Palestine Polytechnic University

College of Information Technology and Computer Engineering
Department of Computer Engineering

Package Delivery Robot

Abdullah Al-Nuaimi (181004)

Mahmoud Harb (217102)

Supervisor : Dr. Hashem Tamimi

May 2023

Acknowledgement

In the name of Allah, the most beneficent and merciful who gave us strength, knowledge and helped us to get through this project. For those who deserve our thanks the most, our parents, we are indebted to you for the rest of our lives for your unconditional love and support. We know that thanks is not enough and there are not enough words to describe how thankful we are.To our families and friends, thank you for your endless encouragement all our lives and especially during the completion of this project.We would like to thank our supervisor of this project, Dr. Hashim Tamimi for the Valuable help and advice during this project.We also thank our faculty and Professors at the College of Information Technology and Computer Engineering for their hard working and support for the students.

Abstract

Recognizing the repetitive nature of order delivery between different locations, we identified the time-consuming and error-prone aspects of the process. Instances of human errors resulting in damage or loss of packages further compounded the issue. Additionally, unforeseen events like the COVID-19 pandemic emphasized the need for automation in this field. Motivated by these factors, we embarked on finding a logical, efficient, safe, and cost-effective solution.

Our solution involved the design of an autonomous mobile robot that assists users in delivering packages to desired destinations. By leveraging a web application, users could load the robot with the package and specify the final destination. The robot swiftly generated an optimal path using a pre-built map, facilitating seamless navigation. Along the journey, the robot encountered unexpected obstacles such as pedestrians, animals, or other obstructions. Our robot possesses the capability to navigate around these obstacles while maintaining its course towards the destination.

To accomplish these tasks, we employed the Neobotix MP-500 robot, which came equipped with an implicit 2D LiDAR sensor. Additionally, we enhanced the robot's capabilities by integrating an Arduino board loaded with an IMU and GPS receiver. This combination of hardware components enabled the robot to successfully fulfill its mission.

الخلاصة

لوحظ لدينا أن عملية توصيل الطلبات بين أماكن مختلفة هي عملية تكرارية تستنزف الوقت والجهد، بالإضافة إلى احتمالية حدوث أخطاء بشرية تؤدي إلى تلف أو فقدان الحمولة المنقولة، مما يؤدي إلى العديد من المشاكل غير المرغوب فيها. بالإضافة إلى ذلك، في بعض الأحيان تحدث بعض الأحداث المفاجئة، مثل جائحة كوفد-19 على سبيل المثال، بالإضافة إلى أن العالم يتسارع في مجال التحكم الآلي بطريقة قوية. لذلك، كل هذه الأمور دفعتنا لإيجاد حلاً منطقيًا سريعًا وفعالًا وآمنًا وغير مكلف لإتمام هذه العملية تلقائيًا من خلال تصميم روبوت متحرك ذاتي القيادة يساعد المستخدم الذي يرغب في إرسال طرد إلى وجهة مرغوبة بواسطة تحميل الروبوت بالطرد، وتحديد الوجهة النهائية التي يرغب المستخدم في إرسال الطرد إليها باستخدام تطبيق الويب الذي سيساعده في ذلك. يقوم الروبوت ببناء المسار الأمثل بسرعة، وكذلك بطريقة سهلة للتنقل من خلاله على الخريطة التي قام ببنائها مسبقًا إلى المكان الذي سيعمل فيه، ثم يبدأ في التحرك نحو الوجهة. من الطبيعي أن يواجه الروبوت عقبات مفاجئة في طريقه قد تمنعه من مواصلة الرحلة، مثل الأشخاص والحيوانات وأي عقبات مفاجئة أخرى على الطريق. يتمتع روبوتنا بالقدرة على تجنب هذه العقبات دون فقدان المسار إلى الوجهة النهائية.

لتحقيق هدفه، سيكمل الروبوت مهمته بنجاح باستخدام روبوت Neobotix MP-500 الذي يأتي مع جهاز استشعار ثنائي الأبعاد (2D LiDAR) ضمنه، وسنزوده بلوحة Arduino المحملة بجهاز استشعار الحركة الداخلية (IMU) ومستقبل GPS.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Preface

The field of autonomous robotics is developing rapidly. Today, robots are everywhere, in homes, factories and military uses. Autonomous robots under development are powerful enough to work alongside humans and perform tasks efficiently.

In this work, we present an autonomous mobile robot that delivers packages without human intervention. Starting from the initial geographic destination provided by the user, the system plans an optimized route and navigates through it independently. How the robot works in an outdoor environment is explained in detail in this work. We have explained the problem of semantic segmentation for road and obstacle detection.A general requirement for autonomous navigation is the availability of very-detailed maps. The topics of robust localization and sensor problems are explained in detail in this work. The need for a vision is also discussed.

1.2 Project Aims and Objectives

In this project, we aim to achieve several objectives:

1. Employ a mobile robot that is autonomously guided through the outdoor-environment.
2. The robots should be able to make a map of the environment and localize itself within it.
3. The robot should be able to take user requests about the destination and navigate to the destination.
4. The robot should be able to avoid obstacles it faces in the road.

1.3 Problem Statement

Delivering small packages costs human time and energy. The indoor delivery robots are limited to one environment, we can replace these alternatives with an automated and safe process done by an autonomous mobile robot.

1.4 Project Requirements

Table 1.1: functional Requirements

Receiving the request	The system must be able to receive the user's request through a smartphone app, so that it knows the desired final destination, which enables him to complete his work.
Select the path	The system must be able to discover the best path from the start point to the end point, through the use of system(GPS) and a built map.
Robot navigation	Our robot must be able to move in all directions on the ground by changing the speed of each of the two rear wheels to change direction.
avoid obstacles	After the obstacle is detected, the robot must be able to change its lane to avoid the obstacle without losing track to reach the final destination.
Build a map	Our robot should be able to build a map while moving, by processing the sensors(LIDAR and IMU) input, using SLAM technique.

Table 1.2: Non functional Requirements

Usability	The user of the robot should be able to use it in an understandable and easy way.
Safety	The robot should follow safety procedures, the robot will keep a safe distance from human in front and back, on the sides the robot will keep on the most right side in the walkways.
Reliability	The robot must be able to reach the goal correctly and within a reasonable time

1.5 System Description

The system is a robot that delivers parcels from point A to point B in an outdoor environment. In order to independently deliver the recommended packages, this robot should be equipped with a GPS receiver and IMU to determine its position in space. For obstacle avoidance it will be loaded with laser sensor (LiDAR), to determine if there's objects around the robot. A camera is optional to only monitor the robot navigation remotely.

