

SWARM

Isaac Kim, Jacob Rohozen, and Tyler Weed

Department of Electrical and Computer Engineering,
Carnegie Mellon University

Abstract—A system capable of distributing large scale search and rescue over a large group of Space-Wandering Autonomous Rescue Vehicles (SWARM). Devices will communicate with each other to segment the search space for a more effective search process. Vehicles have their own localization strategies, and will use an initial triangulation process using Ultra-Wideband (UWB) single-sided two-way ranging (SS-TWR) to agree on a unified coordinate system. Upon finding a target, vehicles will converge to deliver their individual cargo.

Index Terms—Autonomous, design, distributed, path planning, robot, sensor, SS-TWR, UWB

I. INTRODUCTION

The extensive implementation of wheeled mobile robots (WMR) has transformed industrial automation and search-and-rescue operations due to their high operational efficiency and substantial payload capacity. A distributed system of WMRs allow for a faster target search in outdoor environments and heavier weight carriage than humans or drones. This project aims to add scalability and rapid deployment to the functionality of WMRs by implementing a decentralized autonomous swarm to conduct a coordinated search and delivery within a large GPS-denied environment—such as deep caves and dense factories—without the need for fixed-state infrastructure.

The primary users of this system are expected to be emergency responders and industrial inspectors who need an instantly deployable robotic system without the logistical burden. By removing the necessity of fixed-state infrastructure, this system significantly reduces the time and resources required to initialize search and delivery missions.

However, without a central ground station, autonomous operations face challenges regarding efficient space allocation and peer-to-peer navigation. Furthermore, single-antenna Ultra-Wideband (UWB) solutions are often paired with anchors and complex networking stacks to handle localization and communication successfully. We intend to eliminate this overhead by introducing a unified architecture of DWM3001CDK transceivers and Raspberry Pi 4s as the sole communication and computing medium on each WMR. This integration removes the need for heavy communication and computation protocols, allowing the WMRs to coordinate themselves without a centralized computing hub.

State of the art technologies in this area rely on complex Simultaneous and Localization Mapping (SLAM) or heavy computer vision sensors for mapping an environment or processing high-resolution image data. Our system focuses on

peer-to-peer ranging, search, and convergence, allowing each agent to be lighter in terms of computational and sensory payload. This allows each agent to deliver more weight and ensures higher expandability. Theoretically, our design can facilitate a deployment of an arbitrary number of vehicles in a massive area due to decentralized communication and minimal channel occupancy, achieving faster target detection and swarm convergence. We will represent the scalability by simulating the system with three WMRs in this project.

II. USE-CASE REQUIREMENTS

To satisfy the requirements of a prototype of infrastructure-free search swarm, our system must meet the following quantitative measurements. These ensure that the system is efficient, reliable, and scalable.

- 1) Area Search: A swarm of three vehicles must be able to successfully partition and search a 1000ft² room in ten minutes with less than 20% search overlap.
- 2) Target Identification: The target needs to be correctly identified in at least 90% of search operations.
- 3) Delivery: Each WMR needs to be able to carry at least 1lb of payload.
- 4) Convergence: At least $\frac{2}{3}$ of all WMRs must converge at the target after it has been found. Successful convergence at target is identified by less than 30cm of each WMR at the target.

The first requirement is for effective area search. A swarm of three vehicles must be able to partition and fully search a 1000ft² area within 10 minutes. Additionally, the overlap of explored regions between vehicles must be less than 20%. This ensures that increasing the number of vehicles meaningfully decreases search time because exploration will not be redundant.

The next requirement addresses target identification. The swarm must correctly locate the target in at least 90% of search operations. This ensures that users can rely on the system to find a target. Additionally, each vehicle must be able to carry at least one pound of supplies. One pound is important because it is roughly the weight of a water bottle.

Once a target has been identified by a vehicle, it must alert the other vehicles by broadcasting the target location. Then at least $\frac{2}{3}$ of the swarm must converge at the target. This is required because a large number of vehicles should be able to deliver important supplies. Convergence of a vehicle is successful when it is within 30cm of the target or within 30cm of a vehicle that has converged. This is necessary because it may not be possible to fit a large number of vehicles within 30cm of the target.

III. ARCHITECTURE

The SWARM system is composed of multiple identical autonomous vehicles that communicate to carry out a search, converge at the target, and deliver supplies. Each vehicle has sensing, computation, communication, and localization tasks to meet this goal.

