GAZİ UNIVERSITY

EEE492 – EE Engineering Design II

Hasan AĞAÇAYAK 181110001
Mesut ARIKAN 181110012
Mustafa ATEŞ 181112005
Serkan BAŞER 181110016
Yavuz AVCI 201110610

CONTENT

1. Executive Summary

Our project focuses on the design and development of an advanced autonomous land vehicle. In our report, we talked in detail about the requirements for this vehicle and the solutions we found to meet these requirements. With the tests we carried out while developing the solutions, we made them comply with the necessary standards for possible commercialization situations in the future. We also discussed what we could add for ease of use and made some improvements. In addition, we also have a manual driving mode for user intervention to problems that may occur during autonomous driving. The user can provide this change with minor changes and can drive with a joystick. Ultimately, the resulting vehicle, LiDAR for navigation, and combining image processing and image processing algorithms for object detection, can autonomously navigate and adjust its position. The user interface provides real-time camera feeds and map visualization. In its mechanical design, the exterior of the vehicle is closed.

With its versatile mechanics and ability to perform tasks such as agriculture and transportation, the vehicle demonstrates innovation and adaptability. Rigorous testing ensures safety and compliance with standards. This autonomous vehicle paves the way for a smarter future by contributing to efficiency and technological advancement in various industries.

We have developed solution techniques for the 5 requirements given to us for our project. The first of these requirements is an autonomous driving, we can examine the processes we have developed and implemented to realize autonomous driving under four main headings. The first one is ROS (Robot Operating System) framework We developed our own software by creating nodes that allow communication among themselves and Enabled communication between control stations and vehicles. For the second operation we utilize a Raspberry Pi 4B as the main computer for our vehicle, along with an adept HAT serving as the motor driver and power distributor. After that, we carried out mapping and localization processes. We used the Hector mapping algorithm with SLAM for these operations with the LiDAR sensor. Hector mapping is an algorithm that allows us to actively map and localize. Finally, to perform autonomous navigation we utilized move_base package and implemented to our robot. move_base is a high-level navigation system that combines localization, obstacle avoidance, and path planning to enable autonomous robot navigation. After performing these operations, our vehicle has now become autonomous.The second is to give inputs for the route operation of this vehicle with the Rviz interface, these operations can be entered by the user. Using our image processing algorithm, the vehicle detects objects encountered based on their color, determines whether they are friend or foe, and adjusts. Initially, the real-time image data captured by the vehicle's camera is sent to Raspberry Pi. Our fourth request is the realization of control processes with artificial intelligence. The techniques we use for the mapping and autonomous driving we have created are artificial intelligence supported algorithms. These algorithms, on the other hand, perform artificial intelligence supported operations in order to perform global planner and local planner operations within the move_base package.In the last case, the exterior of the vehicle is closed in accordance with this in our mechanical design.

2. Introduction

While designing the advanced autonomous land vehicle (GAZİ), which is our project, our aim was to distinguish between friends and foes with the help of image processing, to easily specify a target and communicate with the vehicle through the user interface, and to obtain real-time output data, in addition to being an autonomous vehicle. It is to create a vehicle with advanced features and not to compromise on important issues such as safety and efficiency. As a result of our research at the very beginning of our project, we determined the Engineering requirements of our vehicle as autonomous, image processing, mapping, user interface creation and mechanical design. In addition, we determined our marketing requirements as easy to use, low cost, accessibility. Through extensive research, we have come across numerous initiatives and articles that highlight the ongoing progress in this field. Notable projects like Delivers.ai, Orca, Ratel, and Boa have emerged, focusing primarily on product transportation. These projects have inspired us to incorporate autonomous features into our own endeavor. In our research, we found products used in the industry such as these, and we also had the opportunity to examine the prototype vehicles that competed in Teknofest and were planned to be used in areas such as agriculture that were successful. Inspired by these examples, we succeeded in overcoming the difficulties and obstacles we encountered during the development of our vehicle.

2.1. Marketing Requirement

2.1.1. Easy to Use

The destination location information can be easily inputted through the user interface we have developed. The camera data and map information are displayed clearly on the screen.

2.1.2. Low Cost

By utilizing software, we optimize more affordable hardware components, thereby reducing the cost of the vehicle. This allows us to present an affordable model that is within reach. In addition, we have designed the vehicle to be as compact as possible and easily modifiable. This approach allows for cost reductions when implementing new additions or modifications.

2.1.3. Availability

The vehicle is capable of performing tasks such as automatic seed planting, plant maintenance, and harvesting in agricultural fields. It also can perform automatic shipping and logistics operations. Additionally, with its autonomous driving features, it can be integrated into future traffic systems. As seen, the vehicle can easily adapt to different use cases with minor adjustments when needed.

2.2. Proposed Concepts:

We can examine the conceptual design in 3 titles (Level 0, Level1, Level 2).

2.2.1. Level 0 Design

The Level 0 design, in which we examined the basic logic of our vehicle as a simple conceptual design, is as follows:

