ANEBIT
Autonomous Unmanned Ground Vehicle

Version 1.0
26/05/2024

Design Studio Section:
[1]

Presented by:
Ata Oğuz Tanrıkulu
Doruk Yazıcı
Ebrar Çakmak
Erkin Atay Toka
Yunus Emre Tüysüz

Table of Contents

Executive Summary

This report will present the final designs by ANEBIT for a smart robot vehicle that can operate in different environments. This isn't just any robot; it's what we call an Autonomous Unmanned Ground Vehicle (UGV), and it's designed to understand its surroundings and move to where it needs to go all on its own. The magic behind this robot is a tool called a 2D LiDAR sensor. This sensor scans the area and helps the robot make a 3D map of the space it's in. Then by labelling this map, one can choose where to send this robot. The robot can then travel to specific spots without crashing into obstacles or getting lost.

Our mission with this robot is to make workspaces like warehouses, factories, and even hospitals safer and more efficient. Imagine sending the robot into areas that might be risky for people, or having it do jobs that are very repetitive jobs that are important but can be tiring or boring for humans. This robot could also be a big help in places where there isn't enough workers. And it's not just about work; our robot is also friendly to the environment because it can be powered by electricity.

We've put together a team of talented individuals, each with their own special skills, to bring this robot to life. They're working on everything the robot needs to be successful: creating maps, navigating through spaces, connecting with computers and phones, and making sure it's built to last and operates safely. As aimed, the project is completed by the end of May 2024. By then, we hope to show off a working robot that can really make a difference in how work is done, making it safer and more efficient for everyone involved.

In this report, you'll find all the details about who's on our team, what each person has been responsible, and how did all parts of the project come together. We'll talk about the challenges we have faced abd expect to face and how we overcame them. Plus, we'll share our hopes for what the robot is able to do in it finalised version.

In our journey to create this autonomous robot, we have set clear goals and have outlined what we plan to deliver by the end of the project. Our main objectives are to ensure that the robot can accurately map its surroundings, move to designated areas without help, and recognize and avoid obstacles. To achieve these goals, we have focused on several key deliverables: a fully functional 3D mapping system, a reliable navigation program, a user-friendly interface for interaction, and a sturdy and practical design for the robot itself. We also made sure that our robot can map out a room quickly—in about 10 to 15 minutes—and keep working for at least 25 to 30 minutes on a single charge. We also kept an eye on the costs because we want this technology to be affordable.

We're committed to delivering a UGV that is not only smart and self-sufficient but also user-friendly. This means that anyone using it, whether they're a tech expert or not, will find it simple to operate. By the project's completion, a comprehensive guide that explains how the robot works, training materials for users, and a support system to help with any questions or issues are presented. We also had a series of tests and demonstrations to show that the robot can do what we promise – these are all part of the important steps we're taking to make sure our robot is ready for action.

We believe our UGV will not just meet the needs of today's industries but also inspire new ideas and improvements for future technology. It's not just about building a robot; it's about creating a future where robots and humans work better together.

Introduction

Background of the project

In recent years, autonomous systems such as self-driving cars, unmanned ground vehicles (UGVs), cleaning robots, and drones have become increasingly prevalent, reshaping various aspects of our daily lives. These systems rely on advanced technologies to perceive their surroundings, make decisions, and carry out tasks without human intervention. The integration of these autonomous systems has opened new possibilities and challenges, necessitating a deeper understanding of their capabilities and limitations. As company ANEBIT, we decided to design a basic UGV system with perception, planning, and control capabilities, so that we can generate solutions to the possible challenges and limitations that we encountered.

Problem statement

The primary objective of this project is to design and develop an autonomous Unmanned Ground Vehicle (UGV) capable of operating effectively in indoor and outdoor environments while establishing the communication with its user in a wireless way, and not requiring any help in physical form. The UGV must integrate advanced technologies for perception, planning, and control, focusing on the use of a single 2D Light Detection and Ranging (LiDAR) sensor. The core problems in this project are stated below.

- **3D Mapping with 2D LiDAR**: The UGV must generate accurate 3D maps of indoor environments containing static objects, solely using a 2D LiDAR sensor. This mapping process is critical for enabling the UGV to understand and navigate in the environment. The ability to perform this task without manual map corrections represents a significant technical challenge, which requires integration and flawless operation of different submodules. According to the mapping system that is decided, different mapping algorithms should be implemented as the data should be manipulated in different ways to generate a 3D map. Also, different mechanical designs should be created to move the LiDAR in the way that is in conformance with the mapping algorithm used.

- **User Interface and Communication:** Throughout the operation, -while the region is labelled manually, the vehicle will be controlled, and the generated map will be visualized from a graphical user interface- there should be wireless and instantaneous communication between the vehicle and user via the graphical user interface. The communication between the PC of the user and the main processor unit of the vehicle should be established.

- **Obstacle Detection and Avoidance and Path Planning:** The UGV must identify and autonomously avoid positive (physical) obstacles. This includes the capability to detect and navigate around dynamic obstacles, adapting its path in real-time toward the destination point defined by the user.

- **Mechanical Design:** Mechanical parts to enable the motion of the vehicle and chassis to accommodate necessary sensors, processor and power supplies must be designed, and tested in various conditions to prevent a failure in the state of operation. The electrical system should be powering the vehicle in a stable way throughout the operation, the gravimetric analysis should be performed to prevent roll-overs, tip-overs and pops that result in the failure of the mobility system.

Final Status

The project is completed by the end of May 2024. The information about the susbsystems and subunits are given throughout the report.

Scope and Organization

This report focuses on the finalized requirements, objectives, and design decisions made throughout the project, the main solution methods that are proposed for the submodules which are being worked on and implemented. Then, test results using the current solution methods are given and concluded. Lastly, risk analysis and cost analysis are done.

Company Organization

Ata Oğuz Tanrıkulu is an undergraduate engineering student specializing in communication and coding theory. He is also a double major student at the Computer Engineering department. He has some experience in programming (C/C++, Python, MATLAB etc.) because of his other major. He will be responsible for the wireless communication, user interface design and programming aspects of the project during this process.

Doruk Yazıcı is an undergraduate student specializing in power electronics and control systems. During his summer internships, he gained experience on design and implementation of power electronics and their control systems as well as control algorithms of larger scale projects. He has an interest in various areas of robot design so that throughout the project, he will be mainly responsible for the mapping, control system, and integration of the subsystems.

Erkin Atay Toka is an undergraduate student specializing in power electronics, electromechanical energy conversion and control theory. He has experience in motor control, design and power systems and electronical analysis. Also, he is willing to learn about path planning algorithms. He will be in charge of mechanical and power design of the vehicle and also, he will be working on path planning and motor drive in this project.

Yunus Emre Tüysüz is an undergraduate researcher in the Heart Research Lab. He works on imaging. During summer internships, he gained practical experience in communication protocols using C# and GUI design using WPF, and he focused on parameter estimation using Python. For the upcoming project, Yunus will use his skills in imaging, GUI design and coding. His previous experience and knowledge in these areas positions him well to contribute effectively to the success of the project.

Ebrar Çakmak is an undergraduate student specializing in solid state devices technologies, photonics, and CMOS circuit design. She is mostly interested in quantum mechanics, memristor devices and optics. Thanks to her years participating in robotics and different technical clubs at university, she gained experience on mechanical design, electronics-sensor subsystem design, and integration of subsystems in several projects. During this project, she will be responsible for electronic and mechanical design, as well as the integration of subsystems.

The overall organizational structure of the company can be seen in Figure 1. It should be noted that this organizational structure shows the main working areas of the members. For each subsystem, work distributions may be done, and other members will help the main member working in that subsystem.

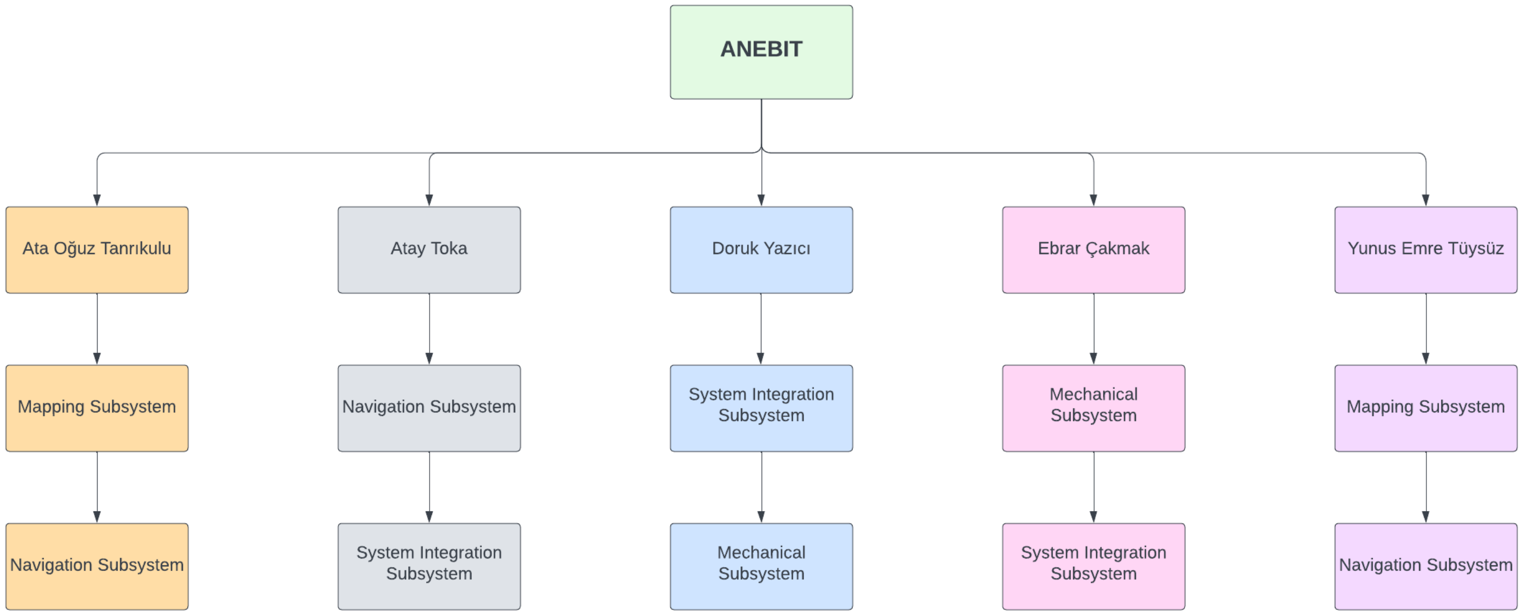


Figure 1. Organizational structure of the company

System Design

The overall system mainly consists of four crucial subsystems which are "Mapping", "Navigation", "Mechanical", and "System Integration". These modules each have different submodules which secures optimum performance. Each sub-system and their sub-units can be seen in Figure 1.

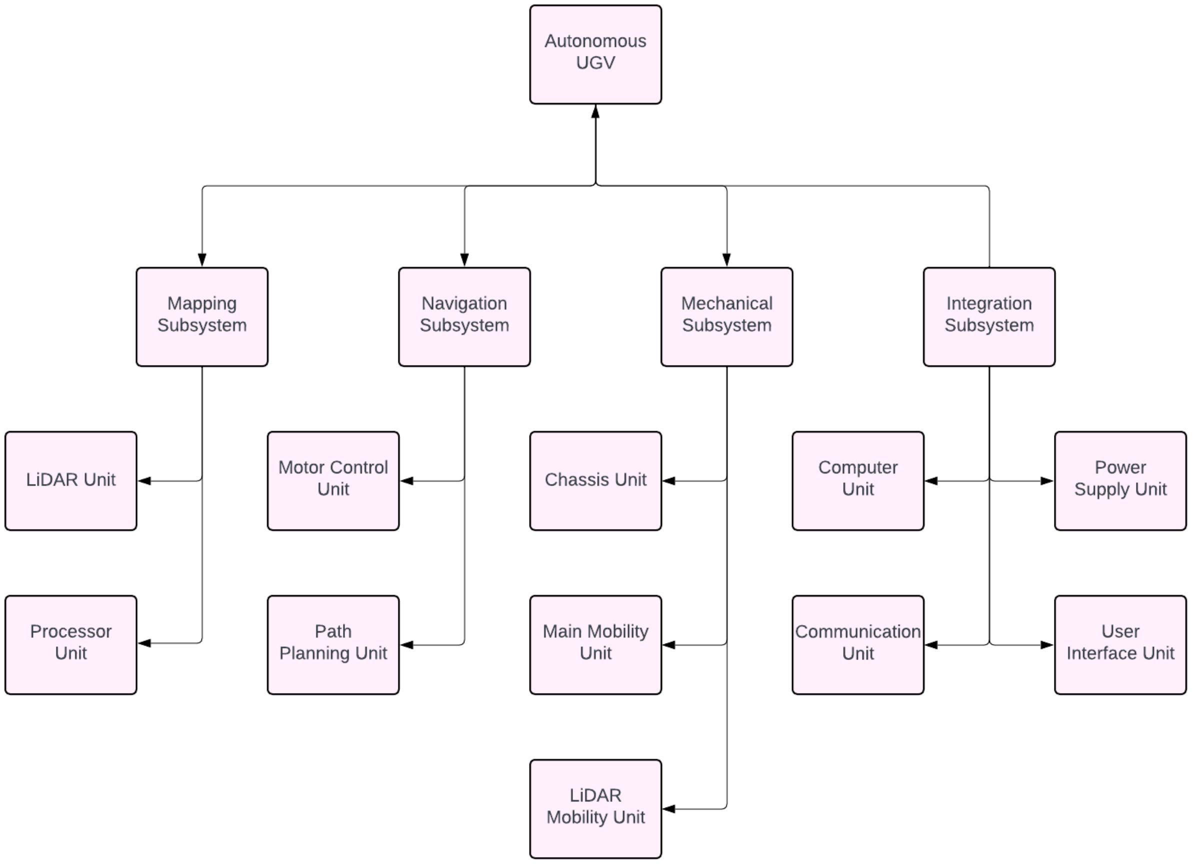


Figure 2. Subsystems and Subunits

Mapping Subsystem

The Mapping Subsystem of the autonomous unmanned ground vehicle (UGV) is engineered to generate both a 3D map of the surroundings and a 2D map at the LiDAR level.

It comprises two primary components: LiDAR unit and a processor module.

The Mapping module operates in symbiosis with the navigation subsystem, providing essential maps for the navigation module's motor controller and localization algorithms, while also utilizing the navigation module to traverse the environment and collect data from various points within it.

1. LiDAR Unit

The primary unit of the mapping system is a LiDAR-based system designed for 3D mapping and 2D localization. At its core is the LiDAR component, essential for producing accurate geographical data.

LiDAR module consists of a 2D LiDAR, its control, and data transfer unit. The LiDAR used is an RPLiDAR A1-M8, which is a 360° LiDAR that can obtain 6000 data points in a second with a rotation frequency of 5Hz. It can measure up to 6 meters in one direction so that a circular area with a diameter of 12 meters can be scanned. It sends hundreds of light beams in a second, and by comparing the time that it takes between sending and receiving these beams, it calculates if there is an object at the angle it looks, and if there is one, it calculates the distance of the object and returns a two-dimensional data. The picture of the RP LiDAR can be seen in Figure 2.

Figure 3. RPLiDAR A1-M8

As it is shipped with a DC motor mounted, the LiDAR controls its own rotation. This also helps the robot to control its surroundings more efficiently when operating and navigating, because it does not need to rotate the whole vehicle to scan its surroundings.

- The control unit of the LiDAR is responsible for the rotation of the DC motor and the LiDAR itself.
- The data transfer unit carries the data that the LiDAR collects to the processing module via a Micro USB cable.

In summary, the operational principle of LiDAR involves the emission of light beams to detect objects and ascertain distances, generating a two-dimensional dataset.

The system's comprehensive scanning ability must be enhanced with the Mechanical Module to ensure mobility of the LiDAR to be able to obtain a 3D map. The submodule is named "LiDAR Mobility System", which adds a critical feature by facilitating LiDAR rotation around the x-axis, essential for capturing detailed 3D data. In other words, mobility system adds an additional dimension to its scanning capability.

Basic idea is rotating the LiDAR 3° at each step and obtaining 6000 data points, for a total of 60 times so that the LiDAR turns 180° in total. The rotation angle and the data obtained from the LiDAR are combined in a specified version of Euler's Transformation that generates the 3D map of the environment scanned. A more detailed design of the mobility unit will be given in the Mechanical Module. The rotation of the LiDAR can be seen in Figure 3.

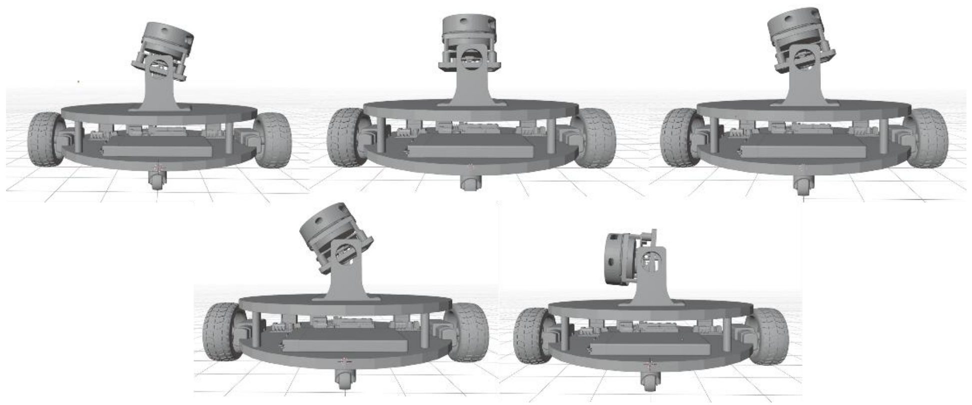


Figure 4. Rotation Examples of the LiDAR Mobility System

Data is collected from multiple points through iterative processes, guided by environmental settings. The user must provide movement instructions for the UGV at each iteration to specify the motor speed. Utilizing the motor control unit of the Navigation Sub-system, the UGV manually navigates to the specified location without a predefined path. Specifics of the manual navigation will be described in more detailed way later on at the Navigation Sub-system part. Upon reaching the target location, it begins gathering data as described earlier, storing this information for further processing.

While the robot is in motion between points, the LiDAR remains parallel to the floor. The UGV employs the 2D SLAM algorithm from the Hector-SLAM library of ROS to determine its new location and orientation relative to its previous position. These updated values are stored at each iteration, allowing the UGV's location and orientation to be compared to its initial position.

During the phase of creating the 3D map from the obtained data, additional transformations are applied to the collected data points based on these location and orientation values. This ensures that all data points are aligned within the same coordinate system for accurate mapping.

The system continues iterating until instructed by the user to halt. Subsequently, it utilizes the accumulated data to generate the desired maps.

2. Processor Unit

The control of this system, and further data processing and communication with the computer are handled by a Single Board Computer (SBC), specifically a Raspberry Pi 4 4GB. It organizes the operations and also processes data from the LiDAR, which is subsequently transmitted to a computer via Wi-Fi. The data is processed on the Raspberry Pi 4 with Python, using "NumPy", and "Matplotlib" libraries of Python, and the algorithm that is found. Also, "RPLiDAR" library is additionally used for gathering data from the LiDAR. The map obtained can be directly viewed on the computer of the user. To better visualization of the 3D map, user can transmit the map files to its own computer and plot the 3D map on Matlab environment.

The processing unit, equipped with the Raspberry Pi 4, conducts simultaneous mapping and navigation (SLAM). It employs the Hector SLAM algorithm, which processes data from the LiDAR. This process is vital for accurately determining the UGV's spatial orientation and angles of view.