Hardware Architecture

The base of each vehicle is a two-wheel chassis driven by DC motors controlled with an L298N motor driver. Encoders are attached to the motors to provide movement data. A Raspberry Pi is responsible for computation, including localization, obstacle avoidance, arbitration, and pathfinding. A Raspberry Pi Camera supports obstacle detection and target identification, enabling the vehicle to search the environment. Additionally, an infrared (IR) sensor serves as a proximity sensor to detect nearby obstacles. Level shifters will be needed as the motor driver and IR sensor logic level is 5V and the Raspberry Pi logic level is 3.3V.

For communication and localization, each vehicle has a DWM3001CDK UWB module for sending and receiving messages and for ranging. The UWB development board also includes an inertial measurement unit (IMU) which can be used for localization.

A dual power system will supply the motors with 12V and the Raspberry Pi with 5V.

A full diagram of the hardware architecture is shown in Figure 1.

libraries, so does not need to be changed once the basic capabilities are in place. The SWARM repository will hold all the vehicle behavior and data processing information, and this will be our main working repository.

Each device will have control over its UWB module, its motor controller, and its fleet of sensors: a camera, an IR sensor, and motor encoders.

System Operation

The software guides the vehicles through three states over the course of a run: Initialization, Wandering, and Convergence. Devices will maintain an understanding of their surroundings through an Environment class, which stores a 2D grid of space with data on whether each cell is explored, whether there are obstacles detected in the cell, and whether the cell is within the vehicle's search space.

In Initialization, devices send out messages to discover each other and establish arbitration indices to elect a leader device. Then, the leader device facilitates the coordinate agreement process by moving to three different coordinates, taking distance measurements to all other devices at each position, and triangulating the positions of each device to calculate and communicate to them their current position. Then, every vehicle will move forward, and the triangulation process will be repeated so that the devices know their orientation in space. Devices will then organize themselves in a starting constellation, likely a circle or a line. Then, wandering will start.

In Wandering, the software will generate a search path through the search space and send the vehicle along it. It will continually facilitate communication between any other devices it encounters while wandering and devices will coordinate to agree on separate search spaces to avoid redundant searching. Upon moving into Wandering from Initialization, the leader device will immediately facilitate one such separation, sending half of the devices to one side, and the other half to the other. It will stay the leader of one group and elect a leader of the other. This process will recurse until each device is the leader of a space that contains no other devices. Through this process, the space will be effectively split among all devices.

In Convergence, which is triggered by a device finding the target, the device will broadcast the target position and all others will move to that position. If a device does not converge, the devices that have converged will go out, find it, and lead it to the target. This retrieval process informs what data will need to be collected during the Wandering phase: devices will need to keep track of the last known location of other devices as well as data about the direction other devices were headed in.

Retrieval in the Convergence phase can be thought of as reentering Wandering with a new goal: establishing communication with the target device. This process will take place in and around the area the target device was last seen, and a success in finding the target leads devices to once again attempt to all converge at the convergence location.

A full control flow diagram for the software system can be seen in Figure 2.

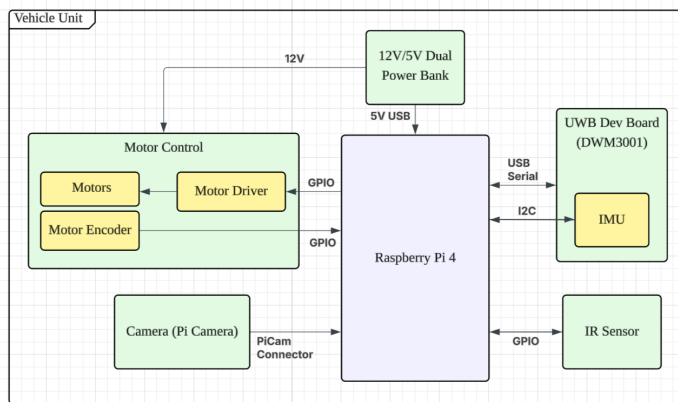


Fig. 1. This block diagram shows the hardware architecture on each WMR.

Software Architecture

We will maintain two repositories for this project: An OS repository and a SWARM project directory. Our Raspberry Pi 4s are each running Ubuntu 25.10. Upon boot, they each pull from the SWARM git repository to get the latest version of the code and run the main Python file.

