that we will make. However, because of the low amount of usb connection ports we will make more use of our raspberry pi for the prototype PCB. This layout will connect directly from the battery and show all connection ports and adaptors needed to make our prototype work. This needs to be able to correctly transfer all the information and knowledge learned from the prototype and make advancements towards our own PCB when we create our own ESP32 board instead of using the dev board. However, even with the few connection ports we can still make good use of the GPIO pins of the ESP32 dev board to power up the ultrasonic sensors, motor controllers, and our spectrometer. This also needs to show how we will connect the two boards together to make them work synchronously in harmony to complete our goals.

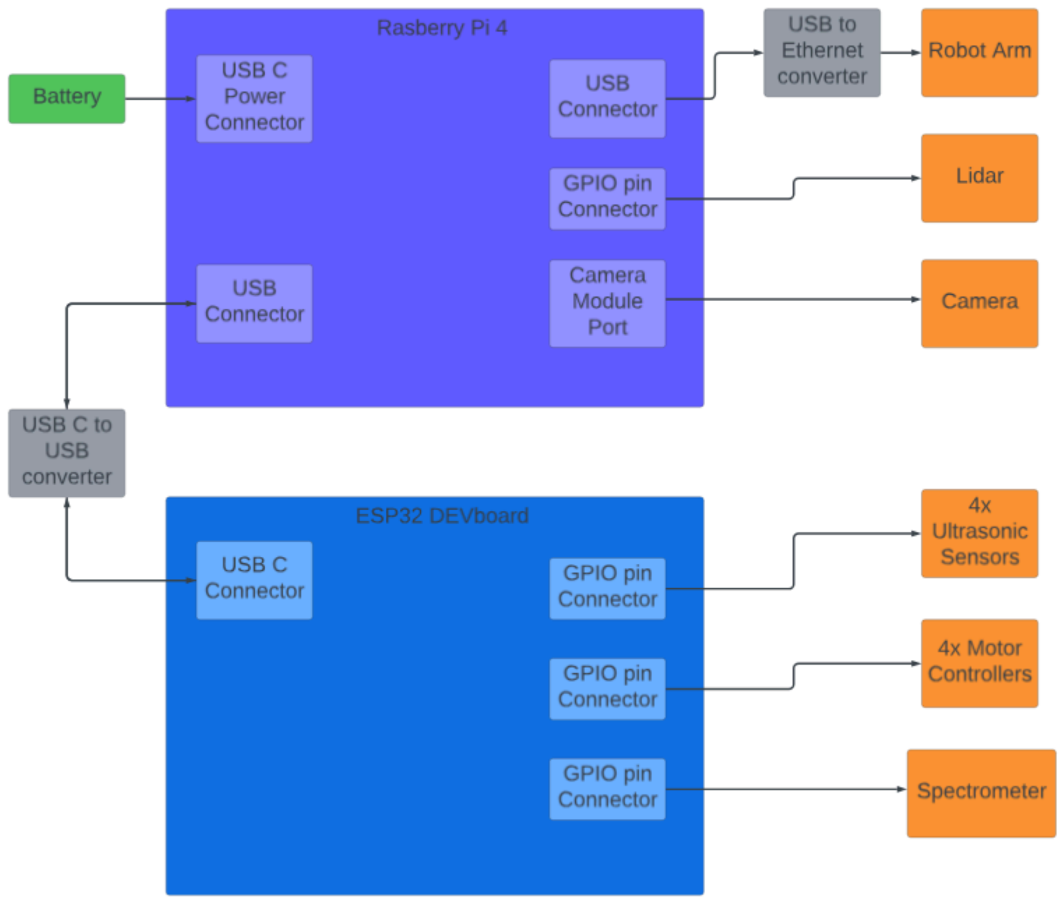


Figure 8.5.1: *PCB Layout for Prototype*

This first issue the figure above solves for us is how each component will connect with one another. It shows the necessary adaptor for both the connection between the two boards and the connection to the robot arm. The boards will need an adapter to connect because the ESP32 only has one usb c port to use for power and data transfer that we will use to connect to the raspberry pi for power from the raspberry pi and to transfer data between. The next adapter will be used for the robot arm because it only has an ethernet cable to receive inputs from and we are uncertain about using the built in ethernet on the raspberry pi. The lidar will make use of 6 of the GPIO pins on the raspberry pi to connect and receive data. The big reason why we chose the raspberry pi was for its camera module port so it's great we will make excellent use of it to power our camera and receive data from the camera. We will then make use of the GPIO pins built into the ESP32 dev board to connect and power the motors, ultrasonic sensors, and the spectrometer for ALiCS. For more specifics on how we will make use of the GPIOs for both boards look back at the hardware design. The last but most important part is that all the power will run through the raspberry pi making sure everything runs efficiently and correctly. With the expectation of the robot arm because the robot arm requires too much power from just the raspberry pi so we will make adjustments to divert energy from the battery straight into the robot arm making sure it receives the correct amount of power.

Chapter 9 System Testing & Evaluation

9.1 Hardware Testing

ESP32

To test the ESP32 WROOM, we can connect the ESP32 board with a USB cable and connect it to the Arduino IDE on the computer. Once connected, there is a blue LED that indicates the ESP32 is receiving power. From there, we can connect to the Arduino IDE and choose the correct port so that it can communicate with the ESP32. We can now program the ESP32 to print out a sketch to make sure it is connected properly. The next test we can make with this is to power the motors and wheels directly from our ESP32 WROOM board and make sure we can get them running without anything else connected. The next test would be connecting the ultrasonic sensors to ESP32 and see if we can at first get them working in unison with our motors without any issues occurring. Then we can test to see if we can get the spectrometer working off our ESP and slowly connect the motors and sensors. If all works correctly then we can move forward with our project to complete

Meca500 Robot Arm

The Meca500 has a lot of components to set up and communicate with to ensure that both the hardware and software systems are working cohesively. The Mecademic Industrial Robot website provides all the information on how to get started and program the robotic arm to reference. First, the industrial arm needs to be set up itself on a breadboard, connected to a power supply, ethernet cable, and computer. Next, the industrial arm and computer must communicate by establishing connection via the ethernet cable and install the control software. Within the control software, we can now communicate with the arm by inputting specific tasks or movements. This test showed that we could continue to make the robot better once we had additional means. Once that claw is added we could start working on picking up objects with raw inputs or coordinates. Then slowly get the movement and positions correct for holding up an object for our spectrometer. Then we worked on positions for the robot arm for each bin slowly but surely. Then we were able to add camera vision and start training the arm for different distances depending on what the camera detects as trash. The next part about the test we found out was all the complications with the power brick. The power brick has a massive power draw and needs to be one of the main focuses when looking into a battery for powering our robot. It also comes with a lock that will need additional space to keep the lock clear so we can keep the robot arm functional during our testing of the robot arm. After these conditions were met we were able to make our robot arm do a simple test run. Showing the movement options of just hard coding.