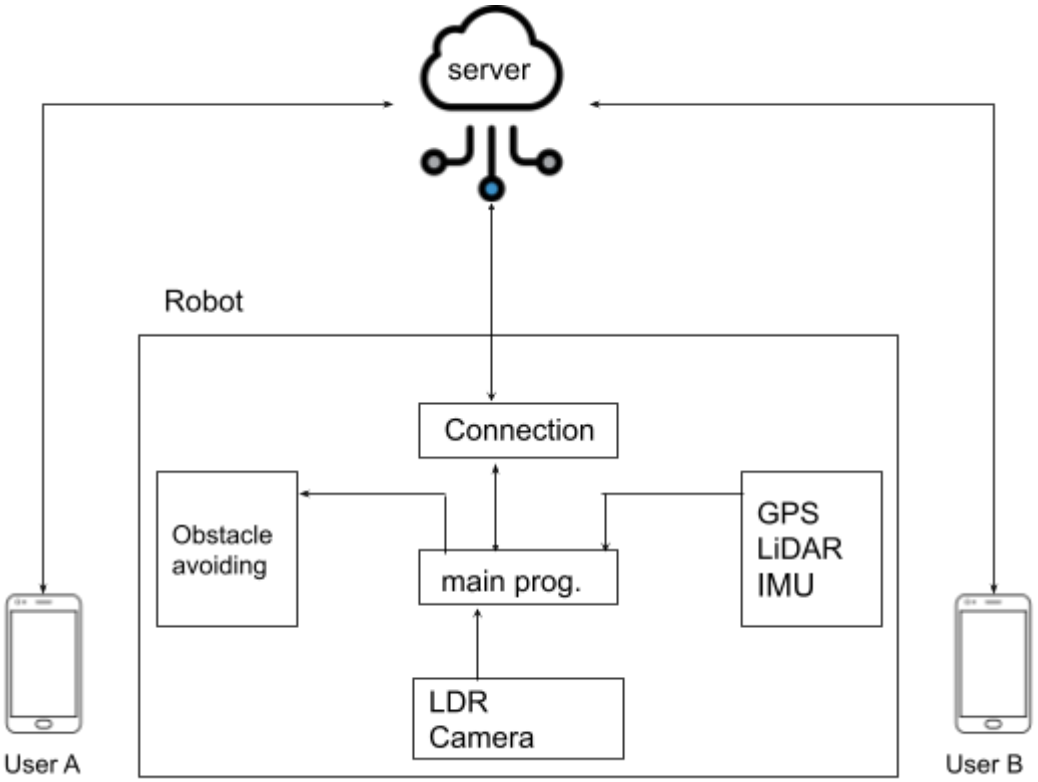


Figure 1.1: main block diagram of the system

1.6 Project Limitations/constraints

- The robot will not function in places that do not have GPS signal available like indoors and near large buildings.
- Distance is limited to the robot battery capacity.
- Maximum package weight should not go over 1 kg.
- Maximum robot speed is 1.5m/s.
- The robot can not use stairs.

- Climate change, rainfall, strong winds may harm the robot and have an impact on the robot's functions. So we assume the robot will not function under these conditions.
- The system takes shortest path as the optimal path, without consideration of overcrowding, D* lite algorithm will be used to determine the optimal path.

1.7 Project Schedule

Table 1.3: Project schedule

task	Time (cell = 2 weeks)
Introduction	
Background	
System Design	
Implementation and Testing	

1.8 Report Outline

The next chapter, "Background," covers theoretical background and Literature Review, software and hardware design options. The third chapter, titled "System Design," contains a full conceptual description of the system (Hardware and Software aspects), detailed design, schematic diagrams, block diagrams, structural diagrams, and any necessary information relevant to the system design.

Chapter 2: Background

2.1 Overview

This chapter introduces the theoretical background needed for this project. It contains the technologies employed in the project, and a description of the system's hardware and software components. Certain design specifications and constraints are described at the end of this chapter. In addition, the chapter reviews other projects that have been done similar or related to this project.

2.2 Theoretical Background

In this section, we will provide a clear idea about the project functionality, and give some information about the general terms.In order for a robot system to navigate through an environment it needs three main things: Mapping, Localization and path planning as shown in previous figure 2.1, which we will demonstrate in the following sections.

2.2.1 Navigation

1. Mapping

Mapping is a technique used to create a map, contains three things: Free space the robot can move in it, occupied space the robot should avoid collision with it, unknown space which is not discovered yet.

There are several representations of the map, one of them is volumetric representation as shown in figure 2.2 in which every pixel on the grid has a status, another way to present a map is feature map which shows the robot trajectory as shown in figure 2.3.

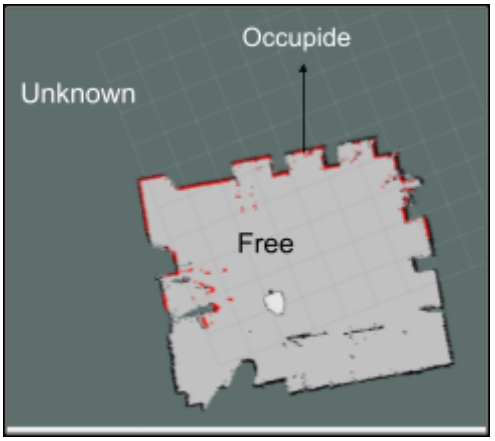


Figure 2.1: volumetric map (i.g. Grid)

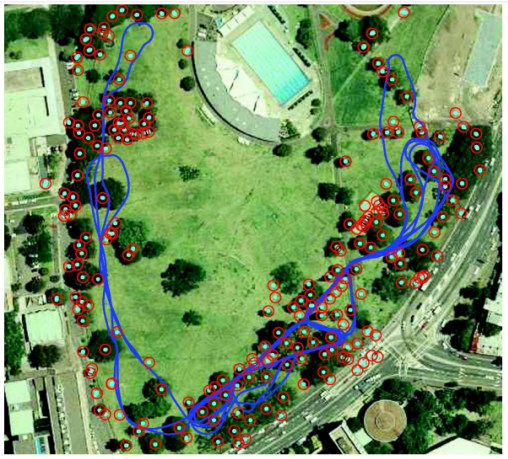


Figure 2.2: feature map [1.1]

The main technique to build a map called SLAM (Simultaneous localization and mapping), it's a computational problem of constructing a map and keep track the robot location within it, it appears to be chicken and egg problem were a map needed for localization; a pose needed for a mapping as shown in figure 2.3.

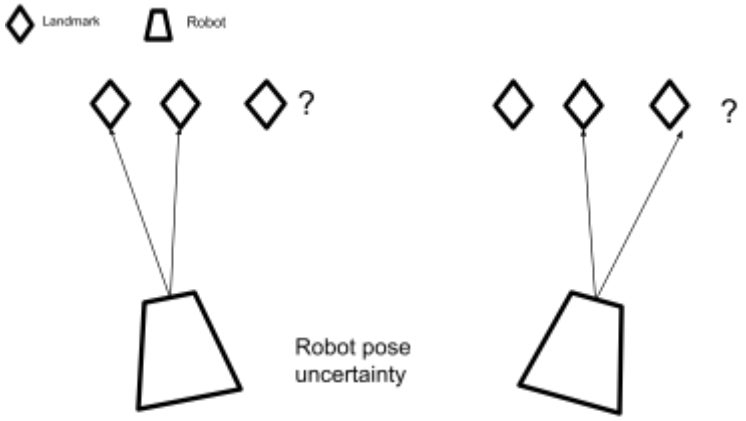


Figure 2.3: SLAM problem

There are various ways to solve SLAM problem like particle filter, extended kalman filter [1] and Gmapping. Algorithm 1 shows the Gmapping solution for the SLAM problem.