The operating system will be its own git repository to simplify the flashing of new SD cards. Devices will only ever be pulling from our SWARM repository, and the operating system repository will be used solely for flashing new vehicles with the OS, libraries, and boot scripts ready to go. The OS simply runs the SWARM repository and has the proper

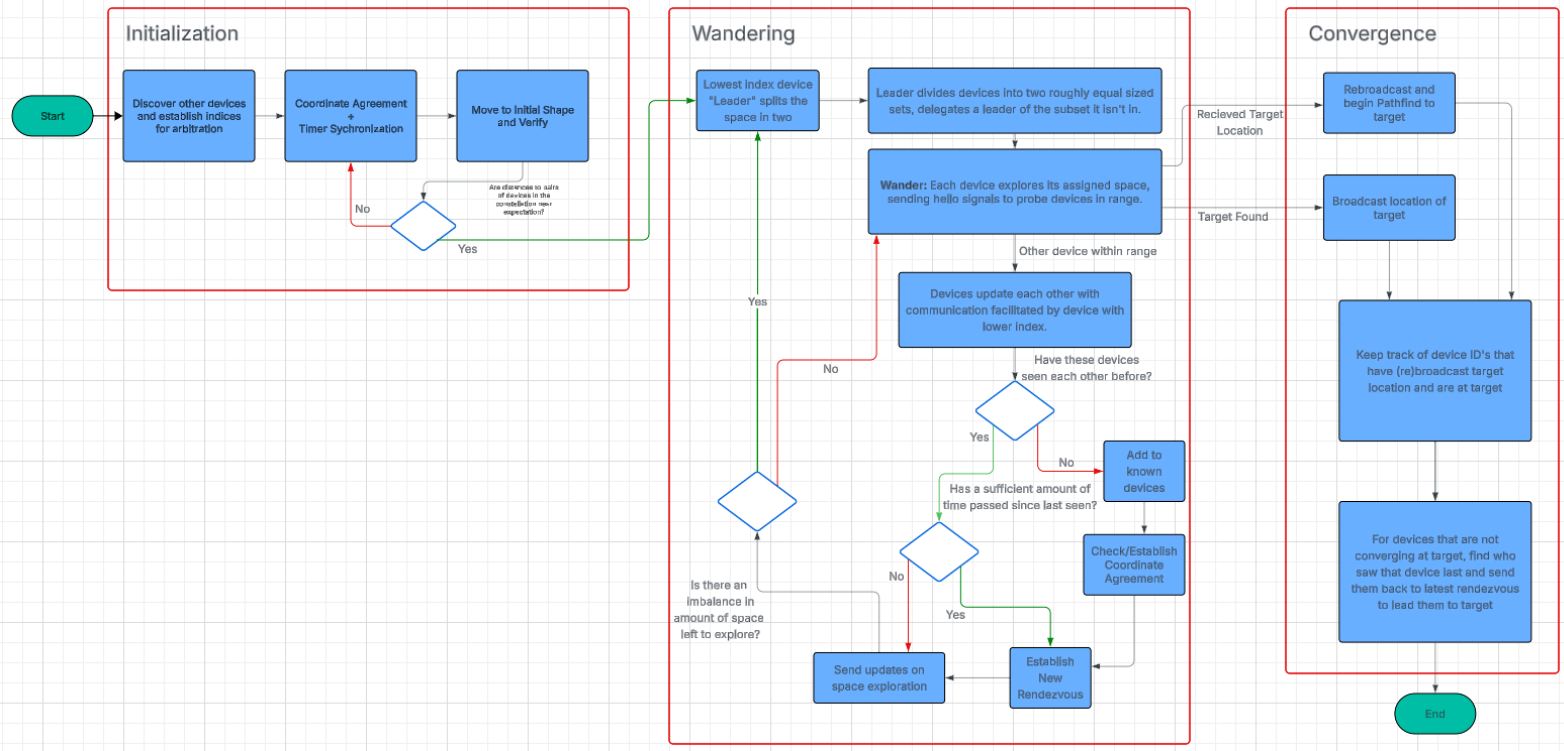


Fig. 2. This diagram shows the software control flow for each WMR.

IV. DESIGN REQUIREMENTS

A. Autonomous

Each vehicle must be completely autonomous. This means that aside from the signal to start a run, there is absolutely no human input until the end of the run. This requires having robust pathfinding algorithms to fully search a space, an automatic arbitration system to allow the vehicles to coordinate among themselves, and effective information exchange between the vehicles so they can reduce the work done by each vehicle. Additionally, the vehicles must be safe. They should stay at least 10cm away from each other to avoid colliding with other vehicles.

