Technological Feasibility Analysis

March 11, 2022

Team Sponsor: **Alexander (Allie) Shenkin**
Team Mentors: **Tom Prys-Jones, Anirban Chetia**

Document produced by

Austin Malmin, **Conrad Murphy**, **and ShanHong “Kyle” Mo**

Table of Contents

1. Introduction

In the world of automation, normal mundane tasks are being replaced by artificial intelligence that can perform these jobs at a higher efficiency and with better accuracy. This precision is starting to see a breakthrough into the field of autonomous robots and drones. Today, humans are currently performing tasks that can easily be automated with self-guided drones, for tasks like search and rescue, package delivery, agriculture, reconnaissance, and mapping inaccessible spaces, the latter being the focus of this project.

Search and rescue missions can prove to be dangerous in places not easily navigated by humans, such as stranded people in cave collapses, house fires, war zones etc. Amazon has already taken use of autonomous drones in their delivery of goods throughout the United States and the United Kingdoms. The dusting of crops with drones and planes piloted by humans can be inaccurate and expensive, whereas autonomous drones offer precision and require no man hours which cuts down on costs. Mapping the upper and lower forest canopy currently requires a researcher to manually move a ground-based sensor through the forest which is time-consuming and difficult. The sensors used are Light Detection and Ranging (LiDAR). LIDAR sensors measure the time for a laser beam to return once it's fired to determine the distance of the object from the sensor. These sensors are also capable of creating 3D models of different surfaces of the earth ranging from ocean floors to jungle canopies. Ecologists perform research focused on how forests around the world respond to climate change and simulate the effects certain changes in the climate would have on the environment.

Dr. Alexander Shenkin is one researcher whose current workflow involves traversing the forest on foot and LiDAR scanning the surroundings from the ground level; this process is painfully slow, requires excessive manual labor, and results in incomplete data. Groups of researchers trudge through the forest floor on foot, then find adequate spots to set up a 360-degree LiDAR device to map the nearby surroundings. This cycle, of fighting through the forest underbrush to find spots for mapping, must be repeated hundreds of times over the course of a week to create a satisfactory map. Moreover, the mapping is entirely restricted to the ground level. It is incredibly difficult or even impossible to see into the canopy, and their details are missed. As a result, forest ecologists receive an incomplete picture of 3D forest structure. Since forest structure dictates forest function, incomplete data results in an incorrect assessment of what the forest is doing.

The solution is an automated drone that can maneuver in the space between the understory and the crowns of trees. By using a drone, Dr. Shenkin would be able to create more accurate and complete maps of the forest while saving many hours of manpower.

In this document, we break the project down into four parts. First is the list of challenges our team expects to face. Secondly, there is an analysis of different solutions, scored on their efficacy at addressing the challenges identified.Thirdly, the wide variety of different options to resolve each issue are detailed and scored based on their perceived efficiency in the context of our solution. Lastly, we cover the integration of each part into one effective solution with a preliminary architecture, as well as highlight the impacts of a successful project.

2. Technological Challenges

The team has determined that there are four different aspects to our solution. For each design decision, we will list the desired characteristics and their importance. Section 3 outlines a custom scoring rubric with which we will evaluate our different options for each technological challenge and reach a final decision.

2.0 Computing Requirements

The drone will need a computer mounted onto it for processing, and there is no reasonable choice except the Raspberry Pi. The only comparable options are Arduino, which is a microcontroller with no inherent computing power, and other “Pi” devices, which serve different purposes from the Raspberry Pi.

2.1 Memory Requirements

When working with a Raspberry Pi, there are a wide variety of storage options that can be used to provide external memory to the users. For this project, Team Mockingbird has identified either an SD card or a Solid State Drive such as an M.2 drive as suitable options for our project due to their portability, write speeds and compatibility with the RaspberryPi. The sponsor expressed that their desired minimum speed would be 100 MB/s. The memory must also be easily transferable for the user to take the information from the drive in the drone and transfer it to an

external computer. This enables our sponsor to take the maps made of the environments and load them into game engines to perform the research.

2.2 Operating System Requirements

An operating system already has a wide range of tasks but one that pilots a robot has been built upon and requires much more. One important, must-have feature in a robot operating system is compatibility with LiDAR and Ultrasonic sensors. Another important aspect of the system is documentation and support that is available surrounding it. The last major factor was the amount of overhead and power that was eaten by the operating system versus computing power. When discussing the different operating systems that could be used to pilot the drone three different systems fitted the needs of our project: ROS, LCM, and ZeroMQ.

2.3 Sensor Requirements

Sensors of various types, such as LiDAR and ultrasonic, will enable the drone to see its environment. Different sensors specialize in different jobs. Some are well-equipped for long-range vision and mapping, but function poorly at close range; other sensors have low range but more thorough and accurate detection within that range.

LiDAR sensors collect data effectively at long range for mapping purposes, but struggle to thoroughly map spaces at close quarters. They fire one-dimensional beams of light at various angles and measure the time it takes for the beams to reflect back onto the sensor. Using this information, the sensor generates clouds of points in 3D space where objects are located. A LiDAR is good with detecting and mapping large obstacles from a longer range; however, since each pulse is one-dimensional, the sensor is likely to miss objects spaced in between the pulses. While LiDAR is suitable for mapping, we cannot rely on one for navigating at close quarters if the sensor is prone to missing some objects entirely.