Hector-SLAM operates by creating a grid-based map where each cell represents the likelihood of occupancy. As the LiDAR scans the environment, the algorithm updates this map in real-time, adjusting the robot's estimated position with each new data point. One of the key strengths of Hector-SLAM is its ability to efficiently process the vast amounts of data produced by LiDAR sensors. It does this by using a scan-matching approach that quickly aligns new scan data with the existing map, enabling the UGV to localize itself accurately and with minimal computational overhead. An example of SLAM interface can be found in Figure 4 in which the red line represents the backward direction of the LiDAR.

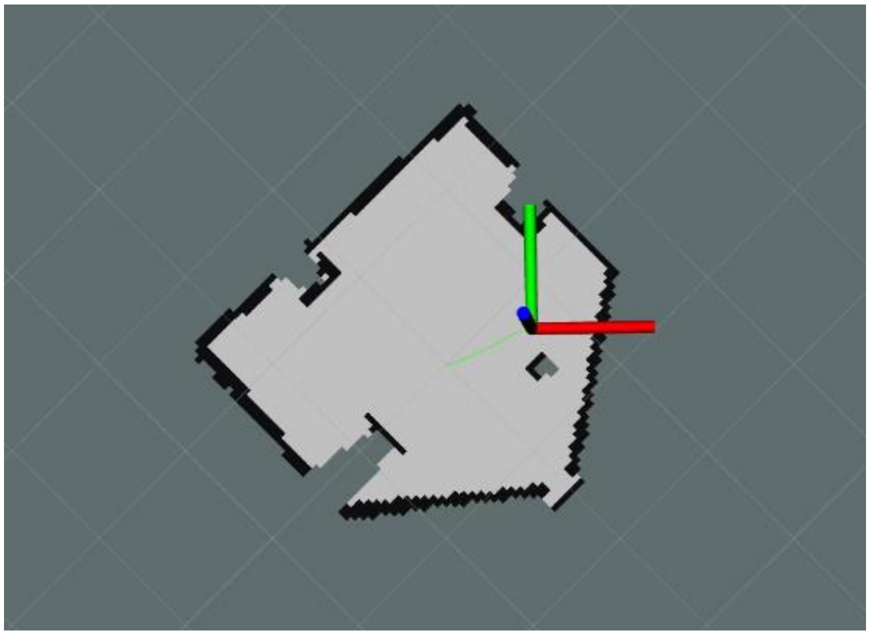


Figure 5. SLAM Interface

Also, the Raspberry Pi 4 has a connection to Arduino UNO to control the rotation of the servo motor and getting the data from the servo motor. By combining the information of the LiDAR, servo motor, Hector-SLAM, and the main mobility system, the detailed point cloud is processed in the Raspberry Pi 4 using Python and the 3D map is generated.

Flowchart of these process is summarized in Figure 5.

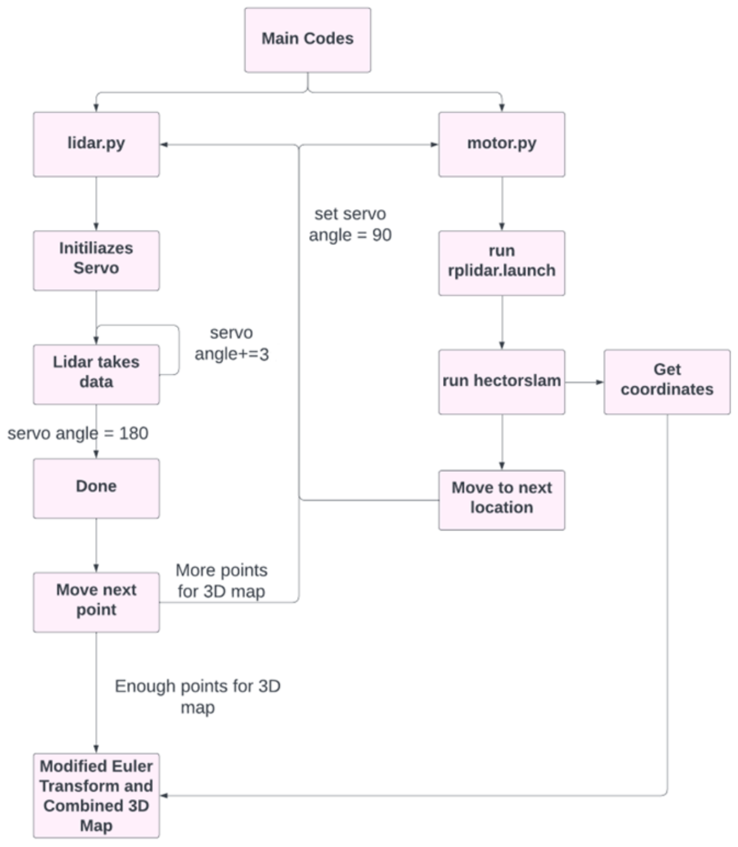


Figure 6. 3D Mapping Flowchart

The UGV employs the Hector-SLAM algorithm to create a 2D map of its surroundings for future autonomous navigation. While utilizing the manual navigation capabilities of the Navigation Sub-system's motor controller unit, the UGV simultaneously initiates the SLAM algorithm. This allows it to generate the 2D map of the environment in real-time as it navigates manually, scanning the entire area using the RPLiDAR. Ultimately, the UGV produces a detailed 2D map of the environment at the LiDAR level. 2D map displayed at Figure 4 could be an example of the product of this process; showing the walls and the obstacles present in the environment.

Navigation Subsystem

The Navigation subsystem is tasked with both user-directed and autonomous navigation of the UGV, while also ensuring static and dynamic obstacle avoidance. As previously mentioned, it relies on the 2D map generated by the mapping subsystem.

This subsystem consists of two primary components: the motor control unit and the path planning unit.

1. Motor Control Unit

Motor control unit is responsible from the autonomous and user directed movement of the UGV. It includes 2 DC motors driven separately using a L298N motor driver and their control is maintened by Raspberry Pi 4 and Arduino Uno. It operates in two different modes for the two different operations.

For user-directed navigation, which is employed between data measurements during map creation as previously described, the Navigation subsystem utilizes the Hector SLAM Library of ROS. The motor controller algorithm and SLAM algorithm are initialized simultaneously, operating in tandem throughout navigation. While the motor controller manages the UGV's movement, the SLAM algorithm assists in determining its pose, comprising both position and orientation on the 2D plane relative to the initial starting location.

The UGV utilizes user input to determine motor speeds, allowing for control via the keyboard. By pressing keys "W," "A," "S," "D," and "X," user can navigate the UGV: "W" moves it forward by increasing both motors' speed on the forward direction, "A" turns left by decreasing the left motor speed and increasing right motor speed, "S" moves backward by increasing both motors' speed on the backward direction, "D" turns right by decreasing the right motor speed and increasing left motor speed, and "X" stops all movement instantly by making both motors' speed 0. Notably, repeated key presses increment motor speed in the corresponding direction. Lastly, user can press "ESC" key to terminate the process. These adjustments on motor speeds are facilitated to PWM signals generated by an Arduino Uno, with oversight from the Raspberry Pi 4. Arduino Uno communicates with the L298N motor driver, and the motors are driven.

This algorithm doesn't provide any obstacle avoidance, user should adjust motor speeds to avoid both dynamic and static obstacles. Flowchart of the process is given in Figure 7.

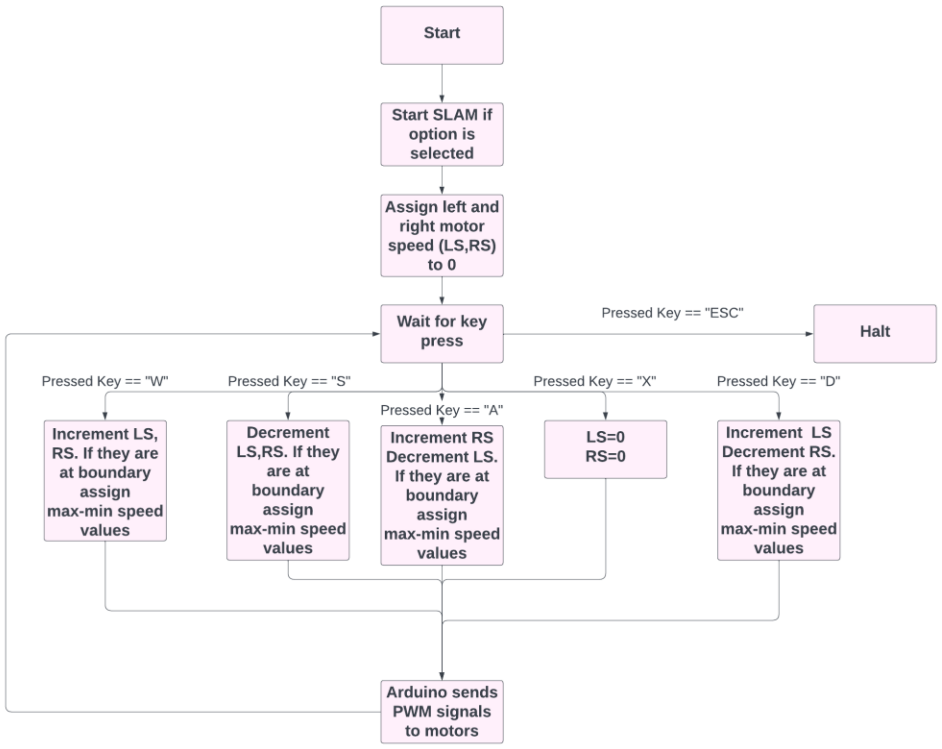

Start
Start SLAM if option is selected
Assign left and right motor speed (LS,RS) to 0
Wait for key press
Pressed Key == "ESC"
Halt
Pressed Key == "W"
Pressed Key == "S"
Pressed Key == "A"
Pressed Key == "X"
Pressed Key == "D"
Increment LS, RS. If they are at boundary assign max-min speed values
Decrement LS,RS. If they are at boundary assign max-min speed values
Increment RS Decrement LS. If they are at boundary assign max-min speed values
LS=0 RS=0
Increment LS Decrement RS. If they are at boundary assign max-min speed values
Arduino sends PWM signals to motors

Figure 7: Manual Navigation Flowchart

For autonomous navigation, a distinct algorithm is employed wherein the UGV follows a path generated by the path planning unit. Unlike the user-directed navigation algorithm, which relies on specified locations, this autonomous navigation algorithm is designed to avoid obstacles in the environment. Additionally, for localization, the AMCL library of ROS is utilized instead of the SLAM algorithm.

The main difference lies in their approach: while SLAM continuously constructs a map while localizing, AMCL relies on an initial 2D map provided to it. Utilizing LiDAR scan data, AMCL estimates the UGV’s location on the given map. The 2D localization map generated in the mapping subsystem is fed into this algorithm to facilitate localization throughout the navigation process.

In the AMCL algorithm, a particle filter is employed to estimate the UGV's pose (position and orientation) based on the sensor measurements, odometry – estimated motion of the UGV - , pose estimation found in previos iterations and the known map. By comparing the sensor readings to the map, the algorithm probabilistically determines the UGV's pose, taking into account uncertainties and inaccuracies in both the sensor data, odometry, previous estimates of the pose and the map. This iterative process refines the UGV's estimated pose over time, enabling accurate localization even in dynamic environments.

Odometry is supplied to AMCL from an outside source, which is another ROS library, Laser Scan Matcher. Laser Scan Matcher uses laser scans coming from the LiDAR and facilitates a process based on an Iterative Closest Point (ICP) algorithm, where it compares iterative laser scans to estimate the movement of the UGV. This information is fed as the odometry data, which AMCL uses for localization.

For the initialization of the AMCL algorithm, it requiers an initial pose estimation for its iterative algorithm to start with. Instead of giving an inteligent estimation for this value, UGV assigns a default value. Later on, user can use manual navigation algorithm described earlier without the use of SLAM to navigate the UGV in the range of 10-20 cm radius for a short period of time, 5-10 seconds. Doing this, AMCL node starts its iterative algorithm and slowly corrects the initial pose estimation, so that UGV localize itself on the environment with the minimum error. After that, UGV is ready for autonomus navigation.

Determining the motor speeds in the navigation process is held by another ROS library called "move_base", which uses the laser scan data coming from the LiDAR, pose estimation data coming from the AMCL and 2D map data generated by mapping subsystem. It gathers a destination target point from the user as a goal. The goal can be given directly from the RViZ graphical user interface, or the graphical user interface generated by ANEBIT. This algorithm is responsible form caluclating the path that UGV should follow to reach the target and also the motor speeds to follow this path. Specifics of the move_base node will be explained in path planning unit, but basically this node publishes the motor velocity commands on some certain frequency as linear and angular speed of the UGV. Autonumus motor controller algorithm reads these velocity commands contiously, transform the linear and angular velocity values to speed values for left and right DC motors, and sends PWM signals for these speed values to both motors using Arduino Uno. Relation between left, right motor speeds and linear-angular motor speeds are given in the below equations.

$$v_{left} = v_{linear} - \frac{v_{angular} * W}{2}$$

$$v_{right} = v_{linear} + \frac{v_{angular} * W}{2}$$

Where W indicates the distance between the two wheels of the UGV in meters. Flowchart of the autonomous navigation is given in Figure 8.

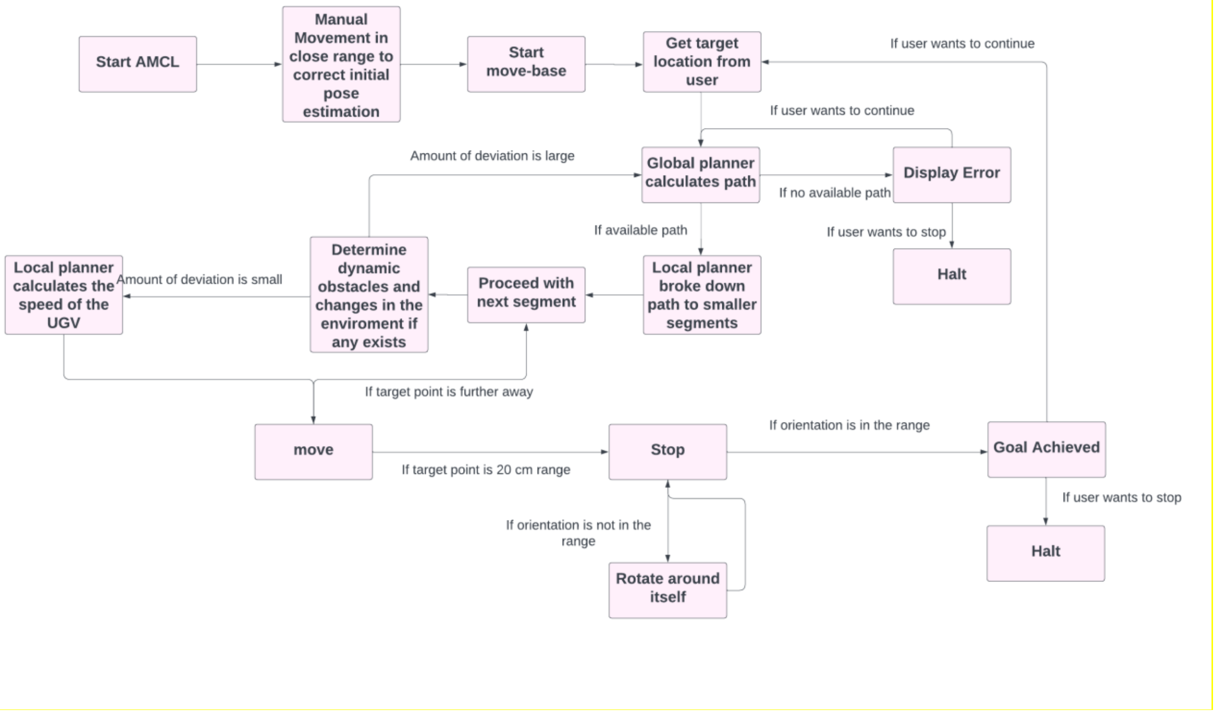

Start AMCL
Manual Movement in close range to correct initial pose estimation
Start move-base
Get target location from user
If user wants to continue
If user wants to continue
Amount of deviation is large
Global planner calculates path
If no available path
Display Error
If available path
If user wants to stop
Halt
Local planner calculates the speed of the UGV
Amount of deviation is small
Determine dynamic obstacles and changes in the enviroment if any exists
Proceed with next segment
Local planner broke down path to smaller segments
If target point is further away
move
If target point is 20 cm range
Stop
If orientation is in the range
Goal Achieved
If user wants to stop
Halt
If orientation is not in the range
Rotate around itself

Figure 8: Autonomous Navigation Flowchart

2. Path Planning Unit

This unit is assigned the task of planning obstacle-free paths for autonomous navigation, handling both dynamic and static obstacles. It utilizes the 2D map of the environment generated by the mapping subsystem to establish a route. Initially, it receives a target location from the user and utilizes AMCL, fueled by the 2D map, to determine its position in the environment.

As previously mentioned, this node is tasked with determining the route to the designated target location and calculating the linear and angular velocities required for the UGV to follow this path. It accomplishes this through two main planners: the local and global planners. Upon receiving the target destination from the user and current location from the AMCL node, the global planner computes the shortest and safest feasible path to this location, taking into account static obstacles within the environment. It ensures that the path devised allows for a safe margin between the UGV and any obstacles, preventing any overlap with the UGV’s boundaries. If a viable path that meets these conditions cannot be found, the move_base node will generate an error message to alert the user that no navigable path is available.

If there is an available path, the local planner takes over for the execution phase, handling real-time navigation by breaking down the global path into smaller, manageable segments that the robot can follow in the short term. It dynamically adjusts the robot's movements and speed in response to unforeseen obstacles or changes in the environment that weren't accounted for by the global planner. To do that it uses sensor data to continuously evaluate the environment and make immediate decisions to maintain a collision-free path. It implements Dynamic Window Approach (DWA) algorithm, which consider the robot's velocity, acceleration limits, and turning capabilities to optimize movement decisions.

During this process, move_base continually monitors the progress of the robot along the path planned by the global planner. If deviations occur due to dynamic obstacles or other changes, the local planner suggests alternate routes or maneuvers to keep the robot on track towards its final destination. The move_base node can trigger a re-planning from the global planner if the local adjustments aren't sufficient to overcome obstacles or if a significantly better path becomes available due to changes in the environment.

The UGV halts its forward motion once it approaches within a 20 cm radius of the target location. Subsequently, it performs a brief rotational maneuver to align itself more precisely with the specified orientation goal. The rotation continues until its orientation falls within a pre-established acceptable range, at which point it comes to a complete stop.

Mechanical Subsystem

The mechanical subsystem of the autonomous unmanned ground vehicle (UGV) is designed to fulfill the functional requirements of stability, durability, mobility, and efficiency in both 2D and 3D mapping operations, as well as navigation purposes. This part outlines the detailed design and considerations for the main mechanical components: the chassis, LiDAR mobility system, and main mobility system. The block diagram of this subsystem can be seen in Figure 9.