B. Distributed

The search must be effectively distributed in a way that avoids redundancies. This again requires a robust communication system so that coordination between vehicles is possible. Through effective communication, vehicles must also make and agree upon decisions about which space they should be exploring. The space partitioning algorithm should ensure that all overlap between any vehicles is less than 20% of the total search area. This enables the swarm to find targets more efficiently.

To test that distribution is effective, we can test whether adding vehicles to the swarm reduces the search time for a given area. Ideally, the time taken to find the target would scale with a $1/n$ relationship, such that upon running many tests with varying numbers of vehicles, attempting to fit the

data to an a/n regression model would find a strong fit for the data.

C. Search Speed

To meet the requirement of searching a 1000ft² space in under 10 minutes with three vehicles, the vehicles must be fast. A minimum speed of one foot per second is required.

Efficient pathfinding and vehicle motion will also be necessary to meet this time requirement. It is anticipated that faster speeds will lead to less accurate motion, which we must keep under control.

D. Localization

Accurate localization is necessary to ensure that local vehicle positioning is correct and agrees with the swarm coordinate frame and for converging when a target is found.

After coordinate agreement, it is up to each vehicle to ensure that they maintain that agreement while moving around the space. First and foremost, this requires an accurate coordinate agreement process. Additionally, it requires effective utilization of the motor encoder and IMU data, as well as integrating some IR data and Camera data to make sure the relative motion of the environment is as expected for a given movement of the vehicle.

Having accurate localization is necessary for our requirements of effective space partitioning, communicating convergence location, and accurate constellation positioning. Without localization, coordination is a much harder problem.

The effectiveness of this localization system will be quantified by making sure the estimated initial vehicle positions are within 15cm of the actual positions for each vehicle. Upon convergence and constellation formation, vehicle distances to each other will be measured. Vehicles

should converge to a location within 30cm of the target and should be within 30cm of the target location in a constellation.

V. DESIGN TRADE STUDIES

A. Localization Methods

1) While UWBs provide distance data, UWB SS-TWR does not provide directions. With only distances, it is difficult to maintain a coordinate system throughout the system's operation. In order to incorporate the orientation movement of each WMR, we are integrating IMUs and Motor Encoders.

IMUs provide acceleration of the WMRs and its rotational orientation. Acceleration data is integrated with the output of the motor encoders for robust movement data. IR sensors and cameras will be used to measure the distance between close WMRs.

2) In order to eliminate the necessity of pre-installed infrastructure, we will not be using UWB anchors for localization. By the triangulation process described in the Software Architecture section, vehicles begin a run by agreeing upon a global coordinate system. Then we are able to maintain global coordinate agreement with per-vehicle localization. This one-time triangulation strategy applies the principles of anchoring without needing the anchors themselves, allowing our system to be quickly deployed in new, anchor-free environments.

TABLE I. UWB METHODS TRADEOFFS

Localization	Pre-deployment	Distance	Direction
Anchors	Necessary	Available	Available
Trirangulation	Not needed	Available	Available
Marco Polo	Not needed	Available	N/A
Hotter-Colder	Not needed	Available	N/A

Table 3. This table shows the tradeoffs between our choice of UWB localization methods.

B. Vehicle Type Tradeoffs

One of the main considerations for our project was what vehicles we would apply our system to. We initially planned to use drones. While drones have higher degrees of freedom in movements, WMRs are more cost effective and have higher payload capacity. As we are focusing on target detection and convergence, WMRs are more suitable for our project.

Since our software strategy generalizes to most conceivable vehicle types, we decided on WMR to keep our deployment simple, cost effective, and capable of carrying more weight.

C. Communication Methods

Due to the distributed nature of our vehicles, we need effective communication between them. This means that we need to maximize the amount of data that can be sent without clogging the communication space. We knew that we would need UWB transceivers for distance measurements, but it took analyzing other communication methods to see whether we should depend on it for overall communication.

The tradeoffs of communication methods between UWB, Bluetooth, and Wi-Fi communication are in Table 2.

TABLE II. COMMUNICATION METHODS TRADEOFFS

Communication Method	Range	Accuracy (Error)	Bandwidth
UWB	~10m	< 10cm	6.8Mbps
Bluetooth	~10m	< 2-5m	2Mbps
Wi-Fi	100+m	< 5-20m	100Mbps

Table 2. This table shows the tradeoffs between our choice of communication methods.

As shown in Table 2, UWB and Bluetooth have similar ranges, but UWB has significantly higher bandwidth. Wi-Fi is long range and fast, but has significantly higher specs than needed for our project. UWB is also more energy efficient than the others, leading to longer battery life. So, UWB works well as a primary communication method.