We considered ultrasonic sensors for more reliable close-range obstacle detection. Ultrasonic sensors emit sound waves and measure the time it takes for the sound to echo back to the sensor. Unlike LiDAR, ultrasonic sensors can project sound waves onto a continuous 3D surface, which means they are less likely to miss obstacles. However, they are prone to interference from other sounds, such as the drone’s rotors, unless working at relatively close range. We will test if ultrasonic is viable once we can buy one during the prototyping phase.

We also considered cameras for our drone's vision. A camera combines the advantages of both sensor types. It has LiDAR's long range for effective mapping, and like an ultrasonic sensor, it leaves no gaps in its vision. However, we suspect that a camera will require massive overhead. A camera-based vision system must process each individual pixel of each individual frame, which expends enormous amounts of processing power. Additionally, the system would also need to recognize shapes and estimate distance, both of which would require machine learning–another resource-intensive process. For these reasons, cameras are a tentative option at best until we can conduct further testing.

2.4 Vision Library Requirements

Whichever sensors we use for our drone, we will also need to find vision libraries which will allow our navigation system to interface with them. Vision libraries are built at various levels of complexity and often serve different purposes. For instance, some LiDAR vision libraries only manage low-level point cloud processing; others have built-in capability for SLAM (Simultaneous Localization and Mapping), which suits our needs for mapping the environment in real time. Also, libraries are often compatible with only a few types of sensors–for instance, only the RPLidar series or only an XVLidar. Our decisions on vision libraries will affect available options for our sensors, and vice-versa.

3.Technological Analysis

This section considers the options for the four main challenges we need to solve. First, choose a hard drive that has a quick enough write speed to satisfy the client's needs as well as compatibility with the Raspberry Pi that we will be using. Second, find an operating system that interfaces well with the drone's directional APIs. Third, choose a sensor or combination of sensors that can accurately detect its environment. Fourth, narrow down the computer vision libraries that we will use to direct the drone's movement.

3.1 Memory Requirements

Figure 3.1a is a comprehensive list of our options, scored by their price, read speeds, and write speeds. All drives inside the figure are 1TB as specified by our sponsor. In order to determine the rating of our memory options, a universal score out of 5 was given to each category for each

item. This was then multiplied by a weight based on the significance of the attribute. For the rating of costs, anything under $150 was given a 5, $150-$175 a 4, $175-$200 a 3, and anything more expensive than that a 0. Cost was given a weight of 1 as it is not the most important factor. Memory write speed rating was determined as follows: over 120 MB/s a 5 as this was the client's minimum request, 100-120MB/s a 4, 80-100MB/s a 3, and anything under 80 a 0. This category was given a weight of 3 as it is extremely important for the drone to keep up with the incoming sensor data and not stall out trying to write into storage. Our client was not concerned about memory read speed, as long as it read above 100MB/s, so we gave a 5 to any drive that exceeded 100MB/s and 0 otherwise. This category's weight is 1. We also considered compatibility with the Raspberry Pi system. With a traditional SD card or MicroSD card, there is already built-in support for the cards inside of a Raspberry Pi. This is not the case for an M.2 drive as it requires a separate cable to connect to the Raspberry Pi. The cable is bottlenecked at around 100MB/s. So while the M.2 drives have a significantly higher write speed, this cannot be utilized as the data bus limits the write speed to 100MB/s. Therefore, 15% reduction of score was taken from the M.2 drives. Finally the scores were multiplied by the respective weights and the mean of the score was taken.

3.1.1 Memory Options

In this section, each memory card will have a description of how its score was calculated. Refer to Figure 3.1a to see a graphical representation of this data and easily compare the scores for each.

SanDisk 1TB Ultra MicroSDXC UHS-I, Memory Card

The SanDisk card costs $129 which places it in the 5 category for costs and this category has a weight of 1. Its write speed is 120 earning it a 5/5 in this category with a weight of 3. The read speed is 120 which satisfies the minimum from the client earning it a max score with a weight of .1 as it's not super important. After computing the calculations, this card earns a final score of 6.834.

SanDisk Extreme® microSDXC™ UHS-I CARD

This card has a cost of $170, getting it just barely inside the 4 category which was multiplied by a weight of 1. The write speed was 90 which was below the request from the client earning it a 4

with a weight of 3. Its read speed of 160 satisfies the client's requirements, placing it in the 5 category with a weight of .1. The final score for this card is 5.5.

Crucial P2 1TB 3D NAND NVMe PCIe M.2 SSD Up to 2400MB/s

This device is an SSD card. Its cost is $89, earning a score of 5 with a weight of 1. The write speed category had a weight of 3 and the device had a speed of 1900 earning it a 5. Its read speed of 2400 also placed it in the highest category. Due to this being an SSD card that requires adapters for the Raspberry Pi, its final score was reduced by 15% for a total of 5.125.

SanDisk 1TB Extreme PRO UHS-I SDXC Memory Card

This card has a cost of $250, earning it the lowest score of 0. Its read speed was also below what the client requested, earning a 4 with a weight of 3. Its read speed was above the requested requirements earning a 5 but was only multiplied by a weight of 1 as this category was not mission critical. Its final score is 4.167.