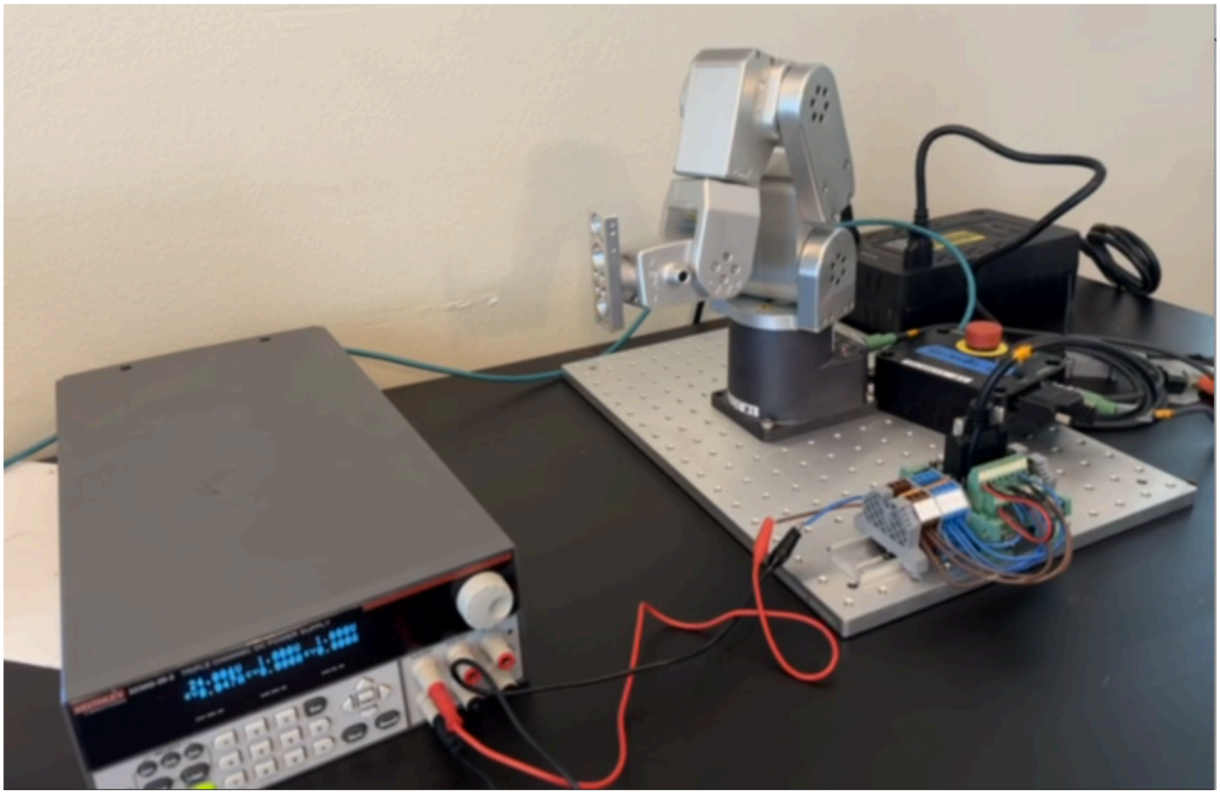


Figure 9.1.1: *Hardware Testing For Robot arm*

The figure above shows our first hardware test with the robot arm showing all the parts just needed to make the arm function, not including the battery that we needed to power it. You can see the massive power brick needed to power the robot will need some dedicated room inside of ALiCS. In addition we needed more room for the override lock that is placed on the power brick to allow it to work. This override will also need its own power going to its function.

Once we attached the Robotiq 2F-85 gripper, we began testing picking and placing objects on a bench top. We were able to set gripping force, speed, and release the objects.

Raspberry Pi 4

To ensure that the Raspberry Pi is functioning correctly, both the software and hardware components need to be checked. Once the Raspberry Pi is connected via the HDMI and USB cords, a LED light will indicate that there is power being supplied to the board. After the board is set up, the ethernet and wifi connection needs to be established. The peripherals, storage, GPIO, software now can all be tested.

Lidar Puck

To test the Lidar puck, we must connect it to a power supply and the processing unit, which can be a raspberry pi or computer via USB or ethernet. Then, the required software and drivers should be installed correctly. Once setup is complete, we can run a simple data acquisition program to ensure that the Lidar puck is transmitting point cloud data. There are various data visualization software available for us to use, ROS, CloudCompare, and PCL(Point Cloud Library). We also must verify that the puck is rotating and able to capture a 360-degree view of its surroundings. To further test its accuracy we can check the data output and compare it to known distances in our controlled testing environment.

Spectrometer

Testing the spectrometer involves testing its accuracy, functionality, calibration etcetera. Once the spectrometer is set up, we can connect it to a power source and a computer to see if it'll connect to the software. Then using a calibration light source, we can record the wavelengths and identify its peaks. We can also measure the sensitivity of the spectrometer such as measuring the noise level. Repeating these tests will ensure the accuracy and precision of the spectrometer. Once the spectrometer is calibrated, we can begin to test on samples commonly found on the beach. We plan to sample 3 materials; red solo cups, beer cans, and beer bottles. A stretch goal would consist of testing cigarette butts seashells.

Rover

For the rover, we assembled the chassis and mounted all the motors. Motors were connected to the controller which will be connected to our processor, and power supply. We began testing with just the motor and the ESP32 WROOM at first. With those two components, we wrote a simple program that makes sure each motor individually can rotate in both directions. After that, in order to test the compatibility of the motors and the wheels, we attached the wheels to the motors to make sure that the motors were capable of moving the wheels. After doing simple tests with the motor and the wheels, to further test the capabilities of the motors, we then rewrote the program (when the rover is partially assembled), to test and see if the rover was capable of turning left, turning right, and turning around. Once the beginning testing was performed, we then modified the program so that it set a small course and route for the rover to follow, and we tested if it reached the destination in a timely manner, and continued to test different paths with varying difficulty to test for reliability and accuracy. All the testing was done with the help of the ROS functions that will be implemented and they are RViz, ROSbag, and rostest. These three functions gave us a lot of options to test, things like how the pathing of the rover looks like, is the parameters given to the function being calculated correctly, as well as allowing us to store data from past attempts in order to review them to see if it produces adequate results.

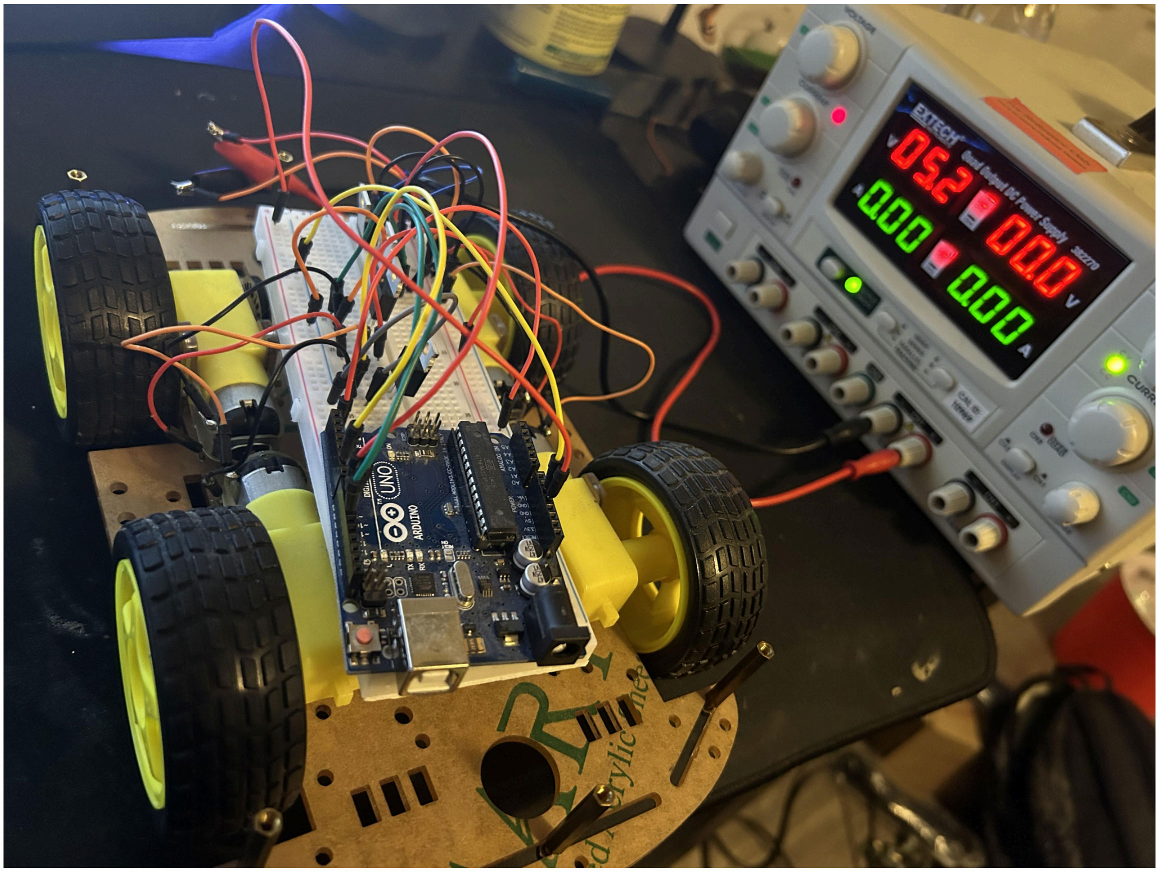


Figure 9.1.2: *Hardware Testing For Rover*

Figure 9.2 shows the early testing we did for the rover. Using a power supply that is giving off a voltage of around 5 volts, it is able to apply enough power to all four motors so that the wheels can turn. Since it is early testing, we are trying and testing it out with just a breadboard at first, but we are not allowed to use a breadboard in our final design so that will not be included later on. Also, we used an arduino to test the rover due to what was currently available during the time we tested it. It should be a simple switch from the arduino to an ESP32, all we would need to do is make some small changes to the circuit and upload essentially similar code we used in the arduino IDE to the ESP one. The code we used is able to turn on the motor for about a second, turn it off for about a second, and repeat it until it is forced to stop.