One other benefit to UWB is that if vehicles have sufficiently synchronized clocks, including the time a message is sent in the message body allows the receiver to calculate the distance to the sender without a second message. This helps reduce the number of messages sent during each exchange, which allows for faster communication between all vehicles as there is less interference. We will want to keep track of relative distances to other vehicles consistently during a messaging process, so the capability for messages to serve the dual purpose of having both communication and distance is a major positive to the use of UWB.

D. Operating Systems

A fundamental consideration for our project was what operating system to run it on. Because we knew we would be using RPi's, a logical consideration was the Raspberry Pi OS, which provides higher hardware optimization and GPIO support, as well as a lightweight experience in incorporating software systems.

However, RPi's can easily be flashed with Ubuntu, which provides superior performance in customizability, updates, library compatibility, as well as robustness. Setting up our RPi's as Ubuntu Servers would have less system overhead than RPi OS. As our software architecture does not need the support of RPi OS, we decided to use Ubuntu as it provides benefits that are more suitable for our project, and we all have more experience with Ubuntu than with RPi OS.

VI. SYSTEM IMPLEMENTATION

A. Communication Subsystem

Communication will be done through DWM3001CDDK UWB transceivers. The internal IMUs on the UWBs will communicate inertial information directly to the UWB. UWBs will be connected to each Raspberry Pi 4 via USB serial interface to transmit and receive data with one another. However, this introduces the problem of multiple UWBs

attempting to broadcast their messages at the same time. To prevent broadcast overlap, we are utilizing Time-Division Multiple Access (TDMA). By allocating a timeframe for one UWB to facilitate communication, the other UWBs will not attempt to broadcast their own messages and interfere with each other.

TDMA will be used in the initial calibration of local coordinates and in space allocation for the three WMRs. The leader WMR is elected by having the lowest index and will move to three distinct points in the initialization phase. At each point, UWB index1 will facilitate the communication by sending out messages and requiring responses from UWB index2 and UWB index3. UWB index2 and UWB index3 will respond within their allocated TDMA timeslots to avoid reception collision. By receiving uninterrupted signals, UWB index1 can calculate its distance to UWB index2 and UWB index3.

Once the initialization phase is complete, UWB index1 will initiate space allocation. By dividing the search area and taking a partitioned space, UWB index1 messages UWB index2 and UWB index3 that the partitioned space is taken. Then UWB index2 facilitates the communication to allocate the rest of the space to itself and UWB index3. Once the partition of search spaces is complete then the system internally recognizes it and transitions to the wandering phase.

B. Movement Subsystem

1) Routing

Routing of the vehicles are decided after the search space has been partitioned and allocated to each WMR in the initialization phase. For a WMR to recognize the space it has to search, our software will have a class of a 2D grid of explored and unexplored cells. Each WMR will only have to search unexplored cells.

Each WMR follows the Boustrophedon (Lawnmower) Patterns, which allows for a full exhaustive search of the unexplored areas. Although there are more efficient search patterns that reduce the total amount of movement, our scalable project allows the lower computation of Boustrophedon Patterns to be within the required timeframe to identify a target within the area. The integration of motor encoders will also favor the straight-line of travel in the Boustrophedon patterns to be the dominant path, as it provides higher accuracy to the integration of motor encoder data.

If a WMR encounters another WMR while it is wandering, they communicate via UWB to re-partition the area and reset their wandering routes and avoid 20% area overlap.

Once a WMR identifies a target using the Pi Camera, the finder broadcasts a message that the other WMRs adhere to and transition into the convergence at rendezvous. The identifier WMR broadcasts its location via TDMA protocol, and the other WMRs begin routing to converge.

In the convergence phase, each WMR stops searching and calculates the shortest path to the destination. While they are converging, Pi Cameras and IR sensors detect any potential obstacles for the WMR to route around to avoid collisions. The distance between the WMR and obstacles will be at least 10cm.

If any WMR fails to converge at the target, the gathered WMRs will initiate a retrieval phase, an extension of the convergence phase. The last-seen locations of the lost WMR will be their target location in reverse chronological order. In the retrieval phase, the WMRs will also follow the shortest path to find the lost WMR.

2) Control

Movement is enabled by a two-wheel differential drive with DC motors and an L298N motor driver. The motor driver is connected to the Raspberry Pi with GPIO. Varying each motor's speed is achieved with pulse width modulation (PWM). Encoders measure the motor rotation rate and provide distance data. The motor driver is controlled by the pathfinding algorithm on the Raspberry Pi. This algorithm takes into consideration the distance data from the IR sensor, and will stop the vehicle at least 10cm away from obstacles and safely path find around them.