Algorithm 1: Gmapping [28].

```
Initialize:
- Create a grid-based map with empty cells
```

```
- Initialize a set of particles representing possible poses
and maps
Loop:
1. Sense environment:
   - Obtain sensor measurements (e.g., laser range readings)
   - Update robot's odometry information (e.g., position and
     orientation)

2. Motion update:
   - Apply motion model to update particles poses based on
     odometry

3. Measurement update:
   - For each particle:
     - Transform sensor measurements to the particle's pose
     - Associate measurements with map features (e.g.,
       occupancy grid cells)
     - Update the probability of occupancy for each
       associated map feature
     - Update particle weights based on the measurement
       likelihood

4. Resampling:
   - Resample particles based on their weights to maintain a
     diverse set

5. Map update:
   - For each occupied cell in the updated map:
     - Increase the occupancy probability
   - For each unoccupied cell in the updated map:
     - Decrease the occupancy probability

6. Estimate robot's pose:
   - Compute the weighted average of the particle poses as
     the estimate
7. Repeat from step 1
```

2. Localization

Localization is simply the ability of a robot to identify its position (location and orientation) within its environment. There are three main types of localization.

1. odometry localization which is a method that estimates the robot's pose by tracking its motion using internal sensors, such as wheel encoders. By measuring the rotation of wheels and the distance traveled, odometry calculates an estimate of the robot's position. However, odometry is prone to accumulating errors over time and may lead to drift in localization.

2. global localization (map based, beacon based) that makes observation of the environment using sensors like lidar and GPS.

3. combination of both odometry and global localization which is most used; to reduce the errors came from sensors [2] .

Various algorithms were developed to solve the localization problem, main algorithms we are going to observe are grid localization (an application of bayes filter) [3] , monte carlo localization (an application on particle filter) [4] and kalman filter [5] a comparison on the three algorithms is observed in table 2.1

Table 2.1: a comparison between grid,monte carlo and kalman filter localization algorithms

	Grid localization	Monte carlo localization	(extended) Kalman filter
Measurements	Raw Measurements	Raw Measurements	Landmarks
Measurement noise	Any	Any	Gaussian
Posterior	Any	Any	Gaussian
Efficiency (memory)	- -	-	+ +
Efficiency (time)	- -	-	+ +
Robustness	++	++	-
Resolution	+	+	++
Ease of implementation	-	++	+
Unknown initial pose	Possible	Possible	Not possible

Algorithm 2: Extended Kalman Filter [29]

//control vector u, measurement vector z, estimation values μ_t(pose) and //Σ_t(covariance).

Prediction in lines 2-3 , correction in lines 4-6.

Function KALMAN FILTER $\left(\left(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t\right)\right)$

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^{\top} + R_t$
3. $k_t = \bar{\Sigma} C_t^T \left(C_t \bar{\Sigma}_t C_t^T + Q_t\right)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_\tau \left(z_t - C_t \bar{\mu}_t\right)$
5. $\Sigma_t = \left(1 - K_t C_t\right) \bar{\Sigma}_t$
6. Return μ_t, Σ_t

end function

3. Path planning and obstacle avoiding

Path planning is an essential capability of autonomous robot, enabling it to determine the most optimal path from its current location to a desired destination within the free configuration space. It involves evaluating various criteria, such as avoiding obstacles, minimizing travel time, conserving energy, or maximizing safety, to generate a trajectory that satisfies these objectives.

Once the path is planned, robot navigation comes into play, referring to the robot's ability to effectively control its actuators and follow the trajectory derived during the planning phase. By precisely adjusting its actuators, the robot can traverse the planned path, maneuvering around obstacles and accurately reaching the desired destination.

The process of path planning and robot navigation are intricately connected, with path planning serving as the foundation for guiding the robot's movements, and robot navigation ensuring the execution of the planned trajectory [6].

A: path planning

There are two main types of algorithms used in path planning.

1.Sampling based search algorithms:

Creating a random sample of nodes in configuration space that helps the robot to find a trajectory leads it to the goal state without considering if the path is

optimal or not, also it is a very good approach in high dimensional search spaces. Rapidly Exploring Random Trees (RRT) and (RRT*) is an example of sampling based search algorithms [7].

2.Search based algorithms:

Find path based on minimum cost in a given grid map, this kind of algorithms mostly used in 2D grid maps, the memory space in this algorithms increases rapidly when the increasing of dimensions like dealing with a 4 or more DOF system.

Main search algorithms:

A* algorithm It is a tracking algorithm that determines the path from the current location of the robot to a specific goal point while avoiding static obstacles in Cobs inside the map and giving greater priority to the goals that are closer with lower costs [8].

D* lite algorithm, It is based on A* so it determines the roadmap with lowest cost, in addition it recalculates the path when the robot faces a new obstacle and avoids it[9].

Algorithm 3: D* lite pseudocode [30]

```
1.  Function Key(s):
2.      return
3.          {min(g(s), rhs(s)) + h(s_start, s) + K_m;min(g(s),rhs(s))]
4.  Function Update Vertex(s):
5.      if s ≠ s_goal then
6.          rhs(s) = min_{s'∈Succ(s)} (cost(s, s') + g(s'))
7.      end
8.      if s ∈ OPEN then
9.          OPEN.remove(s)
10.     end
11.     If g(s) ≠ rhs(s) then
12.         OPEN.insert(s,Key(s))
13.     end
14. Function ComputePath():
15.     while (OPEN.TopKey() < Key(s_start) OR rhs(s_start) ≠ g(s_start))
        do
16.         k_old = OPEN.TopKey()
17.         S = OPEN.Pop()
```

```
18.         If k_old < Key(s) then
19.             OPEN.insert(s,Key(s))
20.         Else if g(s) > rhs(s) then
21.             g(s) = rhs(s)
22.             forall s' ∈ Pred(s) do
23.                 UpdateVertex(s')
24.             end
25.         else
26.         g(s) = ∞
27.         forall s' ∈ Pred(s) ∪ {s} do
28.             UpdateVertex(s')
29.         end
30.     end
31. Function Main():
32.     forall s ∈ S do
33.         rhs(s) = g(s) = ∞
34.     end
35.     s_last = s_start
36.     OPEN = o
37.     rhs(s_goal) = o; k_m = o
38.     OPEN.insert(s_goal, Key(s_goal))
39.     ComputePath()
40.     while s_start ≠ s_goal do
41.         s_start = argmin s' ∈Succ(s_start)(cost(s_start, s') + g(s'))
42.         Move to s_start Scan for cell changes in environment (e.g.
            sensor range)
43.         if Cell changes detected then
44.             K_m = k_m + h(s_last,s_start)
45.             s_last = s_start
46.             Forall s ∈ CHANGES do
47.                 Update cell s state
48.                 for all s' ∈ Pred(s) ∪ {s} do
49.                     UpdateVertex(s')
50.                 end
51.             end
52.             ComputePath()
```

53. end
54. end

B: Obstacle Avoidance

Obstacle avoidance is the use of decisions made by an autonomous vehicle to avoid obstacles. This is critical in many sectors where the unmanned vehicles need to detect and react to any obstacles while it is flying above a city or moving on the ground to deliver the various things to human, Obstacle avoidance is distinguished from path planning because it is normally implemented as a reactive control law, while path planning involves the pre-computation of an obstacle-free path which a controller will then guide a robot along.