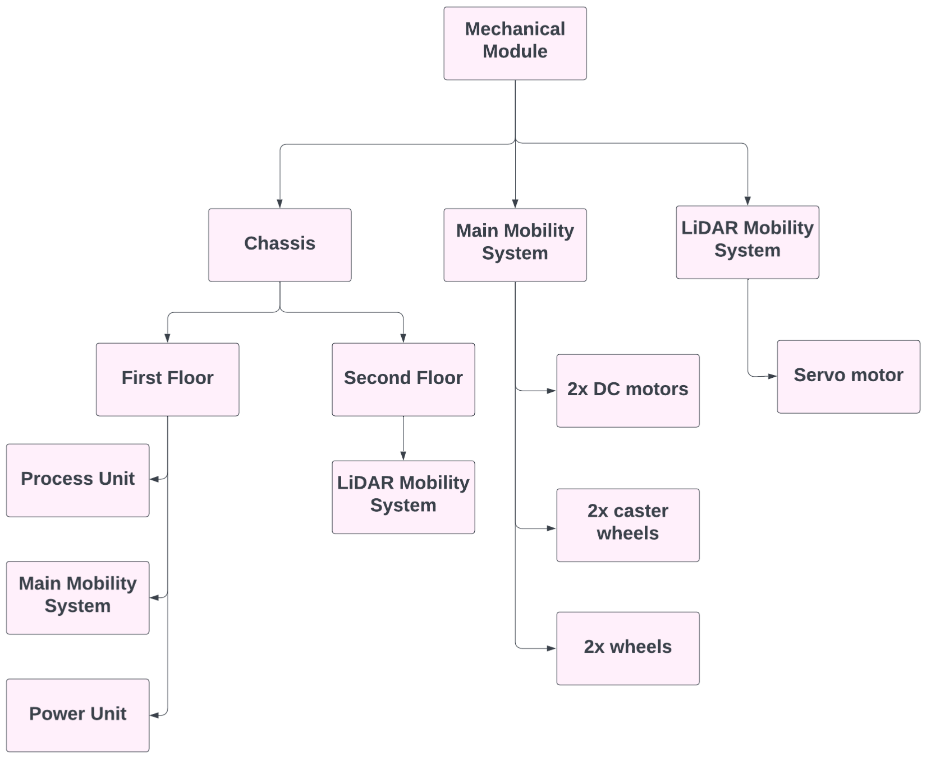


Figure 9: Mechanical Subsystem Flowchart

Chassis Design

The chassis is the skeleton of the UGV, engineered to provide a lightweight yet strong framework that supports all other subsystems. The chassis design has two circular floors with the LiDAR mobility system on top.

- The first floor is the main supporter. It carries DC motors, caster wheels, Arduino UNO, Raspberry Pi 4, powerbank, battery, voltage regulator, and motor driver. These components are connected to the chassis with screws in various sizes, ensuring rigidness.
- The second floor carries the LiDAR mobility system. The floors and their surfaces are utilized efficiently to obtain low centre of gravity and ease of access for maintenance and component replacement. The chassis design from isometric, front, side, and top views can be seen in Figure 10.

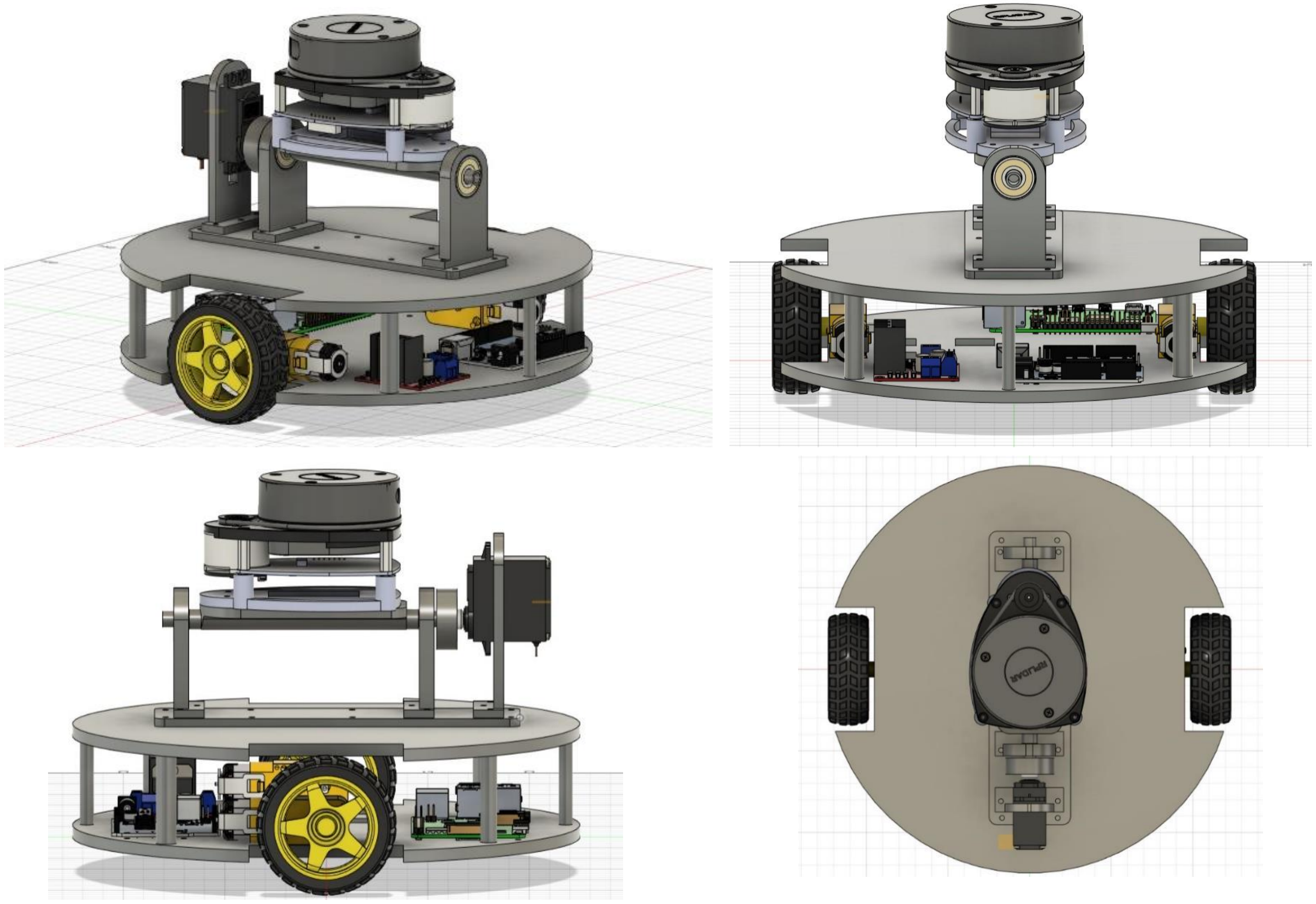


Figure 10: Isometric, Front, Side and Top Views of the Chassis

Moreover,

- The design employs 3D printing techniques. These parts are durable and lightweight with easy access to electronic parts. Moreover, these parts are easily connected to other parts by utilizing screws.
- Special attention has been given to the stability of the UGV, with a maximum height restriction of 30 centimetres to prevent toppling and ensure balance during rapid movements and acceleration. The prototype UGV has reached to a height of 24 centimetres.

Main Mobility Unit

The UGV's mobility is conferred through a thoughtfully designed system consisting of a number of wheels, motors, and a control mechanism. This design employs two main wheels connected to the motors, complemented by two caster wheels.

- The motors are located at the sides of the vehicle. To the front and back of the vehicle, a total of two caster wheels are connected to enhance the stability and balance of the vehicle. This configuration allows the vehicle to execute precise movements and turns by varying the wheel speeds independently.

- Motor selection is done based on the trade-off between brushed DC and brushless DC motors, with considerations including efficiency, maintenance requirements, torque characteristics, power consumptions, and cost. Although BLDC motors are superior in terms of efficiency and maintenance, brushed dc motors are more feasible in terms of cost, availability and simplicity of drive and control. BLDC motors are capable of operate without reduction in the efficiency for wide range of torque and speed. However, the load is almost constant during the whole operation and does not demand high speed or high torque. Brushed DC motors can also operate for this type of load cost effectively and the control is relatively simple compared to BLDC motors. Thus, a brushed DC motor is chosen for main drive system.

- Load of the system is the friction between the tyres and the ground. The friction coefficient can be approximated as 0.6 and mass of the robot is expected to be smaller than 2kg.

$$T_{load} = \frac{1}{2} M\ g\ k_{friction} r = \frac{1}{2}(2)(9.8)(0.6)\frac{43\ 10^{-3}}{2} = 0.12642\ Nm = 1.29\ kg \cdot cm$$

- Selected motor has rated torque of 3.6 kg cm and rated speed of 110 RPM.

$$v_{max} = n_{motor}\frac{\pi}{30}\ r = 0.5\ m/s$$

$$a = \frac{T_{motor} - T_{load}}{M} = 0.1132\ m/s^2$$

$$t_{acc} = \frac{v_{max}}{a} = 4.417\ s$$

It can be seen that the selected motor is adaquate for the drive system.

- A motor driver is incorporated to control the torque of the motor and, consequently, the direction and speed of the vehicle. An Arduino UNO is used to control the motor driver LN298N, which is the bridge between the power supply and the motors. These connections are shown in Figure 12, in System Integration Subsystem part.

LiDAR Mobility Unit

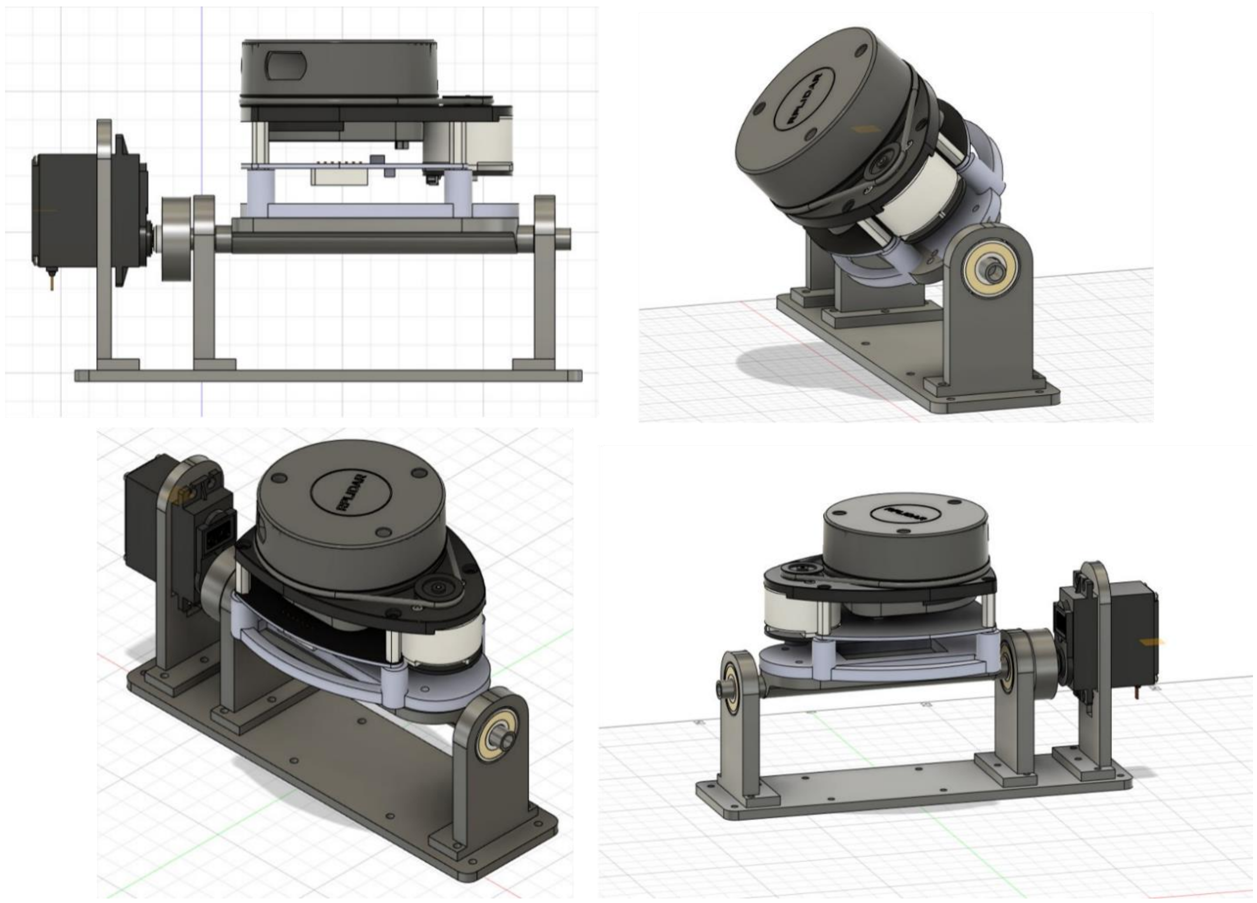


Figure 11. 3D Design of the LiDAR Mobility Unit

Figure 11 presents the modified mobility system of the LiDAR, which includes a servo motor MG995 that has a peak torque of 12 kgcm. Servo motor is directly attached to the platform that is designed to carry the LiDAR.

The purpose of this platform is to not make the servo motor carry the weight of the LiDAR but to make the system carry the weight and use the servo motor only for rotating it. On the both sides of the LiDAR's platform, there are roller mechanisms including bearings. This mechanism is implemented to distribute the load and reduce the torque demand on the servo motor, which can help in preventing mechanical overload and reducing wear.

The main platform serves as the structural component that connects the servo motor and the rotating system to the UGV's chassis. This setup is critical for integrating the mobility system with the UGV and ensuring that the servo motor can effectively support and move the LiDAR sensor as required.

The mechanical module is central to the functionality of the UGV, affecting its operational capabilities in real-world scenarios. The design considerations detailed in this report are crafted to meet the specific demands of the UGV's intended use, from material choices to power management. The modular nature of the design also allows for regular maintanence and adjustments based on wears on the parts.

System Integration Subsystem

The integration subsystem is a crucial facet of the autonomous unmanned ground vehicle (UGV), facilitating interaction between the user and the vehicle through an intuitive user interface and a robust wireless communication system. This part delineates the design and development process for the integration module, emphasizing, control unit, power management system, data visualization, graphical user interface and seamless connectivity within operational constraints.

Computer Unit

The main computer of the UGV is a Raspberry Pi 4 with a RAM of 4GB. The selection of the main computer is made considering the size, price, computational performance, power consumption and number of USB ports mounted on it. It is installed with a Ubuntu Server 20.04 as the operating system -OS-, which is a free and open-source software. One of the main reasons for selecting such OS is its compatability with ROS -Robot Operating System- which utilizes the modules and libraries needed for mapping, localization, and navigation of the UGV so that ROS is also installed. As Ubuntu Server 20.04 does not establish a graphical user interface -GUI-, Ubuntu Desktop is installed on the Ubuntu Server 20.04 so that when the user is connected to the Raspberry Pi 4 directly over a monitor, it displays a user friendly interface. To be able to offer this user friendly interface when the user is connected to the Raspberry Pi 4 wirelessly -the details for the communication will be given in "Communication Unit" part- XRDP is installed. XRDP is also a free and open-source software that enables OS other than Microsoft Windows to establish a "Remote Desktop" interface. To be able to communicate with Arduino devices, Arduino IDE is installed. After installing such software, less than 10GB of the 32GB available storage of the Raspberry Pi 4 is used and it takes less than a minute for the Raspberry Pi 4 to boot itself up. The tryouts of the Raspberry Pi 4 have shown that it is capable of running CPU consuming softwares like ROS, so that it is being used as the hub for data collection and processing, and the user computer is a remote display and control device.

Communication Unit

The UGV is engineered to operate effectively within areas of limited range, which requires a wireless communication system capable of high-speed data exchange between the UGV and the user computer while being power efficient. There are two main communication basis are established in the design: one is the communication between the UGV and the user computer, and the other one is the communication of the hardware within the UGV.

The main communication with the user computer and the UGV is established over Wi-Fi networks. In the config files of the Ubuntu Server 20.04, "network_config.conf" is updated with the SSID and the password of the Wi-Fi network that is used so that Raspberry Pi 4 can be connected to a Wi-Fi network during boot up. When the Raspberry Pi 4 is connected to a network, XRDP server also becomes online which enables remote desktop connections to the

Raspberry Pi 4. A critical point here is that to be able to connect to the Raspberry Pi 4, the user computer and the Ubuntu Server 20.04 should be configured in a parallel manner so that they will be connected to the same Wi-Fi network. When the UGV and the user computer are connected to the same Wi-Fi network, the user computer can connect to the UGV using "Remote Desktop Connection" if it uses Windows or "Remmina" if it uses a Ubuntu version. To be able to connect to a remote desktop, the IP address of the remote desktop is needed. The IP of the UGV is found from the "Advanced IP Scanner" software which can be installed freely to the user computer. It shows the devices connected to the same Wi-Fi network with the user computer so that the IP of the Raspberry Pi 4 can be easily found.

Raspberry Pi 4 is the base of the communication between the hardware used in the UGV with two USB 2.0 and two USB 3.0 ports. RPLiDAR A1-M8 and Arduino UNO are directly connected to Raspberry Pi 4 over two USB 3.0 ports which can establish faster data transfer. Arduino UNO is mainly used for the control of the DC motors used. Using serial communication, Arduino UNO communicates with the MG995 Servo Motor, and L298N Motor Driver which controls the two DC motors used for the mobility. With the help of "pyFirmata" library installed in the Arduino UNO, the Python scripts regarding the control of the motors that are being run in the Raspberry Pi 4 can be directly sent to the servo motor or the motor driver easily. In Figure 12, the communication system in the UGV is represented.

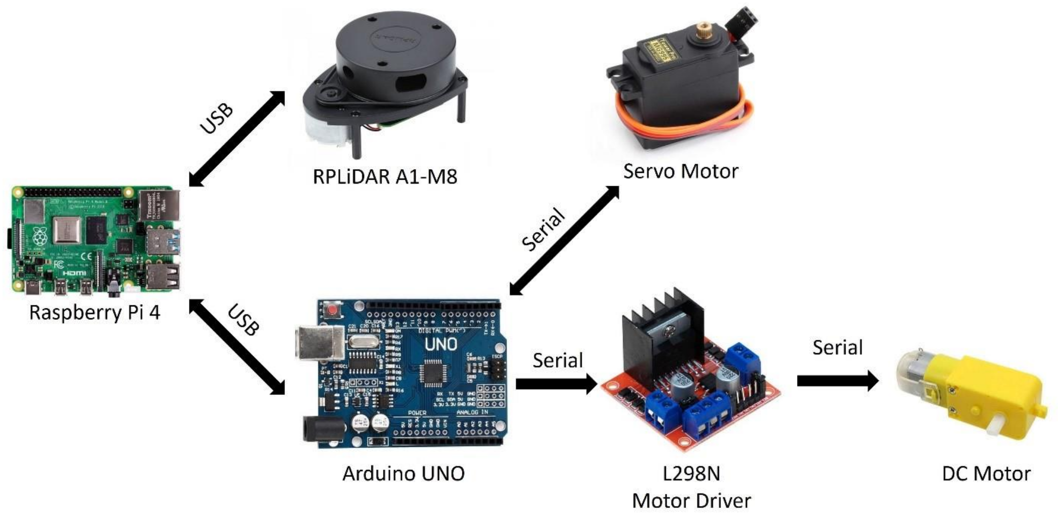


Figure 12. Communication System within the UGV

Power Supply Unit

The power supply unit is essential in ensuring that every element of the system is provided with the necessary power for its operation. It is tasked to distribute energy to all the hardware used in the UGV. For electrical safety, the power supply unit is divided into two parts: LiPo battery, and a power bank.

As the battery, a 14.8V 3300mAh 4S1P LiPo battery from Jetfire is used. It is used to power up the DC motors and the servo motor. As the L298N motor driver can take 5-35V input from its VCC pin, the battery is directly connected to it without any voltage regulator. The servo motor operates with 5-6V, however the output torque increases with the increasing input voltage. To supply 6V to the servo motor, an LM2596 Voltage Regulator is used. It decreases an input voltage between 3.2-40V to 3-35V so that it is safe to use this regulator. The output of it is arranged with a screw on it so that it is arranged to 6.05V, regarding the losses. For safety, a switch is placed between the battery, motor driver and the voltage regulator supplying the servo motor.

The power bank used is a TTEC Powerslim with a capacity of 10000mAh. It powers up the Raspberry Pi 4 which operates nominally with 5V and 3.1A, so that the powerbank is selected to be able to supply at least 15W. Also, using a power bank utilizes better electrical safety for the Raspberry Pi 4 as it can control the output current. Otherwise, the battery may damage the Raspberry Pi 4 as its output current can not be controlled. RPLiDAR A1-M8 and Arduino UNO are powered up from the Raspberry Pi 4, and Arduino UNO powers up the integrated circuit of the L298N Motor Driver from its 5V pin.