ADATA Technology 1TB SWORDFISH M.2 2880 Solid State Drive

This is an M.2 drive with a cost of $90, making it an affordable option for this project and earning a 5 with a weight of 1. Its write speed was also well above the minimum (1200), earning a 5 that was then multiplied by a weight of 3. Its read speed of 1800 earns it a 5 with a weight of .1. Due to being an SSD, its score was decreased by 15% with a final score of 5.125.

SanDisk - Extreme PLUS 1TB microSDXC UHS-I Memory Card

This card costs $270, which earns it a 0 for the cost category. The write speed is 90, which makes it a 4 out of 5 overall with a weight of 3. For read speed, it earned a 5 with its speed being 170. This category had a weight of .1. Its final score is 4.167.

Samsung 1TB 980 PCIe 3.0 x4 M.2 Internal SSD

The cost of this drive is $100, earning it a score of 5 multiplied by 1 for the weight. Its read and write speeds were above the minimum earning a 5 for both categories. Taking into consideration the 15% reduction for SSDs, the final score was a 5.125.

Figure 3.1a Memory Scoring Results

Name	Cost + tax and shipping	Memory Write MB/s	Memory Read MB/s	Mean Score (max 6.83)
SanDisk 1TB Ultra MicroSDXC UHS-I, Memory Card	$129	120	120	6.83
SanDisk Extreme® microSDXC™ UHS-I CARD	$170	90	160	5.5
** Crucial P2 1TB 3D NAND NVMe PCIe M.2 SSD Up to 2400MB/s	$89	1900	2400	5.125
** ADATA Technology 1TB SWORDFISH M.2 2880 Solid State Drive	$90	1200	1800	5.125
** Samsung 1TB 980 PCIe 3.0 x4 M.2 Internal SSD	$100	3000	3500	5.125
SanDisk 1TB Extreme PRO UHS-I SDXC Memory Card	$250	90	170	4.167
SanDisk - Extreme PLUS 1TB microSDXC UHS-I Memory Card	$270	90	170	4.167

***indicates M.2 drive with a 15% reduced score*

Figure 3.1a: Chart that displays different storage options for the system, their cost, read/write speeds and their rating

3.1.2 Memory Final Decision

When comparing the different SD cards using the scoring metric listed above, the SanDisk 1TB Ultra MicroSDXC UHS-I, Memory Card proved to be the obvious choice as it meets exactly the criteria we were looking for in a memory card.

3.2 Operating System

Inside of this section, the following operating systems will be discussed for the drone: ROS / ROS2, LCM, and ZeroMQ. In order to score these systems, we looked at a wide range of deterministic factors and arrived at the following categories: documentation, ease for us to learn

and use, functionality and community support. Of these 4 categories, ease for us to learn and documentation are the most important factors earning weights of 3 each. Functionality was an important factor as well; therefore, it was given a weight of 2. Community support was given a weight of 1. Each category was scored out of 5.

3.2.1 Operating System Options

ROS / ROS2

ROS (Robot Operating Systems) is open-source software developed by Willow Garage in 2006 to enable scientists and researchers to spend their time doing research instead of trying to code and develop the bare bone essentials to render a robot useful. Since then, ROS has become the industry-standard operating system for getting the most out of your robots. This operating system utilizes bag files or logs to easily communicate sensor data to the engine for easy training and testing purposes. This software is so extensive that you can test code alongside it without the physical robot or hardware. This is done by creating a “digital twin” as described on ROS’s website, which can be extremely useful to train and test each robot.

Since ROS has become an industry standard, there is a large amount of community support when it comes to troubleshooting and solving current problems inside of ROS. It is also extremely easy to interface with and contains many APIs which can be called upon to obtain the data and point clouds to pass to the computer vision libraries. ROS also has installs available for Ubuntu, Linux, macOS, and windows making it extremely versatile. Our sponsor has also connected us with a professor on campus who is extremely knowledgeable in ROS and can show us how to work with it. Overall, this is the obvious choice.

ROS also has a second system titled ROS2 which runs natively on embedded microcontrollers and interacts directly with the ROS in real-time. ROS2 supports microROS, but as we will not be dealing with embedded microcontrollers, this is not a necessary feature to consider.

LCM

LCM is very similar to ROS in its approach but has one unique feature that differentiates it from ROS. LCM is extremely lightweight and can be useful if the robot does not have much computing power or overhead. The largest difference between LCM and ROS is the communication to the user. While ROS offers no tool for live communication with the user, LCM

can connect to the drone via WLAN. This is not necessary for our project, since the specifications enable us to perform all data handling on the drone itself. Also, this drone will be largely used inside the forest or jungle where a WLAN connection might not be feasible.

ZeroMQ

ZeroMQ is another alternative to ROS which prides itself on equal performance to ROS. This particular choice has the unique advantage of having a better design for performance scaling if given unlimited power resources. Unfortunately, power is a concern as we will need the drone to last as long as possible to complete the full mapping of the target area. Another feature of ZeroMQ is that it has a zero error rate. This sounds nice but is unsuitable for our project. Its “zero error rate” means that if an image fails to transfer over, the system instead sends nothing. As the drone will be attempting to navigate dense forests and jungles, any partially correct information is essential to be passed to the algorithms that will determine its path instead of leaving the drone in the dark. This implementation of handling image data is simply not ideal for this project, making ZeroMQ a non-starter for Project Glasswing.