One of the main algorithms used to solve obstacle avoidance problem is the artificial potential field (APF).

Algorithm 4: APF pseudocode [31]

---

1. Procedure APF(q_o,q_f,map,k_a,k_r,η,∈,M)
2. safe ⇐ True
3. i ⇐ 0
4. fitValue ⇐ 0
5. $d_a \Leftarrow \|q_f - q_0\|$
6. while d_a > ∈ and i < M and safe do
7. $U_{total}(q(i)) \Leftarrow U_{att}(q(i)) + \sum_{j=1}^{n} U_{rep}(q(i))_j$
8. $F(q(i)) \Leftarrow -\nabla U_{total}(q(i))$
9. q(i + 1) ⇐ q(i) + η * F(q(i))/|| F(q(i)) ||
10. $d_{a \Leftarrow}$ $\|q_f$ - q(i+1) ||
11. fitValue ⇐ fitVale + || q(i + 1) – q(i) ||
12. If $\rho \leq r + r_{obst}$ then
13. safe ⇐ False
14. end if
15. i ⇐ i + 1
16. End while
17. $Q_G \Leftarrow$ [q(0),q(1),q(2),…,q(i)]
18. nConf ⇐ i
19. If $d_{a \leq}$ ∈ then

20. goal ⇐ True
21. else
22. goal ⇐ False
23. End if
24. Return [Q_G, fitValue, nConf, goal]
25. end procedure

The following figure shows an illustration of the APF algorithm in obstacle avoidance application.

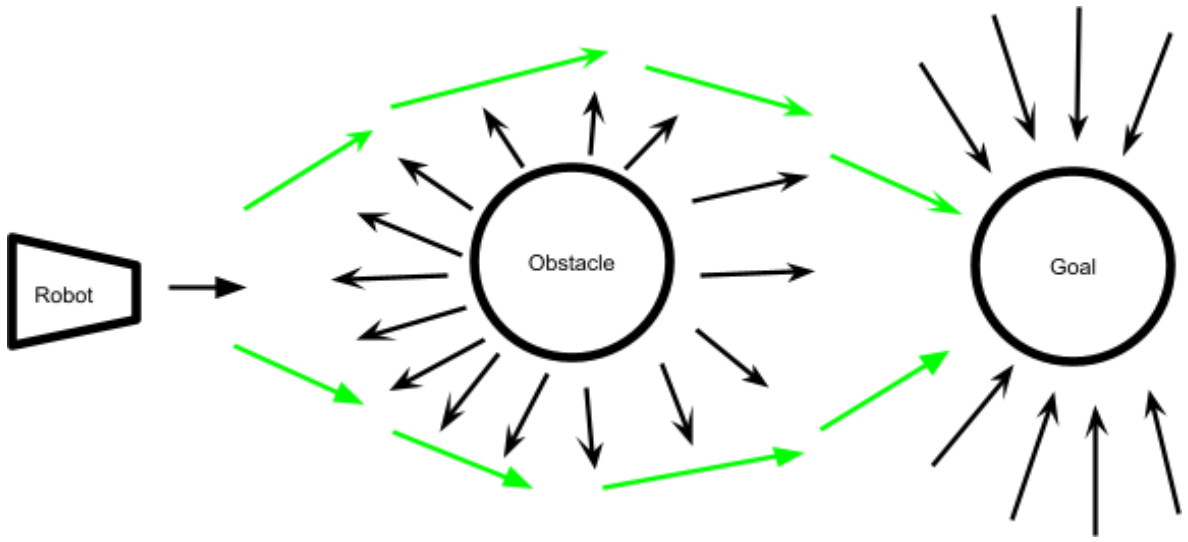


Figure 2.4: illustration of APF algorithm

2.2.2 Sensors

1. GPS sensor

Global Positioning System (GPS) is a navigation system based on satellite, the GPS receiver determines its own location by measuring the time it takes for a signal to arrive at its location from at least four satellites[10], figure 2.5 shows how GPS works

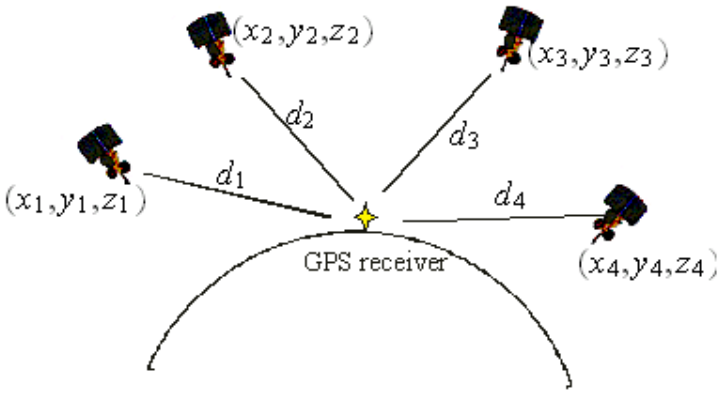


Figure 2.5: Global positioning system

GPS receiver catches the signal came from satellites and analyze it as shown in the following figure

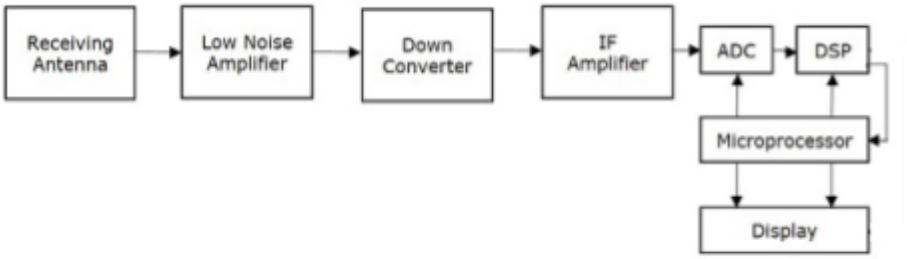


Figure 2.6: GPS receiver components

Receiving Antenna receives the satellite signals.
Low Noise Amplifier amplifies the weak received signal.
Down converter converts the frequency of received signal to an Intermediate Frequency (IF) signal.
IF Amplifier amplifies the Intermediate Frequency (IF) signal.
Analog to Digital Converter (ADC) converts analog signals to digital.
Digital Signal Processor (DSP) generates C/A code.
Microprocessor make the position calculations from C/A code and control timing of the digital blocks in order to give accurate and valid data to use.

2. Inertial Measurement Unit

Inertial Measurement Unit (IMU) is an electronic device that is used to measure and report an object's specific force (acceleration) and angular rate (rotation) with respect to a reference frame.

IMU typically consists of three primary components:

1. Accelerometers: These sensors measure the acceleration forces along different axes. They can detect changes in linear motion or acceleration in any direction. Accelerometers use the principles of microelectromechanical systems (MEMS) to sense changes in capacitance, piezoelectric effects, or other physical phenomena.

