Scalable and Fault-Tolerant Autonomous Hexapod Swarm for Search and Rescue Missions

Kobe Zhang, Akash Arun, and Casper Wong

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**Search and Rescue teams dealing with natural disasters face many challenges. For example: a lack of communication infrastructure, understaffing, harsh conditions, and compromised structural integrity. With modern technological developments, certain groups have been exploring the integration of autonomous systems with the Search and Rescue process. This project furthers this exploration by creating an autonomous swarm of hexapod robots that collaborate to complete search and rescue (SAR) tasks. The hexapod swarm utilizes LAN communication, YOLOv8 and R-CNN object detection, Visual SLAM, and a distributed search algorithm to get around the challenges that human SAR teams face and coordinate their search.

I. Introduction

Our product is a fully autonomous search and rescue system with a scalable number of hexapod robots. Each robot is responsible for mapping its terrain as well as identifying potential survivors. Designated MedBots have the additional responsibility of providing medical supplies / care packages to survivors identified by any robot that is part of the network.

The motivation for our solution is to reduce the need for human intervention in search and rescue as much as possible. Understaffing is a huge problem in search and rescue missions as workers are often not paid sufficiently to compensate them for the risk they undertake. A small rescue force may not be sufficiently large enough to cover a large search surface with adequate efficiency. In search and rescue missions every minute counts and can lead to loss of life. An additional problem is that these missions often involve workers being in precarious situations that risk injury or even death. Casualties and injury from such missions cause the workers and their families to have psychological stress from the vocation itself. Our product is scalable so the number of robots could be scaled up or down in response to the mission requirements and fault tolerant so it can work even if some robots fail. Hence human intervention might only be needed to interface with the robots.

An existing solution that we saw was Inuktun's small robots with tank-like treads that were used after 9/11 at the Twin Towers site and after Hurricane Katrina. These robots were very useful but a key difference between our solution and these was that robots were remote-controlled and needed a human to operate them. In comparison, our solution improves upon it by having our robots completely autonomous. Another difference is the establishment of a local network which helps each robot communicate its information with the other and optimize their collaborative search effort as much as possible.

II. Use-Case Requirements

We define the following Use-Case-Requirements for this project:

1. The hexapod swarm shouldn't need constant signal access to communicate with each other. This comes from the use case where our hexapods enter areas to perform search and rescue tasks without a strong network infrastructure.

2. The hexapod should have an active battery life of at least 1 hour. This is a requirement to ensure that the hexapods can conduct a thorough search of a house or enclosure.

3. The hexapods should have a high accuracy of detecting a possible survivor in the frame with a low false negative rate. This requirement is to ensure that our hexapods can correctly identify survivors in a search and rescue environment. Additionally, we want to have our hexapods lean towards more false positives for survivor detection than false negatives, since we don't want to accidentally miss any real survivor. Moreover, our model should be able to accurately identify many kinds of humans and not have implicit biases based on race, age, and gender.

4. The hexapod swarm should be scalable; it should be able to seamlessly incorporate additional hexapods to make the swarm more efficient and it should also be able to adapt to failures of single hexapods. Our solution would need to be flexible in swarm size for human SAR teams to be able to effectively deploy our swarm. In cases where the search location is larger, the swarm should be able to scale up in numbers to compensate for the increase in search area. Additionally, the swarm should be able to detect and account for failures in case hexapods die in the process of the SAR mission.