Figure 3.2a Operating Systems Scoring Results

O.S.	Documentation (weight: 3)	Projected Learning Curve (weight: 3)	Functionality (weight: 2)	Community Support (weight: 1)	Score (max 11.25)
R.O.S/ ROS/2	5	5	5	5	11.25
LCM	4	2	5	4	8
ZeroMQ	3	3	5	2	7.5

Figure 3.2a: Chart that displays different Operating System options for the system, their documentation, projected learning curve, functionality, community support and score.

3.2.2 Operating System Final Decision

After much research, it has been determined that ROS is what Team Mockingbird will be implementing for this project as it scored highest based on our rating metric.

3.3. LiDAR

This section explains how LiDAR works and ranks LiDAR sensors based on several factors, with cost being one of the most important.

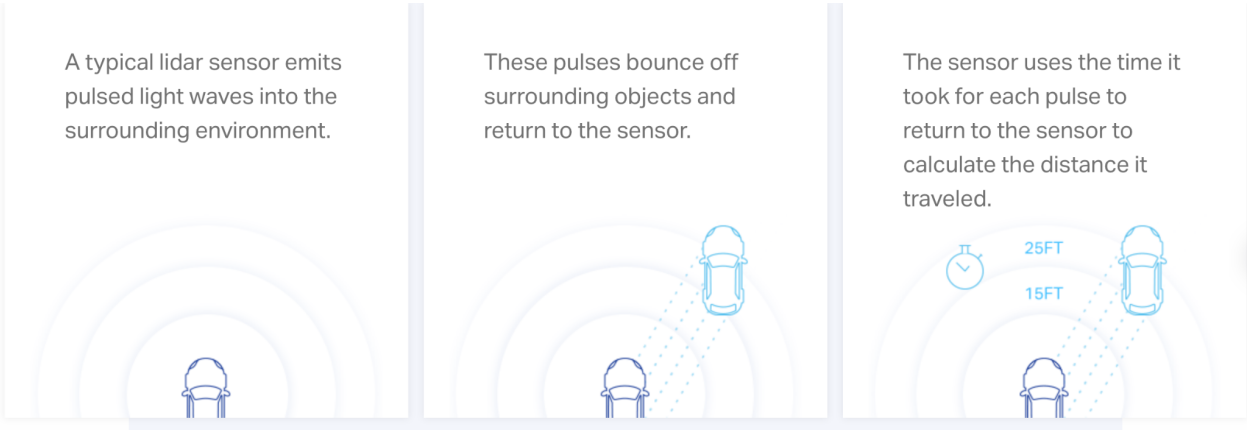
3.3.1 How LiDAR works

There are 3 main components to LiDAR:

- Laser
- Scanner
- Specialized GPS receiver

In general, most LiDAR will emit around a hundred thousand laser pulses of light per second, and when a pulse hits an object, it will bounce back to a scanner and be registered. The time it takes for a pulse to reflect back to the sensor can be used to calculate the point in 3D space off of which the pulse was reflected. Over time, millions of such pulses are collected into a map of the explored space.

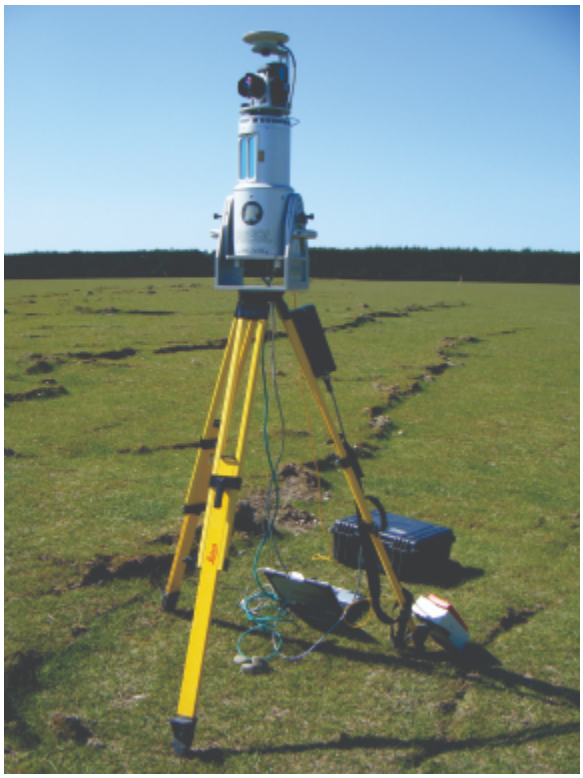
Figure 3.3a How Lidar Works



3.3.2 Types of LiDAR

There are two main classification or types of LiDAR systems:

- **Terrestrial LIDAR** operations happen on the Earth's surface and can either be stationary or mobile. This is currently being used by Dr. Shenkin and his mapping crew.



- **Airborne LIDAR system** is a technology in which a laser scanner, attached to an aircraft, such as helicopters, airplanes, or a drone, creates a 3D point cloud map of the landscape. We will be using this in our project.

3.3.3 LiDAR and Non-LiDAR Options

Key notes:

- Costs of all hardware must be below the allowed budget (around $200), therefore many LiDAR will not be considered in the evaluation below
- Some factors weigh more than others
- Because every LiDAR offers different capabilities, it is difficult to understand what it is truly capable of until we test them in real life; right now it is speculation based on the specs provided

Overall, the rating is based on affordability * (range + accuracy). Affordability has the largest weight because the project is primarily limited by budget. Anything that costs too much to be negotiable with the sponsor earns a 0.