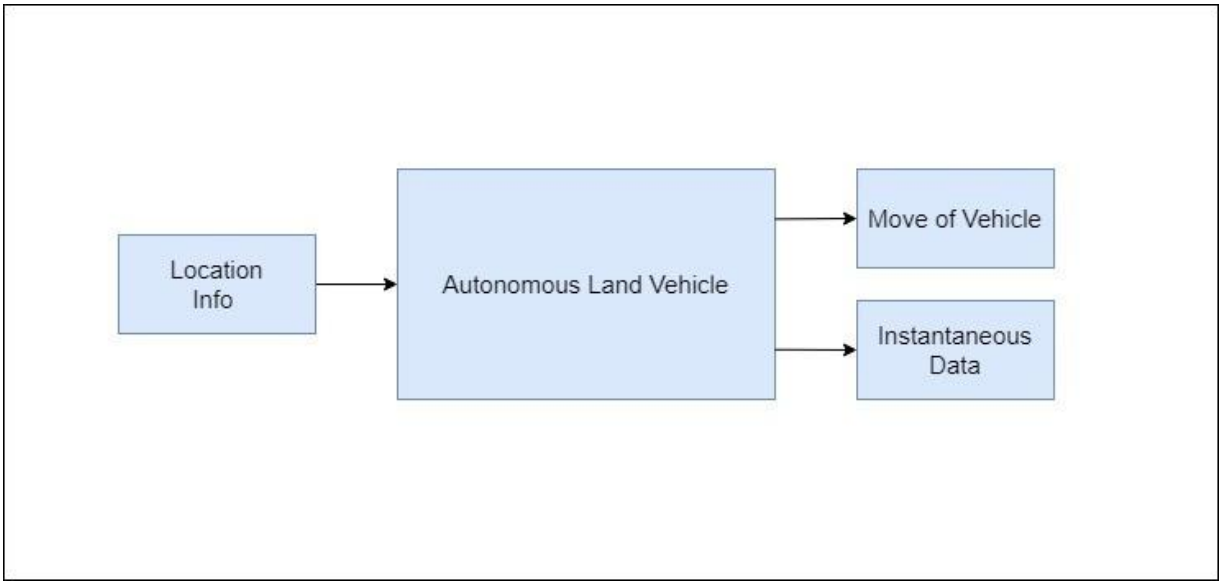

Figure 1

2.2.2. Level 1 Design

Level 1, which is the design process in which we examine our vehicle in more detail compared to Level 0, and divide the operations to be performed into units, is as follows:

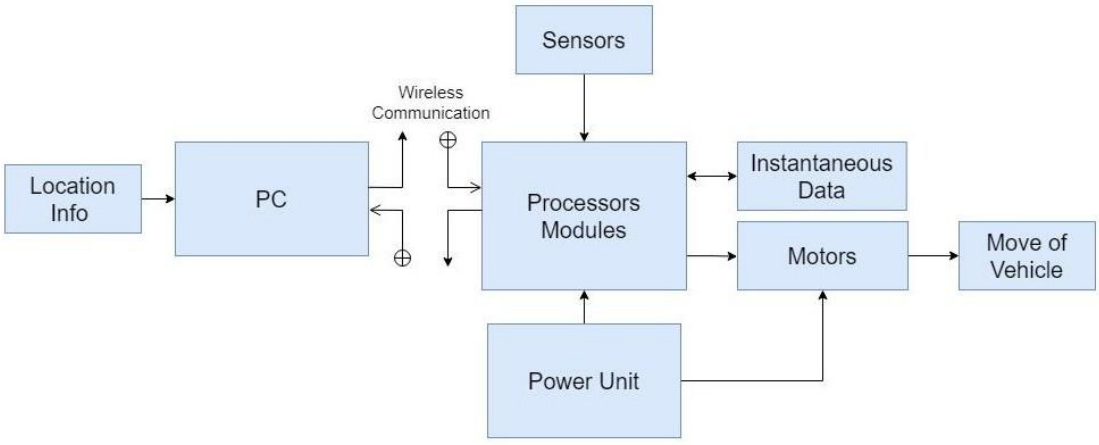

Figure 2

2.2.3. Level 2 Design

Our Level 2 design has changed compared to the previous period. In Level 2 design, the user first enters the location info and then sends the received info to the vehicle via wireless. Our vehicle processes this information in Raspberry. It then sends the data from the LIDAR to the Raspberry computer for mapping. Our Raspberry computer sends the command, which is the result of many processes, to the engine and the movement of our vehicle is ensured. Apart from these, it sends the images we receive from the camera to the Raspberry computer for image processing. It processes and makes sense of the data. Finally, there is instant data that communicates with Raspberry in our Level 2 diagram. With this data, mapping and positioning operations are performed.

Level 2, where we designed our vehicle in the most detailed way, is as follows:

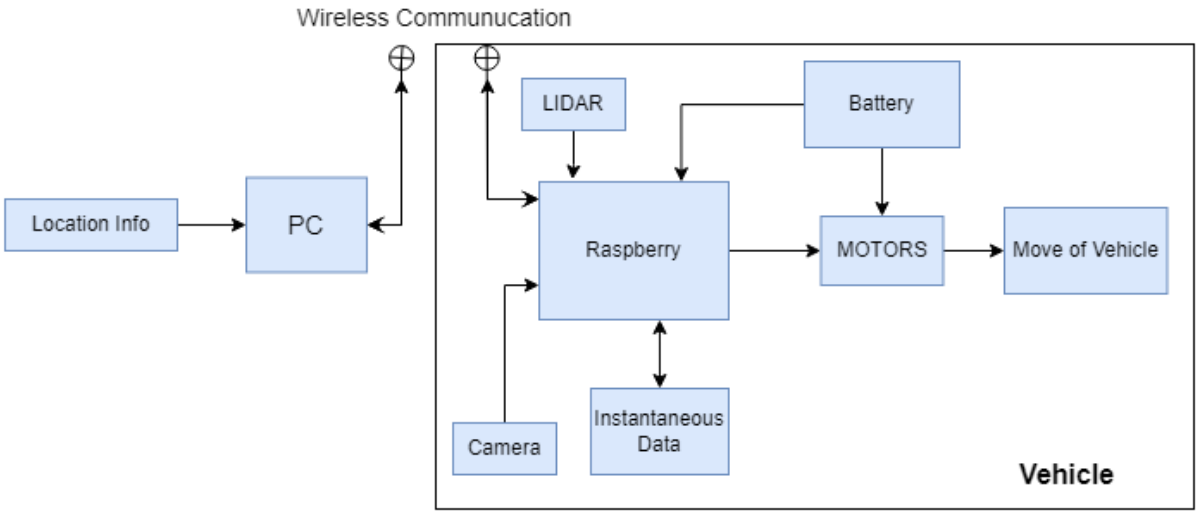

Figure 3

3. Design Details

3.1. General Description

This specially designed vehicle has an advanced technology automatic navigation system. It is capable of moving towards predetermined destinations on its own. It uses a technology called Lidar to generate data that identifies and detects potential obstacles in its surroundings. This technology enables the vehicle to continuously perceive its environment during its journey and relay this information to the user interface. This feature significantly enhances the vehicle's ability to move autonomously.

The vehicle utilizes image processing algorithms to identify objects in its vicinity and adjust its position. The image data captured by the vehicle's camera is sent to Raspberry Pi. Raspberry Pi uses algorithms written in Python's OpenCV library to perform object recognition. As a result, identified objects are displayed to the user within frames on the interface for guidance.

Additionally, the vehicle processes Lidar data to map its surroundings and create a map.

This map is optimized based on predetermined values and continuously updated. It rapidly reports dynamic changes and assists the vehicle in adjusting its path without colliding with obstacles.