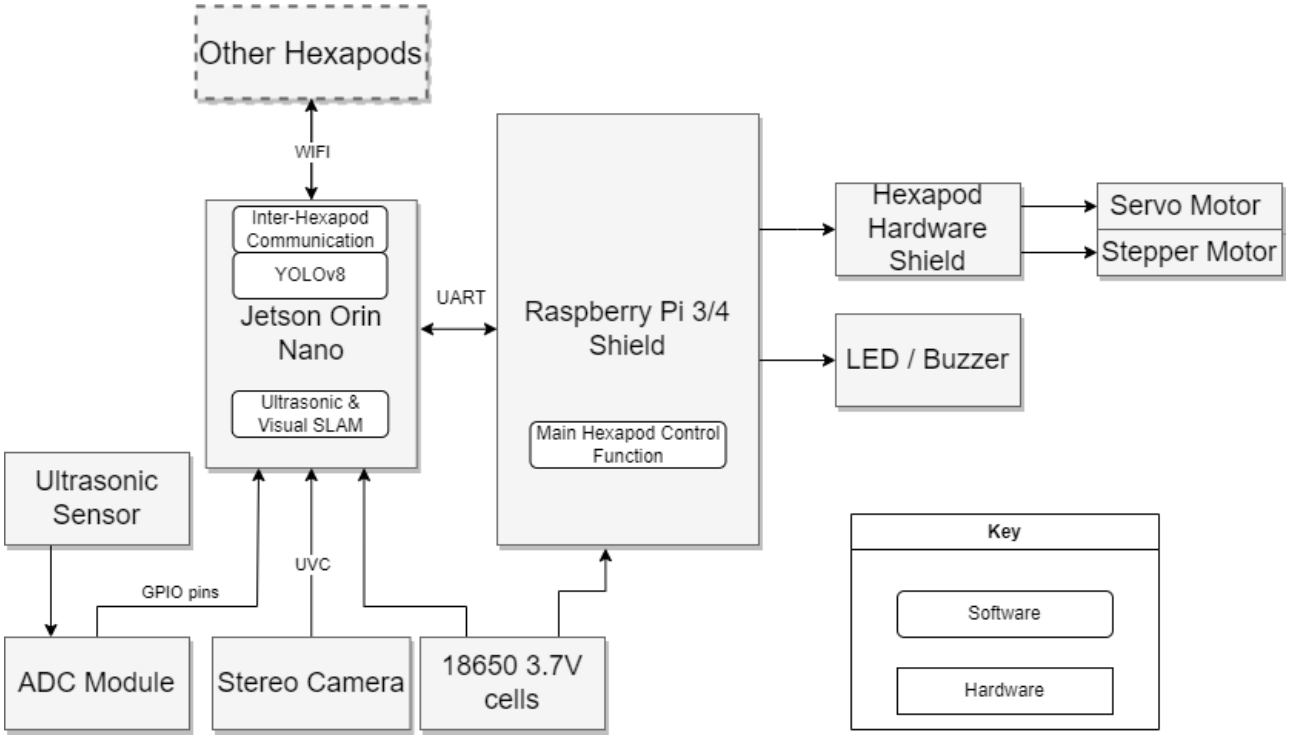

Figure 1: Block diagram of the system with one Hexapod

III. Architecture and/or Principle of Operation

To create our hexapod swarm, there are a few essential subsystems, which can be seen as a block diagram in Figure 1. More in-depth information on the implementation of these subsystems will be discussed in section VI. First, the hexapod control is a subsystem. In this project, we use an off-the-shelf hexapod robot from Freenove. This hexapod is controlled using an RPi that interfaces with a hardware shield that moves around the hexapod using its 6 servo-powered legs. An off-the-shelf hexapod is used to leave more time for the team to implement the desired swarm behavior, communication, object detection, and hexapod localization.

The swarm behavior, inter-hexapod communication, object detection, and localization are all individual subsystems that are implemented with the Jetson Orin Nano and its peripherals. The design went through multiple iterations and landed on the Jetson Orin Nano as the best computation unit for this project due to its relatively small size and weight in combination with its computational strength and software support. The Orin Nano will be interfacing with an IMX219 camera and an ultrasonic sensor via an ADC module to translate analog signals.

An attached Wi-Fi module to the Jetson Orin Nano will allow it to host a local network for communication with other hexapods without access to the internet. This local network will be the cornerstone of the inter-hexapod communication subsystem.

The object detection subsystem on the Orin Nano will run the YOLOv8 object detection algorithm to search for survivors, getting the image data from the IMX219 camera. The camera will be mounted on the head of the original hexapod, which is servo-controlled and can swivel 360º which provides a lot of flexibility for image collection.

The Orin Nano will utilize readings from the attached camera and ultrasonic sensor for Ultrasonic and Visual SLAM in the localization subsystem, which will allow it to localize itself and map its surroundings to remember paths and survivor locations. With this information, the hexapods will coordinate with fellow hexapods and route their search path based on a distributed search algorithm to optimize search area coverage in the swarm behavior subsystem. This Jetson Orin Nano will be mounted on the RPi (Raspberry Pi) and will communicate to the RPi via UART communication.