Note also that a non-LiDAR with a better score than a LiDAR does NOT necessarily mean it is better, as each type of sensor has its own unique advantages and disadvantages. For example, thermal cameras cannot detect objects that don't release heat, so a wall could be an undetectable obstacle. Ultrasonic has the issue that drones are loud and could cause interference.

3.3.3.1 LiDARs

YDLIDAR X2 360° Laser Scanner

The cheapest LiDAR option available, but as a result, also the weakest. We will only use this if we cannot afford anything else; even though it meets the bare minimum requirements, the sponsor wants the product to be long-term.

- Offers 360-degree scanning distance measurement
- Provides stable distance measurement and high accuracy
- Can measure a distance of 0.1 - 8 m
- Uses 5-8Hz adaptive scanning frequency
- Costs: $69
- Score: 40

RPLidar A1M8 - 360 Degree Laser Scanner Development Kit

A small upgrade from the previous LiDAR, with functions above that of YDLIDAR X2 but also a steeper price. It's one of the biggest candidates for our project as it meets the wanted requirements of the sponsor. Overall, this could be our final choice.

- 360° Omnidirectional Laser Scan
- 5.5 - 10Hz Adaptive Scan Frequency
- Sample Frequency: 4000 - 8000Hz
- Distance range: 0.15 - 12m
- Range less than 0.1%
- Works with SLAM
- Costs: $99
- Score: 64

YDLIDAR X4 360° Laser Scanner

Similar specs to RPLidar, with similar range and frequency. Both products are similar enough to be indistinguishable just from reading the specs online. This is another likely candidate for our project as it does meet the wanted requirements of the sponsor.

- 360-degree scanning rangefinder
- 5000 times/sec sampling rates
- 11-meter scanning range
- Clockwise 360-degree rotary ranging module
- Auto mapping and positioning
- Costs: $99
- Score: 64

RPLIDAR S2 360° Laser Scanner (30 m)

The most expensive LiDAR considered by the sponsor, costing $366. Even though it is way above our discussed budget, Dr. Shenkin is considering it because he wants this project to be long-term. We will further discuss whether the desire for a high-quality product outweighs cost.

- RPLIDAR S2 360° Laser Scanner
- Offers 30 meters radius ranging distance
- Adopts the low power infrared laser light

- Provides excellent ranging accuracy
- Works well both in an indoor and outdoor environment
- Proof Level: IP65
- Costs: $366
- Score: 22

RPLidar ($100 - $200 for A1) ($300 - $400 for A2) ($400 - $600 for A3)

Has 3 different versions; all three versions are counterparts to the products above. This means even though it is a good product, other options have better specs for the price. For instance, the A3 costs around $600, way above RPLidar S2 which is half of that price.

- Up to sample rate up to 16000 times per second
- Up to 25 meter range radius
- Up to 360 degree scanning
- 4cm thick
- Many versions A1, A2, A3, depending on performance
- Costs: ($100 - $200 for A1) ($300 - $400 for A2) ($400 - $600 for A3)
- Score: 21

Vu8 Lidar Sensor

Far too expensive and will not be chosen.

- Detection range up to 700 feet (215 meters)
- Compact and lightweight (75 grams)
- 8 independent segments with simultaneous acquisition and lateral discrimination capabilities
- 20°, 48° and 100° beam width options, for optimized field of view
- Rapid refresh rate up to 100 Hz
- Immune to ambient light
- No moving parts, for ultimate robustness
- Easy to integrate, includes Leddar Enabler SDK
- Based on the modular LeddarVu platform for flexible integration and customization
- Best cost/performance ratio
- Costs: $700-$900
- Score: 0

There are many more LiDAR candidates, but they significantly exceed our budget. We included one LiDAR which costs over $500 as an example of what we find too expensive to choose.

Figure 3.3b shows the total scores for each LiDAR option.

Figure 3.3b LiDAR Scoring Results

Name of product (ordered by price from lowest)	Affordability: lower cost scores higher (weight: 0 to 4)	Range of scanning (weight: 0 to 10)	Accuracy: how detailed are the scans (mostly based on frequency and sample rates) (weight: 0 to 15)	Total rating (max 100): Costs * (added value of the other values)
YDLIDAR X2 360° Laser Scanner	4	3	7	40
RPLidar A1M8 - 360 Degree Laser Scanner Development Kit	4	6	10	64
YDLIDAR X4 360° Laser Scanner	4	6	10	64
RPLIDAR S2 360° Laser Scanner (30 m)	1	10	(unclear) Assuming 12	22
RPLidar A series	1	9	12	21
Vu8 Lidar Sensor	0	X	X	0

3.3.3.2 Non-LiDARs

MLX90640 Thermal Camera Breakout

Very good rating with great functions and specs with an acceptable price of $70, but unfortunately, it is a thermal camera, meaning that temperature and seasons greatly influences the function. Will be considered but only as a support camera for the LiDAR at most.

- Melexis MLX90640 far-infrared sensor array (datasheet)
- 32x24 pixels

- Field of view: 55°x35° or 110°x75°
- Up to 64FPS
- -40 to 300°C detection with approximately 1°C accuracy
- I2C interface (address 0x33)
- 3.3V or 5V compatible
- Reverse polarity protection
- Compatible with all models of Raspberry Pi, and with certain Arduino models
- Costs: $70
- Score: 68

Radiolink SUI04 Ultrasonic Sensor

Ultrasonic sensors have the downside that drones are very loud, meaning that sound interference could obfuscate the data. Will be considered only as a support sensor for the LiDAR at most.