Raspberry Pi Camera

The raspberry pi camera goes hand-in-hand with the raspberry pi board. Once the camera is connected to the camera port, ensure that the connection is secure and the cable is properly oriented. After turning the raspberry pi on, we can test the camera using the terminal and running the Raspberry Pi Software Configuration Tool. From here, we should be able to capture images as well as record a video.

9.2 Software Testing

Testing is a crucial phase in the development of ALiCS, as we must make sure that the robot performs its tasks consistently and effectively in real-world conditions. Our first stage of testing, unit testing, focuses on the individual components of the software and hardware. Each software module we incorporated in the robot such as the communication protocols, the sensor data processing, and SLAM algorithms were tested on their own. There are several tools we can use to automate the testing such as PyTest which handles the Python aspects of our software development and Google Test for C++. Ensuring these all work individually before blending them all together is vital in the development of ALiCS to avoid and fix any bugs early before daunting problems occur in the late stages. For example, we can test the Meca500's physical capabilities to make sure it meets its specifications by programming predefined movements, and instructions to measure its precision, reach, payload, and repeatability.

Once we tested and validated all individual components, our next step in the testing process was to see how well the components work together. We used incremental integration to test these components in harmony, where we start by integrating the SLAM module with the robotic arm control module, then integrating the rover navigation module, and so on. This will help tremendously in identifying issues at each step.

SLAM

To thoroughly test SLAM for our Lidar Puck, we can test both unit and integration testing, simulation testing, real-world testing, performance testing, robustness testing, and user acceptance testing. To focus on unit testing, we can perform tests on the map updating function to confirm that it reflects changes in the environment accurately. We can ensure that SLAM is integrated with the LIDAR data and ensure that the data flow is correct by simulating the robot's movement and verify that the software correctly maps the environment and localizes the robot. SLAM can be tested in a simulation to see how it will operate in a real-world scenario and then be tested in a real atmosphere. In addition, we can measure the efficiency and resource utilization of the software to check the performance. We can monitor processing times for different functions and evaluate memory and CPU usage. Checking to see the robustness of the SLAM system is important as well. Introducing noises and disturbances, we can see how sensitive the system is and how it responds to dynamic changes.

Camera Interface YOLO

For the camera software, we used YOLO for object detection. To test YOLO, we can look at it from a systematic approach evaluating both component-level and system-level testing. For component testing, we can test specific components such as image preprocessing and box prediction. Verifying that the camera is properly integrated with YOLO, YOLO should be able to detect objects once it receives images from the camera. The performance is a crucial aspect and can also be tested by measuring the detection accuracy and speed of identifying an object. To ensure the robustness of YOLO, we can introduce diverse environments such as cluttered backgrounds, brighter areas, etcetera to test how well YOLO can still identify objects.

Meca500 Robot Arm

The Meca500 came with its own software system that made it easy to communicate with the arm. Within MecaPortal, the arm can be tested by inputting simple move-to commands and rotations. The initial setup of the Meca500 involves connecting the device to a computer via Ethernet and launching the software by inputting the device's dedicated IP address. The MecaPortal is user-friendly and does not require extensive programming knowledge to operate the arm. MecaPortal allows users to input simple movement and rotation commands to test the robot. We tested the arm by moving it in a 3-D cube shape with these commands:

```
movelin (140, -100, 250, 0, 90, 0)
movelin (140, 100, 250, 0, 90, 0)
movelin (270, 100, 250, 0, 90, 0)
movelin (270, -100, 250, 0, 90, 0)
movejoints(0,0,0,0,0)
```

Taken from ***Programming Manual for Mecademic Industrial Robots***

These commands allow us to ensure the arm can move precisely to specific coordinates and rotate properly, confirming its 3-D mobility. We can use MecaPortal to program and test the arm's ability to pick up objects and move them from one location to another. The following pick-and-place commands test the gripper's ability to open and close and move objects to a specified location:

```
movelin(150, 0, 150, 0, 90, 0)
grip(close)
movelin(150, 0, 250, 0, 90, 0)
```

Now the robot arm moves to other location and releases object:

```
movelin(200, 0, 250, 0, 90, 0)
grip(open)
movelin(150, 0, 250, 0, 90, 0)
```

Taken from ***Programming Manual for Mecademic Industrial Robots***

To further test the Meca500 we can perform an accuracy evaluation and assess the repeatability. With an extremely low rated specification of ±0.005mm, we are working with a very precise device. We can test this by performing the same pick-and-place programs over and over and ensuring the arm is returning to the same coordinates and the object is not displaced with every movement. Its speed and responsiveness is another evaluation we can perform by measuring response time to commands, and cycle time per operation. Proper testing and optimization is crucial in ensuring ALiCS will collect trash consistently without fail.

Table 9.2.1: *Testing for Robot Arm Movements*

Test ID	Test Description	Expected Outcome	Actual Outcome	Status
1	Basic Movement Test	Arm moves to specified coordinates with no error.	Arm Moved Correctly.	Pass
2	3-D Cube Path Test	Arm follows 3-D cube path without deviation	Arm follows 3-D path with no deviations	Pass
3	Precision Test	Arm identifies object and picks up	Arm hovers over object and moves down to pick up	Pass
4	Speed Test	Arm moves at maximum speed without loss of precision or control	Speed, control, and precision maintained.	Pass
5	Load Test	Arm operates smoothly under maximum payload (0.5kg)	Operated with no error.	Pass

Raspberry Pi 4

The Raspberry Pi is serving as the central unit of this entire system so it's important that the software is set up properly and tested. The Raspberry Pi needs to be correctly installed and updated before using the board. The board should be able to connect to the internet either by Wi-Fi or Ethernet by using basic network commands like 'ping'. Verify that all software packages and libraries are installed using the package manager. In the terminal, we can run unit tests for individual software components and test how they interact with each other. We can also test the GPIO pins in testing to ensure that they are functioning properly by writing and reading values to and from the pins.

Communication Software (R.O.S.)

To test that ROS (Robot Operating System) is functioning and integrated with the robot, we can test multiple components. We tested the Unit Test Framework ensuring that individual components such as libraries or services are working correctly. The first thing to test would be making sure that the files needed to run roslaunch are correctly installed. That can be tested by running a few simple lines of code provided by the ROS website that checks the launch files to make sure there are no errors or missing items. The same could be repeated for all nodes that we installed such as rostest, RViz and others. For all nodes, it is best to test them to make sure that the files needed are correctly downloaded and working as intended so that way there will not be any problems in the future. To continue on testing roslaunch, we then ran a basic node that is already in the roslaunch library to double check if everything was working correctly. The basic node should most likely work if all is set up correctly. Once roslaunch is good to go and every other node that needs to be added is added and tested, we can then begin to use rostest. As mentioned in chapter 7, rostest is able to allow us to do full integration testing across multiple nodes, which is extremely important for our design since the nodes need to be compatible with each other for our design to work as intended. This could allow us to test certain connections for nodes to see if they are working together, for example, the LiDAR and camera node should be able to communicate with the global pathing node (the A* algorithm) in order to create a global map for the global pathing node. Rostest should be able to determine which node, if any, was giving errors so that we were able to change only the nonworking nodes. To be extra safe, we could've added specific alerts for each node to determine which node is not working just in case something does not go well with rostest. Once we have determined that every possible node is working correctly on its own as well as together, we began testing with real life examples. This was towards the end, when everything was all set. The table below shows some of the possible ways we could have tested our ROS software.

Table 9.2.2: *ROS Commands*

Method	ROS Command	What It Does
Running a Node	rosrun node	Starts the node which allows us to verify that it can run correctly
Launching Multiple Nodes	roslaunch package_name file.launch	Is able to open multiple nodes at once to see if they work well together
List Nodes	rosnode list	List nodes that are currently running
Pinging a Node	rosnode ping /node_name	Allows us to ping a node to see if it is responsive and

		working
Recording	rosbag record -a	Allows us to record data using the ROS Bag

Spectrometer Software

The spectrometer software needs to be tested so that the device correctly captures, processes, and displays the data. Once the spectrometer is correctly installed and configured, we can run unit tests for individual components such as testing data capture, processing and visualization. We can also test how each component interacts with each other and in real-world cases. Measuring the time taking to capture, process, and display data can show how fast the performance of the spectrometer is and we can adjust accordingly. The software can also test the error handling to see how robust the spectrometer is in handling situations such as being disconnected or receiving invalid data.

9.3 Performance Evaluation

Performance evaluation is a crucial step to ensure that the system meets its design goals and operates effectively in its environment.