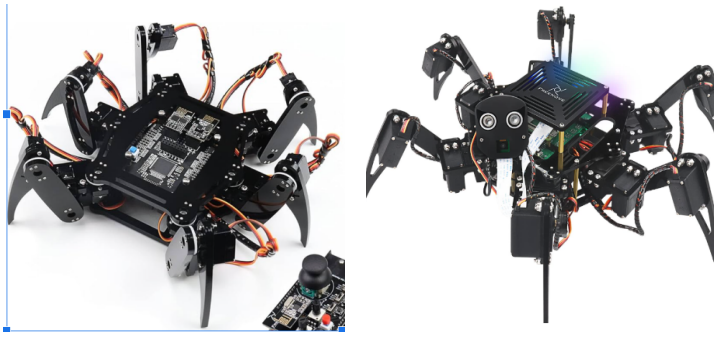

Figure 2: Images of off-the-shelf hexapod robot. (left) Arduino version. (right) RPi version with rotatable head.

IV. Design Requirements

For the project's design requirements, the focus is on 4 main aspects of our hexapod swarm function that were described in the use-case requirements: 1) Inter-hexapod communication, 2) Hexapod survivor identification, 3) Swarm scalability, and 4) Hexapod battery life.

1. For inter-hexapod communication, hexapods should be able to send packets to each other at varying distances up to 20m with <5% packet loss. The average global house is $20m^2$ hence we came to a max distance figure of 20m. This should be achievable as we plan on using 2.4GHz LAN which has a range of approximately 50 feet indoors.

2. For hexapod survivor identification, our team preferred to err on the side of caution and create a model that results in false positives rather than make one that neglects potential survivors. As a result, we want to be strict about having a greater than 5% rate of false negatives. This thought process led us to arrive at a figure of less than 80% mAP (mean average precision) for the detection accuracy of different kinds of human and non-human objects. 80% also represents a good balance between detection accuracy as well as the limitations of our hardware and real-time detection needs. When we ran experiments with YOLOv8 with images that had humans clearly in them it still had an accuracy of around 85-90%. This also influenced the selection of our figure since we can't predict if our accuracy will be worse or better when we train it with a custom dataset. We also need to are required to use a very diverse training dataset consisting of people of different races, genders, and ages so that we can maintain this 80% accuracy across the board and not have our model pick up unwanted biases.

3. For swarm scalability, the swarm should have a 1.5x search completion time speedup with 3 hexapods compared to a single hexapod. This requirement is so that the additional cost of having more hexapods is justified with a corresponding improvement in search efficiency.

4. For hexapod battery life, the battery duration for a hexapod should be >1 hour under an active load (i.e. constant movement, running a Jetson…etc.). The average search and rescue mission lasts for 31 hours, in a real-life use case where our solution is used the hexapod wouldn't be useful if it wasn't able to search for at least an hour before getting substituted with another hexapod.

V. Design Trade Studies

To get to our current solution, our team examined a lot of possible approaches to create our hexapod swarm. We first explored the various options for the hexapod itself. Freenove offered a variety of options for hexapod robots with varying costs and functionalities. The first hexapod we looked at was a smaller model that was controlled using an Arduino. While the low cost of this model was desirable, we were concerned about the ability of the smaller hexapod to carry larger loads including the weight of the Jetson Orin Nano and various needed peripherals. Additionally, we were unsure of the ability of the Arduino to handle communications from the Jetson Orin Nano while simultaneously running all the necessary controls for the hexapod movement. Due to these concerns, we decided to look at the largest hexapod available that is also controlled through an RPi. This hexapod, which is our current one, offers the additional advantage of having a head module that could swivel 360º, a desirable trait for our object detection tasks. While this hexapod came in at a much larger price than the original model, it also came with an ultrasonic sensor, camera module, and a higher weight capacity. Ultimately, we decided to move forward with this hexapod model for the benefits of computational power, head mobility, and weight carry capacity. The trade-off was an increase in cost per hexapod and an increase in power consumption.