Power flowchart of the design is illustrated in Figure 13.

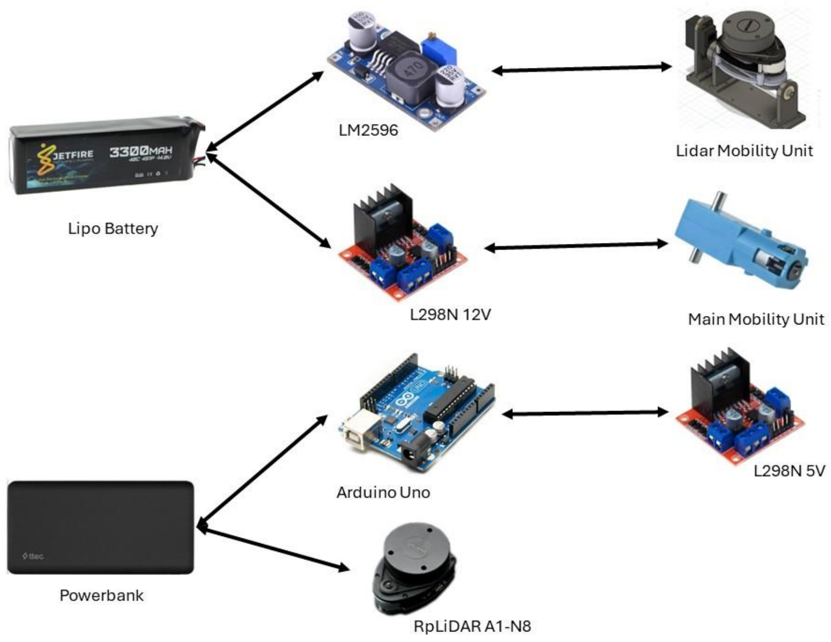


Figure 13. Power Distribution Diagram

User Interface Unit

To maximize the utility of the LiDAR data, the development of a user-friendly Graphical User Interface (GUI) is essential. The GUI enables users from diverse technical backgrounds to easily interact with the UGV and access comprehensive data. It is pivotal in enhancing the navigation module by allowing users to input selected coordinates for vehicle guidance and must integrate seamlessly with the wireless communication module. A desktop application on Raspberry Pi designed with Python's PyQt5, MATLAB's 3D plotting interface and Rviz - visualization tool within the ROS environment - are identified as the main frameworks for the GUI, with references to successful applications in related projects.

The desktop application oversees the primary process flow of the UGV, providing users with an easier interface to input their commands directly. It features a main page that allows users to navigate to three sub-pages: "Navigate," "Create a New 3D Map," and "Create a 2D Map via Hector-SLAM." Main page and each sub-page have the "Help" option where user could reach an explanation for the use of page and the functionalities. Main page is illustrated in Figure 14.

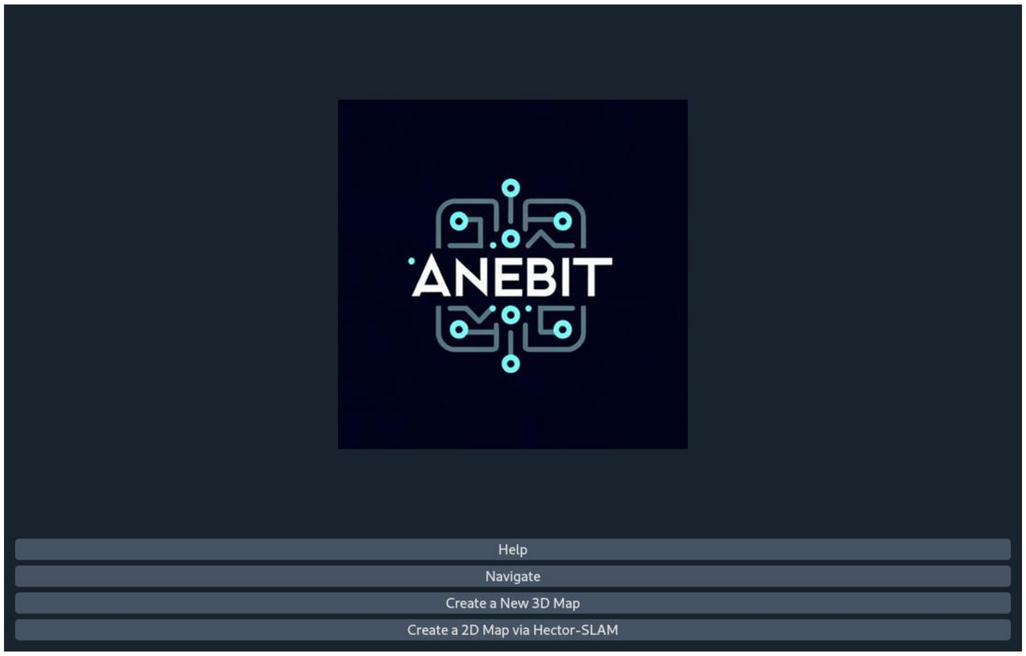


Figure 14: Main Page of the Application

"Create a New 3D Map" page is for the 3D map generation, as the name suggest. User can oversee the map generation stage via this page. He/she can trigger a data acqusition or navigate the robot manually through the environmnet. It offers two main options for navigation: "Use SLAM", "Don't Use SLAM". User can choose one of these options for manual navigation. It is illustrated in Figure 15.

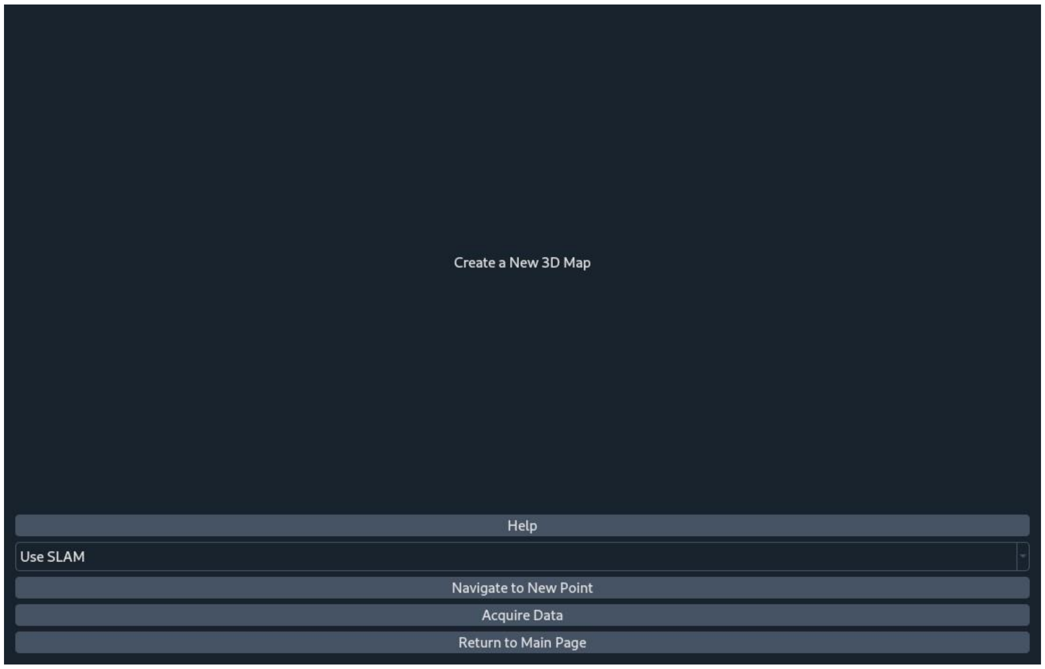


Figure 15. Create a New 3D Map Sub-page

3D map generation also interacts with the other two GUI components: RViZ and MATLAB. When "Use SLAM" option is selected for the navigation, Rviz screen opens to display the pose and the path of the UGV follows on the environmnet. It also displays the detected obstacles on the environment, enabling user to remotly control UGV's motion without direct visual connection.

To view the 3D map data, users can utilize the Matlab environment by transferring recorded map files to a computer with Matlab installed and running the designated software. In Matlab's 3D plotting interface, a scatter plot representing the point clouds, corresponding to the obstacles in the environment, will be displayed. This allows users to interact with the map visually. Additionally, the software facilitates the measurement of Euclidean distances between two selected points on the plot, enabling easy environmental measurements. An illustrative example of such a map is depicted in Figure 16.

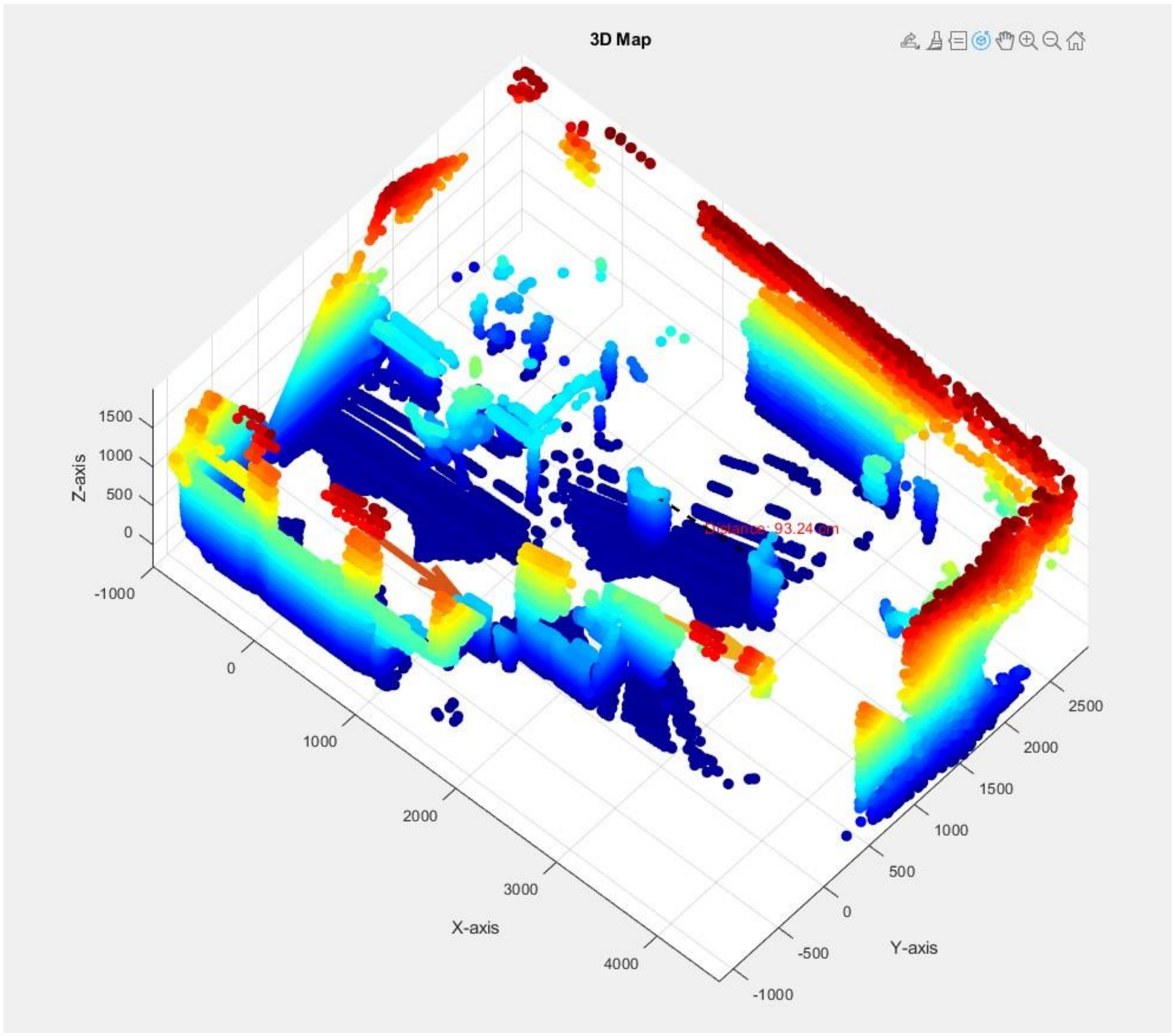


Figure 16: 3D Map Display on MATLAB

Similar to the "Create a New 3D Map" page, the "Create a 2D Map via Hector-SLAM" page is also responsible for map generation. It utilizes the Hector-SLAM library of ROS for both navigation and map creation. Users are presented with two options for map generation: "Save" and "Don't Use SLAM." Selecting the "Save" option triggers the SLAM algorithm to run during navigation and saves the generated 2D map for later use in autonomous navigation. On the other hand, selecting "Don't Use SLAM" allows for manual navigation in the environment without map generation. It is illustrated in Figure 17.

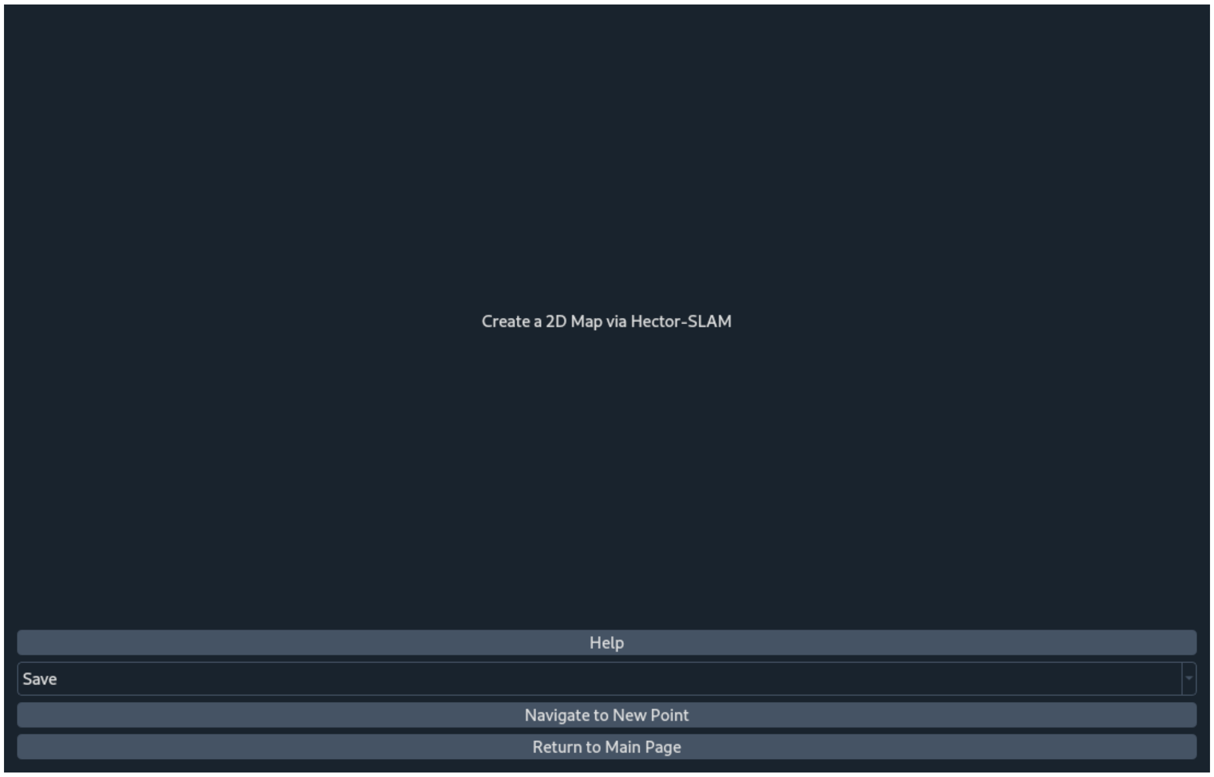


Figure 17. Create a 2D Map via Hector-SLAM Sub-page

The "Navigate" sub-page is dedicated to the autonomous navigation of the UGV. It presents the generated 2D map to the user and allows them to label obstacles, set or update specific label locations, delete labels, or command the UGV to "Go to Label" for autonomous navigation. It provides both autonomous and manual navigation options to the user. During autonomous navigation, the page interacts with RViZ, where clicking the "Start Autonomous Navigation" button opens an RViZ page.

RViZ displays various elements including the 2D map, UGV's pose estimation (indicated by a red arrow) with covariance (represented by a pink area), orientation angle covariance (depicted by a yellow area), 2D Navigation Target Pose (shown by a yellow arrow if provided), laser scans from the Lidar (displayed as red lines), and the path followed by the UGV during navigation (illustrated as a green line). This interface enables users to monitor the UGV's status, set 2D Navigation Goals, or measure distances between selected objects for convenient measurements. The Rviz screen closes and autonomous navigation halts when the user clicks the "Stop Autonomous Navigation" button, exits the "Navigate" page, or terminates the application. “Navigate” sub-page and RViZ screen for autonomous navigation are displayed in Figures 18, and 19.

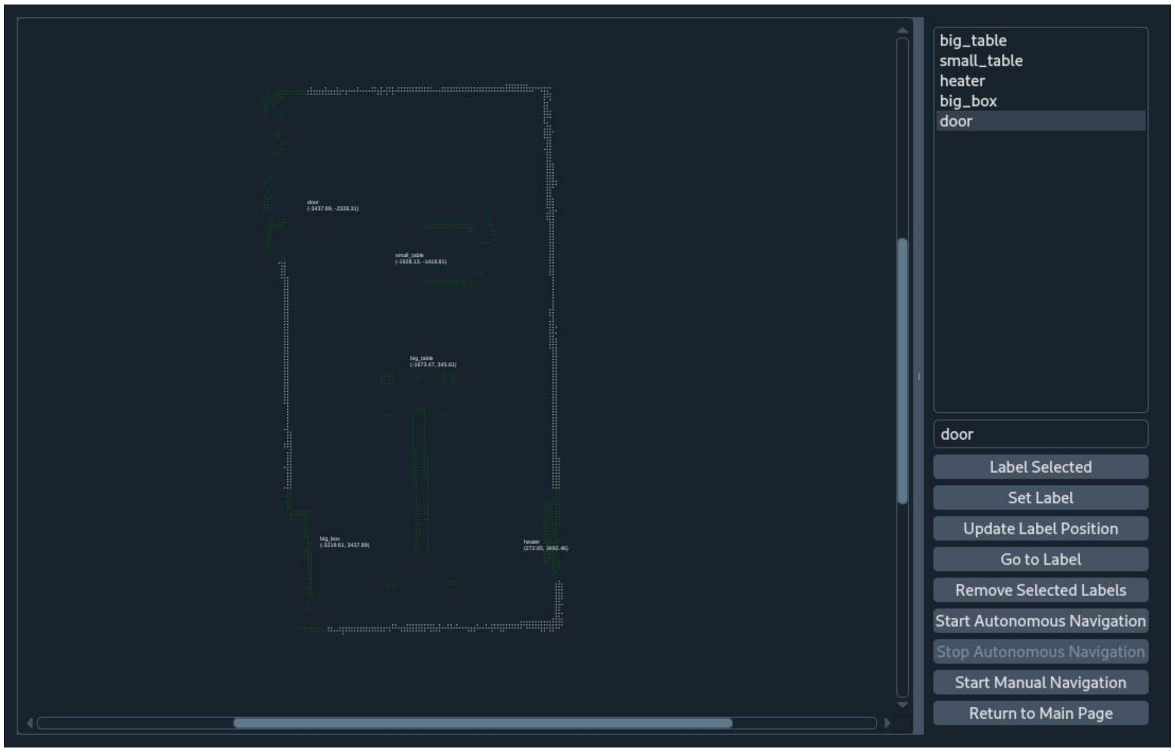
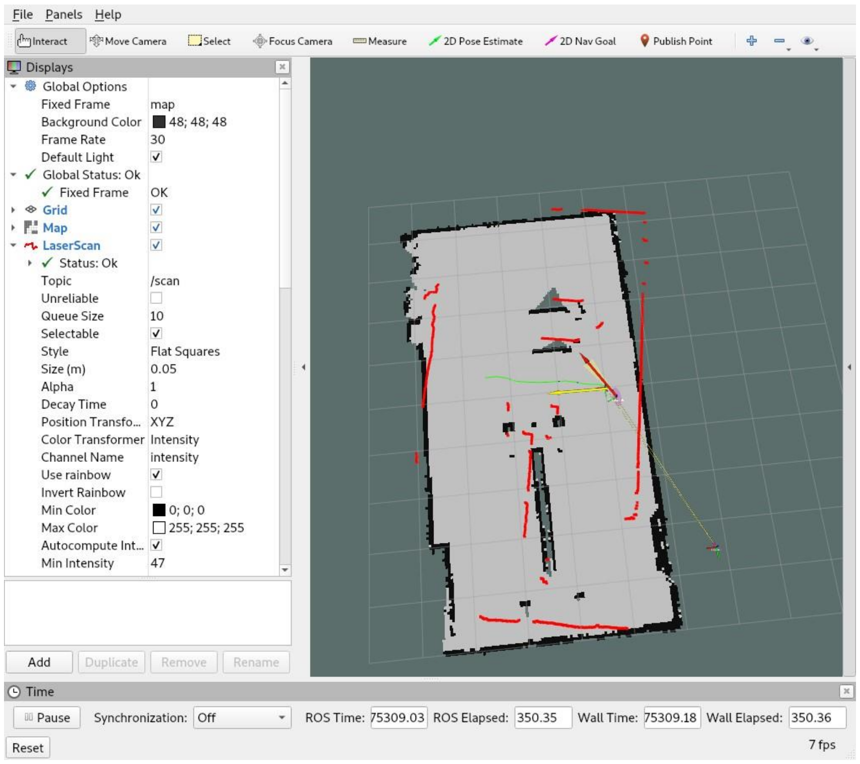


Figure 18. Navigate Sub-Page



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
Map	✓
LaserScan	✓
✓ Status: Ok	
Topic	/scan
Unreliable	☐
Queue Size	10
Selectable	✓
Style	Flat Squares
Size (m)	0.05
Alpha	1
Decay Time	0
Position Transfo...	XYZ
Color Transformer	Intensity
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐
Min Color	■ 0; 0; 0
Max Color	□ 255; 255; 255
Autocompute Int...	✓
Min Intensity	47

Synchronization: Off ROS Time: 75309.03 ROS Elapsed: 350.35 Wall Time: 75309.18 Wall Elapsed: 350.36

7 fps

Figure 19. RViZ Screen for Autonomous Navigation

Design Requirements

The design requirements of the project are classified into three categories: Performance Requirements, Functional Requirements, and Physical Requirements.