We can evaluate the hardware performance by robustness, sensor accuracy, and power consumption. We can test the robustness of the Meca500 robot arm, raspberry pi, rover, and other hardware components in different environments such as different temperatures, humidity, and sediments. This will test how these components will perform in a beach environment. The LiDAR puck and proximity sensors can be evaluated to validate the precision and accuracy by performing different tests. The sensors can be tested to measure detection range, resolution, and response time. The Raspberry Pi camera can be evaluated based on image quality, resolution, accuracy and varying light performance. It's ideal that the entire system has low power consumption. Therefore, measuring the entire system's power consumption and making sure that it's within the acceptable requirements is crucial.

The algorithm efficiency, response time, and error handling can be tested for software performance evaluation. We can evaluate the SLAM algorithm used for navigation, focusing on the mapping accuracy of the interface and the speed at which it processes data. The camera software, YOLO, can be tested for object detection and its accuracy for identifying different trash. It's important that ALiCS is efficient in time so measuring the response time of the rover picking up trash, we can ensure that the robot's response time is acceptable. Testing the software's troubleshooting abilities, we can introduce simulated faults in the system to evaluate how well the software responds to the issue.

We can conduct begin to end testing of the integrated system to ensure that hardware and software components work seamlessly together. The system can be run for a period of time for stability and reliability. Data flow between different components can be evaluated to ensure that the data from the sensors is accurate. To improve the rate at which the robot completes a task, we can conduct multiple tests under different conditions to assess ALiCS's effectiveness. This will ensure that the robot is more effective and accurate when picking up objects. We can also take the system's throughput into consideration, measuring the time frame of the robot detecting the trash, picking it up, and sorting without damage to the environment. Conducting a cohesive and comprehensive performance evaluation gives the user a better understanding of how the whole system works and components relying on each other in different environments.

9.4 Overall Integration

Even though these specific hardware and software platforms are being tested individually, it's imperative that they are able to work cohesively and merge them all into one system.

Hardware Integration

For hardware integration, it's important that each component communicates seamlessly and can function correctly. The Raspberry Pi is the heart of the central unit which is running the ROS and managing the overall communication of the entire system. It is intertwined with LIDAR, the spectrometer, and the camera which all handles the mapping of the environment and object detections. Once the spectrometer and LIDAR map enough of the environment out, the rover will be able to move and adapt to its surroundings. The ESP32 microcontroller communicates with the Raspberry Pi and handles the control tasks and real time data. The Meca500 robot arm is controlled with its own software but with high-level commands from ROS to enable the arm to precisely pick up objects and move. In order for everything to function perfectly, the Raspberry Pi, the ESP32, and the Meca500 software all have to be able to communicate without any issues on any part of the system. It is also implied that for each of those three, they would have to be able to communicate without any issues to the components that rely on them, for example, it is important that the motors are able receive orders from the ESP32.

Software Integration

The software integration uses ROS as a middleware which facilitates the communication between the modules and hardware. The software integration leverages ROS (Robot Operating System) as the middleware to facilitate communication between various modules and hardware components. Each functional module, such as SLAM for LIDAR data processing, YOLO for object detection, and the control panel for the Meca500 robotic arm, is implemented as a ROS node. These nodes interact through ROS, allowing for efficient data exchange and synchronization across the entire system. By subscribing to relevant ROS topics, these components can share information and coordinate their actions seamlessly. Establishing clear communication protocols and robust data management strategies within ROS ensures that the integrated software functions cohesively.leveraging ROS's capabilities, we can achieve a high level of modularity and scalability, making it easier to manage and extend the system as needed. Ultimately, this integrated software architecture will significantly improve the operational efficiency and effectiveness of ALiCS in its environmental monitoring and cleanup tasks.

9.5 Plan for SD2

With all preliminary testing completed, it is time to create and test a complete prototype during SD2. Our plan is to schedule initial team meetings to discuss the objectives, timelines, and individual responsibilities so that everyone is clear on what they need to work on. We plan on having a gantt chart to keep track of everyone's responsibilities as individuals as well as a team. If there needs to be reassigned roles based on feedback and the weight of certain tasks, that will also be done in the beginning. There will be a walk through with the whole team to finalize design specifications to make sure everyone is aware and agrees on the hardware and software specifications. All of the materials and components will be acquired and tested for compatibility.

Once all of the parts were acquired, the hardware components were connected and integrated. This includes the wheels, motors, robotic arm, power supply and sensors. After the system is built, we can evaluate the mechanical stability and verify that all of the connections are secure and properly integrated. Then, the components will be tested as well as the integration of the hardware. The motors and wheels will be tested on various surfaces and mobility of the rover. The main area that will be tested for the motor and the wheels will be the sand, we want to make sure that the wheels are perfectly capable of driving on the sand with no problems at all. The robotic arm was tested for a range of motion and precision in picking up the trash. We tested the arm with multiple trashes ranging from small to large to see how well it would perform. We would also check and see how stable the mount of the robotic arm is to make sure it will not fall off on its own because that will be pretty pricey if something were to happen to the robotic arm. Here, we made sure that the robot is not front-heavy that will cause the rover to tip over when in motion. The power supply was checked for proficient power distribution and battery life.

After the hardware development is constructed, the software implementation can begin. The SLAM software interface can be implemented for the LiDAR puck to see real-time navigation and avoid obstacles. The Meca500 will be programmed using its interface, MecaPortal and the YOLO interface will be implemented for the trash detection through the camera. The sorting algorithms will be developed and categorize the trash into glass, aluminum and plastic. Each software interface will be tested individual and together to ensure that they function properly. ROS will be implemented to ensure smooth communication between the robot and the entire system using different software modules.

It's crucial that both the hardware and software components are integrated seamlessly to build a functional prototype. The ESP32 and Raspberry Pi 4 must be able to communicate together and with all of the sensors and actuators. Once ALiCS is ready to be tested, the system will be tested in a controlled environment to see how it will respond to picking objects and ensure that all components work well together. If any problems arise, we can troubleshoot these issues before testing on a beach.

Once ALiCS is functioning, we can assess the performance of the system and improve in different areas. While collecting data after each test, we could improve the accuracy, speed, detection, battery life etcetera to improve ALiCS. If there is extra time, our stretch goals could be implemented to improve our beach robot.

Chapter 10 Administrative Content

10.1 Budget

Although there are no external sponsors for this project, the group was able to source a substantial amount of supplies for the design build through donations and “on loan” from one of the team members’ employers. The Mecademic Meca500 Industrial Robot Arm was an essential component that will allow the team to retain capital and focus on sourcing less expensive parts. In addition, the team will have access to 3D printing and a small machine shop for custom brackets/fixtures. The table below details the critical components necessary to have a successful design. For the remainder of the items that aren’t donated, the group will be dividing the cost of parts between the 5 members. This will help gain experience in finding cost-effective, off the shelf solutions throughout the design process. The group has agreed to set the budget to $500, this will cover the supplies listed below with some cash flow to cover unforeseen expenses.

Table 10.1.1: *Preliminary Design Budget*

Part	Quantity	Total Price
Wheels	4	$120
57BLF03 Motors	4	$144
Hc-SR04 Sensor	2	$18
Battery EXP12200	2	$80
ESP32	1	$10

Rasp Pi 4	1	$60
Robotic Arm	1	On Loan
Robot Claw	1	On Loan
LiDAR Puck	1	$98
Raspberry Pi Camera	1	$36
Enclosure	1	Donation
Misc wiring/connectors	1	Donation
Spectro Slit	1	$113
Spectro Mirror	2	$133
Spectro Grating	1	$129
	Total =	$941

10.2 Distribution of Worktable

Below is a table that outlines each specific task that is expected from each group member in which it ensures a clear division of responsibilities. There are integration meetings to ensure that each section flows seemingly together and test the system as a whole.