2. Gyroscopes: Gyroscopes are used to measure angular velocity or rotational rate. They detect changes in rotational motion around the axes of the device. Like accelerometers, MEMS-based gyroscopes are commonly used due to their small size and cost-effectiveness

3. Magnetometers (optional): Some IMUs also incorporate magnetometers to measure the strength and direction of the magnetic field. Magnetometers are used to determine the orientation of the IMU relative to the Earth's magnetic field and can provide a heading reference. This is particularly useful for applications such as compass navigation[11].

IMUs are frequently used in combination with other sensors like GPS (Global Positioning System) to enhance navigation accuracy. By combining data from different sensors, IMUs can compensate for the limitations of individual sensors and provide a more robust estimation of motion parameters figure 2.7 illustrates the working concept of IMU.

Also calibration is needed to make the magnetometer read the true magnetic field of the earth neglecting all noises.

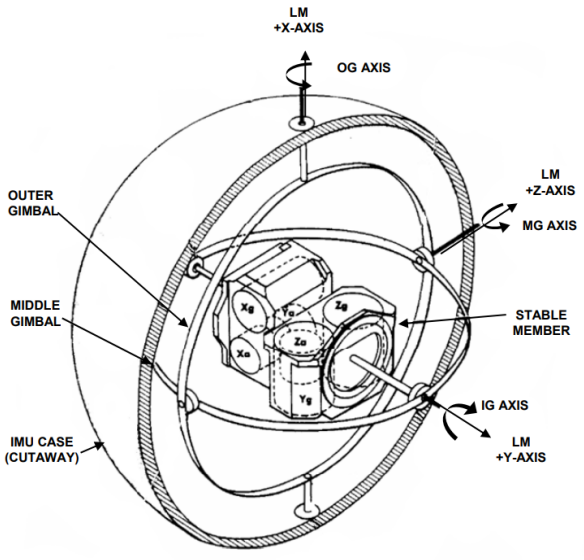


Figure 2.7: Inertial Measurement Unit[11.1]

3. LiDAR sensor

Light Detection and Ranging (LIDAR), LiDAR can detect the distance by sending a laser beam from transmitter then receive the reflected beam and calculating the time that light takes to reflect from surrounding objects[12], figure 2.8 shows LiDAR components and how they work.

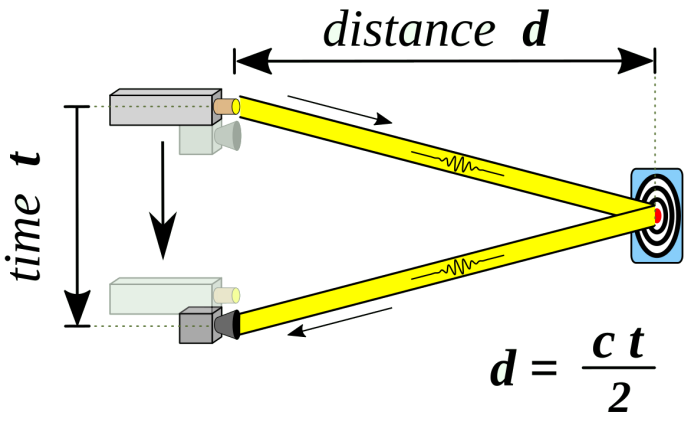


Figure 2.8: LiDAR

In order for LiDAR to scan the environment it sends laser beams in a specific range, usually 360 degrees, and combines the reflected beams dots to create a clear map of the environment as shown in the following figure.

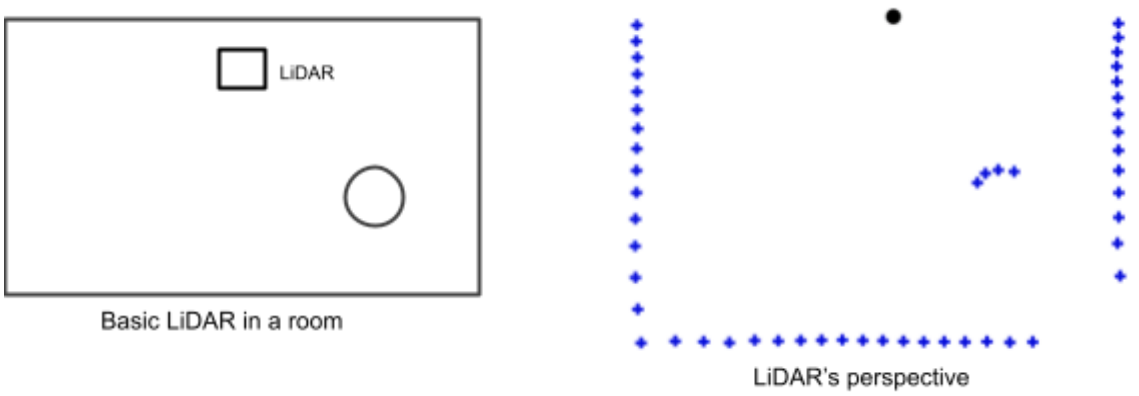


Figure 2.9: illustration of how LiDAR works

2.3 Literature Review

In this part, we show various projects that are related to the concept of our project.

1. Real-world postal services auxiliary delivery bot application

The authors of this project [14] provide a robotic system designed to assist postal workers by carrying heavy parcels in a complex urban environment such as an apartment complex. Since most of these regions do not have access to reliable GPS signal reception, we propose a 3D cloud map based on translation matching with robust location estimation along with a visual servo-based algorithm. The delivery robot is also designed to communicate with the control center so that the operator can monitor the current and past status, the postal worker can choose between autonomous driving mode and follow-up mode using his mobile device. To validate the performance of the proposed robot system, as a conclusion the robot can navigate with reliable position accuracy and ability to avoid obstacles.

2. Autonomous outdoor robot design

The authors of this study [15] present the design of a six-wheeled external mobile robot. The six-wheeled robot platform is equipped with some sensors, such as GPS, HD webcam, light detection and ranging (LiDAR) and rotary encoders. A personal laptop computer and an 86Duino ONE microcontroller were used as the algorithm computing platform. In terms of control, the lateral offset and head angle offset of the robot were calculated using a differential GPS or camera to detect the structured and unstructured road boundaries. Lateral offset and head angle offset were fed to a fog controller. The control inputs are designed by Q-Learning for the speed differential between the left and right wheels. This made the robot follow a reference path so it could stay in its own lane. 2D LiDAR was also used to measure the relative distance from the frontal obstacle. The robot will stop immediately to avoid collision when the distance between the robot and the obstacle is less than a specified safety distance. The specially designed swing arm gave the robot the ability to climb a low step. Body balance can be maintained by controlling the angle of the rocker arm when the robot changes position. (Kao, I et al. #).

3. Autonomous social robot navigation in unknown urban environments using semantic segmentation

The authors of this paper [16] present an independent navigation approach for unknown urban environments that combines the use of semantic segmentation and LiDAR data. It does not require a prefabricated map approach and provides a 3D understanding of the safe zone for travel, enabling the robot to plan any route through the pedestrian path. Thus, it is better utilized with a success rate greater than 91% outdoors, and more than 66% indoors. their method has enabled the robot to remain on a safe travel path at all times, and has reduced the number of collisions.

4. Campus Delivery Robot

Authors of this project [17] aim to develop an autonomous delivery robot for the WPI campus, capable of transporting packages, food, equipment, and other items from one location to another. To complete deliveries, the robot uses an innovative reversible belt conveyor system and navigates using routing, trajectory tracking, and collision avoidance.