The vehicle features a user interface developed using PyQt5. This interface allows users to access live camera feeds, real-time recorded maps, and the vehicle's position on the map. Users can also provide movement commands to the vehicle through the interface.

The mechanical structure of the vehicle includes a 4-wheeled chassis, RPLIDAR, Raspberry Pi camera, power distribution board, a 7.4V battery, and 4 motors. By adjusting the power and direction given to the wheels, the vehicle has a 360-degree maneuvering capability.
Finally, with minor modifications to the system, the vehicle can be equipped to carry out a range of tasks, including automatic seed planting, plant maintenance, and harvesting in agricultural fields. It is important to note that for automated transportation and logistics operations, additional modifications and enhancements would be necessary. The vehicle's modular design allows for flexibility and customization, making it possible to adapt to different use cases with ease. Each component of the vehicle serves a specific function, enabling tasks to be easily distributed and creating a highly flexible and customizable system structure. Ultimately, this versatile vehicle offers the potential to revolutionize agricultural practices and improve operational efficiency in various industries.

3.2. Technical Details

3.2.1. Software Technologies:

3.2.1.1. ROS

ROS, which stands for Robot Operating System, is a flexible and powerful framework extensively used in robotics research and development. One of its fundamental capabilities is mapping, which enables robots to build representations of their environment. ROS provides various mapping algorithms, such as SLAM (Simultaneous Localization and Mapping), allowing robots to navigate and localize themselves within unknown or dynamic surroundings. Additionally, ROS employs a node-based structure, where each node represents a specific functionality or computation. These nodes communicate with each other by passing messages, which are standardized data structures facilitating seamless information exchange. ROS also supports inter-device communication, enabling robots to interact with sensors, actuators, and other devices. This communication is facilitated through topics, services, and action servers, which enable efficient data exchange and execution of complex actions. Overall, ROS provides a comprehensive framework for developing and integrating robotic systems, offering numerous tools and libraries to facilitate tasks such as perception, planning, control, and coordination.

3.2.1.2. Rviz

Rviz, abbreviation for ROS visualization, is a powerful 3D visualization tool for ROS. It allows the user to view the simulated robot model, log sensor information from the robot's sensors, and replay the logged sensor information. By visualizing what robot is seeing, thinking, and doing, the user can debug a robot application from sensor inputs to planned (or unplanned)

actions. Rviz can display various data such as 3D sensor data from stereo cameras, lasers, Kinects and 2D sensor data from webcams, RGB cameras, and 2D laser.

3.2.1.3. OpenCV

OpenCV is an open-sourced and well optimized library for computer vision. It has support for both Python and C++. OpenCV gives many optimized algorithms to use in image processing and with the machine learning library inside, it speeds up the development process significantly.

3.2.1.4. PyQt5

A robust foundation for building graphical user interfaces (GUIs) is offered by the Python module PyQt5. It is built on the well-liked Qt framework, a toolkit with a wide range of features. With PyQt5, programmers can create interactive, aesthetically pleasing programs that work on a variety of operating systems, such as Windows, macOS, and Linux. With PyQt5, programmers can create user interfaces that are simple to use and responsive by utilizing a variety of widgets, layouts, and event handling techniques. Additionally, it gives users access to Qt's vast tool and module library, allowing for the incorporation of cutting-edge technologies like multimedia, networking, and database connection into applications. Overall, PyQt5 is a useful tool for Python developers to quickly and easily construct high-quality GUI applications.

3.2.1.5. Python

Python is a high-level, interpreted programming language. It also features many libraries that have been optimized by the Python community. Python was chosen because it provides easy development environment and open-source resources.

3.2.1.6. Cpp

C++ is a mid-level compiled language that supports Object-Oriented Programming. C++ is utilized in microcontroller boards and numerous ROS packages. Image processing and autonomous navigation algorithms that need high computation processes work more efficiently and faster than high-level programming languages.

3.2.1.7. Communication Protocols (i2c, spı,uart)

The I2C, UART, and SPI protocols are commonly used for communication between different electronic devices. I2C is a protocol that uses clock and data pins to facilitate communication between slave and master controllers. It is advantageous for reducing cabling but is suitable for short distances and has speed restrictions. UART is an asynchronous protocol that enables communication between computers and microcontrollers without requiring a clock signal. It sends data in order of importance and can include start and stop command bits. SPI, on the other hand, operates in a master-slave structure and allows bidirectional communication. It requires four pins for communication: MISO, MOSI, CS, and Clock. Data can be sent using MOSI and received using MISO, while the CS pin selects the slave device, and the Clock pin

synchronizes the data transfer between the two devices. Each protocol has its own advantages and use cases, providing flexibility for different communication needs in electronic systems.

3.2.2. Technical Design:

3.2.2.1. ROS

Through the ROS framework, we benefit from its communication channels, inherent structure, and its own packages. We develop our own software by creating nodes that enable communication among themselves. Typically, we employ the publisher-subscriber model to facilitate this communication. Messages generated by nodes are transmitted between different nodes under of name called topics. These topics can encompass various content types, such as sensor data or output from code. As for the structure, we organize our own package structures within the workspace. Each package contains the code we write and the launch files we use to run them. We establish communication between packages ourselves. Additionally, we adapt and utilize ROS's pre-built packages, especially those related to mapping and navigation. By defining ROS master and connected devices, we enable communication between control stations and vehicles. Once connected within the same local network and accessed via SSH, all messages are processed through the ROS framework. Visual representation of data in the user interface is achieved by extracting messages from ROS and performing the necessary visualization tasks.

3.2.2.2. Embedded Software

We utilize a Raspberry Pi 4B as the main computer for our robot, along with an adeept HAT serving as the motor driver and power distributor. The adeept HAT directly connects to the Raspberry Pi 4B through pin sockets, providing the signals required to drive the motors. In order to communicate at a low level, we have written Python code that utilizes the RPi.GPIO library. This code includes functions to control the robot's movements in different drive conditions, such as forward, backward, left, and right. The code also defines the specific pins used for the motor control. Additionally, we have another Python code that imports these functions and is responsible for listening to a ROS message called cmd_vel. Upon receiving this message, certain conversions are made. The cmd_vel message contains six attributes, but we focus on two of them: linear_x and angular_z. These values determine the direction and speed for the drive functions. The cmd_vel topic can be generated manually using a joystick or autonomously generated through our autonomous algorithm.