Another critical design choice we made was for the main computational unit of our hexapod. We originally chose the NVIDIA Jetson Nano due to its low cost and GPU support. After a few rounds of initial trials with the software that we wanted to run, it was evident that the JetPack versions that the NVIDIA Jetson Nano was able to support were not enough for our project's needs, specifically for our object detection tasks. This was because the Jetson Nano could not support JetPack 4.7 and higher so it could not run Python 3.7 and above, which was critical to our object detection subsystem. After using a virtual machine to get around software dependencies, we were able to run the object detection algorithm we wanted but the detection process was too slow (~30 seconds) for our purpose. Thus, we decided to switch to using a Jetson Orin Nano. The original Jetson Nano was released in 2019 whereas the Orin Nano came out in 2023. The difference in computation power in comparison to their size difference is representative of these last 4 years of hardware innovation. Once again, the tradeoff is an increase in price ($150 vs $500) and an increase in power consumption (max 10W vs max 15W, 5V input vs 7-20V input), for better support, more modern software, and higher JetPack version support, faster computation (approx. 80x), and a lot more benefits. We realized that the Orin Nano supports the usage of Issac-ROS which helps us better utilize the GPUs of the Nano to perform object detection and SLAM more efficiently. The Orin Nano is also approximately the same size as the Jetson Nano, making it feasible to use with our hexapod without needing a substantial change in the design of our harness.

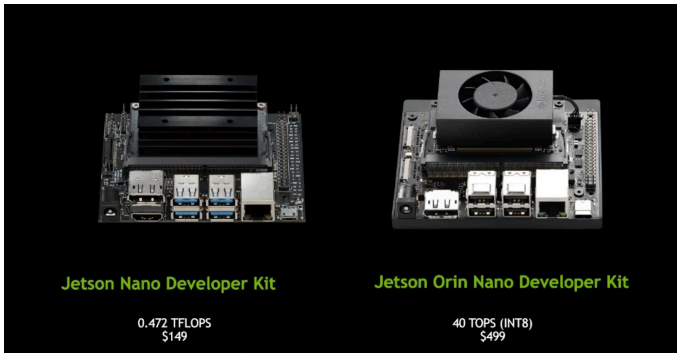

Figure 3. Comparison of Jetson Nano and Jetson Orin Nano

For our object detection subsystem, we compared various versions of the YOLO object detection algorithm for speed, accuracy, and ease of use. Since we upgraded from the Jetson Nano to the Jetson Orin Nano, we decided to continue with YOLOv8 which is one of the newest versions of YOLO that offers one of the highest accuracy ratings. While we were considering using YOLO-Nas since it utilizes quantization-aware training and post-training quantization to reduce the size of the model and increase performance, we valued the increased accuracy of YOLOv8 more. This decision was also motivated by the upgrade of our central computing unit to the Orin Nano which could run YOLOv8 with fast speeds. Additionally, we found that Isaac-ROS supported YOLOv8 and allowed for the hardware acceleration of our object detection through their NITROS optimization.

For the inter-hexapod communication subsystem, we chose to go with a local area network (LAN) with Wi-Fi over nRF and UWB. This is because upon conducting some deep research on different forums that discuss the applications of these protocols in various robotics projects, we discovered a couple of key challenges that we would have to face if we used UWB or nRF over Wi-Fi. One challenge stem from our usage of an operating system (Jetson Linux 36.2) with a scheduler rather than having a microcontroller that runs bare metal code. UWB or nRF have very tight timing/latency requirements that need to be met for it to function properly. This wouldn't be an issue in the case of Wi-Fi as the Wi-Fi card handles this requirement, but UWB Modules expect the user of the module to deal with the requirement, which the Linux scheduler would be unable to meet. A workaround could be to connect our Jetson to an Arduino or another microcontroller and write a UWB driver that helps us meet our timing needs. They also have lower communication bandwidth in comparison to Wi-Fi which might cause us issues down the road. Wi-Fi modules also have a lot more built-in support with drivers, etc. in comparison. The big advantage of these protocols over Wi-Fi, however, is that they consume very little power. After considering these facts, we decided to choose Wi-Fi because of how easy it is to work with and create a LAN. A convenient plus with this decision is that Jetson Orin Nano's come with Wi-Fi cards built in which saves us additional expenditure.

A key design decision in our project is to build our solution by leveraging ROS (Robot Operating System) which is an open-source middleware framework. This decision gives a handful of advantages. For one, ROS is very modular so it's easy to plug and play different software packages. It has an extensive network of researchers worldwide who contribute to its software packages which will further speed up our development time. This gives us access to pre-existing implementations of various object detection, SLAM, and control algorithms so we won't have to reinvent the wheel. ROS also supports simulations through tools like Gazebo and RViz so we can test our integrated system before we deploy it on the hexapod hardware.