5. Development of Reduced Human Intervention Delivery Robot for Covid-19 Pandemic

Authors [18] used Abstract-Artificial Intelligence to create a delivery robot for the Covid-19 epidemic. In order to eliminate human interaction, they developed a robot to deliver items. Their delivery robot uses artificial intelligence to identify the best route.

6. Design and Development of Autonomous Delivery Robot

Authors [19] aim to make the robot fully autonomous and competent to work with humans, the robot must be able to perceive the situation and devise a plan for smooth operation, considering all the adversities that may occur while carrying out the tasks. In this thesis, they present an autonomous mobile robot platform that delivers the package within the VNIT campus without any human intercommunication. From an initial user-supplied geographic target location, the system plans an optimized path and autonomously navigates through it.

2.4 Summary

We came to the following conclusion after reviewing studies: that the delivery of small parcels is a repetitive and tedious process as well as it costs time and human energy, so it has become less efficient, in addition to the possibility of not care health and safety of the parcel is high ,and at the same time there are robots that carry out the transportation process, but they need help Humans to monitor and correct their behavior and sometimes complete the task manually, and some of them have the ability to climb obstacles and pass over them, and this matter is complex and requires high costs to accomplish, and there are some artificial intelligence is added to it to choose the best path to reach, but this requires a long period of training for the robot and remains prone to errors there Studies are looking at the ability of the robot to walk in the place designated for walking without entering places where it cannot be walked, so we will do our project in a simple way that provides the required task to be appropriately effective and at the lowest cost so that it moves completely independently in the external environment inside the university campus, our robot He will find a group of paths and choose the shortest path possible, which increases his efficiency.

Chapter 3: System Design

3.1 Preface

In this chapter we will introduce the system design starting with system components (software and hardware), design options, conceptual system description and algorithms and system methodology.

3.2 System components and design options

3.2.1 software components

1. Robot Navigation Technique

The robot navigation is denoted as a combination of three main parts:
Map-building (mapping), Localization and Path planning[20]. So the robot need to build a map that describes the environment around the robot, that's the map-building part, and it need to know its position (location and orientation) inside the map, and that called the localization, finally the robot should be able to determine a path leads to a goal point in the map and it's called path planning.

In the following sections we will demonstrate each step (Mapping, localization, path planning), and we are going to explain our approach to navigate the robot to its destination.

A. Map-building(Mapping)

Mapping is a technique used to create a map, update it, and estimate the location of the robot in order to map the surroundings.

There are various mapping packages used in ROS framework, each mapping package has its own approach, such as RTAB-MAP(Real-Time Appearance-Based Mapping) package[21], hard to implement in our project, Another package called Gmapping fits more, the slam_gmapping node takes in sensor_msgs/LaserScan and odometry data messages and builds a map (nav_msgs/OccupancyGrid) [22].

Our approach to build an initial map by controlling the robot with joystick and rover it all over the environment (university campus in our case) that will give us a detailed grid map with respect to stationary obstacles, we still have a coordinate problem and that's because the grid coordinates of the local map will interfere with the GPS coordinates reading.

To solve this problem we have to do kind of fusing the GPS and odometry readings to get the x,y coordinates in our local map represented from the GPS latitude and longitude, in order to do that there's a node in *robot_localization* package called *navsat_transform_node*[23], which takes the GPS raw inputs and change them to our local grid map.

B. Localization

Localization is simply the ability of a robot to identify its position (location and orientation) in a certain map. We have a local stored map that was manually built from the previous section(Mapping).
In order to find the robot's location at any time from odometry and GPS readings, the best choice is to work with the robot_localization package from ROS.

Robot_localization package consists of various nodes that work together to provide nonlinear location estimation, it is using ekf_localization_node(extended kalman filter localization node) to determine robot location inside the map, in addition there is navsat_transform_node which helps in data integration when GPS is used.

In our case the GPS readings will be transformed to grid coordinates via navsat_transform_node then it will get integrated with odometry data via ekf_localization_node to generate the location estimation.

C. Path planning

The Robot can construct the ideal path between the current Robot position and the destination while avoiding obstacles by searching for the target location by using the produced map.

Move_base is a software module used in robotics for autonomous navigation. It enables mobile robots to plan paths, avoid obstacles, and reach desired goals. It incorporates global and local planning algorithms, sensor integration, and motion control [24].

2. Web application

Web app is essential part of our project, it links the high end user with the Robot, the application interface will be designed using React [25], and linked with a web server to transfer the data between the robot and the user via REST Api, app layout will provide the user with robot location, availability and list of users, For the backend database MongoDB [24.1] will be used.

3.2.2 hardware components

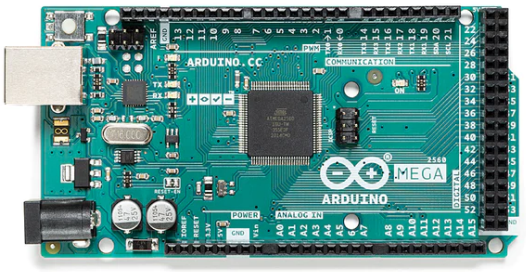
We will show hardware components we are going to use in the project, and other alternatives, our characteristics to choose a certain component will be as following:

1. Availability
2. Efficiency
3. Cost

1. Microcontroller

Microcontroller is an essential part of the project, we need it to transfer data from external sensors to the robot

We will demonstrate 2 microcontrollers in table 3.1, to choose one of them



(a) *(b)*

Figure 3.1: (a): arduino uno *(b): arduino mega 2560*

Table 3.1: Microcontroller options

	Arduino uno	Arduino mega 2560
Dimensions cm	6.858 x 5.334	10.16 x 5.334
processor	Atmega328p	Atmega2560
Flash memory KB	32	256
EEPROM KB	1	4
SRAM KB	2	8
Voltage support Vs	5	5
Digital I/O pins	14	54
Digital I/O with PWM pins	6	15
Analog pins	6	16
USB connectivity	Standard A/B USB	Standard A/B USB
Cost	~ 21$	~ 40$

Both Microcontrollers are available, and efficient in this thesis
We had chosen Arduino Uno due to the lower price.

2. Mobile Robot

(a)

(b)

Figure 3.2 :(a) Neobotix MP-500 (b) Neobotix MP-400

Table 3.2: robot options

	Neobotix MP-500 [26]	Neobotix MP-400 [27]
description	Compact, industrial grade robot for material flow and intralogistics. Very good system for robotics research when equipped with a robot arm.	Small, agile robot for material flow and intralogistics in industrial applications. Automatic charging station and load handling system available.
speed	1,5 m/s	1,5 m/s
Working environment	outdoor	indoor
Drive system	Differential drive	Differential drive
Dimensions (in mm)	814 x 592 x 361 (LxWxH)	590 x 559 x 411 (LxWxH)
Payload	80 kg	100 kg

Table 3.3: characteristics for mobile robot

	MP-500	MP-400
availability	Available	Available
Efficiency	Efficient	Inefficient
Cost	~20000$	~20000$

MP-500 was chosen based on the previous table, due to efficiency.