Performance Requirements for the Project

Perf.1. **Mapping Accuracy**: The 2D LiDAR system must generate a 3D map of the indoor environments with a mean-square error less than 17 cm.

Perf.2. **Servo Accuracy:** Error in the servo motor's rotation angle should be less than 2 degrees.

Perf.3. **Mapping Time:** The UGV should be able to accurately map indoor environments in a little amount of time. Depending on the environment set up, it should be able to map an average 25 m^2 room less than 15 minutes so that there is enough t ime for the user until the UGV needs a recharge.

Perf.4. **Navigation Accuracy:** The UGV should be able to move to the destination point with less than 10 cm error (Euclidian distance).

Perf.5. **Obstacle Detection**: The UGV should be detecting moving obstacles closer than 50 cm on its path.

Perf.6. **Self-Localization Accuracy:** The UGV must possess a highly accurate self-localization capability, enabling it to determine its position within the mapped environment with a precision of 10 cm in root mean square error. The self-localization system should maintain this precision consistently across various indoor environments and operational conditions.

Perf.7. **Battery Life**: The vehicle should have a sufficient battery life, higher than 30 minutes to complete mapping of an indoor environment and allow enough usage time for the user without frequent recharging.

Functional Requirements for the Project

Func.1. **Autonomous Navigation**: The vehicle must autonomously navigate to user-defined destinations using the pre-mapped environment.

Func.2. **Dynamic Path Planning**: The UGV should be capable of real-time path planning and adjustment in response to environmental changes or obstacles. The UGV must reliably detect and avoid both static and dynamic obstacles, including steps and uneven surfaces.

Func.3. **Navigation Efficiency**: The vehicle should navigate to defined destinations efficiently, calculating the ideal way without any observable computing time to minimize travel time while ensuring safety.

Func.4. **User Interface Functionality**: The GUI should allow users to easily input destinations, visualize the 3D map, and monitor the UGV's progress.

Func.5. **Data Processing and Storage**: The UGV should be able to process and store mapping data efficiently for real-time and future use.

Func.6. **Integration with User Devices**: The system must integrate seamlessly with various user devices for control and data transmission.

Func.7. **Wireless Communication Range and Stability**: The communication system should offer a stable and sufficient range enough to communicate in an indoor environment and to maintain constant contact between the UGV and the user's device.

Func.8. **Modular Design and Testability for Maintenance and Upgrades**: The design of the UGV should allow for easy maintenance and potential future upgrades. Also, each of the main modules could be easily testable to correct design errors.

Physical Requirements for the Project

Phys.1. **Compact Size**: The UGV should not exceed a height of 30 cm, ensuring it is compact enough for indoor navigation.

Phys.2. **Structural Integrity**: The vehicle must be robust and durable to withstand the wear and tear of indoor navigation.

Phys.3. **Weight Limitations**: The UGV's should be less than 4 kg so that it is enough to be easily transported and manoeuvrable, yet more than 2 kg so that it is enough to maintain stability.

Phys.4. **Power Source**: The vehicle should have an energy-efficient power source, such as a rechargeable battery, suitable for prolonged operational periods.

Phys.5. **Sensor Integration**: The 2D LiDAR sensor must be integrated seamlessly into the vehicle's design without significantly impacting its size or manoeuvrability.

Table 1. Requirements of the UGV

Performance Requirements	Value	Unit
Mapping Accuracy	17	cm
Mapping Time	15	min
Obstacle Detection	50	cm
Navigation Accuracy	10	cm
Self-Localization Accuracy	10	cm
Battery Life	30	min
Functional Requirements	**Value**	**Unit**
Autonomous Navigation		
Dynamic Path Planning		

User Interface Functionality		
Data Processing and Storage		
Integration with User Devices		
Modular Design and Testability for Maintenance and Upgrades		
Physical Requirements		
Compact Size	30	cm
Structural Integrity		
Weight Limitations	2000-4000	gram
Power Source		
Sensor Integration		

Table 2. Subsystems vs Requirements

Requirements	**SUBSYSTEMS**			
	MAPPING	**NAVIGATION**	**MECHANICAL**	**SYSTEM INTEGRATION**
Perf.1	✓			
Perf.2			✓	
Perf.3	✓			
Perf.4	✓	✓		
Perf.5		✓		
Perf.6	✓	✓		
Perf.7		✓		✓
Func.1	✓	✓		
Func.2		✓		
Func.3		✓		
Func.4				✓
Func.5	✓			
Func.6				✓
Func.7				✓
Func.8	✓	✓	✓	✓
Phys.1			✓	
Phys.2			✓	
Phys.3			✓	

Phys.4				✓
Phys.5			✓	

Compatibility Analysis

1. **Signaling and Mechanical/Physical Interfaces:**

   - Communication between the Raspberry Pi 4 and the LiDAR mobility system, is compatible. By using a micro-usb cable, LiDAR is connected to the Raspberry Pi 4 and servo motor is connected to the Arduino UNO using jumpers. Also Arduino UNO is connected to the Raspberry Pi 4 via a USB cable.

   - Raspberry Pi 4 and the user computer are connected to each other using a common WiFi network. By using Remote Desktop Connection app found in Windows OS, or the Remmina app found in Ubuntu OS, Raspbery Pi 4 screen is projected to the user computer screen.

   - The LiDAR Mobility System is situated on a separate floor of the robot's chassis to prevent interference with other components. Our design ensures that physical dimensions, mounting points, and attachment mechanisms align properly without any interference. Additionally, efforts have been made to minimize cable clutter.

2. **Software/Firmware Versions:**

   - All the libraries are compatible with Python 3.9 and Ubuntu Server 20.04. Codes are developed using the libraries that are compatible with Python 3.9. Also the Ubuntu version selected as 20.04 such that ROS Noetic can be used.

3. **Timing Analysis:**

   - The communication between the LiDAR, servo motor, and Raspberry Pi 4 is synchronized. Additionally, the remote desktop connection between the Raspberry Pi 4 and the computer experiences minimal delay depending on the strength of the WiFi signal.

4. **Compliance with Relevant Standarts:**

   - Wireless communication between the Raspberry Pi 4 and the user computer complies with Wi-Fi standards to ensure secure and reliable data transmission.

   - Electrical safety is achieved through the use of a switch button between the battery and the main mobility system and a buck converter. Also, sensitive components like LiDAR, Raspberry Pi 4 and Arduino UNO are powered from the powerbank as the current of it can be controlled so that those components will be safe.

   - The accuracy rate of a project is around 93% for mapping. The maximum error rate we have for 3D mapping is around 15%, which shows us we are on the right track [1].

Moreover, the mechanical system is controlled using the mapping and navigation unit via Python3. The servo and motor controllers are adjusted using Python and LiDAR information. The LiDAR determines the speed of the motors, and the angle of the servo is determined using Arduino UNO. Each communication method is explained in more detail in previous sections. Since all communication inside UGV are done via USB cables, the system experiences negligible delay. All the systems work properly with each other and provide feedback to maximize the performance of the UGV.

Table 3: Subsystem Level Compatibility Analysis

Subsystem	**Input(s)**	**Output(s)**	**Interface**	**Environment**
Mapping	Laser Scan Data	2D map "robot_location"	RviZ, hector_slam	Raspberry Pi 4
Mapping	Laser Scan Data	3D map "servo_angle"	MATLAB	User PC
Navigation	Laser Scan Data	"cmd_vel" "amcl_pose"	RviZ amcl	Raspberry Pi 4
Mechanical	"cmd_vel"	DC motor speed	Arduino L298N	Raspberry Pi 4
Mechanical	"servo_angle"	Servo angle	Arduino LM2596	Raspberry Pi 4
System Integration	User Commands	"Target location" "Start mapping"	GUI	Raspberry Pi 4

"cmd_vel" refers to the linear and angular velocity commands directed to the UGV, which are genertated by the "move_base" library that is used.

Compliance with Requirements

In the following section, the solutions that are proposed for each of the design requirements are represented.

Perf.1 Modified Euler Transform is introduced to increase accuracy.

Perf.2 Servo mobility system is designed to increase accuracy.

Perf.3 DC motors with high RPM values are selected. Number of data points obtained in each angle with the LiDAR is decreased to decrease the overall mapping time.

Perf.4 Path planning algorithm is implemented to increase navigation accuracy.

Perf.5 2D map and AMCL will be used for obstacle detection.

Perf.6 Hector SLAM and AMCL is used for accuracy.

Perf.7 Power bank is introduced to increase battery life and electrical safety but it introduces more weight to the robot which conflicts with the design requirement **Phys.3**.

Func.1 Path planning algorithm is introduced.

Func.2 When a dynamic obstacle is seen, the path will be recreated, which represents dynamic path planning.
Func.3 Path planning algorithm calculates the shortest distance.
Func.4 Raspberry Pi 4 screen and a GUI which is created using Python is introduced.
Func.5 Raspberry Pi 4 is used.
Func.6 Devices that can connect to a WiFi network will be able to use this UGV.
Func.7 Wifi communication protocol is introduced.
Func.8 All the pieces are portable since screws are used.

Phys.1 A neater and a smaller design solution is implemented.
Phys.2 2 caster wheels are introduced in the main mobility unit for stability and integrity.
Phys.3 Lightweight materials are used to minimize weight.
Phys.4 A LiPo battery, a power bank and voltage regulators are introduced.
Phys.5 LiDAR is placed in the second floor of the robot which increases the height of the robot and conflicts with the design requirement **Phys.1.**

At the end, all of the requirements are satisfied. However, utilizing the LiDAR Mobility System and a two-floor robot approach can increase the height and may impact the physical conditions of the design requirements Phys1 and Phys2. Additionally, larger implementations of Wi-Fi connections may require extra effort. While using two power sources maximizes battery life, it also results in a heavier robot and increases the number of cables and converters, thus making the robot larger.

A table representing the compliance with requirements is given in Table 4.

Table 4. Compliance with Requirements

Requirements	LiDAR	Processor Module	Motor Control	Path Planning	Chassis	Main Mobility System	LiDAR Mobility System	Commu-nication	Power Supply	User Interface Design	Data Visualization and Interaction	Synchronous Communication and Feedback
Perf.1	√	√										
Perf.2							√					
Perf.3		√										
Perf.4	√	√	√									
Perf.5				√								
Perf.6	√	√	√	√								
Perf.7									√			
Func.1	√	√	√	√								
Func.2				√								
Func.3			√	√								
Func.4											√	
Func.5		√										
Func.6								√				√
Func.7								√		√		
Func.8	√	√	√	√	√	√		√	√			√
Phys.1					√	√						
Phys.2					√	√						
Phys.3					√							
Phys.4									√			
Phys.5							√					

Test Results

A number of tests were conducted with the prototype of the UGV to be able to tests the subsystems.

Test 1

Test 1 was conducted to assess the accuracy of the mapping subsystem and the effectiveness of data collection from multiple points within a room. The experiment took place in a room measuring 10-15 m^2, containing various static obstacles such as tables, chairs, and trashes.

The UGV was positioned at one location within the room and initiated data collection. While the chassis remained stationary, the LiDAR sensor was rotated using the LiDAR mobility system, capturing data, and recording both the LiDAR readings and the corresponding servo angles in a text file. This scanning process continued until the LiDAR had scanned 120 degrees. Once completed, the UGV was relocated to another position within the room with manual navigation system, and data acquisition was recommended. This cycle repeated three times.

Each measurement was recorded along with the location's pose relative to the initial position, and the time duration. Pose information was obtained by using the SLAM algorithm during navigation. After all data points, along with their respective location poses, were collected, they were processed using MATLAB and plotted to create a 3D map visualizing the environment. Processing time was also measured.

Subsequently, specific static points within the environment were selected, and their distances from each other were measured using a ruler in the physical environment, providing the actual results. These distances were then calculated on the 3D map generated in MATLAB, yielding experimental results.

Upon reviewing the results, it's evident that despite the error rate generally increasing with actual distance, it consistently remains below 17 cm. Also, we observed that data acquisition takes approximately 1 minute for each location while data processing does not exceed 5 seconds. These results suggest that the mapping subsystem operates efficiently and accurately, meeting the corresponding requirements satisfactorily. Test results and corresponding sub-system requirements are given in Table 5.

Table 5. Subsystem Requirements vs. Test Results

Subsystem Requirements	**Error Rates are Under the Threshold**	**Data acqusition and processing duration**
Perf.1	✓	
Perf.3		✓
Perf.4	✓	
Perf.6	✓	
Func.1	✓	
Func.5		✓

Func.8	✓	✓

Detailed description of the test procedure and results can be found at Appendix-1.

Test 2

Test 2 is conducted to assess the accuracy of the navigation subsystem and the efficiency of self-localization of the UGV. The experiment took place in a room measuring 10-15 m^2, containing various static obstacles such as tables, chairs, and trashes.

If the 2D map of the environment is not present, it is generated moving the UGV around the environment by using manual navigation system and SLAM algorithm. 2D map generated by the SLAM algorithm is saved at end of the operation for later use.

Later on, the UGV placed on a random location in the environment and localization algorithm, AMCL is initiated. To correct the initial localization, manual navigation system is started and the UGV is navigated manually around a small range. After localization accuracy exceeds a certain range, manual navigation terminated, and automatic navigation initiated. The pose estimation of the UGV is measured via graphical interface and actual position is measured by a ruler and results are recorded. This procedure is repeated 3 times.

After that, 2D navigation goals are given to the UGV through the GUI a number of times. After navigation is done and the UGV reaches the target point, its actual location is measured with a ruler. Lastly, the target position is determined through GUI and results are recorded. This procedure is repeated 3 times.

We observed the GUI manages to localize itself with the acceptable error while navigating autonomously on the environment without crashing any static obstacle and reaching the target location with an acceptable error rate.

However, we observed that the GUI is unable to avoid dynamic obstacles if they are present on the environment.

Upon reviewing the results, it's evident that the error rate is consistently remains below 10 cm for localization and 15 cm for navigation. Also, we observed that the UGV is not capable of dynamic obstacle avoidance. These results suggest that the navigation subsystem operates efficiently and accurately, meeting the majority of the corresponding requirements satisfactorily, except the fact that it cannot fully satisfy obstacle detection and dynamic path planning requirements because of the lack of dynamic obstacle avoidance. Test results and corresponding sub-system requirements are given in Table 6.

Table 6. Subsystem Requirements vs. Test Results

Subsystem Requirements	**Localization Error Rates are Under the Threshold**	**Navigation Error Rates are Under the Threshold**	**Lack of Dynamic Obstacle Avoidance**
Perf.4	✓	✓	
Perf.5	✓	✓	✕
Perf.6		✓	

Perf.7		✓	
Func.1	✓	✓	✕
Func.2	✓	✓	✕
Func.3	✓	✓	
Func.8	✓	✓	

Detailed description of the test procedure and results can be found at Appendix-2.

Test 3

Test 3 aimed to show that power resources, which are the LiPo battery and the powerbank, are capable of supply power to complete mapping and navigation procedures. The voltage limits for the Lipo battery is from 16.8 V to 15.6 V for healty battery life time.

First, the voltage of the Lipo battery and charge percentage of the powerbank are measured. Then, navigation procedure is conducted according to the Test 2. After completing the navigation procedure, voltage of the Lipo battery and charge percentage of the powerbank are again measured. I has seen that voltage of the Lipo battery dropped to 16.3 V from 16.4 V and charge percentage of the powerbank dropped by 10%. Navigation procedure hold 9 minutes. According to these results, Lipo battery and powerbank can operate before running out of charge for 216 minutes and 90 minutes respectively. The determining source is powerbank for navigation. Therefore, navigation procedure can be repeated 10 times.

Secondly, mapping procedure is conducted according to the Test 3. After completing the mapping procedure, voltage of the Lipo battery and charge percentage of the powerbank are again measured. I has seen that voltage of the Lipo battery dropped to 16.2 V from 16.3 V and charge percentage of the powerbank dropped by 6%. The mapping procedure hold 7 minutes. According to these results, Lipo battery and powerbank can operate before running out of charge for 168 minutes and 116 minutes respectively. The determining source is powerbank for mapping. Therefore, mapping procedure can be repeated 16 times.

According to these results, the determining source for operation duration is power bank for both mapping and navigation procedure. Powerbank can endure for approximately 100 minutes.

Table 7. Subsystem Requirements vs. Test Results

Subsystem Requirements	**Error Rates**
Perf.2	✓
Perf.7	✓

Resource Management

Cost Analysis

The money spent on the final version of the robot components is provided in Table 8.

Table 8. Finalized Robot Cost Analysis

Component	Price (₺)	Price ($)	
RPLidar A1-M8	2872,00	88,71	$/₺ = 32,3745
Raspberry Pi 4	1862,00	57,51	
3300 mAh Battery	1323,00	40,87	**Total:**
10000 mAh Powerbank	445,00	13,75	8098,70 ₺ / 250,16 $
Arduino Uno	163,00	5,03	
MG995 Servo Motor	150,00	4,63	
LiDAR Mobility Unit	110,00	3,40	**Remaining Money:**
L298N Motor Driver	64,00	1,98	1613,65 ₺ / 49,84$
LM2596 Voltage Regulator	26,00	0,80	
2 x DC Motor	174,00	5,37	
2 x N20 Wheels	57,60	1,78	
2 x Caster Wheels	46,80	1,44	
Chassis	400,00	12,35	
25W Type-C to Type-C Cable	139,90	4,32	
Micro USB to USB Data Cable	114,90	3,55	
Type-A to USB Data Cable	25,80	0,79	
Screws, Standoffs	55,00	1,69	
Power Cables, Jumpers	60,90	1,85	
Switch	8,80	0,27	

The total money spent throughout the design and engineering process of the project is given in Table 9.