- Need 6 sensors to be put on drone to allow 360 degree scanning
- Small size: 0.017Lb and 0.78*0.86*0.75in
- High accuracy: Support 0.4cm high detection accuracy
- Costs: 20 each
- Total costs: $120
- Score: 66

Note that there aren't many non-LiDAR options that are open for sale online and compatible with drones, so the options that we found have limited information on their prices and hardware specs. We found other sensors as well, but not their prices and specs.

Figure 3.3c shows the total scores for each non-LiDAR option.

Figure 3.3c Ratings of Non-LiDARs

Name of product (ordered by price from lowest)	Affordability/Costs *higher values means cheaper (weight: 0 to 4)	Range of scanning (weight: 0 to 10)	Accuracy: how detailed are the scans (mostly based on frequency and sample rates) (weight: 0 to 15)	Total rating (max 100): Costs * (added value of the other values)
MLX90640 Thermal Camera Breakout	4	7	10	68
Radiolink SUI04 Ultrasonic Sensor	3	10	12	66

3.3.4 Conclusion

Based on what we discussed with the sponsor, **YDLIDAR X4 LiDAR** and **RPLidar A1M8** are the best candidates for our project with possible support camera/sensor that goes along with it. We decided the **YDLIDAR X4** would be a slightly better pick, as it has better functions than the **RPLidar A1M8.** We can support the LiDAR sensor with the **Radiolink SUI04 Ultrasonic** because we believe that we can eliminate the noise from our calculations.

3.4 Vision Libraries

It is necessary to find existing software libraries that interact with the sensors that we plan to use. Vision libraries can be used to pull data from the LiDAR and ultrasonic sensors, such as point clouds and ultrasonic frequency. Some libraries also support SLAM (Simultaneous Localization and Mapping), a feature which could expedite our drone's mapping process.

Our chosen vision libraries must be compatible with our system specs (ROS, Raspberry Pi), should be compatible with many different brands of sensors, should support SLAM, and should be easy to use and understand. For system compatibility: Compatibility with ROS includes support for one of its native languages (Python and C++). We would prefer to use Python over C++ as we are more comfortable with the former. A library scores 3 out of 5 points if it is written in Python and an additional 2 out of 5 points if it has special compatibility with the system specs. System compatibility will have a weight of 2. For sensor compatibility: Many libraries only work

for a select few brands of sensors, which could limit our options for picking sensors; this is particularly true for LiDARs. A library which works for only a couple brands of sensors receives a 1; a library which works for any sensor receives a 5. Sensor compatibility has a weight of 3. For SLAM support: A library receives a 0 if it does not provide clear evidence that it supports SLAM, and a 5 if it supports outdoor SLAM. SLAM support has a weight of 3. For ease of use: Our team members will all need to set up the library software on their own machines, which may have different platforms (Windows, MacOS, iOS), as well as study and understand the software. A library which is overly complex in its setup or operation, has little cross-platform support, and/or is simply hard to understand receives a 0 or 1; a library with an easy setup and intuitive API, cross-platform support, and well-understood code receives a 4 or 5. Ease of use has a weight of 2.

3.4.1 Library Options

RPLidar

This low-level LiDAR library is a package for ROS. It is compatible with the RPLidar-A series of sensors, ranging from A1 to A3. It's written in C++, though modules written in Python also exist. The RPLidar library's primary function is to read the raw scan data from the LiDAR sensor and convert it to a usable format for ROS. RPLidar also has SLAM functionality and appears simple to use. Since RPLidar is directly built into ROS and supports SLAM, it scores well compared to the other libraries. However, RPLidar is a bit limited in its compatibility with different LiDAR sensors, which may pose an issue.

- System Compatibility: 5
- Sensor Compatibility: 2
- SLAM: 5
- Ease of Use: 4
- Total: (5*2 + 2*3 + 5*3 + 4*2)/10 = 3.9

HDL Graph Slam

Built for ROS, this is a library for 3D LiDAR sensors. It is written in C++, though it also interfaces with Python. Its main function involves taking in a scanned point cloud as input and procedurally generating a 3D map, which can be saved as a PCD file for permanent storage. HDL mentions that the library is compatible with LiDARs with 6 DOF movement–that is, one that can rotate along all three axes of 3D space. We do not plan on using a LiDAR with 6 DOF movement,

though this library might also work with simpler LiDARs. The authors have successful tests with HDL Graph SLAM in outdoor environments, which is crucial for our purposes. However, the library looks quite complex, with no less than 9 dependencies on other libraries and ROS packages. We like the options that this library provides, but we may consider looking for something else if the library ends up being too complicated and incompatible with our choice of LiDAR.

- System Compatibility: 5
- Sensor Compatibility: 3
- SLAM: 5
- Ease of Use: 1
- Total: (5*2 + 3*3 + 5*3 + 1*2)/10 = 3.6

Breezy Slam

This SLAM library is primarily for Python users. It looks to have the most minimal and lightweight interface of all LiDAR SLAM libraries researched. Breezy Slam includes support for several types of LiDAR sensors, including the RPLidar-A1 and XV-Lidar; however, the authors of BreezySLAM note that “I and several users had trouble using BreezySLAM with the RPLidar-A1.” This may be problematic for sensor compatibility.