3.2.2.3. Mapping

Mapping is the process of creating a representation of the environment, while localization is determining the robot's position within that map. Mapping provides knowledge about the surroundings, enabling effective planning, while localization ensures accurate navigation and collision avoidance. Both mapping and localization are crucial for autonomous robot operation in complex environments.

Mapping requires sensors and we choose to use a 2D lidar. The YDLIDAR X4 is a popular lidar sensor commonly used in robotics and autonomous systems. It provides 360-degree

scanning capabilities, allowing accurate measurement of distances and generating detailed point cloud data of the surrounding environment. The YDLIDAR X4 is known for its affordable price, compact size, and reliable performance, making it suitable for a variety of robotic applications.

Hector Mapping is a popular algorithm used in Simultaneous Localization and Mapping (SLAM). SLAM is a technique used by robots to map their environment while simultaneously estimating their own position within that map. Hector Mapping specifically focuses on mapping indoor environments using laser range finders like the YDLIDAR X4. It utilizes scan matching and occupancy grid mapping to generate highly accurate and real-time maps. Hector Mapping is widely used in robotics research and applications, providing a robust solution for mapping and localization tasks.

We achieved better results by updating the parameters in the mapping algorithm and conducting various tests. During this process, we examined the factors that affect map creation and experimented with optimizing the parameters. For example, we adjusted parameters such as scan frequency, scan angle, cell size, or error tolerance. These updates and tests helped us improve the performance of the mapping algorithm and obtain more accurate and precise results. As a result, by customizing the parameters to our specific needs and conducting tests, we achieved better mapping outcomes.

Before: After:

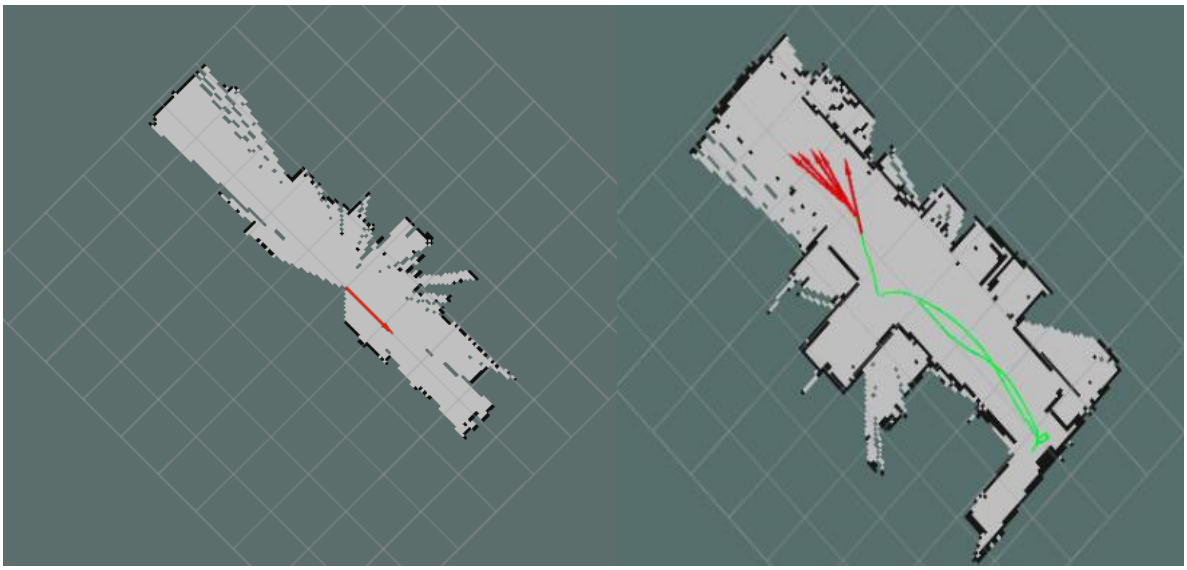

Figure 4

The green line represents trajectory. Red arrow represents last five odometry data.

3.2.2.4. Autonomous Navigation

We utilized move_base package and implemented to our robot. move_base is a high-level navigation system in ROS (Robot Operating System) that combines localization, obstacle avoidance, and path planning to enable autonomous robot navigation. It is commonly used in mobile robot platforms for tasks such as reaching a specific goal location or following a predefined trajectory.

The move_base system works by integrating several components. Firstly, it utilizes a global planner based on A*, to generate a global path from the robot's current position to the goal location. This global planner operates on a static map of the environment.

Secondly, move_base employs a local planner called Dynamic Window Approach (DWA), to navigate the robot through the environment while avoiding obstacles dynamically. The local planner takes into account real-time sensor data which is lidar to compute safe and feasible trajectories.

Furthermore, move_base utilizes a costmap, which is a 2D grid representation of the environment. The costmap assigns different costs to different areas based on obstacles, restrictions, and preferences. This information is used by the global and local planners to generate paths and avoid collisions.

To configure move_base, several parameters can be adjusted based on the robot's specific characteristics and the environment it operates in. Some key parameters include the inflation radius of the costmap, which determines how much space the robot should keep from obstacles, the maximum velocity and acceleration limits for the robot, and the planner-specific parameters like the goal tolerance or lookahead distance. Fine-tuning these parameters can significantly impact the navigation performance of move_base in different scenarios.

The move_base navigation stack in ROS is designed to handle goal-based navigation. By selecting a destination point on the map through the user interface, we provide move_base with the input it needs. The algorithm then continuously provides feedback throughout the navigation process. This feedback includes notifications such as goal reached, generation of a new path, or failure to generate a path. These updates allow us to monitor the progress of the navigation and make informed decisions based on the algorithm's responses.

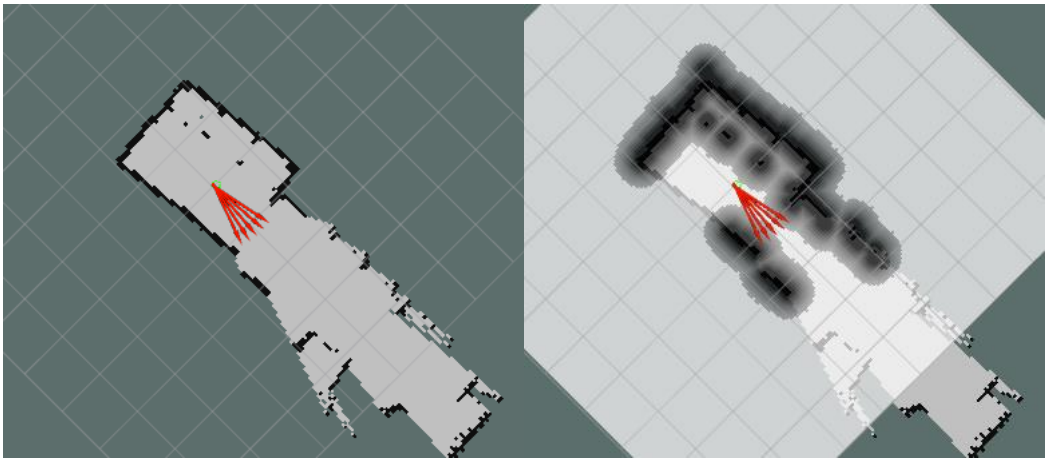

Figure 5

The map on the right is Costmap.