Table 9. Total Cost of the Project

Component	Price (₺)	Price ($)	
Robot Cost	8098,70	250,16	**$/₺ = 32,3745**
Burned Components	250,00	7,72	
LiPo Battery Charger	1198,00	37,00	**Total:**
Charger Adapter	249,90	7,72	10853,87 ₺ / 335,26 $
2 x 12V Micro DC Motors	256,68	7,93	
2 x 6V Micro DC Motors	256,68	7,93	
2 x Yellow DC Motors	62,72	1,94	

3m Micro USB Cable	240,90	7,44
Yellow Wheels	45,90	1,42
Blue Slim Wheels	70,55	2,18
Other Materials	380,55	11,75

Power Management

As explained in the Power Supply Unit part, motors and processors are powered separately for safety reasons. Since motors can draw excessive amount of current in a short period of time, the processors might be harmed. Therefore, the mobility system is powered by the 3300 mAh 14.8V LiPo battery, and the processors are powered from 10000 mAh Ttec Powerbank.

LiPo battery supplies energy to the +12V pin of the motor controller L298N, which supplies the necessary power to the DC motors that are used in Main Mobility Unit. Lipo battery also supplies power to the servo motor in that is used in the LiDAR Mobility Unit. Since MG995 servo motor operates with 6V, an LM2596 buck converter is used to regulate the LiPo battery voltage.

10000 mAh Ttec Powerbank directly supplies energy to the RPLiDAR A1-N8 and to the Arduino Uno which runs the motor control code and connected to the L298N motor driver's 5V pin. Power distribution system is depicted in Figure 20.

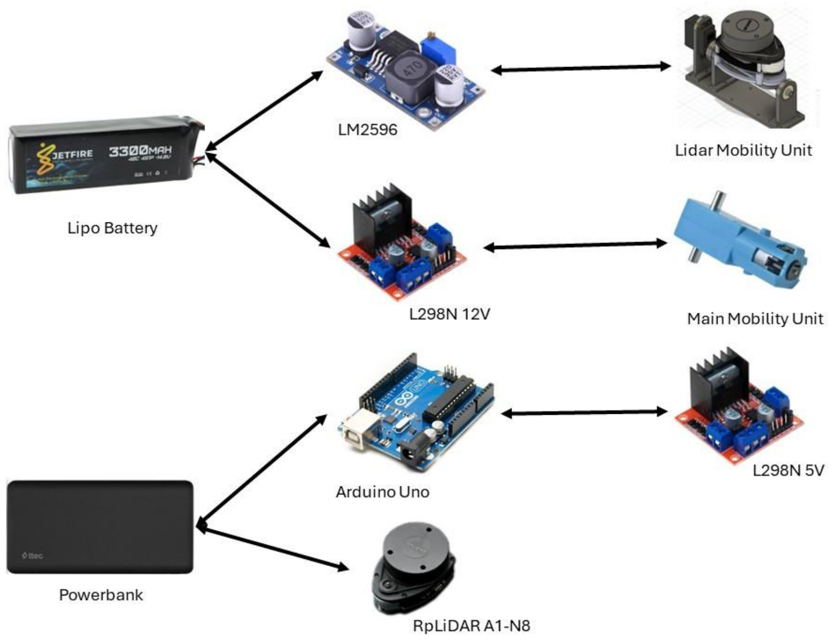


Figure 20. Power Distribution System

In Test-3, the state of charge change of the Lipo battery and power bank are measured for both Lipo Battery and power bank. It is calculated that the vehcile can complete navigation for 10 times and complete mapping 16 times without both sources runnign out of charge. Also, it has been seen that power bank is run out of charge before the Lipo battery. Hence, the powerbank is the determining factor. It has seen that both of the supplies can operate much more than 30 minutes without running out of charge.

Deliverables

In the deliery phase of the Autonomous Unmanned Ground Vehicle (UGV), our team will provide a full set of products, services, and guides to make sure the UGV works well in different indoor places. We'll deliver a working UGV prototype, 3D mapping software, a connectivity kit to link the UGV with devices, a user-friendly interface, an explanatory user manual and testing tools.

1. UGV Prototype:

Size and Design: The UGV prototype is a compact, versatile robot, standing at a height of 24 centimeters. Its size and design are optimized for maneuverability and functionality in various indoor settings.

2D LiDAR Sensor: Equipped with a 2D LiDAR sensor, the UGV can perform intricate mapping tasks and navigate autonomously. The sensor plays a crucial role in creating detailed spatial maps and detecting obstacles, ensuring the UGV's efficient and safe operation in indoor environments.

2. 3D Mapping Software:

Data Processing: The software acts as a vital component of the UGV system, processing the data captured by the LiDAR sensor. It converts this data into a detailed 3D map, providing a comprehensive visual representation of the UGV's surroundings.

Features: The software package includes features for mapping, where the user guides the UGV to generate the 3D map of an area. Waypoint marking enables users to set specific points for the UGV to navigate to.

3. Connectivity and User Interface:

Connectivity Kit: A kit containing all necessary cables for connecting the UGV to computers will be provided. This ensures ease of integration and accessibility for users to interface with the UGV.

User Manual: A comprehensive user manual will accompany the UGV. This manual will detail the operational capabilities of the UGV, providing users with the knowledge to effectively utilize the operations of the robot.

In conclusion, the deliverables for the Autonomous UGV project will span both the tangible, in the form of the UGV prototype equipped with a 2D LiDAR sensor and a user interface, with a suite of documentation and live demonstrations. These elements comprise a complete package designed to ensure a deep understanding of our UGV, its capabilities, and its operation, ultimately delivering a cutting-edge solution for autonomous indoor navigation and mapping.

Safety Issues and Precautions

In the design and development of our autonomous unmanned ground vehicle (UGV), safety is paramount and is considered the foremost priority for both users and manufacturers. From the initial concept through to the final implementation, every step of the project incorporates both manual and coded precautions to ensure the highest level of safety. This commitment to safety is reflected in the detailed design of each subsystem and the comprehensive safety measures integrated throughout the development process. Below, we detail the potential safety issues associated with the UGV and list the specific precautions taken for each, ensuring that both functionality and safety are addressed in harmony.

1. **Collision Risk with Static and Dynamic Obstacles:**
    - **Precautions:**
        - We implemented robust obstacle detection algorithms using 2D LiDAR data to detect both static and dynamic obstacles in real-time.
        - We used "dwa_local_planner" for navigation, which stands for Dynamic Window Approach. This package dynamically reconfigures the path to avoid obstacles.
2. **Reliance on Environmental Conditions:**
    - LiDAR's effectiveness is influenced by environmental conditions, including the physical state of the surroundings such as opacity and thickness.
    - **Precautions:**
        - Maximum speed of the robot is limited.
        - Cover mirrors if present and avoid using the robot in rainy environments.
3. **Interference with or by Humans:**
    - **Precautions:**
        - We have a stop mechanism in thto halt the UGV if it behaves unpredictably or approaches a human too closely.
        - We have an algorithm in the user interface for immediate interaction when necessary, providing controls to manually guide the UGV.
4. **Electrical Hazards:**
    - **Precautions:**
        - All power cables are wrapped with a heat shrinking tube to prevent potential electrical risks.
        - A switch is connected to the output of the LiPo battery, to be able to connect and disconnect the battery easiliy.

Additional Considerations

- **Testing and Validation:** Extensive testing should be conducted in controlled environments before deployment. This includes stress testing the software and hardware to handle unexpected scenarios and ensuring the UGV can safely stop or navigate around unexpected obstacles.
- **Compliance with Regulations:** Ensure the design complies with local safety regulations and standards for autonomous vehicles, including those specific to indoor environments.

By addressing these safety issues with comprehensive precautions, the autonomous UGV system can operate safely and efficiently within indoor environments, and potentially outdoors, with minimal risk to people and property.

Widespread Applications and Societal Impacts of the Autonomous UGV System

The autonomous unmanned ground vehicle (UGV) system developed in this project has the potential for widespread application across various sectors, significantly impacting society in numerous positive ways. Here are some of the key applications and their societal impacts:

1. **Emergency Response and Disaster Management:**
    - **Application:** The UGV can be deployed in disaster-stricken areas where it might be too dangerous for human responders. Its ability to navigate through rubble and avoid obstacles can be crucial in search and rescue operations. Similar case is understanding its location in unknown area and navigate through the exit.
    - **Impact:** Enhances the effectiveness and speed of emergency response operations, potentially saving lives while reducing the risk to human responders.
2. **Healthcare and Elderly Care:**
    - **Application:** In healthcare facilities or homes, the UGV can be used for delivering medications and food to patients or the elderly, especially in isolation conditions, such as during pandemics. Like labelling the humans in a floor, tie up a food to UGV, and give a command "Go to Granny Zeynep".
    - **Impact:** Increases the quality of care and reduces the workload on caregivers, while minimizing physical contact, which is particularly beneficial during health crises especially in COVID.
3. **Industrial and Warehouse Operations:**
    - **Application:** The UGV can be employed in warehouses for inventory management, transporting goods, and performing repetitive tasks that are typically labor-intensive. For the known maps, it can move for the desired location using its GUI

- **Impact:** Increases operational efficiency and safety in industrial environments, while reducing operational costs and human error.

4. **Security and Surveillance:**
    - **Application:** Equipped with the necessary sensors, the UGV can patrol predetermined routes for surveillance purposes, monitor secure facilities, or even detect unauthorized intrusions.
    - **Impact:** Enhances security measures with constant monitoring capabilities, reducing the need for human guards in potentially hazardous situations.

5. **Environmental Monitoring:**
    - **Application:** The UGV can be used in environmental conservation projects, such as monitoring wildlife, tracking environmental changes, or detecting pollutants in difficult-to-access areas.
    - **Impact:** Contributes to environmental protection and research by providing valuable data that can help in making informed decisions about conservation strategies.

6. **Assistance for Visually Impaired Individuals:**
    - **Application:** The UGV can be specially adapted to assist visually impaired people by guiding them in various environments such as shopping centers, airports, since they are mapped beforehand. Equipped with its 2D LiDAR and sophisticated navigation algorithms, the UGV can safely guide blind or partially sighted individuals through crowded or complex areas, avoiding obstacles and navigating efficiently.
    - **Impact:** The technology reduces the risk of accidents and injuries, offering users a sense of security and confidence when moving around independently. Moreover, by facilitating easier and safer travel for visually impaired people, the UGV helps to promote greater inclusion in social, educational, and occupational activities, contributing to a more accessible society.

Societal Impacts

- **Job Creation and Transformation:** While automation can lead to the displacement of certain jobs, it also creates opportunities for new roles in robotics maintenance, system development, and more specialized areas that require human oversight.
- **Accessibility and Inclusion:** Automated systems like the UGV can provide services that are particularly beneficial for individuals with disabilities or mobility issues, enhancing their independence and integration into society.

- **Ethical and Privacy Concerns:** The deployment of autonomous systems raises concerns about surveillance, data privacy, and decision-making in critical situations, necessitating clear regulatory frameworks and ethical guidelines.

Overall, the UGV system developed in this project holds promise for transformative impacts across multiple sectors, driving innovation and efficiency while also presenting challenges that need careful management.

Enviromental Effects

The widespread use of autonomous unmanned ground vehicles (UGVs) offers a range of environmental benefits but also poses potential challenges that need careful management. Here’s a detailed examination of the potential environmental effects of UGVs:

Positive Environmental Impacts

1. **Reduction in Greenhouse Gas Emissions:**
    - UGVs, particularly if powered by electric or hybrid systems, can significantly reduce greenhouse gas emissions compared to traditional gasoline or diesel-powered vehicles. This is crucial in efforts to combat climate change.
2. **Decreased Environmental Disturbance:**
    - UGVs designed for tasks like environmental monitoring or agriculture can operate with minimal ecological disturbance. For example, in wildlife areas, these vehicles can collect data or perform tasks without the noise and physical disruption typically caused by human presence.
3. **Efficient Use of Resources:**
    - By optimizing routes and schedules, UGVs can reduce fuel consumption and improve the efficiency of tasks such as logistics and delivery, leading to less environmental impact per task.

Potential Negative Environmental Impacts

1. **Resource Intensive Manufacturing:**
    - The production of UGVs, especially their batteries and electronic components, can be resource-intensive. Mining for lithium, cobalt, and other minerals needed for batteries may cause environmental degradation if not managed responsibly.
2. **Waste Generation:**
    - At the end of their lifecycle, the disposal of UGVs and their components, particularly batteries, could lead to significant electronic waste. Proper recycling and disposal mechanisms need to be in place to mitigate this impact.

3. **Impact on Wildlife:**
    - If not properly managed, UGVs operating in natural habitats could disrupt wildlife, especially if they are used extensively or if their operation is not carefully regulated to avoid sensitive times or areas.
4. **Energy Consumption:**
    - While electric UGVs reduce emissions during operation, the source of electricity (e.g., coal, natural gas, renewables) will significantly influence their overall environmental impact. Regions reliant on fossil fuels for electricity generation may see lesser environmental benefits.

Mitigation Strategies

1. **Sustainable Manufacturing Practices:**
    - Implementing eco-friendly manufacturing practices, using recycled materials, and ensuring that UGVs are built to last can reduce their overall environmental footprint.

2. **Regulated Use in Sensitive Areas:**
    - Enforcing regulations that limit the use of UGVs in ecologically sensitive areas or during critical periods for wildlife can prevent potential disruptions.

By addressing these environmental considerations through responsible practices and regulations, the widespread deployment of UGVs can contribute positively to sustainability goals while minimizing their ecological footprint.

Conclusion

In conclusion, this report has detailed the design of an Autonomous Unmanned Ground Vehicle (UGV) with an emphasis on efficient 2D and 3D mapping utilizing a single 2D LiDAR sensor, and autonomous localization and navigation in a known area. It has systematically explored the various modules—mapping, navigation, mechanical, and system integration—highlighting the substantial progress made towards developing a compact, robust, and user-friendly UGV. Each subsystem has been thoroughly discussed, along with both the conducted tests and those planned for the future. The current status and forthcoming tasks have been outlined, showcasing the UGV's tested performance in meeting its core requirements. The collaborative efforts of the team underscore the UGV's potential to enhance operations across various industries, affirming the project's relevance to current technological trends and market needs. Overall, the project represents a significant advancement in the field of autonomous navigation, poised to deliver valuable insights and capabilities to the industry.

Appendix 1: Test 1

Location	METU Electrical and Electronics Engineering Department Block E – Technician Room
Date	15 May 2024
Time	17.00
Description	Unmanned Ground Vehicle (UGV) 3D Mapping Test
Aim	Testing the 3D mapping ability of the Unmanned Ground Vehicle by ANEBIT where more than one measurement of the room will be taken and merged together.
Expected Outcome	The Unmanned Ground Vehicle by ANEBIT is expected construct a 3D map of the desired environment using the measurements taken from more than one point with an error less than 17 cm considering the objects present.
Participants	Ata Oğuz Tanrıkulu, Doruk Yazıcı, Ebrar Çakmak, Erkin Atay Toka, Yunus Emre Tüysüz

<u>Test Devices, Tools & Software</u>

1. **LiDAR Sensor:** A primary tool used for mapping, localization, and distance measurement in the environment, RPLiDAR A1-M8.
2. **UGV Mobility System:** The system that enhances the mobility of the UGV. Consists of 2 DC motors, L298N motor driver, 2 caster wheels, and a 14.8V battery for powering up the system.
3. **LiDAR Mobility Unit:** The system that is responsible for the mobility of the LiDAR throughout the operation.
4. **User Computer System:** The computer that the user will be using to control the robot wirelessly, through the graphical user interface designed. If the operating system of the user computer is Windows, then "Remote Desktop Connection" software must be installed. If the operating system of the user computer is a variant of Linux, "Remmina" software must be installed beforehand.
5. **Raspberry Pi 4:** The main computer of the UGV which runs the graphical user interface, localization, mapping, and navigation algorithms, and Robot Operating System -ROS-.
6. **Arduino Uno:** The secondary computer of the UGV which serves as an interface between the main computer and the LiDAR Mobility Unit and the UGV Mobility System.
7. **Wi-Fi Router:** A device that will serve as the hotspot between the user computer and the Raspberry Pi 4 mounted on the UGV. The username of the router must be "SUPERONLiNE_WiFi_2870" and the password must be "FTUV7NJYMXNL" to ensure the connection of the Raspberry Pi 4. The user computer must be connected to the same Wi-Fi network.
8. **JetFire 3300mAh 14.8V Battery:** The battery that will be one of the two power sources of the UGV, powering up the UGV Mobility System and the LiDAR Mobility Unit.

9. **Ttec PowerSlim 10000mAh Powerbank:** The battery that will be the second power source of the UGV, powering up the Raspberry Pi 4, RPLiDAR A1-M8, and Arduino Uno.
10. **UGV Chassis:** The 3D printed, 2-level chassis that holds all the UGV together.
11. **Tape Measure:** Used for taking real-life measurements for calibration and comparison.
12. **Data Recording Tools:** Notebooks are used for recording measurements and observations.
13. **Multimeter:** The voltage of the JetFire 14.8V 3300mAh battery should be measured using a multimeter.
14. **Slamtec RoboStudio Software:** To check the properties of the LiDAR such as calibration, health, sample rate, and rotation frequency, the LiDAR is utilized in the mentioned software before a test is conducted.
15. **MATLAB Software:** To create a 3D map from the data acquired, MATLAB will be used.
16. **Chronometer:** A chronometer is used to measure the total mapping time.
17. **Protractor:** An angle measurement device that will be used in the validation of the units.

Calibration of LiDAR: Before starting the tests, using Slamtec RoboStudio, the calibration of the LiDAR is checked. As the Slamtec Robostudio is the official software of the RPLiDAR A1-M8, it checks the calibration, orientation, health status, and many more properties of the LiDAR. If there are no faults seen, the LiDAR can be used in the tests.

Calibration of LiDAR Mobility Unit: The unit composed of electrical and mechanical components should be checked and verified to make sure that it will be working as expected throughout the operation. To check the system, using the software in the UGV, the unit should be rotated to 30, 60, 90, 120, 15030, 60, 90, 120, 150 degrees one-by-one and using a protractor, the angles should be measured at each step. If the measured angle is in the range of ±3 degrees from the required angle at each measurement, then the unit can be approved for test.

Ground Truth Establishment: After the calibration and the checking of the LiDAR, using the live 2D map shown in Slamtec Robostudio, the distance values for 2-3 arbitrary points will be taken from the map and they will be compared with the real values taken via a tape measure. If the measured value is in the 10 cm error range of the LiDAR measurement, the LiDAR is validated.

Validation Checks: During the test, the tape measure will be used to verify the distances measured by the LiDAR system. Adjustments will be made if significant discrepancies are observed.

Test Environment

The test environment is the technician room that is located in the E Block which consists of walls, tables, chairs, garbage bins and closets. The 2D map of the test environment is obtained beforehand with the UGV.