VI. System Implementation

Object Detection

We will be using an object detection model to detect the presence of search and rescue survivors. This will predominantly include people, who may be partially obscured under rubble. Other potential objects of interest include pieces of clothing and domestic animals. We will train our model such that it commits virtually no type II errors so that the Hexapods do not accidentally ignore any survivors. To avoid false negatives, we will train our model on a diverse dataset that includes different ethnicities, genders, and other demographic variables.

To maximize the performance of the object detection model, we plan to combine the usage of a single-stage detector with a two-stage detector. During regular operation, the Hexapod will be using the fast object detector, such as YOLOv8 (You Only Look Once v8) and will use a slower but more powerful object detector, such as RCNN (Region-based Convolutional Neural Network), when the former detects an object of interest. We plan to run YOLOv8 on Isaac-ROS, which NVIDIA developed specifically to use GPU acceleration on their products, such as the Jetson Orin Nano.

Visual Simultaneous Localization and Mapping

Hexapods need to map their surroundings to remember paths and survivor locations – this is important for the designated Medbot to travel to other robots that have located survivors. To achieve this, we will be using the Isaac ROS Visual SLAM library developed by NVIDIA, which again utilizes GPU acceleration to provide low-latency results in a robotics application. Using an IMU and stereo camera, VSLAM combines visual-inertial odometry, which visually estimates the position of a robot relative to its start position, with SLAM, which creates a map of key points to determine if an area is previously seen. A demonstration of the library running on the Jetson Orin Nano can be found here. Using this library, the Hexapods can quickly map out obstacles such as walls and unsurmountable rubble, as well as retain the path that they took to get to their current locations.

Search Algorithm

We will be adapting and iterating upon a previously implemented cooperative search algorithm for distributed autonomous robots. It is a simple algorithm that almost fully eliminates communication between robots to reduce overhead when scaling up. Each robot follows 5 behavioral rules, prioritized with 1 being the highest:

1. Avoid obstacles and fellow robots – Using camera and ultrasonic sensors.
2. Find targets and alert neighboring robots – After finding a target, the robot stops and broadcasts to other robots.
3. Response to neighboring robots' messages – Robots will move away from others to avoid redundant searching.
4. Follow external commands – Robots listen to global communication on WAN for start and stop.
5. Wander in the environment.

Previous literature demonstrated that 5 robots can lead to a 60% increase in search efficiency.

As we can see in the original algorithm, robots do not need to know either their position or environmental layout. Although this prioritizes simplicity, it also leads to a lot of search redundancies, where multiple robots might search the same area over some course of time. So, we plan on improving this algorithm after implementation. Since we will have the location of robots using SLAM, we hope to keep track of previously visited locations and communicate this across robots. In doing so, the robots can follow a new rule of avoiding previously searched areas.

General Algorithm – State Diagram

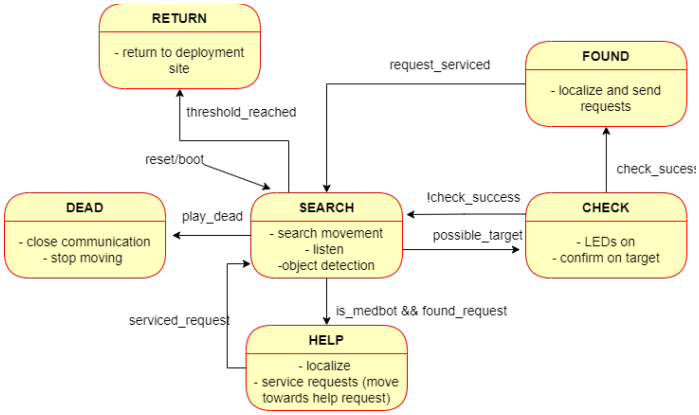

Figure 4. state diagram for hexapod behavior.

Upon start-up, the Jetson Nano Orin will run the algorithm shown in Figure 4, beginning in the search state.

In the search state, Hexapods will wander, using a provided actuation library, and actively run object detection to search for survivors as well as VSLAM to map out the environment and their path. Upon finding an object of interest with YOLOv8, the robot will run the stronger RCNN network to confirm whether the object is indeed a survivor. The robot will then be in either the help state if it is the Medbot, where it will help the survivor; or it will be in the found state, where it will signal for the Medbot to come and help the survivor. Finally, when Hexapods are running low on charge, they will return towards the deployment site so that they can be easily assessed and recharged.