3.2.2.5. Image Proccessing:

Using our image processing algorithm, the vehicle detects objects encountered based on their color, determines whether they are friend or foe, and adjusts the vehicle's position accordingly. Initially, the real-time image data captured by the vehicle's camera is sent to Raspberry Pi. Then, algorithms implemented with Python's OpenCV library are utilized on Raspberry Pi to process the image data and perform friend-foe distinction. Firstly, we convert the captured image from the BGR format to the more suitable HSV format. To ensure the object is detected most accurately, we fine-tune the Hue, Saturation, and Parameter values to their optimal settings and apply lower and upper limits for object recognition. Subsequently, contouring is performed within the range defined by these lower and upper values. If there is a contour area smaller than the specified region, it is excluded from consideration. Finally, the boundaries of the final result are determined and displayed on the interface with frames to guide the user. All these processes are applied to both friend and foe objects based on their color.

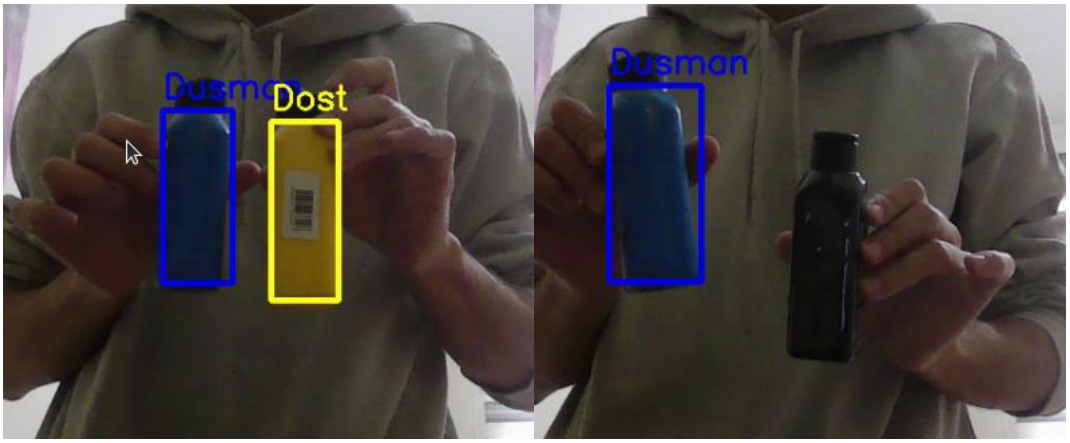

Figure 6

3.2.2.6. Communication:

Ensuring data integrity and security is of utmost importance during the communication between the vehicle and the ground station. Any disruption in this communication can lead to undesirable consequences. SSH, a secure remote access protocol, serves as an effective solution for establishing a secure connection between the vehicle and the ground station. By establishing an SSH connection, commands and files can be securely transmitted between these two entities.

To establish an SSH connection, a shared wireless network is configured between the vehicle and the ground station. Through this network, the ground station, acting as the master device, initiates a request to access the Raspberry Pi on the vehicle. Once access is granted, the ground station gains control over the vehicle.

During SSH communication, encryption, and authentication mechanisms are employed to ensure the confidentiality and integrity of the transmitted data. These measures prevent unauthorized access and tampering with the communication.

In summary, utilizing SSH in the communication between the vehicle and the ground station offers a secure and reliable means of transmitting commands and data, thereby guaranteeing the integrity and security of the communication channel.

3.2.2.7. Interface:

The interface, developed using PyQt5, is designed to facilitate vehicle operation for users at the ground station. Through this interface, users can access various data, including live camera feeds (including detected objects), date and time information, real-time recorded maps, and the vehicle's position on the recorded map. Additionally, users can control the vehicle's movement to selected points on the field through the interface. In the background, the initial design of the interface was created using the Qt Designer application. After creating the skeleton of the interface design, the integration of key operations such as date-time display, camera functionality, and visualization was implemented in the main code. In this process, two frames were created on the layout to place the camera and map data. The captured image from the camera hardware, which underwent image processing, was placed in the camera frame. The map frame was populated by converting the matrix data obtained through the "nav_msgs/OccupancyGrid.msg" message into a visual representation. Additionally, the position of the vehicle on the map was displayed in the map frame using the Pose and Twist data obtained from the "nav_msgs/Odometry.msg" message.

3.2.2.8. Batteries:

Lithium-ion batteries are a type of secondary battery in which lithium ions move from the cathode to the anode during the discharge process. Li-ion batteries are known for their high voltage and charge storage capacity per unit mass and volume. Another advantage of Li-ion batteries is that they do not require full discharge before recharging. They can be charged at any time and can be disconnected from the charging process at any level. As a result of these benefits, the vehicle was equipped with two 3.7V and 2000mAh lithium-ion batteries.