Table 10.2.1: *Distribution of Worktable*

Computer Engineering	**Responsibility**	**Tasks**
Andre Reveles	Program Robot Arm	Program the robotic Arm: write a programming algorithm using MecaPortal that will allow the arm to pick up trash and move
		Integrate the Camera: Collaborate with Mary so that the arm's movements align with the camera detections
		Simulate and Test: Simulate the program in a controlled environment and test on the beach
Computer Engineering	**Responsibility**	**Tasks**
Mary Bartlinski	Program Camera Vision	Program the Camera: implement YOLO for object detection that will determine trash from objects

		Integrate the Camera: Collaborate with Mario and Andre so that the camera works seamlessly with the arm and LiDAR puck.
		Simulate and Test: Simulate the YOLO algorithm and test the camera to distinguish between trash.
CREOL: Photonics	**Responsibility**	**Tasks**
Mario Puesan	LiDAR Puck and Spectrometer	Image Processing Algorithm: Develop and implement an algorithm that measures distances and provides an updated map in real-time to localize the robot. Design and implement a spectrometer that will allow the robot to measure distances of objects
		Integrate the Arm: Work with Andre and Mary to ensure that the LiDAR puck and spectrometer works simultaneously with the robotic arm and camera for object detection and rover movement.
		Test: Test the camera in various lighting and different beach conditions
		Trash differentiation algorithm: create and test an algorithm that is able to use the spectrometer to split up the trash into different types
Computer Engineering	**Responsibility**	**Tasks**
Luiz Hernandez	Program the Self-Driving Car	Car Selection: Research and select parts for the base of the robot.
		Navigation System: Develop an algorithm that will allow the car to self drive on different surfaces and detect objects, making sure that it interacts with the camera and arm

		Testing: Test the car in different beach conditions making sure it's well structured
Electrical Engineering	**Responsibility**	**Tasks**
Sean Waddell	Supply Power to Robot and PCB design	Power System Design: Design a system that will allow power to flow throughout the whole robot selecting batteries and circuits that provides sufficient and efficient power to provide the most efficient battery life and performance
		PCB design: Design a PCB that will match our test microcontroller and have all the peripheral, memory, and other specified requirements.
		Safety and Testing: Implement safety features to prevent overheating and exhaustion as well as ensuring no water damage to the system. Test in different temperatures.

10.3 Milestones

Below are two tables that tell the meeting and event times that are required to create good time management between our group members and give us plenty of time for each step of the process. These milestones will lead us down the right path to success for our project. The first table will describe the time spread out for making the project idea and paper. The second table will describe making the prototype for the project.

Table 10.3.1 *Project Initialization Milestone*

Project Design				
Start Date	**Planned Date**	**End Date**	**Task**	**Description**
5/14/24	5/14/24	5/16/24	Recruitment	Members recruited: Mario Puesan (PSE)**,**Mary Bartlinski (CPE), Andre Reveles (CPE),

				Luiz Hernandez (CPE), Sean Waddell (EE)
5/16/24	5/25/24	5/27/24	Brainstorming and Decision	Talking over zoom and deciding from everyone presented projects which one to do
5/16/24	5/30/24	5/31/24	Divide and Conquer Paper	First 10 pages of our final document
5/16/24	7/4/24	7/5/24	60-Page Paper	60 pages made for our final draft of the paper
5/16/24	7/22/24	7/23/24	Final Paper	150 pages for the final draft of our project in Senior Design 1

Table 10.3.2: *Project Prototyping Milestone*

Project Prototyping				
Start Date	**Planned Date**	**End Date**	**Task**	**Description**
8/19/24	8/20/24	8/23/24	BOM/ Part Purchase	Making a BOM based on the parts we need with room to grow in case we need more or less for the project. When BOM is finished, we will begin purchasing parts
8/23/24	8/24/24	9/13/24	Individual Research/Design	Once parts have arrived, we will begin working individually on our parts while also communicating our

				progress with each other.
9/13/24	9/14/24	10/27/24	System Research/Design	Designing the system together with all the parts connected and first look at what the final design will accurately look
9/13/24	9/13/24	10/9/24	PCB Design	Design a PCB to fulfill the requirements of our test PCB with all the required specifications needed to run our project
10/14/24	10/15/24	10/20/24	PCB Testing	Testing the PCB out to make sure it works as well or better than our test Microcontroller for our project
10/27/24	10/29/24	12/1/24	Prototype Completion	The completion of our Prototype and the end of Senior Design 2

Various tasks overlap as expected due to how the project is done with previous work being important to future work and some tasks are TBD. These milestone tables are a guideline for reference to make a successful project.

Chapter 11 Conclusion

The Autonomous Litter Collection Sorting (ALiCS) is a self-driving rover that is able to scan the environment for any kind of obstacles and trash that is located on the beach. It will then use the robotic arm to pick up the trash and then use a spectrometer to scan and detect what type of material the trash is made of. The three types of trash that we looked for were aluminum cans, plastic bottles, and glass bottles. Our motivation for creating this design is because we think that technology should be used to help have a positive effect on our environments. Not only that, but we were also inspired by some of the current existing technologies that are closely related to what we are trying to accomplish. The first one is the BeBot, which is a beach cleaning robot that essentially performs the same task that our design should accomplish, but the main difference is that it is able to use a rake to pick up any waste from the beaches, where as ours is using a robotic hand to pick up specific trash on the beaches. The second technology is the iRobot Roomba, and what it does is that it zigzags around a room randomly while vacuuming trash off the floor. The thing we want to use as something to learn from the roomba, is the navigational capabilities that it has. The reason why is because we aim for our rover to be autonomous with machine learning and the roomba is able to do something very similar. The last piece of technology that we talked about was the beach metal detectors, and how they work is that they are able to use either very low frequency or pulse induction to find metal that is laying underneath the sand. The reason why this inspired us was because we wanted to consider implementing this as a stretch goal because we believed that it could be important.

With that being said, we decided that our area of focus should be the beach and that is how we came up with the idea of ALiCS. Through our best efforts at researching about and choosing parts, we were able to settle on the parts we had. We decided that the components we included in our design will be a LiDAR Puck, a raspberry pi camera, a Meca500 robotic arm, beach balloon wheels, brushless DC motors, ultrasonic sensor, raspberry pi, and an esp32. We were able to choose a good selection of components that fit within our budget and as well as meeting our requirements.

After choosing the hardware, we then wrote about our considerable options for the software we plan on using. We believe that we should use YARP in order to help with the software framework for developing robot applications, ZeroMQ which is a general-purpose library that could help support and make communicating patterns easy to implement, and Robot Operating Systems (ROS) which is a software specifically designed for robotic applications. With the help of these three softwares, it would help us make it easier to implement the other softwares we plan on using. SLAM is able to use the sensors that we have (LiDAR puck and camera), and it is able to use it to scan the nearby area in order to collect data about the environment. YOLO is another software that is specifically used for the camera that is able to detect objects that appear in the camera in real time. Another software is the MecaPortal, which is the interface for the Meca500 robotic arm that we used. It is designed to make it easy to help control, program, as well as monitor the robot’s movement.

Going deeper into the specifics of ROS, there are some functions of ROS that we focused on, because they helped us greatly with what we planned on designing. The first one is roslaunch and it is basically the main function of ROS, anything that we use from ROS will all fall under roslaunch. Rostest is another function that allows us to test any kind of ROS node and we can also test multiple nodes at the same time. RViz is a tool we can use that will help visualize what our software is displaying and ROSbag records any sensor data and can display it to us when we wish to see it. The three ROS functions above will be able to help us test our design throughout the entirety.

Two other ROS functions that we used that helped us with the global and local pathing of the robot. The global pathing function we be used was the A* algorithm, which was able to find an optimal path that reaches from the starting point to a goal (in our case, trash) all while avoiding obstacles. It creates an optimal path using the data that was received from the LiDAR puck as well as the camera. When the optimal path is determined, the local path will then begin following that path, but make any slight adjustments based on changes, whether it be obstacles or possibly change in the sand. The local path we used is called Timed-Elastic Band and it works by taking the parameters of our design, and being able to use it to calculate an optimal trajectory that should be taken based on the area.

We also then began researching all the possible standards that would apply to our design as well as the constraints of our design. We made sure to adhere to any robotics and automations standards since our robot will be autonomous, optical standards since we plan on using optics with both the LiDAR and the camera, laser product standards since we used the LiDAR puck, and also portable and rechargeable battery standards since we decided to use rechargeable batteries. Another standard to adhere to is soldering standards when we need to solder components to the PCB board as well any other things we need to solder. The constraints we considered important towards our design were design constraints, time constraints, ethical constraints, economic constraints, and also environmental constraints.

Another thing we went over was the concerns of ChatGPT and other LLMs. ChatGPT has grown to be a very powerful resource that will most likely be capable of knowing and understanding too much in today's society. Proven by our case studies, it is able to see just how capable ChatGPT is when it can provide things like skeleton code, recommendations for lenses, and also understanding what microcontrollers are compatible with raspberry pi. There are some concerns as to how accurate it is as well as how ethical it would be if we were to use it in our project.

The next thing that was written about was the hardware designs using schematics and diagrams to show how we plan to design our robot hardware wise. As well as software design where we demonstrated our plan using diagrams on how the software will be implemented. Then, using hardware and software designs we had, we were able to write about the system fabrication, meaning how we would plan to place everything within our PCB.