To ensure the rotation information from the encoders are accurate, vehicles will only travel in straight lines. This means that they will rotate in place to the desired angle before moving forward.

C. Localization Subsystem

Localization integrates motor encoders, UWB, IMUs, cameras, and IR sensors. In the initialization phase, the leader WMR will move around the other WMRs for initial distance measurements. Then the other WMRs will move in two directions while the leader WMR performs another calibration to calculate the direction of the distances. The directions will be measured by internal IMUs of the UWBs and displacement will be measured by the motor encoders.

Cameras and IR sensors will be used to provide proximity data to assist UWB distance measurement when the WMRs are close together. Equations for calculating the distance between UWBs by reading the signal time are below.

$$ToF = \frac{(T_{RR} - T_{SP}) - (T_{SR} - T_{RP})}{2} \quad (1)$$

$$T_{RR} = \text{Time of Receiving Response (Initiator)} \quad (2)$$

$$T_{SR} = \text{Time of Sending Response (Responder)} \quad (3)$$

$$T_{RP} = \text{Time of Receiving Poll (Responder)} \quad (4)$$

$$T_{SP} = \text{Time of Sending Poll (Initiator)} \quad (5)$$

$$d = c \times ToF \quad (6)$$

In addition to measuring raw distance, combining three distance measurements taken from three known locations can give a relative position by triangulating the measurements. Because each vehicle has its own localization through its own sensor data, we can use this triangulation to communicate the Leader's coordinate system to all other devices. The Leader can keep track of its own position in its local space. Define

positions p_1, p_2, p_3 as three positions from which the leader device measures a distance to device A. Define these distances as d_1, d_2, d_3 , where d_x is the distance measured to device A from p_x . For each position, knowing that device A is some distance away from that position means we can draw a circle with center p_x of radius d_x and know that device A lies somewhere on this circle. For each set of p_x, d_x define circle C_x to be the circle with center p_x and radius d_x . To estimate the position of device A in the Leader's local space, find the closest point on C_3 to an intersection of C_1 and C_2 . Because these measurements may not be exact, we must also account for the case where C_2 and C_3 do not intersect. In this event, we draw a line between C_1 and C_2 . This line will intersect both circles. Treat the center of the segment of the line that lies between the two intersections to be the "intersection point", and find the closest point on C_3 to this simulated intersection point instead.

Through this process, we can use a Leader device to determine the position of another (Follower) device in the Leader's local space.

D. Camera Subsystem

The camera will mainly be used for target detection, but can also be used for varying levels of obstacle detection.

Target detection integrates the Raspberry Pi Camera and the IR sensor for target identification and proximity detection. Once close enough to a potential target, the vehicle will decide if the target has been successfully found. For MVP, the target will simply be a brightly-colored object. This will be done by setting a threshold distance from some target color and checking whether some threshold number of pixels in the image are within that color difference to the target.

This simplifies the computer vision necessary for target identification, and keeps our project centered around autonomous vehicles and distributed systems instead of implementing CV models.

One stretch goal for the vision subsystem is to use it for obstacle avoidance. We can run images through an edge detection convolution and, using the direction to those pixels and the vehicles localization data, define their position along some ray in space. The intersection of rays should correspond to points on these edges, which we can treat as obstacle points.

VII. TEST, VERIFICATION AND VALIDATION

A. Tests for Speed

These are tests on both the vehicles speed themselves and on the speed of finding a target in a given space. Vehicle speed should be at least one foot per second, and vehicles should be able to search for and find a target in a 300sq ft space within 10 minutes.

B. Test for Battery Life

This is a test that the vehicles can run with full movement and communication for over 30 minutes. We will test this by running the motors and the communication subsystem from full battery to when the battery runs out and ensuring that it lasts longer than 30 minutes.

C. Test for Autonomy

This test will ensure that the autonomous routing and control system works as expected. Upon placing one vehicle and defining a search radius, ensure the vehicle can perform an exhaustive search on that whole area. Using tape on the floor, we can ensure that the vehicle moves along the entire perimeter of the space and does not leave its search area.

D. Tests for Convergence

A test that upon one vehicle finding the target, successfully starts the convergence process, going to find other vehicles if none immediately respond. This will test that at least 2/3 of all running vehicles end up converging at the target.