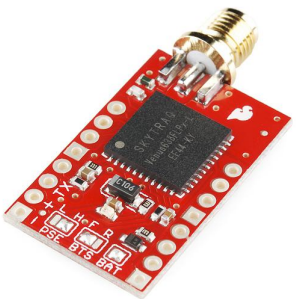
3. LiDAR sensor

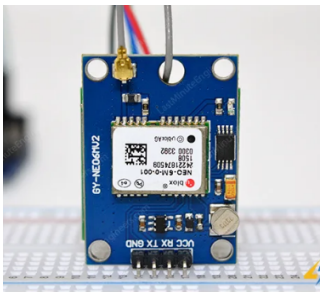
Figure 3.3: S30B-2011BA

Table 3.4 : LIDAR sensor options

	S30B-2011BA
Working range in meters	30
Working range in degrees	H~360° V~0°
Angular resolution	0.5°
Supply voltage Vs	24 V DC (16.8 V DC ... 30 V DC)

5. GPS sensors





(a) *(b)*

Figure 3.4:(a) Venus-GPS (b) NEO-6M

Table 3.5: GPS sensor options

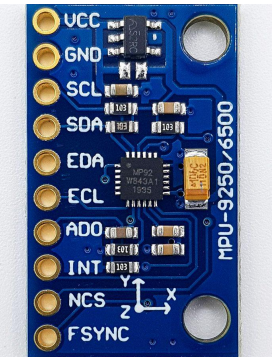
	SparkFun Venus GPS	Interface ublox NEO-6M GPS Module
Operating Temperature	-40°C ~ 85°C	-40°C ~ 85°C
Number of Channels	22	50
Navigation Sensitivity	-165dBm	-161dBm
Accuracy	~2.5m	~2.5m

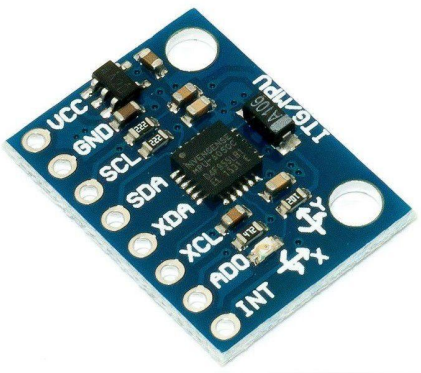
Table 3.6: characteristics for GPS receivers

	SparkFun Venus GPS	Interface ublox NEO-6M GPS Module
Availability	Not available	Available
Efficiency	Efficient	Efficient
Cost	~50$	~10$

We had chosen NEO-6M due to availability.

6. Inertial measurement unit (IMU)





(a) *(b)*

Figure 3.5: (a): MPU 9250 IMU (b): MPU 6050 IMU[32]

Table 3.7: IMU options

	mpu 9250	mpu 6050
communication	I2C interface	I2C interface
Sensor Integration	Accelerometer, gyroscope and magnetometer(compass)	Accelerometer and gyroscope
Supply voltage Vs	3.3V-5.0V	h 3.3V-5.0V
Output rate Hz	1 kHz	1 kHz
cost	~ 8$	~ 9$

Table 3.8: characteristics for IMU

	mpu 9250	mpu 6050
Availability	Available	Available
Efficiency	Efficient	Efficient
Cost	~ 7.29$	~ 8.16$

They don't have too much difference, we had chosen mpu 9250 due to the additional compass sensor.

3.4 Conceptual system description

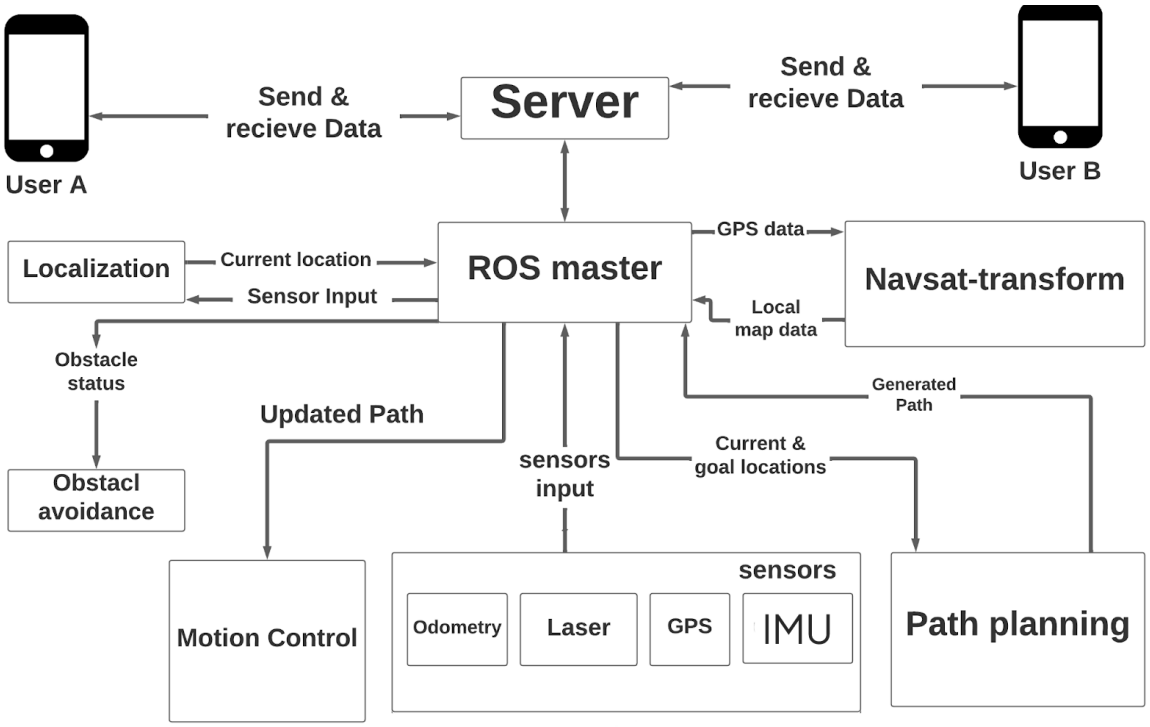


Figure 3.6: System Block diagram

The previous figure shows detailed block for the system were the hardware components are the motors in the **motion control component** takes the the path from the main program **ROS master component** and the **sensors component** send the robot observations for the **ROS master,** the software part starts with **user A** sends a request to **user B** via **server** if the robot available and user B accepted the request, the user B location's will be sended to **ROS master** via **server, ROS master** sends target location and GPS reading to **navsat-transform** component to mirror this data to the local map, after that **ROS master** sends **sensors**(modified GPS, Laser, Odometry,IMU) readings to **Localization** component to respond with Robot's location, then **ROS master** sends start and goal points to **Path planning** component to acknowledge with a generated path that will be sended to **Motion control** component to start moving, if the robot observe an obstacle during the journey **ROS master** will provide **Path planning** with the new data to regenerate a path within the free space and return it to **motion control** via **ROS master, meanwhile ROS master** component keep updating robot location to both users via **the server,** if **ROS master** receive any object breaks the safety distance around the robot from **sensors**(Laser, ultrasonic), it will order **motion control** to stop the robot

motion for a while and start move away from the object till the safety distance is clear to avoid collision, then deal with the object as an obstacle