The last few things that were written about were how we planned to test our design, the budget we have planned, and the distribution of the workload. Planning is an essential part of the design to make sure that every part is working on its own as well as working together. That is why the first thing we want to test is every part individually to make sure that they are working and will not be a problem later. Soon after, we tested the components when they were combined to make sure there are no issues with the compatibility of them. Lastly, we demonstrated an expected budget of how much we would pay for everything we have and also how we plan on splitting up the workload.

Appendix A - References

ResearchGate Article
"Rechargeable Batteries and Their Management," ResearchGate, 2011, https://www.researchgate.net/publication/224225487_Rechargeable_Batteries_and_Their_Management.

MathWorks SLAM
"Simultaneous Localization and Mapping (SLAM)," MathWorks, 2024, https://www.mathworks.com/discovery/slam.html.

MathWorks SfM
“What is Structure from Motion,” MathWorks, 2024, What is Structure from Motion? - MATLAB & Simulink (mathworks.com)

YARP Learning
"YARP Learning," YARP, 2024, https://www.yarp.it/latest/#yarp_learning.

Meca500 Robot Arm | Mecademic Robotics
"Meca500 Robot Arm," Mecademic Robotics, 2024, https://hs.mecademic.com/hubfs/Meca500/Meca500%20Documentation%20%2B%20FW%20(Latest)/User%20Manual%20for%20the%20Meca500%20(firmware%2010.2).pdf.

User Manual for the Meca500
"User Manual for the Meca500 (firmware 10.2)," Mecademic Robotics, 2024, https://hs.mecademic.com/hubfs/Meca500/Meca500%20Documentation%20%2B%20FW%20(Latest)/User%20Manual%20for%20the%20Meca500%20(firmware%2010.2).pdf.

Programming Manual for Mecademic Robots
"Programming Manual for Mecademic Robots (firmware 10.2)," Mecademic Robotics, 2024, https://hs.mecademic.com/hubfs/MC%20Programming%20Manuals/Programming%20manual%20for%20Mecademic%20Robots%20(for%20firmware%2010.2).pdf.

GeeksforGeeks Filter Circuit
"Filter Circuit," GeeksforGeeks, 2024, https://www.geeksforgeeks.org/filter-circuit/.

BeBot
“BeBot- Beach Cleaning Robot,” serial-cleaners, 2024, https://searial-cleaners.com/our-cleaners/bebot-the-beach-cleaner/

UL 4600

"Evaluation of Autonomous Products", shopurlstandards, 2024,
https://www.shopulstandards.com/ProductDetail.aspx?productId=UL4600_3_S_20230317&ShowFreeviewModal=1

UL 4600
"Standard for Safety for the Evaluation of Autonomous Products," users.ece, 2024,
https://users.ece.cmu.edu/~koopman/ul4600/index.html

Wheels Comparison
"Polyurethane Wheels vs Rubber Wheels: Which Material is Better?," stealtho, 2024,
https://stealtho.store/polyurethane-wheels-vs-rubber-wheels-which-material-is-better/#:~:text=Polyurethane%20wheels%20are%20more%20durable,cause%20damage%20to%20rubber%20wheels.

Wheeleez Wheels
"Polyurethane Wheel," Wheeleez, 2024,
https://wheeleez.com/product/wz1-30uc/

Ballon Wheels
"VEVOR Balloon Wheels," Amazon 2024,
https://www.amazon.com/VEVOR-Balloon-Wheels-Replacement-2-Pack/dp/B0B17HKXZS?th=1

Ballon Tires
"Extreme Max Balloon Sand Tires," Amazon, 2024,
https://www.amazon.com/Extreme-Max-3006-8425-Balloon-Tires/dp/B0BJ33M7HQ?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&smid=ATVPDKIKX0DER&th=1

Brushless DC Motor Info
"What are Brushless DC Motors," Renesas, 2024,
https://www.renesas.com/us/en/support/engineer-school/brushless-dc-motor-01-overview

Stepper Motors Info

“Stepper Motors Basics,” monolithicpower, 2024, https://www.monolithicpower.com/en/learning/resources/stepper-motors-basics-types-uses

Ultrasonic Sensor Info

“How Ultrasonic Sensors Work,” MaxBotix, 2024, https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work

Infrared Sensor Info

“Understanding Infrared Sensors:,” getsadeandsound, 2024, https://getsafeandsound.com/blog/infrared-sensor/

Soldering Standard Info

“IPC J-STD-001 Standard Soldering Requirements,” protoexpress, 2024, https://www.protoexpress.com/blog/ipc-j-std-001-standard-soldering-requirements/

Battery Standard Info

“Battery Standards,” epectec, 2024, https://www.epectec.com/batteries/battery-standards.html

Infrared Sensor Info

“Infrared Reflective Sensor,” waveshare, 2024, https://www.waveshare.com/wiki/Infrared_Reflective_Sensor

HC-SR04 Info

“Ultrasonic Sensor HC-SR04 and Arduino,” howtomechatronics, 2024, https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/#:~:text=The%20HC%2DSR04%20is%20an,an%20inch%20to%2013%20feet).

Ultrasonic Info

"Ultrasonic Sensor HC-SR04 with Arduino," elementzonline, 2024, https://www.elementzonline.com/blog/ultrasonic-sensor-hc-sr04-with-arduino

Floyd Algorithm

"Floyd Warshall Algorithm," geeksforgeeks, 2024, https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/

HC-SR04 Component Choice

"Ultrasonic Distance Sensor -5v(HC-SR04)," sparkfun, 2024, https://www.sparkfun.com/products/15569

Ultrasonic Sensor Choice #2

"MB1232 I2CXL MaxSonar EZ3," maxbotix, 2024, https://maxbotix.com/products/mb1232?variant=41051365310515&currency=USD&srsltid=AfmBOopuaAQ5p5S8HYLhSsTPQ89vXEbakUF38vVgm6LpgVfBU3zPSWmukbM

Ultrasonic Sensor Choice #3

"MA40S4S," mouser, 2024, https://www.mouser.com/ProductDetail/Murata-Electronics/MA40S4S?qs=dyqxRLCA4pZUsZeqTN%2FO6A%3D%3D&mgh=1&srsltid=AfmBOoo6J5LSqsmKJYx6mGsAY6lgEEUaCniWEoPWj3fBwaKYoBNHccGDE5E

BLDC Motor Choice #1

"Brushless DC Motor, Built-in Driver," robotshop, 2024, https://robotshop.com/products/e-s-motor-42d-60l-brushless-dc-motor-built-in-driver-24v-8000rpm?srsltid=AfmBOooH-wmDsgUqyKQ-wSrihQTbQPzCvzDpR4WduDccYkiWamwsIuSPW94

BLDC Motor Choice #2

"24V DC 57BLR BLDC Motor," robotdigg, 2024,

https://www.robotdigg.com/product/1235/spec/NTdCTFItMTM4Rg==/24V-or-36V-DC-57BLR-BLDC-Motor

BLDC Motor Choice #3
"High Torque Brushless DC Motor," oyostepper, 2024,
https://www.oyostepper.com/goods-1333-12V-24V-High-Torque-Brushless-DC-Motor-BLDC-Motor-Built-in-Drive-500gcm-5000RPM.html

Motor Driver
"L6230Q," estore, 2024,
https://estore.st.com/en/l6230q-cpn.html

Motor Info

"An Introduction to Stepper Motors, DC Motors, and Motor Drivers," monolithicpowers, 2024,

https://www.monolithicpower.com/learning/resources/an-introduction-to-stepper-motors-dc-motors-and-motor-drivers#:~:text=Motor%20drivers%20have%20a%20few,%2Dtemperature%20protection%20(OTP).

BLDC Motor Info

"3-phase 6-step control", st, 2024,

https://www.st.com/en/applications/industrial-motor-control/3-phase-6-step-control.html#:~:text=A%20six%2Dstep%20drive%20(also,sensorless%20control%20can%20be%20achieved.