- System Compatibility: 3
- Sensor Compatibility: 1
- SLAM: 5
- Ease of Use: 5
- Total: (3*2 + 1*3 + 5*3 + 5*2)/10 = 3.4

Laspy

This is a Python library. Its main function is to parse LiDAR scan data that has been written to a LAS or LAZ file rather than directly interfacing with the sensors. While this library may be suitable for reading the maps saved on our SD card after mapping is complete, the documentation makes no mention of real-time SLAM. Given these facts, it will not be easy to use this library for SLAM.

- System Compatibility: 3
- Sensor Compatibility: 5

- SLAM: 0
- Ease of Use: 2
- Total: (3*2 + 5*3 + 0*3 + 2*2)/10 = 2.5

GPIOZero

This library supports an HC-SR04 ultrasonic distance sensor for the Raspberry Pi. Even though the library is only compatible with one brand of sensor, the HC-SR04 appears to be the most common one on the market. There appears to be no SLAM support for ultrasonic sensors. The library is quite lightweight and has an intuitive interface.

- System Compatibility: 5
- Sensor Compatibility: 4
- SLAM: 0
- Ease of Use: 4
- Total: (5*2 + 4*3 + 0*3 + 4*2)/10 = 3.0

Adafruit CircuitPython

Adafruit is largely similar to GPIOZero. It supports the HC-SR04 and can be installed to a Raspberry Pi and has a similarly light interface. However, Adafruit has some additional dependencies that GPIOZero does not seem to have, so we slightly prefer GPIOZero.

- System Compatibility: 5
- Sensor Compatibility: 4
- SLAM: 0
- Ease of Use: 3
- Total: (5*2 + 4*3 + 0*3 + 3*2)/10 = 2.8

ORB SLAM2

ORB SLAM2 is a vision library for an RGB camera which supports SLAM. This library is built for ROS in C++. Supported cameras include the Intel RealSense r200 and d435, the Mynteye S, and the Stereolabs ZED 2. ORB SLAM2 has a command line interface with no direct API in a development environment, which may make it annoying to use in code. Documentation mentions nothing about camera SLAM overhead, so we will have to hold off on testing that until we decide to purchase a camera during the prototyping phase.

- System Compatibility: 2
- Sensor Compatibility: 3
- SLAM: 5
- Ease of Use: 2
- Total: (2*2 + 3*3 + 5*3 + 2*2)/10 = 3.2

3.4.2 Scoring Results

Figure 3.4.2a shows the total scores for each vision library.

Figure 3.4a Vision Library Scoring Results

Library	**System Compatibility (Weight: 2)**	**Sensor Compatibility (Weight: 3)**	**SLAM (Weight: 2)**	**Ease of Use (Weight: 3)**	**Total (out of 5)**
RPLidar	*5*	*2*	*5*	*4*	*3.9*
HDL Graph Slam	5	5	5	1	3.6
Breezy Slam	3	2	5	5	3.4
Laspy	3	5	0	2	2.5
GPIOZero	*5*	*4*	*0*	*4*	*3.0*
Adafruit	5	4	0	3	2.8
ORB SLAM2	*2*	*3*	*5*	*2*	*3.2*

Figure 3.4.2a: Vision library scoring. Italicized entries are our top candidates per type of sensor.

3.4.3 Vision Libraries Final Decision

Many of the libraries score similarly to each other, and the libraries will be easy to test further once we have physical sensors on hand, so our final decision is not crucial and may be subject to change. Since one of our candidates for a LiDAR of choice is the RPLidar A1, we will likely use the RPLidar library for our project. HDL Graph Slam is also an option if we decide to use the YVLIDAR instead, but we will need to invest time into learning its interface. GPIOZero and ORB SLAM2 are our primary options if we decide to implement ultrasonic sensors and/or cameras, respectively.

4. Technology Integration

Due to this project all being housed on the drone itself, the integration and flow of data does not get extremely complicated. The Lidar and ultrasonic sensors will pass their input to ROS which will parse the data into a point cloud. This will then be passed to the computer vision libraries, which will process the data and chart a safe flight plan given current information. This cycle of receiving new information and changing flight paths will be happening quite frequently inside the system. The main difficulty inside of this project will be connecting the wide variety of APIs ROS has to offer with the correct APIs of the computer vision libraries that are chosen for this project. Once the libraries have processed the data and determined the best route to take it is then communicated back to ROS as shown in Figure 3a. While this is taking place, the system will write the mapping to the hard drive which the sponsor of this project can then take and piece together a single complex map of the environment. This environment will then be plugged into a game engine where the sponsor can perform various tests and experiments.

Figure 4a shows a basic diagram of our system architecture.