The dead state is a custom state used in testing, in the case that a robot is destroyed during a real deployment.

VII. Test, Verification and Validation and Risk Mitigation

Building upon the design decisions outlined in the previous section, a crucial aspect of ensuring the effectiveness of our hexapod swarm solution lies in testing, verification, and validation. This stage involves a series of controlled and real-world evaluations to assess the functionality, performance, and suitability of the developed system. We have 4 main categories of design specifications: Communication, Identification, Scalability, and Battery Life.

A. Tests for Communication

Different messages will be sent between hexapods from distances ranging from 0.5m-20m. This test exists to verify that our solution is competitive in the average global household's dimensions.

For each of these distances, we will compare the percentage packet loss. We aim to have an overall average packet loss of less than 5%. If we cannot achieve this, we will consider trying other protocols like UWB or nRF. This test is required so we can affirm our first use case requirement which requires that the hexapods shouldn't be dependent on a consistent internet connection to communicate to the outside world. The packet loss requirement we enforce helps us attest to the reliability of our communication system.

B. Tests for Identification

We will create a test dataset consisting of a variety of human and non-human images which we will use to evaluate our model.

If we have more than a 5% rate of false negatives or less than 80% mAP then we fail our testing requirement. In this situation we will try to meet it by tweaking our training data set, having even more layers of object detectors, or using other model frameworks such as FOMO, detect-net, etc. as our main detector.

C. Tests for Scalability

We will run our search algorithm in a sample test environment that is $5m^2$ and randomly scatter rescue targets across the environment. We will do this for just 1 hexapod, 2 hexapods, and 3 hexapods to compare how long it takes to find all the targets in each case.

We aim for at least a 1.5x times speedup as we scale up from 1 to 3 hexapods. If we don't achieve this, we will use different more involved search algorithms / improve our communication and collaborative search scheme.

D. Tests for Battery Life

We will run our hexapods under maximum stress (max speed, SLAM, and Object Detection running) and evaluate how long it takes for the hexapods to be unable to function from a full charge. We will vary the number of 18650 Lithium-Ion cells to see how the number of cells affects the duration it lasts.

We require our system to last for at least one hour hence we will see how many cells are required for us to meet this threshold. If we fail to accomplish this, we will experiment with alternative power sources such as power banks, etc.

VIII. Project Management

A. Team Member Responsibilities

Casper is mainly in charge of the Harness and Structural Setup since he is most comfortable with CAD and 3D Printing. Kobe and Akash are responsible for Inter-robot communication and SLAM. Akash and Casper oversee the Distributed Search Algorithm. All 3 team members are responsible for Object Detection and overall testing.

B. Schedule

As seen from the schedule on the next page, we have planned to front load most of our project to be done before Spring break. This entails the ordering and assembly of the Hexapods, as well as setting up the Jetsons with object detection models and ROS 2. After Spring break, we will set up VSLAM and the search algorithm, then begin to test and integrate all the various components of our project to ensure that it functions as expected.