E. Tests for Communication

This is a test that ensures correctness of the messaging system. We will perform many tests of the different types of communication: A general test with a set of messages to ensure that ACKs are properly sent and all messages are exchanged; a test of the communication that happens at the start of a run, and a test of the exchange of messages between vehicles interacting mid-run. We will put the vehicles in each of these scenarios and monitor messages sent and received to make sure they are as expected.

F. Test for Collision Handling

This is a test that the system can handle an environment with obstacles, either by avoiding them or by routing around them after it hits them. This test will be conducted by ensuring that the search and convergence processes still succeed as in the previous tests despite the presence of obstacles.

G. Test in a Maze Environment

This is an extension of the collision handling test in which we create a large maze out of cardboard boxes and test the vehicles in this environment.

H. Test of Synchronized Movement

This is a test in seeing how accurately the vehicles can organize themselves into constellations for a "synchronized swimming" effect of vehicles breaking off from the organized group to explore their own spaces. Once this system is enabled, test that vehicles line up when beginning exploring and that the space distribution process gives the next space to the vehicle at the end of the line.

VIII. PROJECT MANAGEMENT

A. Schedule

We will begin the project with RPi setup and vehicle construction. Then we will work on UWB distance measurement and communications while working on the system to control vehicle movement. Then we will work on

B. Team Member Responsibilities

B. Team Member Responsibilities

Tyler is responsible for the basic setup of the RPI's as well as the space parameterization and separation algorithms.

Isaac is responsible for the construction of the vehicles and the pathfinding algorithms.

Jacob is responsible for sensor setup and data collection.

Tyler and Jacob will work together on communication and UWB distance algorithms.

Tyler and Isaac will be responsible for generating and moving to next positions.

Jacob and Isaac will be responsible for localization and ensuring accuracy of vehicle movement.

C. Bill of Materials and Budget

The materials needed for each vehicle are shown in the bill of materials in Table4. The total cost for each vehicle is \$168.46.

This means our plan to initially use 3 vehicles costs \$505.38.

Because the RPi 4's are part of the ECE inventory and not included in our budget of \$600, scaling to 6 cars is possible within the budget, as removing the price of the RPi's puts the price per vehicle under \$100.

D. TechSpark Use Plan

We do not plan on using TechSpark for our project.

E. Risk Mitigation Plans

The power supply we intend to use is dual output so it can be used at 12V for the motor driver and 5V for the Raspberry Pi. We aren't sure that the dual output will work as intended. Our backup plan is to use 18650's and a buck converter for more controlled power management.

We are anticipating some risks with using the motor encoders for localization. This will be mitigated by resorting to a combination of IMU, IR sensor, Camera, Motor Encoder, and Ultrasonic data until results are accurate enough.

Similar to localization, we also are unsure how accurate the UWB coordinate agreement will be. If coordinate systems are not accurate enough, we will either augment the UWB with a directional antenna or we could resort to using solely distance maximization/minimization techniques with no requirement for accurate positioning.



Fig 1. Schedule example with milestones and team responsibilities

IX. RELATED WORK

This project is similar to previous ECE capstone projects that focus on the use of autonomous robots for search and rescue.

An example of one such ECE team is Team A0 of Fall 2025. They had a similar autonomous search and rescue wheeled robot, but their method was not distributed as ours is.

We have also seen a search and rescue drone from team A6 in the Fall of 2024. Once again, this was not a distributed solution, with only one drone doing the searching.

There is a lot of work on anchored UWB positioning as well as on localization techniques with IMU and Motor Encoders, but rarely do they combine techniques to treat one localized vehicle as an anchor, taking multiple measurements from different locations to determine positions in a local space.

X. SUMMARY

Our design allows rovers to autonomously search for a target and converge once the target is detected. All rovers are decentralized to allow for this project to be scalable to the requirements of stakeholders—such as emergency responders and industrial inspectors. Each rover acts as a computing hub and does not require any anchors for localization, which eliminates the need to pre-install any infrastructure. Rapid deployment and scalability will allow stakeholders to utilize this project for any size of environments that require search and delivery.

Some of the challenges we face are power management of our batteries and localization accuracy. We have backup plans of buck converters and the Hotter-Colder method to enable the functionality of our system if any of the subsystems turn out to operate inaccurately.