Test Parameters

Table 10. 3D Mapping Test Parameters

Parameter	Range	Step Size	Number of Measurements
Distance between two static points	10-250 cm	60 cm	4

Test Procedure

1. The Wi-Fi router should be turned on with the designated username and password.
2. The USB Type-C to Type-C cable should be connected between the relevant ports of the Raspberry Pi 4 and the ttec PowerSlim Powerbank. The LEDs of the Raspberry Pi 4 should be turned on, which are to be seen from the side.
3. The USB Type-A to USB Micro cable should be connected between the USB 3.0 port of the Raspberry Pi 4 and the green connector of the LiDAR.
4. The USB Type-A to Type-B cable should be connected between the USB 3.0 port of the Raspberry Pi 4 and the Arduino Uno.
5. Connections of the jumper cables at the bottom of the UGV should be checked. If any of the cables have an open connection, they should be connected to the relevant ports.
6. The voltage of the battery should be checked. If the measured voltage is less than 15.00V, the battery should be charged up before any tests.
7. The connection using the XT-60 connectors between the battery and the switch should be made.
8. The switch located at the bottom of the UGV should be turned on. The blue LED should be lid in the bottom of the UGV.
9. The LiDAR should be placed parallel to the ground.
10. The user computer should be connected to the same Wi-Fi router with the designated username and password.
11. From the Wi-Fi router, the IP address that the Raspberry Pi 4 has connected to should be noted.
12. Using the obtained IP address, the wireless communication to the UGV should be established using the “Remote Desktop Connection” or the “Remmina” applications, depending on the operating system of the user computer.
13. Once the remote connection is established, the UGV can be placed in a desired position in the room.
14. The calibration and verification of the units and systems should be completed according to the instructions given.

15. The robot starts from the graphical user interface of the robot, which is located at the desktop of the Raspberry Pi 4 with an icon labeled as "ANEBIT.exe".

16. From the GUI, "Create a New 3D Map"button should be selected.

17. After making sure that no person is in the range of the LiDAR, the chromometer should be started and the "Acquire Data" button should be clicked. It starts the process of rotating the LiDAR and acquiring data. When the data obtaining process finishes, the LiDAR returns back to its original position of being parallel to the ground.

18. Then, returning back to the graphical user interface, "Use SLAM" option must be selected and the "Navigate to New Location" button should be clicked. It should be waited until the Hector SLAM starts running in the background.

19. The UGV should be moved to a new location to acquire a second set of data for 3D mapping. The UGV can be moved to a new location using several keys on the user computer such as "W" to move forward or speed up, "A" to turn left, "S" to move back or slow down, "D" to turn left, and "X" to stop all the movement suddenly. It should be noted that pressing a key several times increases the speed of the motor in the direction that the key is pressed. If the new position is redeemed as appropriate, pressing "ESC" stops the movement algorithm.

20. In the new position, again "Acquire Data" button should be clicked from the graphical user interface. It starts the process of rotating the LiDAR and acquiring data. When the data obtaining process finishes, the LiDAR returns back to its original position of being parallel to the ground.

21. After finishing the data acquisition process, using the "Mozilla Firefox" software that is preinstalled in the Raspberry Pi 4, the files "map_info.txt" and "map_data.txt" files should be sent to the user computer via any software that the user prefers to use.

22. In the user computer, the files "map_info.txt" and "map_data.txt" should be downloaded and placed in the same directory as the "tilt.m" file that generates the 3D map.

23. tilt.m file should be opened in MATLAB, and it should be directly run. After the data processing is completed, the 3D map of the environment can be seen on the user computer. Using MATLAB controls, it can be rotated, zoomed in or out, and positions of the objects can be detected by clicking on the points.

24. To check the performance of the 3D mapping algorithm of the UGV, measurements should be made with tape measure according to the parameters given in Table 1. According to the step sizes, static objects in the area will be selected and the distance between them will be first measured by tape measure, and then it will be compared with the distance measured from the 3D plot obtained from MATLAB. All of the measurements should be recorded.

25. After all of the measurements are recorded according to the parameters in Table 1, the test is completed.

26. To shut down the robot, all of the open terminals and the GUI should be closed, and the Raspberry Pi 4 should be closed from the "Power Off" button located at the top-right of the remote connection screen.

27. After the shutdown of Raspberry Pi 4, it should be disconnected from the ttec PowerSlim Powerbank.

28. The battery should be disconnected using the switch.

29. The tests and the shutdown of the robot are completed.

Test Data

Table 11. 3D Mapping Test Results

Distance to be Measured (cm)	Tape Measure Measurement (cm)	RViz Measurement (cm)
~60	52.5	59.32
~120	111	117.46
~180	188	195.74
~240	223.5	211.89

Results and Discussion

We observed that UGV is capable of generating an accurate and complete 3D map of the environment while obstacle placement and the relative position of the objects on the environment while keeping the error under the determined threshold.

We also see that the 3D mapping lasts approximately 7-10 minutes depending on the number of data acquisition locations.

Appendix 2: Test 2

Location	METU Electrical and Electronics Engineering Department Block E – Technician Room
Date	15 May 2024
Time	17.00
Description	Unmanned Ground Vehicle (UGV) Localization and Navigation Test
Aim	Testing the localization and navigation abilities of the Unmanned Ground Vehicle by ANEBIT in a pre-mapped room.
Expected Outcome	The Unmanned Ground Vehicle by ANEBIT is expected to localize itself in a pre-mapped room with an error less than 10cm, and it can autonomously navigate the designated point with an error less than 15 cm.
Participants	Ata Oğuz Tanrıkulu, Doruk Yazıcı, Ebrar Çakmak, Erkin Atay Toka, Yunus Emre Tüysüz

Test Devices, Tools & Software

1. **LiDAR Sensor:** A primary tool used for mapping, localization, and distance measurement in the environment, RPLiDAR A1-M8.
2. **UGV Mobility System:** The system that enhances the mobility of the UGV. Consists of 2 DC motors, L298N motor driver, 2 caster wheels, and a 14.8V battery for powering up the system.
3. **LiDAR Mobility Unit:** The system that is responsible for the mobility of the LiDAR throughout the operation.
4. **User Computer System:** The computer that the user will be using to control the robot wirelessly, through the graphical user interface designed. If the operating system of the user computer is Windows, then "Remote Desktop Connection" software must be installed. If the operating system of the user computer is a variant of Linux, "Remmina" software must be installed beforehand.
5. **Raspberry Pi 4:** The main computer of the UGV which runs the graphical user interface, localization, mapping, and navigation algorithms, and Robot Operating System -ROS-.
6. **Arduino Uno:** The secondary computer of the UGV which serves as an interface between the main computer and the LiDAR Mobility Unit and the UGV Mobility System.
7. **Wi-Fi Router:** A device that will serve as the hotspot between the user computer and the Raspberry Pi 4 mounted on the UGV. The username of the router must be "SUPERONLiNE_WiFi_2870" and the password must be "FTUV7NJYMXNL" to ensure the connection of the Raspberry Pi 4. The user computer must be connected to the same Wi-Fi network.
8. **JetFire 3300mAh 14.8V Battery:** The battery that will be one of the two power sources of the UGV, powering up the UGV Mobility System and the LiDAR Mobility Unit.
9. **Ttec PowerSlim 10000mAh Powerbank:** The battery that will be the second power source of the UGV, powering up the Raspberry Pi 4, RPLiDAR A1-M8, and Arduino Uno.
10. **UGV Chassis:** The 3D printed, 2-level chassis that holds all the UGV together.
11. **Tape Measure:** Used for taking real-life measurements for calibration and comparison.
12. **Data Recording Tools:** Notebooks are used for recording measurements and observations.

13. **Multimeter:** The voltage of the JetFire 14.8V 3300mAh battery should be measured using a multimeter.
14. **Slamtec RoboStudio Software:** To check the properties of the LiDAR such as calibration, health, sample rate, and rotation frequency, the LiDAR is utilized in the mentioned software before a test is conducted.
15. **Protractor:** An angle measurement device that will be used in the validation of the units.

Calibration of LiDAR: Before starting the tests, using Slamtec RoboStudio, the calibration of the LiDAR is checked. As the Slamtec Robostudio is the official software of the RPLiDAR A1-M8, it checks the calibration, orientation, health status, and many more properties of the LiDAR. If there are no faults seen, the LiDAR can be used in the tests.

Calibration of LiDAR Mobility Unit: The unit composed of electrical and mechanical components should be checked and verified to make sure that it will be working as expected throughout the operation. To check the system, using the software in the UGV, the unit should be rotated to $30, 60, 90, 120, 150$ degrees one-by-one and using a protractor, the angles should be measured at each step. If the measured angle is in the range of ± 3 degrees from the required angle at each measurement, then the unit can be approved for test.

Ground Truth Establishment: After the calibration and the checking of the LiDAR, using the live 2D map shown in Slamtec Robostudio, the distance values for 2-3 arbitrary points will be taken from the map and they will be compared with the real values taken via a tape measure. If the measured value is in the 10 cm error range of the LiDAR measurement, the LiDAR is validated.

Validation Checks: During the test, the tape measure will be used to verify the distances measured by the LiDAR system. Adjustments will be made if significant discrepancies are observed.

Test Environment

The test environment is the technician room that is located in the E Block which consists of walls, tables, chairs, garbage bins and closets; the environment can be seen in Figure 21. The 2D map of the test environment is obtained beforehand with the UGV.

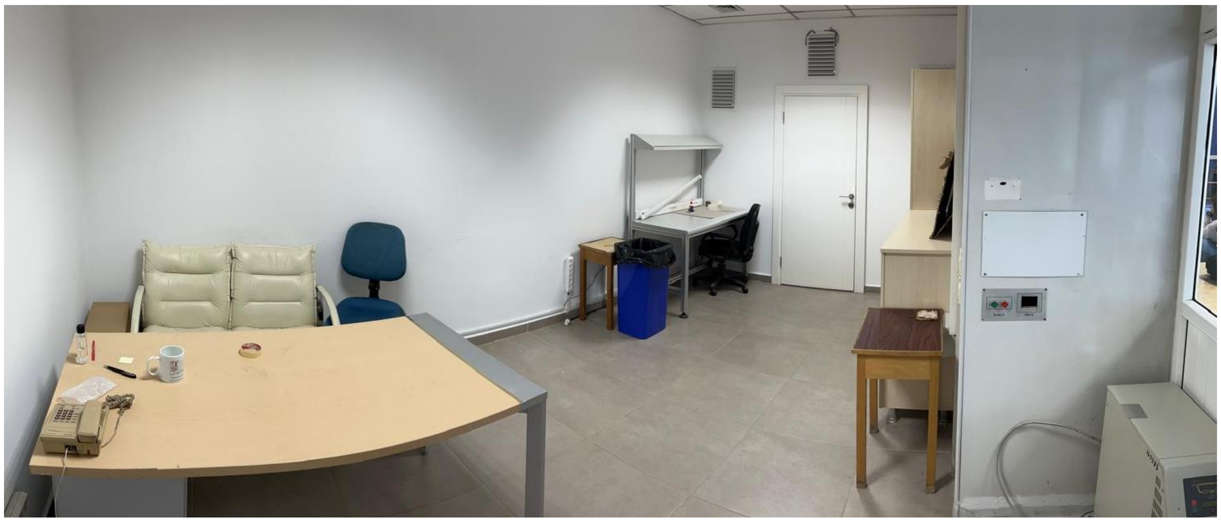


Figure 21. An exemplar test environment for the localization and navigation tests

Test Parameters

Table 12. Localization Test Parameters

Parameter	Range	Step Size	Number of Measurements
Distance of UGV from two obstacles or walls	10-250 cm	80 cm	3

Table 13. Navigation Test Parameters

Parameter	Range	Step Size	Number of Measurements
Distance between the labeled point and navigated point	10-250 cm	80 cm	3

Test Procedure

1. The Wi-Fi router should be turned on with the designated username and password.
2. The USB Type-C to Type-C cable should be connected between the relevant ports of the Raspberry Pi 4 and the ttec PowerSlim Powerbank. The LEDs of the Raspberry Pi 4 should be turned on, which are to be seen from the side.
3. The USB Type-A to USB Micro cable should be connected between the USB 3.0 port of the Raspberry Pi 4 and the green connector of the LiDAR.
4. The USB Type-A to Type-B cable should be connected between the USB 3.0 port of the Raspberry Pi 4 and the Arduino Uno.
5. Connections of the jumper cables at the bottom of the UGV should be checked. If any of the cables have an open connection, they should be connected to the relevant ports.
6. The voltage of the battery should be checked. If the measured voltage is less than 15.00V, the battery should be charged up before any tests.
7. The connection using the XT-60 connectors between the battery and the switch should be made.
8. The switch located at the bottom of the UGV should be turned on. The blue LED should be lid in the bottom of the UGV.
9. The LiDAR must be placed parallel to the ground.
10. The user computer should be connected to the same Wi-Fi router with the designated username and password.
11. From the Wi-Fi router, the IP address that the Raspberry Pi 4 has connected to should be noted.

12. Using the obtained IP address, the wireless communication to the UGV should be established using the “Remote Desktop Connection” or the “Remmina” applications, depending on the operating system of the user computer.

13. Once the remote connection is established, the UGV can be placed in a desired position in the room.

14. The calibration and verification of the units and systems should be completed according to the instructions given.

15. The UGV starts from the graphical user interface of the robot, which is located at the desktop of the Raspberry Pi 4 with an icon labeled as “ANEBIT.exe”.

16. If the 2D map of the room is not obtained beforehand, the “Create a 2D Map via Hector SLAM” option should be selected. In the new screen that comes up, “Save” should be selected, and then “Navigate to New Point” should be clicked. Then, it should be waited until the Rviz logo shows up and the actual Rviz application starts. In the Rviz graphical user interface, an incomplete map of the environment can be seen. To complete the map, the robot should be moved in the environment using several keys on the user computer such as “W” to move forward or speed up, “A” to turn left, “S” to move back or slow down, “D” to turn left, and “X” to stop all the movement suddenly. It should be noted that pressing a key several times increases the speed of the motor in the direction that the key is pressed. After some movement, the map should be completed. When the user decides that the map is complete, pressing the “ESC” key closes down the Rviz, saves the current map for further use, and returns back to the GUI. Lastly, “Return to the Main Page” button should be clicked to go to the main page of the graphical user interface. If the map obtained does not satisfy the user, this step should be conducted again.

17. The robot is placed in an arbitrary position in the room.

18. From the GUI, “Navigate” should be selected.

19. In the new screen, the 2D map is seen on which the labeling can be made. To label a selected obstacle, user should zoom in to the obstacle, holding the left mouse button the obstacle should be selected, the name of the obstacle should be entered in the box named as “Enter label name”, and the button “Label Selected” should be clicked. If everything is done right, the label should be placed on the obstacle with the coordinates. To remove a label, the desired label should be selected from the label list and the “Remove Selected Labels” button should be clicked.

20. To change the position of a label, the desired label should be selected from the list of the labels placed on the top right of the screen. After selecting the label, the “Set Label” button should be clicked, and the new desired position of the label should be clicked on the map. Then, the label should be again selected from the label list and the “Update Label Position” button should be clicked. The label position should be changed on the map.

21. To start the localization and navigation process, “Start Autonomous Navigation” button should be clicked. Then, it should be waited until the Rviz logo shows up and the actual Rviz application starts. In the Rviz graphical user interface, the 2D map obtained beforehand and the current LiDAR scan should be seen.

22. To make the robot localize itself, the algorithm needs some movement and rotation in any of the possible directions. Like in the 2D map creation case, the robot moves using the same controls. Using the controls the UGV should be rotated minimally until the map and scan look alike. To use the movement software, the "Start Manual Navigation" button should be clicked and wait until the software starts. Then, to match the scan with the map, the UGV should be moved forward and backward until the scan matches with the map, to complete the localization. "ESC" key should be pressed on the keyboard to stop the manual navigation.

23. To check the performance of the localization algorithm of the UGV, after the localization of the UGV is completed from the Rviz, measurements should be made with tape measure. First, two walls or obstacles located nearly 1 meter away from the UGV must be selected and their distance to the top of the LiDAR should be measured perpendicularly. Then, the same distance should be measured from the Rviz tools, and all these measurements must be recorded. The same process should be repeated by increasing the step sizes.

24. To navigate to a label, the desired label should be selected from the label list located at the top right corner of the GUI, and the "Go to Label" button should be clicked.

25. To navigate to an arbitrary position, using the graphical user interface of Rviz, "2D Nav Goal" button should be selected, and the desired position should be marked on the map.

26. To check the performance of the navigation algorithm of the UGV, after the goals are selected and the UGV moves to the goal points, measurements should be made with tape measure. First, two walls or obstacles located nearly 1 meter away from the UGV must be selected and their distance to the top of the LiDAR should be measured perpendicularly. Then, using Rviz tools, the same measurements should be made for the labeled or the "2D Nav Goal" points, and all of these measurements must be recorded. The same process should be repeated by increasing the step sizes. The distance between the navigated points and the goal points should be recorded.

27. Ater measurements are done, an extra obstacle can be placed in the environment to test dynamic obstacle avoidance property of the UGV. For example, a trash bin present in the laboratory can be placed in the environment. Then, an arbitrary navigation goal can be given to the UGV.

28. If any other navigation goals are present, they can be sent to the robot using the GUI or the Rviz.

29. After the tests are completed, the Rviz can be closed using the "Stop Autonomous Navigation" button located in the GUI.

30. To shut down the robot, all of the open terminals and the GUI should be closed, and the Raspberry Pi 4 should be closed from the "Power Off" button located at the top-right of the remote connection screen.

31. After the shutdown of Raspberry Pi 4, it should be disconnected from the ttec PowerSlim Powerbank.

32. The battery should be disconnected using the switch.

33. The tests and the shutdown of the robot are completed.

Test Data

Table 14. Localization Test Results

Distance to be Measured (cm)	Tape Measure Measurement (cm)	RViz Measurement (cm)
~80	74.5	78.34
~160	167	172.12
~240	238.5	244.45

Table 15. Navigation Test Results

Distance to be Measured (cm)	Tape Measure Measurement (cm)	RViz Measurement (cm)
~80	83.5	77.46
~160	171	180.71
~240	253	264.49

Results and Discussion

We see that the UGV is capable of localize itself on the given environment with the required threshold, while it can navigate autonomously on the environment keeping the min distance with the target location in the determined range.

However, we observed that the GUI is unable to avoid dynamic obstacles if they are present on the environment.

Appendix 3: Test 3

Location	METU Electrical and Electronics Engineering Department Block E – Technician Room
Date	15 May 2024
Time	6 PM
Description	Power Consumption Test
Aim	To show the power consumption during navigation and mapping
Expected Outcome	To show that power resources are capable of delivering power to the UGV for a sufficient duration.
Participants	Ata Oğuz Tanrıkulu, Erkin Atay Toka, Doruk Yazıcı, Ebrar Çakmak, Yunus Emre Tüysüz

Test Devices & Tools

1. Prototype Vehicle: The tested vehicle.

2. JetFire 3300mAh 14.8V Battery: The battery is the power supply to the main and Lidar Mobility Unit

3. Ttec PowerSlim 10000mAh Powerbank: The powerbank supplies power to Aurdino Uno, Raspberry Pi and RpLiDAR A1-N8. The percentage of the state of charge of the powerbank is visualized in its screen.

4. User Computer System: The computer that the user will be using to control the robot wirelessly, through the graphical user interface designed. If the operating system of the user computer is Windows, then "Remote Desktop Connection" software must be installed. If the operating system of the user computer is a variant of Linux, "Remmina" software must be installed beforehand.

5.Data Recording Tools: Notebooks are used for recording measurements and observations

6. Multimeter: A multimeter is used to measure the voltage of the Lipo Battery.

7. Choronometer: To measure the time elapsed during mapping and nacigation procedured.

Ground Truth Establishment:
Voltage measurements taken from the multimeter and state of charge readings of the powerbank are taken as ground truth.

Test Environment

The test environment is the technician room that is located in the E Block which consists of walls, tables, chairs, garbage bins and closets; the environment can be seen in Figure 22. The 2D map of the test environment is obtained beforehand with the UGV.