Figure 4a: Architecture Diagram

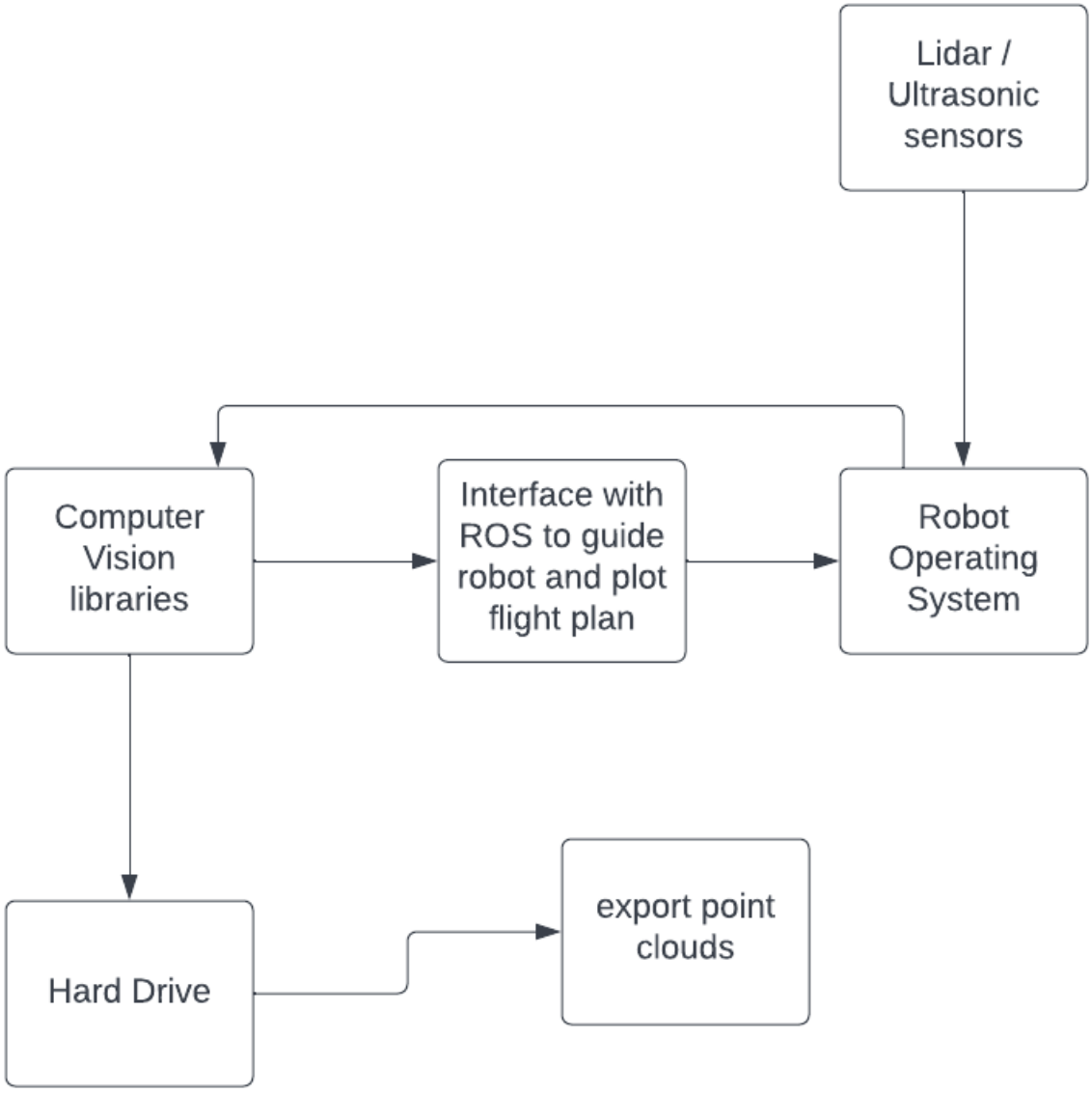


Figure 4a: Diagram of the architecture of our system, depicting how each technological component will blend together.

5. Conclusion

Our autonomous drone will navigate the different levels of the jungle and forest ecosystem. In order to do this we need accurate sensors to detect both near and far obstacles. This sensor data will be processed by our operating system, ROS, which will then guide the drone. This technology has the ability to transform not only the way jungle and forest ecosystems are mapped but also the wide-variety of fields mentioned above. We will build a navigation and mapping system for an autonomous drone which utilizes LiDAR and ultrasonic sensors. The system will interact with the drone through ROS, both by receiving its LiDAR / ultrasonic sensor data and by giving commands for how to fly without crashing. The system will also upload its data to a SanDisk 1TB Ultra MicroSDXC UHS-I Memory Card for permanent storage and future analysis.

Our mapping system will greatly improve Dr. Shenkin's workflow in his study of the rainforest, allowing for faster and more efficient data collection from our ecosystems. This data will help us understand our environment much more intricately, revealing important conclusions such as the effects of climate change and the mitigation of carbon in the atmosphere.

If we can fulfill our base-level goals, we will have a powerful module for automated navigation and mapping which could serve as the foundation for other data analysis tools. Dr. Shenkin could mount the drone with other types of sensors which measure light, humidity, heat, and other environmental factors, allowing for a more complete understanding of the environment's structure and function. Thanks to the drone's maneuverability, this comprehensive data would be quickly and easily obtainable. It would eliminate the trouble of sending a workforce of researchers to map by hand, saving hundreds of man-hours of labor.

For our next steps as a team, we will begin drafting our requirements document, where we will choose which of the sponsor's end goals to consider as requirements for our project. The sensors utilized for this project will be purchased by April 6th, at which point we can begin assembling the system and fine-tuning it during our prototyping phase. We will also discuss the content of our project's mini video overview.