GLOSSARY OF ACRONYMS

UWB– Ultra-Wideband
 RPi – Raspberry Pi
 IMU – Inertial Measurement Unit
 IR – Infrared
 ACK – Acknowledgement
 SS-TWR – Single-Sided Two-Way Ranging
 WMR – Wheeled Mobile Robot

REFERENCES

- [1] “Fall 2025 ECE 18-500 Capstone Design: Team A0,” Carnegie Mellon University. <https://course.ece.cmu.edu/~ece500/projects/f25-teama0/>
- [2] Qorvo, “DWM3001CDK Product Brief: Development kit for DWM3001C module,” Rev. B, May 2022. [Online]. Available: www.qorvo.com
- [3] Handson Technology, “Handson Technology User Guide L298N Dual H-Bridge Motor Driver.” Available: <https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf>
- [4] [Multi-Sensor Fusion Framework for Reliable Localization and Trajectory Tracking of Mobile Robot by Integrating UWB, Odometry and AHRS - PMC](#)
- [5] H. Choset and P. Pignon, “Coverage path planning: The boustrophedon cellular decomposition,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, Albuquerque, NM, 1997, pp. 1203-1209.

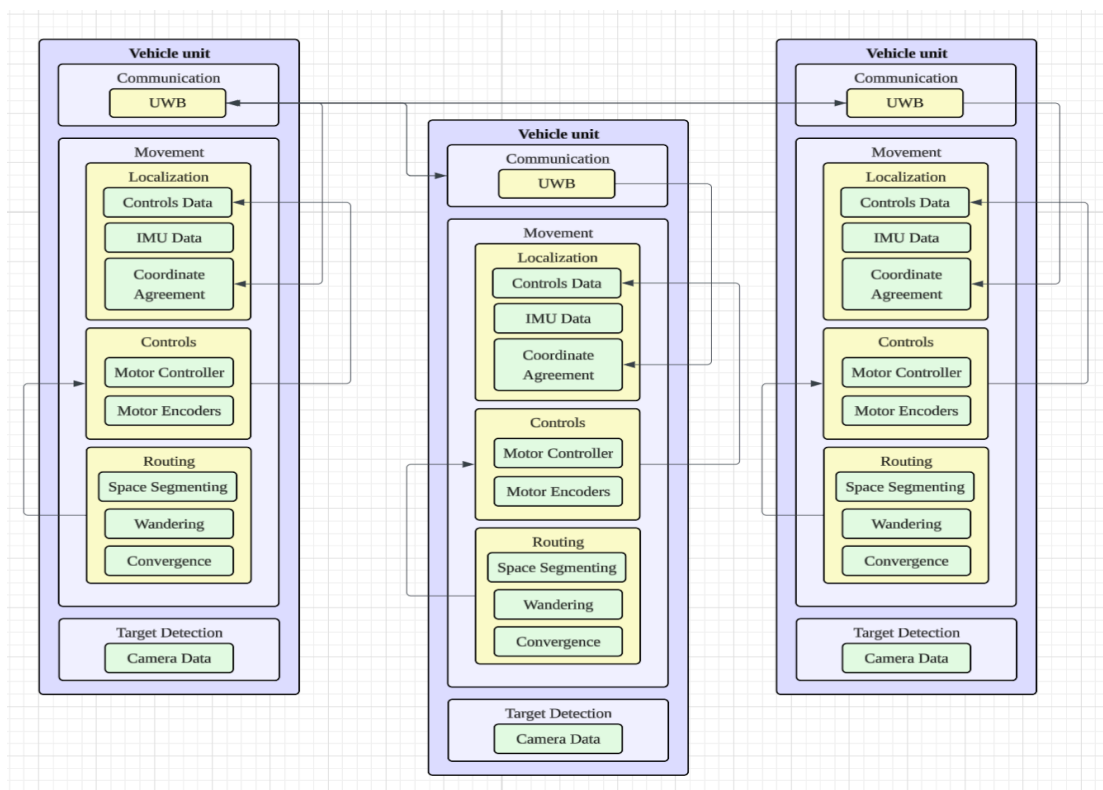


Fig. 3. This block diagram shows the system architecture of 3 WMRs.

BILL OF MATERIALS

Description	Manufacturer	Quantity	Cost @
Cart kit	DIYables	1	\$20.99
Pi Camera	HiLetgo	1	\$7.99
Raspberry Pi 4	Raspberry Pi	1	\$75.00
UWB(DWM3001CDK)	Qorvo	1	\$29.50
Motor Encoders	EC Buying	2	\$1.99
Battery	Talentcell	1	\$32.99
Full WMR	N/A	1	\$168.46

BILL OF MATERIAL TABLE 4. THIS TABLE SHOWS THE BILL OF MATERIALS FOR ONE VEHICLE.