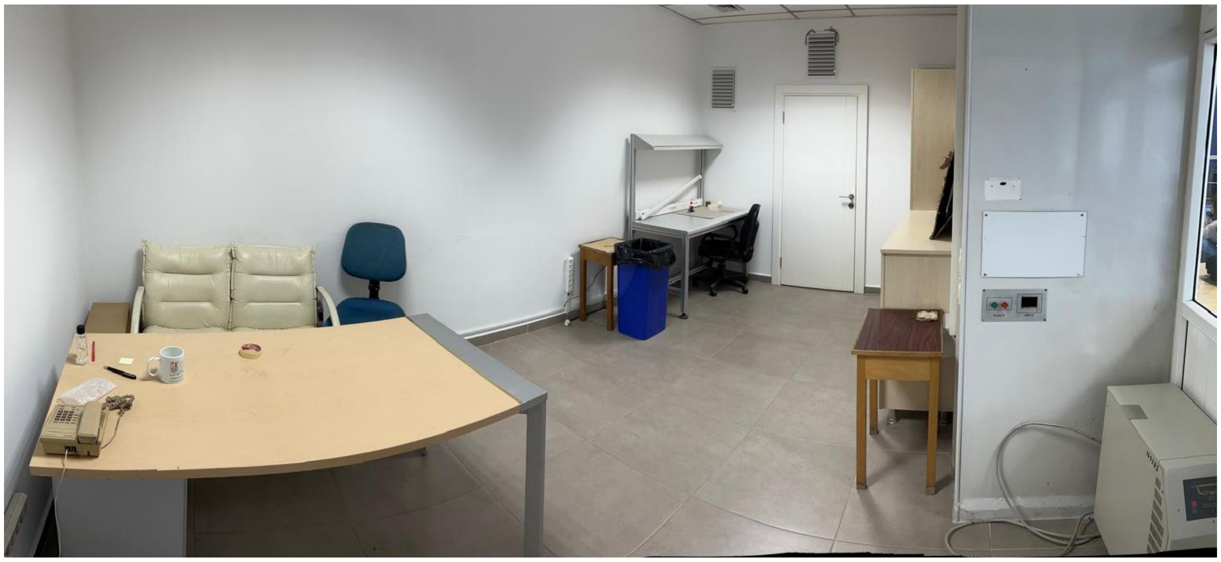


Figure 22. An exemplar test environment for the localization and navigation tests

Test Procedure

1. At first the power consumption during the navigation will be inspected. Therefore, complete the steps from 1 to 5 in Test Document 1.
2. Measure the voltage of the Lipo Battery via multimeter. If the measured voltage is less than 15 V, recharge the battery. If not record the battery voltage.
3. Record the power bank charge percentage.
4. Continue to the navigation procedure from step 7 to 29.
5. Measure and record the voltage of the Lipo Battery via multimeter.
6. Record the power bank charge percentage.
7. Now, it is time to measure the power consumption during the mapping procedure. Implement the steps from 13 to 30.
8. Measure and record the voltage of the Lipo Battery via multimeter.
9. Record the power bank charge percentage.
10. To shut down the robot, all of the open terminals and the GUI should be closed, and the Raspberry Pi 4 should be closed from the "Power Off" button located at the top-right of the remote connection screen.
11. After the shutdown of Raspberry Pi 4, it should be disconnected from the ttec PowerSlim Powerbank.
12. The battery should be disconnected using the switch.
13. The tests and the shutdown of the robot are completed.

Test Data

Table 16. Power Consumption Test Results

	Battery Voltage (V)	**Powerbank Charge Percentage (%)**	**Time Elapsed (min)**
Initial State	16.4	100	
After Navigation	16.35	90	9
After Mapping	16.3	84	7

Data Analysis

The operation duration without Lipo Battery to running out can be calculated as follows.

$$t_{Lipo\ Battery} = \frac{Maximum\ allowable\ voltage\ drop\ * \Delta t_{procedure}}{Voltage\ Drop\ in\ one\ Procedure}$$

Maximum allowable voltage drop is the difference between the maximum battery voltage (16.8V) to minimum allowed Lipo battery voltage (15.6V).
The operation duration without Power Bank to running out can be calculated as follows.

$$t_{power\ bank} = \frac{100\ * \Delta t_{procedure}}{Charge\ Percentage\ change\ in\ one\ Procedure}$$

Navigation:

$$t_{Lipo\ Battery} = \frac{1.2\ * 9}{0.05} = 216\ min$$

$$t_{power\ bank} = \frac{100\ * 9}{100 - 90} = 90\ min$$

$$\#procedures = \frac{min(t_{Lipo\ Battery}, t_{power\ bank})}{\Delta t_{procedure}} = 10$$

Mapping:

$$t_{Lipo\ Battery} = \frac{1.2\ * 7}{0.1} = 168\ min$$

$$t_{power\ bank} = \frac{100\ * 7}{90 - 84} = 116\ min$$

$$\#procedures = \frac{min(t_{Lipo\ Battery}, t_{power\ bank})}{\Delta t_{procedure}} = 16$$

Results and Discussion

It can be seen that the determining supply is the power bank for both mapping and navigation procedures. It can be seen that navigation and mapping can be repeated 10 and 16 times respectively.

Appendix 4: Objective Tree

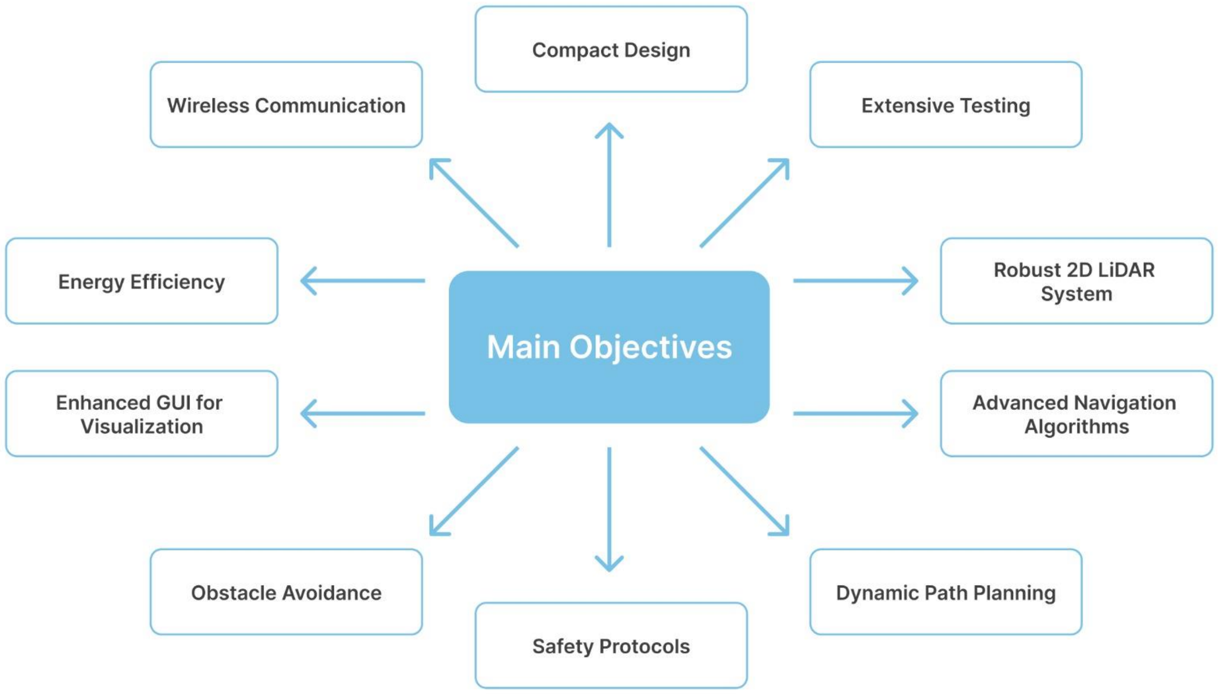
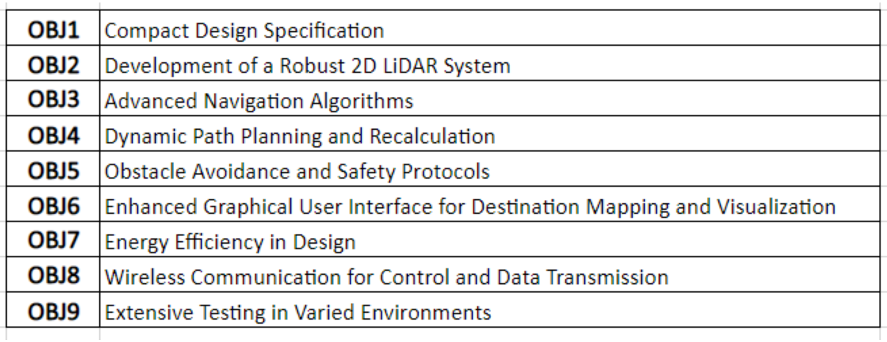
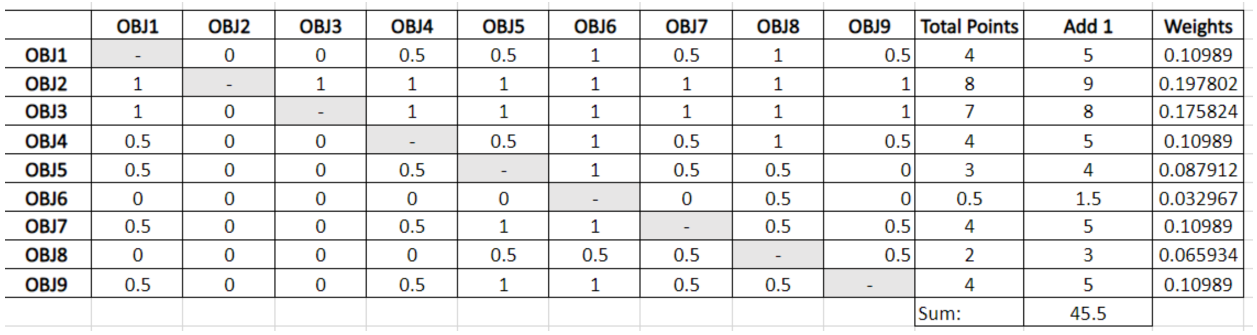


Figure 23. Main Objectives



OBJ1	Compact Design Specification
OBJ2	Development of a Robust 2D LiDAR System
OBJ3	Advanced Navigation Algorithms
OBJ4	Dynamic Path Planning and Recalculation
OBJ5	Obstacle Avoidance and Safety Protocols
OBJ6	Enhanced Graphical User Interface for Destination Mapping and Visualization
OBJ7	Energy Efficiency in Design
OBJ8	Wireless Communication for Control and Data Transmission
OBJ9	Extensive Testing in Varied Environments

Figure 24. Main Objectives



	OBJ1	**OBJ2**	**OBJ3**	**OBJ4**	**OBJ5**	**OBJ6**	**OBJ7**	**OBJ8**	**OBJ9**	**Total Points**	**Add 1**	**Weights**
OBJ1	-	0	0	0.5	0.5	1	0.5	1	0.5	4	5	0.10989
OBJ2	1	-	1	1	1	1	1	1	1	8	9	0.197802
OBJ3	1	0	-	1	1	1	1	1	1	7	8	0.175824
OBJ4	0.5	0	0	-	0.5	1	0.5	1	0.5	4	5	0.10989
OBJ5	0.5	0	0	0.5	-	1	0.5	0.5	0	3	4	0.087912
OBJ6	0	0	0	0	0	-	0	0.5	0	0.5	1.5	0.032967
OBJ7	0.5	0	0	0.5	1	1	-	0.5	0.5	4	5	0.10989
OBJ8	0	0	0	0	0.5	0.5	0.5	-	0.5	2	3	0.065934
OBJ9	0.5	0	0	0.5	1	1	0.5	0.5	-	4	5	0.10989
										Sum:	45.5	

Figure 25. Weighted Objectives

Appendix 5: User Manual of the Unmanned Ground Vehicle

1. Introduction

Welcome to your new Autonomous Unmanned Ground Vehicle (UGV) by ANEBIT. This guide provides essential information for the operation and maintenance of the UGV, designed to navigate autonomously and perform complex mapping tasks. Moreover, the Localization and 3D Mapping Quick Starts Documents can be found in the guide.

2. Safety Information

- Ensure that the UGV is used in environments free of water and excessive dust.
- Keep clear of the UGV’s moving parts while in operation.
- Do not handle the UGV’s electrical components with wet hands.
- Make sure that all electrical connections are made and staying properly.

3. Start Guide

- The Wi-Fi router should be turned on with the designated username and password.
- The USB Type-C to Type-C cable should be connected between the relevant ports of the Raspberry Pi 4 and the ttec PowerSlim Powerbank. The LEDs of the Raspberry Pi 4 should be turned on, which are to be seen from the side.
- The USB Type-A to USB Micro cable should be connected between the USB 3.0 port of the Raspberry Pi 4 and the green connector of the LiDAR.
- The USB Type-A to Type-B cable should be connected between the USB 3.0 port of the Raspberry Pi 4 and the Arduino Uno.
- Connections of the jumper cables at the bottom of the UGV should be checked. If any of the cables have an open connection, they should be connected to the relevant ports.
- The voltage of the battery should be checked. If the measured voltage is less than 15.00V, the battery should be charged up before any tests.
- The connection using the XT-60 connectors between the battery and the switch should be made.
- The switch located at the bottom of the UGV should be turned on. The blue LED should be lid in the bottom of the UGV.
- The LiDAR must be placed parallel to the ground.
- The user computer should be connected to the same Wi-Fi router with the designated username and password.
- From the Wi-Fi router, the IP address that the Raspberry Pi 4 has connected to should be noted.
- Using the obtained IP address, the wireless communication to the UGV should be established using the “Remote Desktop Connection” or the “Remmina” applications, depending on the operating system of the user computer.

- Once the remote connection is established, the UGV can be placed in a desired position in the room.
- The UGV starts from the graphical user interface of the robot, which is located at the desktop of the Raspberry Pi 4 with an icon labeled as "ANEBIT.exe".

4. Navigation and Mapping

- 2D Mapping: Useful for simple navigation tasks and obstacle avoidance.
- 3D Mapping: Best suited for complex environments where detailed spatial awareness is required.
- Operation: Select the mapping mode on your control interface, direct the UGV to start the mapping process, and monitor progress on your screen.

5. Maintaining and Troubleshooting

Routine Checks

- Check battery levels before each use.
- Inspect wheels and sensors for dirt and debris.

Common Issues and Solutions

- UGV Does Not Start: Ensure the battery is charged and properly connected.
- Mapping Errors: Reset the mapping system through the interface.

6. Technical Specifications

- Dimensions: 30 cm x 30 cm x 24 cm
- Weight: 3 kg
- Battery Life: Up to 30 minutes of continuous use
- Mapping Accuracy: Within 10 cm

7. Customer Support

For any issues not resolved by this guide, please contact our support team at [annebitti@yahoo.com] or call +905076275355 or send a message to [anebit.official] using Instagram.

8. Localization and Navigation for Autonomous UGV

Introduction

This section of the manual details the steps required to perform localization and navigation tasks using the ANEBIT Autonomous UGV. It assumes the environment has been pre-mapped using the UGV's LiDAR system.

Requirements

- A pre-mapped room.
- UGV equipped with LiDAR, Raspberry Pi 4, and Arduino Uno.

- User computer with "Remote Desktop Connection" or "Remmina" installed.

Connecting and Setting Up

1. **Ensure Connectivity:**
    - Ensure the UGV and the user computer are connected to the same Wi-Fi network.
    - Turn on the UGV and confirm that the Raspberry Pi 4 is active (LED indicators should be on).
2. **Establish Remote Connection:**
    - From the user computer, use the designated IP address to connect remotely to the Raspberry Pi 4 using "Remote Desktop Connection" or "Remmina".

Performing Localization

1. **Launch Interface:**
    - Start the graphical user interface by double-clicking the "ANEBIT.exe" icon on the desktop of Raspberry Pi 4.
2. **Position the UGV:**
    - Place the UGV at a desired starting position within the pre-mapped room.
3. **Start Localization:**
    - From the GUI, select "Navigate".
    - Click "Start Autonomous Navigation" and wait for the Rviz application to start.
    - To help the UGV localize itself within the map, initiate manual movement using the controls (W, A, S, D, X) until the UGV's real-time scan aligns with the pre-mapped environment. Minimal rotation and movement may be required.
    - Confirm localization by matching the LiDAR scans with the environment in the Rviz interface.

Setting Navigation Goals

1. **Labeling and Navigating to Points:**
    - To label an obstacle or point of interest on the map within the Rviz interface, select the obstacle, enter a name in the "Enter label name" box, and click "Label Selected".
    - To navigate to a labeled point, select the label from the label list and click "Go to Label".
2. **Navigating to Arbitrary Points:**

- In the Rviz interface, select "2D Nav Goal" and click on the desired position on the map to set a new navigation target.

3. **Monitoring and Adjustments:**

   - Monitor the UGV's movement towards the target. Adjustments can be made in real-time if the UGV deviates or encounters obstacles.

Completing the Navigation

1. **End Navigation Session:**

   - Once the navigation tasks are complete, click "Stop Autonomous Navigation" in the GUI to end the session.
   - Close all open applications and shut down the Raspberry Pi 4 using the "Power Off" button.

2. **Disconnect and Power Down:**

   - Shutdown the UGV from the Remote Desktop Connection.
   - Disconnect the UGV from the power sources.
   - Ensure all systems are turned off and the UGV is secured.

9. 3D Mapping With Autonomous UGV

Introduction

This section provides detailed instructions for creating a 3D map of an environment using the ANEBIT Autonomous UGV equipped with LiDAR technology.

Setup and Initial Configuration

1. **Ensure Connectivity:**

   - Confirm that the UGV and the user computer are connected to the same Wi-Fi network.
   - Connect remotely to the UGV using "Remote Desktop Connection" or "Remmina", depending on the user computer's operating system.

2. **Launch Interface:**

   - Open the UGV interface by double-clicking the "ANEBIT.exe" icon on the desktop of the Raspberry Pi 4.

Creating a New 3D Map

1. **Start 3D Mapping:**

   - From the UGV's GUI, select "Create a New 3D Map".
   - Ensure the area around the UGV is clear to avoid any interference with the LiDAR scan.

2. **Data Acquisition:**

    - Click "Acquire Data" to start the LiDAR data collection. Ensure the LiDAR is initially positioned parallel to the ground.
    - Move the UGV to various locations within the room to capture data from multiple perspectives. Use keyboard controls (W, A, S, D, X) for movement and press "ESC" to stop the UGV in a new location.
    - At each new position, click "Acquire Data" again to continue gathering additional mapping data.

Processing and Visualizing 3D Data

1. **Transfer Data Files:**

    - After completing data collection, use "Mozilla Firefox" on the Raspberry Pi 4 to send "map_info.txt" and "map_data.txt" files to the user computer.

2. **Generate 3D Map:**

    - On the user computer, save the received files in the same directory as the MATLAB script "tilt.m".
    - Run the "tilt.m" script in MATLAB to process the data and generate the 3D map of the environment.

3. **Interact with 3D Map:**

    - Use MATLAB to view and interact with the 3D map. You can rotate, zoom, and explore different views of the mapped environment to analyze the layout and object placements.

Completion and Shutdown

1. **Conclude Mapping Session:**

    - Once all data has been captured and validated, close all applications on the Raspberry Pi 4.
    - Use the "Power Off" button in the remote connection interface to safely shut down the Raspberry Pi 4.

2. **Disconnect and Power Down:**

    - Shutdown UGV from Remote Desktop Connection.
    - Disconnect the UGV from all power sources and ensure that the system is completely powered down.

References

[1] M. A. Markom, A. H. Adom, E. S. M. M. Tan, S. A. A. Shukor, N. A. Rahim and A. Y. M. Shakaff, "A mapping mobile robot using RP Lidar scanner," *2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS),* Langkawi, Malaysia, 2015, pp. 87-92, doi: 10.1109/IRIS.2015.7451592. keywords: {Robot sensing systems;Reliability;laser rangefinder;RP Lidar;mapping;pre-processing;mobile robot},