Motor Driver Info

"DMOS driver for three-phase brushless DC Motor," st, 2024,

https://www.st.com/resource/en/datasheet/l6230.pdf

Machine Learning Info

" What is Machine Learning," ibm, 2024,

https://www.ibm.com/topics/machine-learning

Path Planning Info

“What is Path Planning,” mathworks, 2024,

https://www.mathworks.com/discovery/path-planning.html

Path Planning Info

“Path Planning,” sciencedirect, 2024,

https://www.sciencedirect.com/topics/engineering/path-planning

Global Planner Info

“Gloabl_planner,” wiki.ros, 2024,

https://wiki.ros.org/global_planner

A* algorithm info

“A* Search Algorithm,” geeksforgeeks, 2024,

https://www.geeksforgeeks.org/a-search-algorithm/#

Path Planning Info

“Global Dynamic Path Planning Fusion Algorithm Combining Jump-A* Algorithm and Dynamic Window Approach,” ieeexplore.ieee, 2024,

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9328551

Timed Elastic Band Info
“An Improved Timed-Elastic Band Algorithm of Autonomous Ground Vehicle in Complex Environment,” mdpi, 2024,
https://www.mdpi.com/1424-8220/21/24/8312#metrics

_Timed Elastic Band
“teb_local_planner,” wiki.ros, 2024,
https://wiki.ros.org/teb_local_planner

RViz Info
“Rviz,” wiki.ros, 2024,
https://wiki.ros.org/rviz

Rostest Info

"Rostest," wiki.ros, 2024,
https://wiki.ros.org/rostest

Rosbag Info
"Rosbag," wiki.ros, 2024,
https://wiki.ros.org/rosbag

Roslaunch
"Roslaunch," wiki.ros, 2024,
https://wiki.ros.org/roslaunch

Rostest Nodes
"rostest/ Nodes," wiki.ros, 2024,
https://wiki.ros.org/rostest/Nodes

ASR Info
"Ast_ftc_local_planner", wiki.ros, 2024,
https://wiki.ros.org/carrot_planner

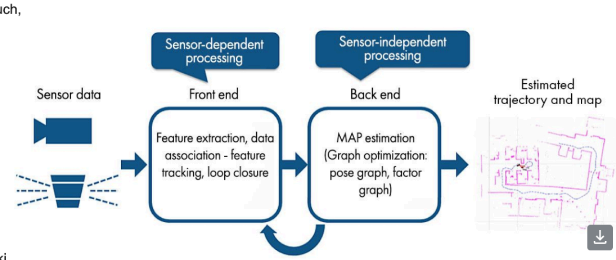
IPC standards
"Ipc standards for pcb manufacturing and assembly", pcbelec.com, 2024,
https://www.pcbelec.com/ipc-standards-for-pcb-manufacturing-and-assembly.html

Appendix B - Copyright

Permission for "SLAM Processing Flow" Image

Mary Bartlinski <bartlinskimary@gmail.com> 3:29 PM (0 minutes ago)
to service

Hello,
My name is Mary Bartlinski and I am an undergraduate student at the University of Central Florida working on my senior design project. I am researching and developing an autonomous beach robot clean up and we are incorporating a Lidar puck and spectrometer for real time mapping. I read your article about SLAM and was thinking of choosing it for our software tool to create the map. I would like to request permission to use an image from the article to use in my report. The document is for educational purposes at my university.
I have included a screenshot of the image and I will credit the image.
Thanks so much,



Mary Bartlinski

Vanessa Yamzon Orsi <vanessa@openrobotics.org>
to me

Hello Luis,

You may use the images from the ROS website along with proper attribution to the source site. Good luck with the paper!

Kind regards,

Vanessa Yamzon Orsi
CEO
Open Robotics

Appendix C - ChatGPT Case Studies

Case Study 1: "What are good lenses to use for a spectrometer?"

"Choosing the right lenses for a spectrometer depends on several factors, including the specific application, wavelength range, resolution, and cost. Here are some general recommendations for good lenses to use in a spectrometer:

1. Achromatic Lenses

- **Description**: Achromatic lenses are designed to limit the effects of chromatic and spherical aberration.
- **Advantages**: They provide better focusing of different wavelengths (colors) of light, making them ideal for applications requiring high precision across a range of wavelengths.
- **Use Case**: Suitable for UV, visible, and near-infrared spectrometry.

2. Aspheric Lenses

- **Description**: Aspheric lenses have a more complex surface profile that reduces spherical aberrations.
- **Advantages**: They can improve resolution and focus light more accurately than traditional spherical lenses.
- **Use Case**: Beneficial in applications where high precision and minimal optical distortion are required.

3. Plano-Convex Lenses

- **Description**: These lenses have one flat surface and one convex surface.

- **Advantages**: They are simpler and less expensive while providing good focusing properties.
- **Use Case**: Commonly used in basic spectrometry setups, particularly where cost is a concern.

4. Biconvex Lenses

- **Description**: These lenses are convex on both sides.
- **Advantages**: They provide better performance than plano-convex lenses in applications where the object and image distances are equal or nearly equal.
- **Use Case**: Useful in compact spectrometers and other optical instruments where space is limited.

5. Fresnel Lenses

- **Description**: Fresnel lenses are flat lenses that use a series of concentric rings to focus light.
- **Advantages**: They are lightweight and can focus light similar to a much thicker lens.
- **Use Case**: Good for portable and handheld spectrometers where weight and size are critical factors.

6. Cylindrical Lenses

- **Description**: These lenses focus light into a line instead of a point.
- **Advantages**: Useful for applications that require focusing light along one dimension, such as slit imaging in a spectrometer.
- **Use Case**: Often used in spectrometers that require high sensitivity and precision in one dimension.

Material Considerations

- **Fused Silica/Quartz**: Excellent for UV and visible light due to its high transmission and low absorption.
- **BK7 Glass**: Common and economical, suitable for visible and near-infrared applications.
- **Calcium Fluoride (CaF_2)**: Ideal for UV, visible, and infrared applications due to its wide transmission range and low birefringence.

Coating Considerations

- **Anti-Reflective (AR) Coatings**: Reduce reflection losses and improve transmission efficiency.
- **UV Coatings**: Enhance performance in ultraviolet spectrometry.
- **Infrared Coatings**: Optimize performance in the infrared spectrum.

Selecting the right lenses for a spectrometer involves balancing performance requirements with cost considerations. Achromatic and aspheric lenses are excellent for high-precision applications, while plano-convex and biconvex lenses offer good performance at a lower cost. Fresnel and cylindrical lenses provide unique benefits for specific use cases. Additionally, material and coating choices are crucial to optimize lens performance for the desired wavelength range." (OpenAI 4.0, 2024)

"There are two main types of lenses used in spectrometers: collimating lenses and focusing lenses.

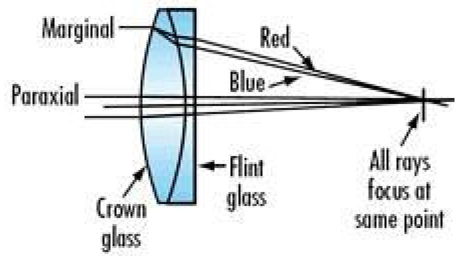
- Collimating lenses are used to collect light from the source and collimate it into a parallel beam. This is important because the diffraction grating, which separates the light into its constituent wavelengths, works best with parallel light.
- Focusing lenses are used to focus the collimated light onto the detector. The detector measures the intensity of the light at each wavelength, which is used to create a spectrum.

The specific type of lens that is best for a particular spectrometer will depend on the wavelength range of the spectrometer and the desired resolution. Here are some general guidelines:

- For spectrometers that operate in the visible and near-infrared (NIR) range, achromat lenses are a good choice. Achromat lenses are designed to minimize chromatic aberration, which is the tendency of a lens to focus different wavelengths of light at different distances.
- For spectrometers that operate in the ultraviolet (UV) range, quartz lenses are a good choice.
  pen_spark
  Quartz lenses transmit UV light well, while glass lenses absorb UV light.
- For high-resolution spectrometers, aspheric lenses are a good choice. Aspheric lenses have a more complex shape than traditional spherical lenses, which can help to reduce aberrations and improve image quality.

Here are some examples of specific lenses that can be used in spectrometers:

- **Achromat lenses:** These are a good general-purpose lens for spectrometers that operate in the visible and NIR range. You can find achromat lenses from various retailers.

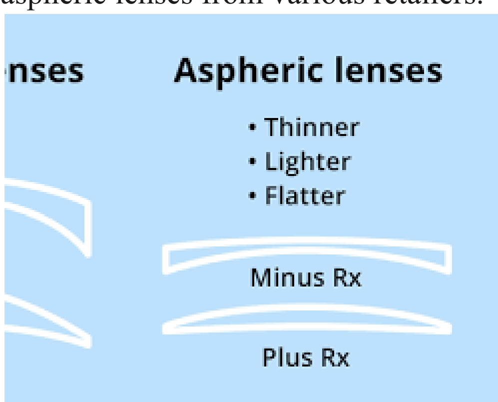


Achromat lenses

- **Quartz lenses:** These are necessary for spectrometers that operate in the UV range. You can find quartz lenses from various retailers.