4. Performance Analysis

ID	Test Plan	Test Result
Balance Test	The vehicle must remain stable and not lose its balance when driving on a flator one-sided elevation road.	Due to the balanced placement of components in the vehicle, considering the center of gravity, the legs not being excessively long, and their proximity to each other, it has been observed that the vehicle remains stable in such challenging conditions.
Drop Test	No appreciable deformation should occur when the vehicle is released from a certain height.	The joints remained rigid after dropping due to flexible tires are absorbing the shock.
Traverse Test	The vehicle should be able to climb a 25-degree slope and descend a 25-	In this test, no unexpected results were observed, and it was observed that the vehicle

	degree incline with ease.	successfully climbed and traversed these slopes. However, it was noticed that the engine was insufficient for steeper inclines. Therefore, a more powerful engine should be preferred for steep slopes.
Vibration Test	The chassis and fasteners must remain rigid in the maximum vibration to which the vehicle can be exposed in motion.	The vehicle has been tested on gravel terrain and no issues have been observed with the chassis or connections.
Wheel Test	The wheels should provide traction on soft ground and not sink into the soil.	As a result of this test, it was observed that the vehicle is not suitable for soft, damp, and muddy terrains. Issues such as slipping and spinning were observed. Additionally, being a low-lying vehicle also poses risks in terms of components. Wheels made of different materials that are suitable for the terrain could be manufactured. However, due to budget constraints, these wheels were chosen.
Safety Test	There is an always active emergency button in an easily accessible position on the vehicle. When an emergency occurs, the power of the system is cut off by pressing the emergency button.	The emergency button has been tested on different days and has no problems with its operation.
Environmental Test	It tested whether it works properly in task environments by running the vehicle on different days and in different environments.	The system worked properly in various environmental factors.
Battery Capacity Test	The effectiveness of the vehicle's battery during the expected operating time is tested. The expected operating time is 30 minutes.	As a result of these tests, which we tried on different days and environments, it was seen that the battery met the system requirements throughout the entire period. After approximately 30 minutes, there were fluctuations in the motor speed, and after a certain period of time, it completely shut down. In conclusion, since it met the expected time, the battery capacity was considered sufficient.
Short Circuit Test	Using a multimeter, the system is	It has been observed that there is no problem in the system

	carefully checked for short circuits.	connections.
Object Detection Test	The object detection algorithm's ability to accurately distinguish between friend and foe based on color should be verified.	After conducting numerous tests, efforts were made to determine the most suitable values and the upper and lower limit ranges for the given colors in the HSV format. Successful results were achieved through these tests.
Mapping Test	The mapping algorithm should be fast responsive to dynamic changes on environment	There are two parameters that allow adjustment for the confirmation count required to label an obstacle as previously unseen or to label it as no longer present when it was previously identified. Multiple tests were conducted to optimize these parameters. As a result, successful attainment of optimal values has been achieved.
Interface Test	The interface is tested to ensure that the output of object detection on live camera footage and the real-time updated map are displayed correctly.	As a result of the conducted tests, no delays or connection disruptions were observed in the data retrieved to the interface.
Navigation Test	A navigation test was conducted to observe how a robot can autonomously navigate to a specific location while effectively avoiding obstacles and staying on course.	The test results indicated that the robot successfully reached the given location with very minimal margin of error and was able to effectively avoid obstacles. However, it was observed that when there were numerous objects in its vicinity, the robot became confused and entered a recovery mode.

Table 1

Rotation Test Error:

SLAM Test Error:

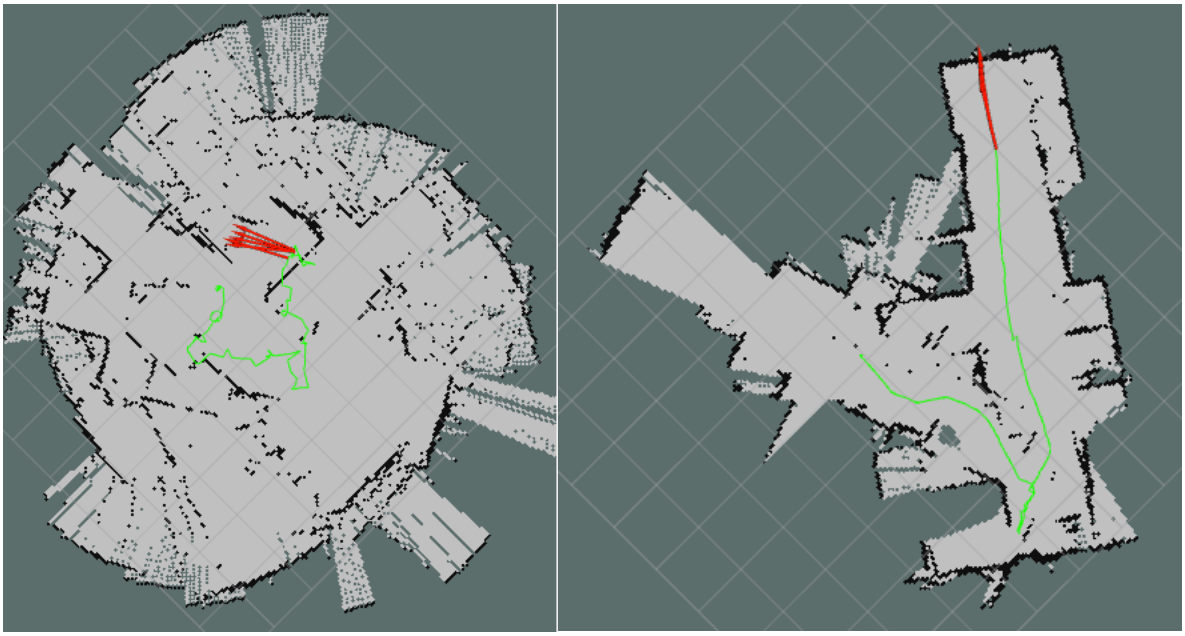

Figure 7

After Correcting Parameters:

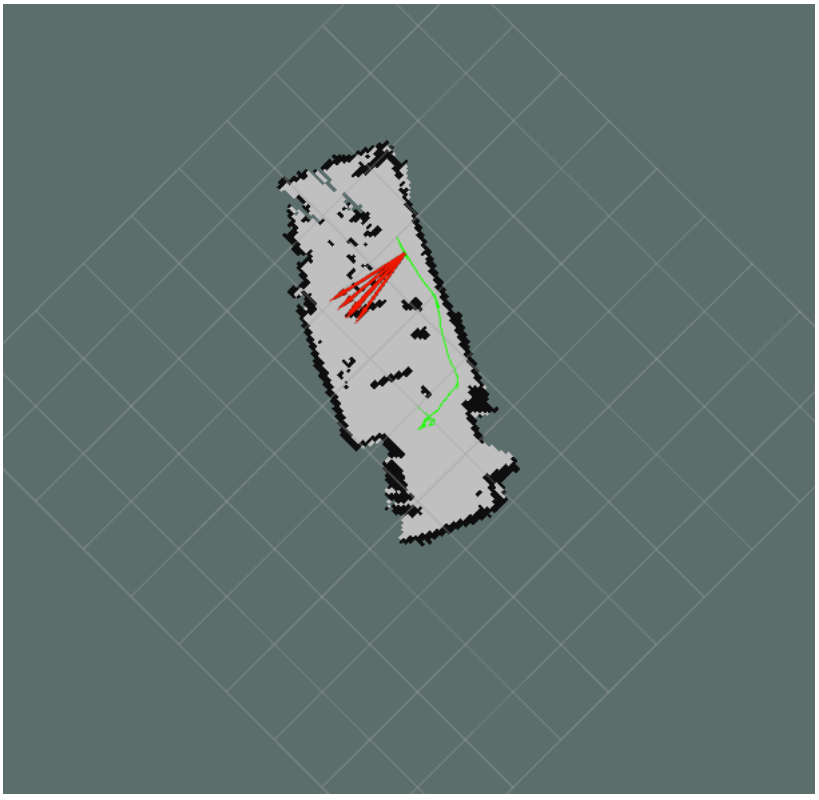

Figure 8

5. Cost Analysis

Module Name	Component Model	Unit	Price per Piece (₺)	Total Price (₺)
Microcontroller	Raspberry Pi Model 4GB	1	4.654	4.654
Lidar	YDLIDAR X4	1	2.858	2.858
Motor + Wheels	Geared DC Motor+ RC Car Wheel	4	79.02	316.08
Motor Control Board	Adeept Motor HAT	1	498	498
Battery	3.7V 2000mAh Li-ion Battery	2	133,04	266,08
SD Card	Adata 32GB microSD Card	1	126	126
Sensor	Distance sensor	1	29.28	29.28
Camera	Raspberry Pi Camera	1	254.61	254.61
Frame	Structure of the vehicle	1	500	500
Display Screen	Samsung 19" Wide	1	500	500
Total	-	-	-	10.002,05

Table 2

Development and production costs are estimated as 10.002,05 ₺ as seen in the table above. (Products that were damaged during the test were not added.)

The vehicle will be constructed using the students' funds. No fixed costs will be incurred, such as rent or market research expenses. Variable costs are limited to raw materials only, and there will be no labor expenses. Therefore, for us, the breakeven point is equivalent to the total raw material costs, amounting to 10.002,05 ₺.

According to our report last semester, we made changes in the materials we planned in our

cost analysis. With these changes, we released the autonomous module, ESC module, Wireless Communication and GPS products. Instead, we added products such as Lidar, SD Card and Motor Control Board. This change is the result of our research and as we progressed, we realized that some products are not needed, but we need some products. In addition, thanks to these changes, we have met the same requirements at a lower cost.

6. Discussions

While designing the vehicle, we made some priorities in the design according to where it can be used for commercialization. While making these prioritizations, we cared our project is primarily how our design meets standards, certification, and safety regulations. In our tests when working with autonomous robots, we always encounter problems such as the disruption of the autonomy, which is a problem that we encounter more often due to the SSH connection being broken and sometimes the Raspberry Pi data processing late. We have developed appropriate solution methods and we have fixed the problems autonomously in our tests, but when our product will be commercialized, we may encounter unforeseen errors. As a solution to such situations, we also wrote a deactivated code that can drive our vehicle manually with a joystick. In a problematic situation, we made a manual correction and then allowed autonomous driving again, so we took precautions for user-based errors, which is one of the biggest problems in commercialization. Thanks to this measure, we have also eliminated the safety problems that may arise in the event of the vehicle getting out of control. Apart from that, we used as few parts as possible in the structure of the vehicle, and their assembly process is very easy and suitable for some integrations. For example, the lidar sensor is one of the most power-consuming modules of the vehicle, if we want, we can solve the power of the lidar with a power bank that we can add to the front of the vehicle.

If we look at the impact of our design on society, first, we need to consider how we create mapping and location information. Prices are increasing with the effect of today's economic developments. To solve this problem, we preferred to use LIDAR instead of using GPS, so we got rid of common problems such as inability to give accurate data, which is the biggest problem in cheap and poor-quality GPS, and incorrect location determination when we process the data it receives. If we consider the other case, this problem is less or not at all-in high-quality GPS, but if we compare the price of LIDAR with these GPS, we see that the price of GPS is almost 5 times or more than LIDAR. Thanks to this solution, users do buy a vehicle that can do the same job at a cheaper price.

Let's take a look at the Standards, certifications, and safety regulations for our project.

1- Standards:

- ISO (International Organization for Standardization): ISO has developed several standards related to autonomous vehicles, such as ISO 21448 (Safety of the Intended Functionality, or SOTIF) and ISO 26262 (Functional Safety for Road Vehicles).

- SAE (Society of Automotive Engineers): SAE has developed a comprehensive set of standards known as SAE J3016, which defines six levels of driving automation from Level 0 (no automation) to Level 5 (full automation). [Also our vehicle is Level 4 (High Automation) the user only takes over control in specific situations.

- IEEE (Institute of Electrical and Electronics Engineers): IEEE has various standards relevant to autonomous vehicles, including IEEE 2846 (Guide for Safety and Performance of Autonomous Vehicles).

2- Certification:

- Safety assessments: Evaluating the design and functionality of the autonomous system to ensure it meets the necessary safety standards and requirements.

- Testing and validation: With the tests we have done in the test part, we have done the necessary actions for both good work and security.

3- Safety Regulations:

    - Functional safety: Requiring autonomous systems to meet specific safety requirements, such as fail-safe mechanisms, redundancy, and risk mitigation strategies.

    - Data privacy and security: The wireless we choose ourselves in our code of the network to be connected for SSH and the password are connected to the predetermined network, only in this way we have ensured the data security.

7. Appendix

7.1. Product Design Document

1. Title: Advanced Autonomous Land Vehicle

The advanced autonomous land vehicle we have developed is a cutting-edge system that offers autonomous navigation capabilities for a variety of applications. With its state-of-the-art technology and robust design, it provides a reliable and efficient solution for tasks in various industries.

2. Purpose: How this product will be used, and who will use it?

This autonomous land vehicle is designed to revolutionize operations in several sectors. Primarily, it is tailored for use in agricultural fields, where it can perform tasks such as seed planting, plant maintenance, and harvesting. By automating these processes, it enables increased productivity, reduces labor requirements, and optimizes resource utilization.
Furthermore, the vehicle has the potential to be employed in transportation and logistics operations. With minor modifications and enhancements, it can be adapted to handle automated transportation tasks, including the movement of goods and materials within industrial and commercial settings. This versatility makes it a valuable asset for streamlining operations and improving efficiency in various industries.