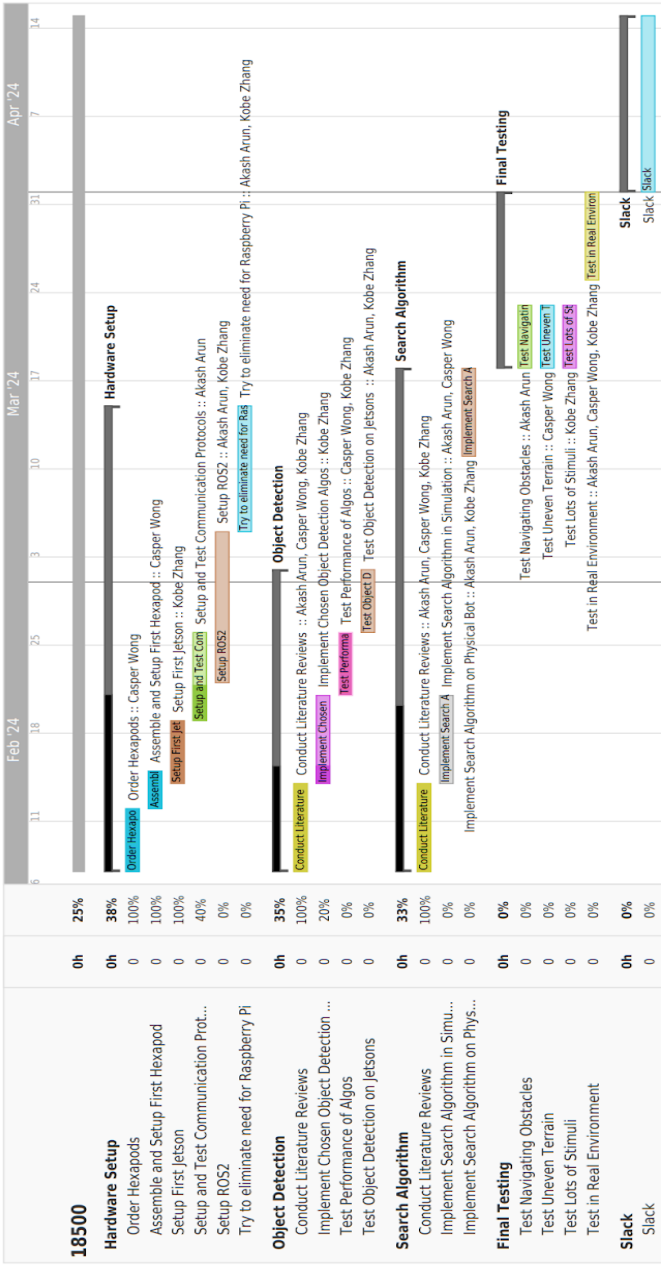

Figure 3: Schedule with milestones and team responsibilities.

C. Bill of Materials and Budget

Part	Supplier	Unit Price	Quantity	Total Price
FreeNove Big Hexapod	Amazon	$169.95	2	$339.90
FreeNove Big Hexapod	Marios Savvides	$0.00	1	$0.00
Raspberry Pi 3 Model B	ECE Inventory	$0.00	3	$0.00
Jetson Orin Nano	ECE Inventory	$0.00	3	$0.00
18650 Batteries + Charger	Amazon	$32.99	3	$98.97
64GB Class10 SD Card (2 pack)	Amazon	$14.99	3	$44.97
eYs3D Stereo Camera - EX8036	ECE Inventory	$0	3	$0
			Total	$483.84

IX. Related Work

- RoboBees - autonomous flying robots with bee-like behavior developed at Harvard.
- Swarmanoid - heterogeneous swarm of robots that work together to perform tasks like exploration and mapping.
- SAFFiR - The Shipboard Autonomous Firefighting Robot - developed by US Navy, focusing on creating autonomous robots to assist firefighting on ships.
- Swarm-SLAM - Sparse Decentralized Collaborative SLAM Framework for Multi-Robot Systems developed by MISTLab
- Inuktun - Rescue robots used in 9/11 and Hurricane Rescue Operations with tank-like treads.

X. Summary

In a world that has a multitude of wars and disasters, search and rescue operations will be vital to reuniting families and bringing hope in dark times. The integration of sophisticated autonomous robots in the search and rescue process is key to future of search and rescue since it can help make missions more effective and efficient. Our hexapod swarm provides a fault tolerant and scalable swarm of independent autonomous robots that can be deployed with search and rescue teams in critical areas. The core components of our design include swarm behavior, object detection, local communication, localization and mapping, and scalability. We predict that a central challenge for our design will be to ensure that all our software can run together and allow the hexapod to run efficiently. Another challenge will be to ensure that the hexapod localization is accurate given the lower-cost sensors that we currently are using. Lastly, providing enough power for both our hexapod controlling RPi as well as our Jetson Orin Nano will be challenging since we want to meet our battery life requirement.

References

[1] Rigging Lab Academy. "The 5 Common Challenges Facing Any Search and Rescue Team." Rigging Lab Academy, [https://rigginglabacademy.com/the-5-common-challenges-facing-any-search-and-rescue-team/].
[2] Thunder Said Energy. "Average Home Sizes." Thundersaid Energy, [https://thundersaidenergy.com/downloads/average-home-sizes/].
[3] Eddyfi. "Versatrax Inspection Crawlers." Eddyfi, [https://www.eddyfi.com/en/product/versatrax-inspection-crawlers].
[4] NVIDIA Developer Blog. "Take AI Learning to the Edge with Jetson." NVIDIA, [https://developer.nvidia.com/blog/take-ai-learning-to-the-edge-with-jetson].