Quartz lenses

- **Aspheric lenses:** These are used in high-resolution spectrometers. You can find aspheric lenses from various retailers.



Aspheric lenses

It is important to note that these are just general guidelines. The best way to determine the best type of lens for a particular spectrometer is to consult with the manufacturer of the spectrometer.

(Gemini 2024)

Case Study 2: "What microcontroller is more compatible with the raspberry pi 4?"

"The Raspberry Pi 4 is a powerful single-board computer that can interface with a variety of microcontrollers to extend its capabilities. Here are some microcontrollers that are particularly compatible and commonly used with the Raspberry Pi 4:

1. Arduino

- **Arduino Uno, Nano, Mega, etc.**
  - **Compatibility**: Easily interfaces with the Raspberry Pi via GPIO pins or USB.
  - **Advantages**: Large community support, extensive libraries, and well-documented examples.
  - **Use Case**: Ideal for prototyping, sensor integration, and control applications.

2. ESP8266 and ESP32

- **ESP8266 (NodeMCU), ESP32**
  - **Compatibility**: Can communicate with the Raspberry Pi via Wi-Fi, serial communication, or GPIO.
  - **Advantages**: Built-in Wi-Fi and Bluetooth (ESP32), powerful and low-cost.
  - **Use Case**: Great for IoT applications, wireless communication, and edge processing.

3. Microchip PIC Microcontrollers

- **PIC16, PIC18, etc.**
  - **Compatibility**: Interfaces via serial communication (UART, SPI, I2C).
  - **Advantages**: Wide range of options, reliable and robust for industrial applications.
  - **Use Case**: Suitable for embedded systems, control applications, and automation.

4. STM32

- **STM32F0, STM32F1, STM32F4, etc.**
  - **Compatibility**: Can interface via various communication protocols (UART, SPI, I2C).
  - **Advantages**: High performance, extensive peripheral support, and scalability.
  - **Use Case**: Ideal for complex applications requiring high processing power and peripheral integration.

5. ATtiny Series

- **ATtiny85, ATtiny84, etc.**
  - **Compatibility**: Can interface via serial communication (I2C, SPI).
  - **Advantages**: Very small form factor, low power consumption.
  - **Use Case**: Suitable for small, low-power applications, such as simple sensors or actuators.

6. RP2040

- **Raspberry Pi Pico**
  - **Compatibility**: Native support from Raspberry Pi Foundation, easy to interface via GPIO and USB.
  - **Advantages**: Dual-core ARM Cortex-M0+ processor, programmable in MicroPython and C/C++.
  - **Use Case**: Perfect for projects requiring tight integration with the Raspberry Pi ecosystem, educational purposes, and prototyping.

Key Considerations

When choosing a microcontroller to pair with the Raspberry Pi 4, consider the following factors:

- **Communication Interface**: Ensure the microcontroller can communicate via a protocol supported by the Raspberry Pi (e.g., UART, SPI, I2C).
- **Power Requirements**: Verify that the Raspberry Pi can supply the necessary power to the microcontroller or if additional power sources are needed.
- **Software and Libraries**: Check for available libraries and examples that facilitate integration.
- **Performance and Features**: Match the microcontroller's capabilities to the requirements of your project.

For ease of integration and community support, **Arduino** and **Raspberry Pi Pico (RP2040)** are excellent choices. For IoT and wireless projects, **ESP8266** and **ESP32** provide robust solutions. For high-performance applications, **STM32**microcontrollers are well-suited. Each of these options can be effectively paired with the Raspberry Pi 4 to extend its functionality and enable a wide range of projects." (OpenAI 4.0, 2024)

"The Raspberry Pi itself is a single-board computer (SBC) with a central processing unit (CPU). It already has processing power and doesn't necessarily need another microcontroller for basic tasks.

However, if you want to add some specific functionality to your Raspberry Pi 4, here are some factors to consider when choosing a microcontroller:

- **Functionality:** What specific task do you want the microcontroller to handle? Microcontrollers are good for low-power tasks that don't require a lot of processing power, such as reading sensors, controlling LEDs, or communicating with other devices.

- **Communication Interface:** How will the Raspberry Pi communicate with the microcontroller? Some common communication interfaces include SPI, I2C, and UART. Make sure the microcontroller you choose has a compatible communication interface.
- **Development Environment:** What programming language and development environment are you comfortable with? Some microcontrollers are easier to develop for than others.

Here are two examples of microcontrollers that are compatible with the Raspberry Pi 4:

- **Raspberry Pi Pico:** The Raspberry Pi Pico is a low-cost microcontroller board that is based on the RP2040 microcontroller chip. The RP2040 chip is a dual-core ARM Cortex-M0+ processor that runs at up to 133MHz.The Raspberry Pi Pico has a variety of built-in features, including SPI, I2C, UART, and ADC. It is also programmable in C++ and MicroPython.
- **Arduino Uno:** The Arduino Uno is a popular microcontroller board that is based on the ATmega328P microcontroller chip. The ATmega328P is an 8-bit AVR microcontroller that runs at 16MHz. The Arduino Uno has a variety of built-in features, including SPI, I2C, and UART. It is also programmable in Arduino C, which is a variant of C++.

These are just two examples, and there are many other microcontrollers that are compatible with the Raspberry Pi 4. The best microcontroller for you will depend on your specific needs. (Gemini, 2024)

Case Study 3: Raspberry Pi 4

“The Raspberry Pi 4 is a powerful single-board computer that can be used for a wide variety of applications. It is a popular choice for hobbyists, educators, and professionals alike. The Raspberry Pi 4 is the latest iteration of the Raspberry Pi series and offers significant improvements in processor speed, multimedia performance, memory, and connectivity compared to prior models.” (Gemini, 2024)

Figure 5.2.1 *Case study #4 by Gemini, Google AI*

“The Raspberry Pi 4 is a popular single-board computer developed by the Raspberry Pi Foundation. It's known for its versatility and affordability, making it suitable for various projects ranging from hobbyist electronics to educational uses and even some small-scale server applications. Is there something specific you're interested in or need help with regarding the Raspberry Pi 4?” (OpenAI 4.0, 2024)

Study Case 4: “How can I program the meca500 robot arm to pick up objects with a raspberry pi?”

“Programming the Meca500 robot arm to pick up objects using a Raspberry Pi involves several steps, including setting up the Raspberry Pi, communicating with the Meca500, and writing the control code. Here’s a step-by-step guide to get you started:”

1. Setting Up the Raspberry Pi

- **Install Raspbian**: Ensure your Raspberry Pi has the latest version of Raspbian installed.
- **Connect to the Internet**: Ensure your Raspberry Pi is connected to the internet for downloading necessary packages and updates.

Update Packages: Run the following commands to update your system:

```
sudo apt-get update
sudo apt-get upgrade
```

2. Install Required Software

Python: Ensure Python is installed. It is pre-installed on Raspbian, but you can update it using:

```
sudo apt-get install python3
```

PIP: Install pip for managing Python packages:

```
sudo apt-get install python3-pip
```

Mecademic Robot Library: You need the Mecademic robot library for Python. You can install it via pip:

```
pip3 install mecademicpy
```

3. Connect to the Meca500 Robot

- **Ethernet Connection**: Connect the Meca500 robot to your Raspberry Pi via an Ethernet cable. Assign a static IP address to your Raspberry Pi in the same subnet as the robot arm.
- **Robot IP Address**: Note the IP address of the Meca500 robot.

4. Write the Control Code

Create a Python script to control the Meca500 robot arm. Below is an example code to pick up an object:

```
import mecademicpy.robot as mecademic

# Define robot IP address
robot_ip = '192.168.0.100'   # Replace with the actual IP address of your Meca500

# Initialize the robot
robot = mecademic.MecademicRobot(robot_ip)

# Connect to the robot
robot.Connect()
robot.ActivateRobot()
robot.Home()

# Function to pick up an object
def pick_object():
    # Move to the pick position (replace with your coordinates)
    # Close the gripper
    # Move to a safe position
```

5. Running the Code

- Save the script as pick_object.py.

Run the script using the following command:
python3 pick_object.py

Additional Considerations

- **Calibration**: Ensure that the Meca500 is properly calibrated and the coordinates for picking objects are accurate.
- **Error Handling**: Add error handling in your code to manage exceptions and ensure safe operation.
- **Safety**: Ensure that the workspace is clear and the robot operates within safe parameters to prevent damage or injury.

(OpenAI 4.0, 2024).

Appendix D - Data Sheet

Raspberry Pi 4 Pin layout