3. Composition: List all parts of the product

The advanced autonomous land vehicle consists of the following components:

- 4-wheeled chassis for stability and maneuverability
- RPLIDAR for precise laser-based sensing and obstacle detection
- Raspberry Pi camera for capturing images of the surroundings

- Power distribution board to manage electrical power distribution
- 7.4V battery for reliable and long-lasting operation
- 4 motors for enabling 360-degree maneuvering capability
- Lidar technology for generating data to facilitate navigation and obstacle detection
- Image processing algorithms for real-time object detection and identification
- User interface developed using PyQt5, providing access to live camera feeds, recorded maps, and vehicle position

4. Development Skills Required: Skills required to develop the product

Developing this advanced autonomous land vehicle requires expertise in several areas:

- Proficiency in the ROS (Robot Operating System) framework for integrating and controlling the various components
- Knowledge of mapping and navigation algorithms, particularly SLAM (Simultaneous Localization and Mapping)
- Experience with image processing techniques and object recognition using OpenCV
- Competence in Python programming language for developing software modules
- Familiarity with hardware integration and control, including Raspberry Pi and motor drivers

5. Quality Criteria: Quality specifications and quality measurements for the product

The product is expected to meet the following quality criteria:

- Compliance with industry standards and regulations for autonomous systems
- Accurate and reliable obstacle detection and avoidance capabilities
- Precise positioning and navigation accuracy
- Robust and responsive user interface for seamless control and monitoring
- Durability and stability in various operating conditions

6. Quality Tolerance and Method: Details of quality criteria range and quality method

The product's quality tolerance is defined within specific ranges for each quality criterion. These ranges ensure that the vehicle operates within the expected performance parameters and meets the required standards.

To ensure quality, various quality methods are employed, including:

- Design verification to confirm that the vehicle meets the specified requirements and functionalities
- Pilot testing in real-world scenarios to evaluate performance and identify potential issues or improvements
- Inspection of each component and subsystem to verify their integrity and functionality

7. Quality Skills Required: Skills required to undertake the quality method

Ensuring quality for this advanced autonomous land vehicle requires individuals with the following skills:

- Expertise in quality assurance and inspection techniques
- Proficiency in testing methodologies and data analysis
- Ability to identify and resolve performance and functionality issues
- Knowledge of industry standards and compliance requirements for autonomous systems
- Collaboration with engineering and development teams to ensure quality standards are met

7.2. Proofs

Figure 9

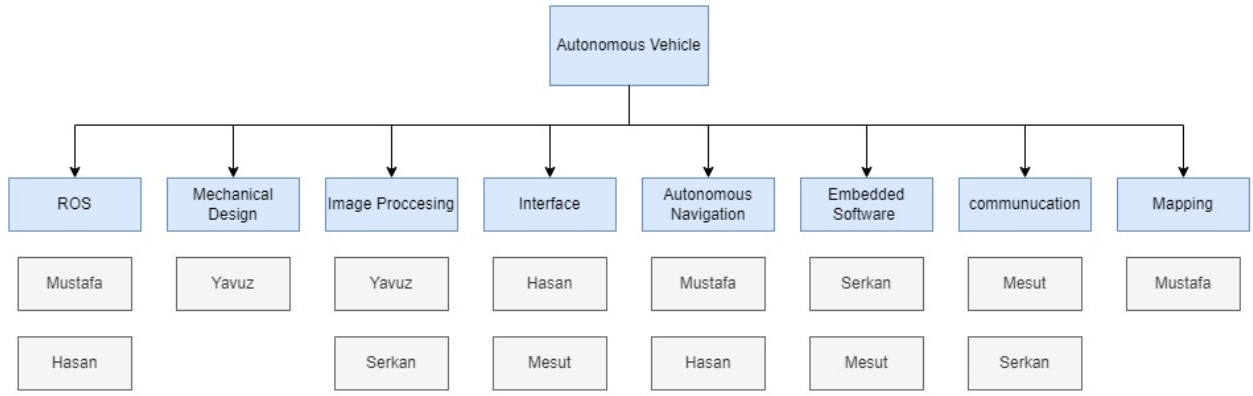

Figure 10

Also we have a Youtube Video of the Vehicle: https://www.youtube.com/watch?v=u234mMnt26w

Note: During the project process, some of our friends were affected by the earthquake. For this reason, the subject of BMS was removed from our project by talking with our project mentor teacher

8. References

Literature references:

1- Silva O.D., Mann G.K.I, Gosine R.G., An ultrasonic and vision-based relative positioning sensor for multirobot localization, IEEE Sens. J., 15 (3), 1716-1726, 2015.

2- Wyglinski A.M., Huang X., Padir T., Lai L., Eisenbarth T.R., Venkatasubramanian K., Security of autonomous systems employing embedded computing and sensors, IEEE Micro, 33 (1), 80-86, 2013. 3- Bruch H, Gilbreath G. A. (2002). Muelhauser J.

3- Bok-Joong Yoon, Jung-Hun Na, Jung-Ha Kim, (2007). Navigation method using multisensor for UGV (Unmanned Ground Vehicle), Control, Automation and Systems. ICCAS '07. International Conference on , vol., no., pp.853,856.

4- K. Kerimoğlu. (2011). Sabit kanatlı bir insansız hava aracı için düşük bütçeli otopilot sistemi tasarımı. Master thesis, TOBB Ekonomi ve Teknoloji Üniversitesi

5- Bellian, J.A., Kerans, C., Jennette, D.C. Digital out crop models: applications of terrestrial scanning lidar technology in stratigraphic modeling. J. Sediment Res., 2005; 75(2): 166-176.

6- Akyol, S. Rp-lidar ve mobil robot kullanılarak eş zamanlı konum belirleme ve haritalama, Yüksek Lisans Tezi, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ, 2017

URL References

1- https://razbotics.wordpress.com/2018/01/23/ros-distributed-systems/
2- http://wiki.ros.org/hector_slam
3- http://wiki.ros.org/gmapping
4- https://phoenixnap.com/kb/ssh-to-connect-to-remote-server-linux-or-windows
5- https://www.youtube.com/playlist?list=PLU0kJa57QNHS9YqFKJe_ZXNBCnzuwSGGa
6- https://www.slamtec.com/en/Lidar/A1
7- http://en.wikipedia.org/wiki/Inertial_measurement_unit