3.5 Algorithms and methodologies

Pseudocode for the system

```
1.  start
2.  Reached = false
3.  Goal = target location
4.  Location = Initial localization
5.  Path planning
6.  If (target reachable) then //global path planning
7.      while (!Reached)
8.          Location = localization
9.          Follow the path
10.         Avoid obstacles
11.         If (Location == Goal)
12.             Reached =  true
13.         Delay (small time)
14.     End while
16. end
```

3.6 Schematic diagram

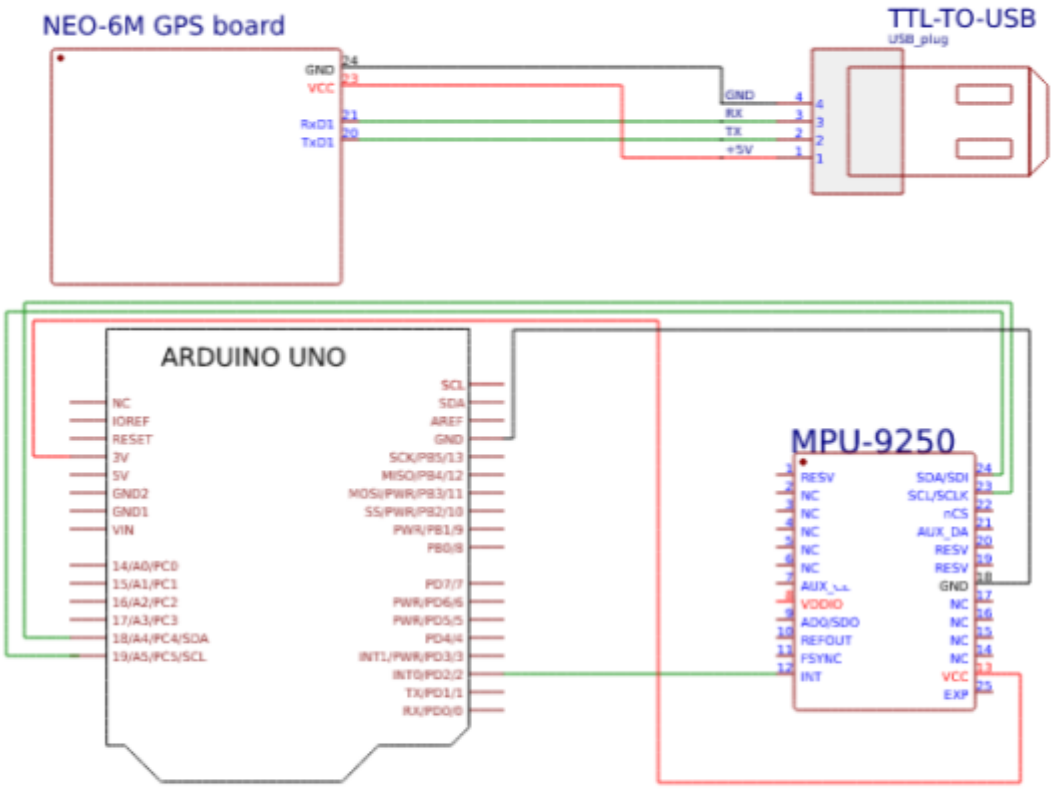


Figure 3.7: schematic diagram

3.7 Summary

We have come to the conclusion that the best framework for a robot is ROS Provides packages and tools to assist software developers in developing robotics applications And it has many advantages, after we compared a range of options we had come up with the best options that perform the required with the best efficiency and hardware performance possible, and we have come up with a clear and detailed picture of how the robot works with the sequential steps and algorithms that it needs from the moment the user communicates with our robotic system, the way he interacts with the surrounding.

Chapter 4: System Implementation

4.1 Preface

This chapter provides a comprehensive overview of the software and hardware implementation of the project, along with a detailed discussion on the essential components and tools required for constructing the robot.

4.2 Hardware implementation

The main part is the Robot which is connected to the other components as follows:

4.2.1 laptop linking

We connect our laptop to the robot via a local network to use the robot as a remote desktop through a wifi access point, next figure shows the access point that was used.

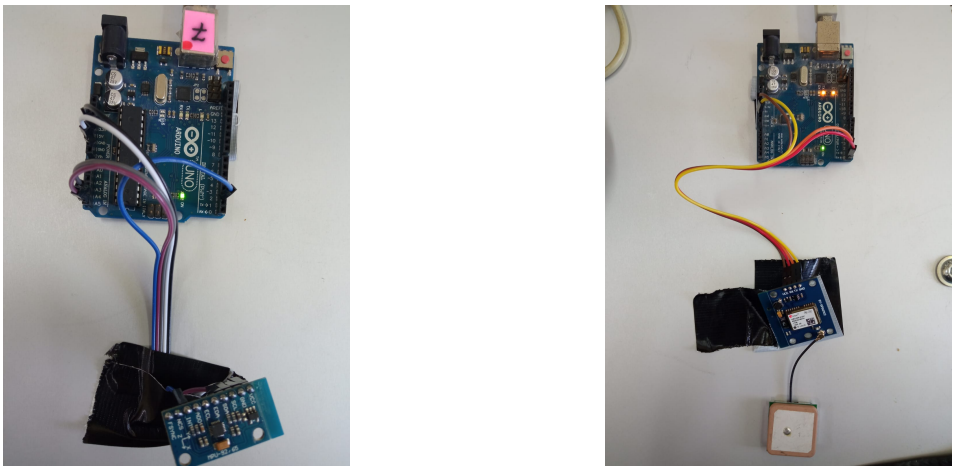
Figure 4.1: access Point link laptop with the robot

4.2.2 IMU module wiring

We connect the mpu 9250 imu module to arduino board as mentioned in schematics, the serial part from module to arduino respectively as follows: SCL to A4 SDA to A5 INT to 2 and Vcc to 3.3V, GND to GND, then we connect arduino board to the robot through serial usb cable, figure 4.2 shows the wiring.

4.2.3 GPS module wiring

We connect the NEO 6M GPS module to ttl-to-usb board as mentioned in schematics, the serial part from module to ttl-to-usb respectively as follows: RX to RX, TX to TX, Vcc to 5V, GND to GND, then we plug the ttl-to-usb to the robot through a usb port, figure 4.2 shows the wiring.



(a) (b)

Figure 4.2:(a) IMU wiring,(b) GPS wiring

4.3 Software implementation

The main part is the ROS master which is which run and control all the software components as follows:

4.3.1 Operating system and Framework

Ubuntu 20.04 LTS (focal fossa) was used as an operating system, ROS noetic was used to control the robot.

4.3.2 Robot modeling

Robot modeling refers to the process of creating a representation or simulation of a robot using mathematical and computational techniques.

The robot model is usually described using the Unified Robot Description Format (URDF), which is an XML-based file format, Luckily neobotix provides a robot description in URDF for their robots.

We imported the mp-500 model from neobotix repositories then we visualized it on Rviz, a 3D visualization tool provided by ROS, RViz allows you to display the robot's geometry, joints, and other sensor data, the next figure shows how the robot model is represented inside Rviz.

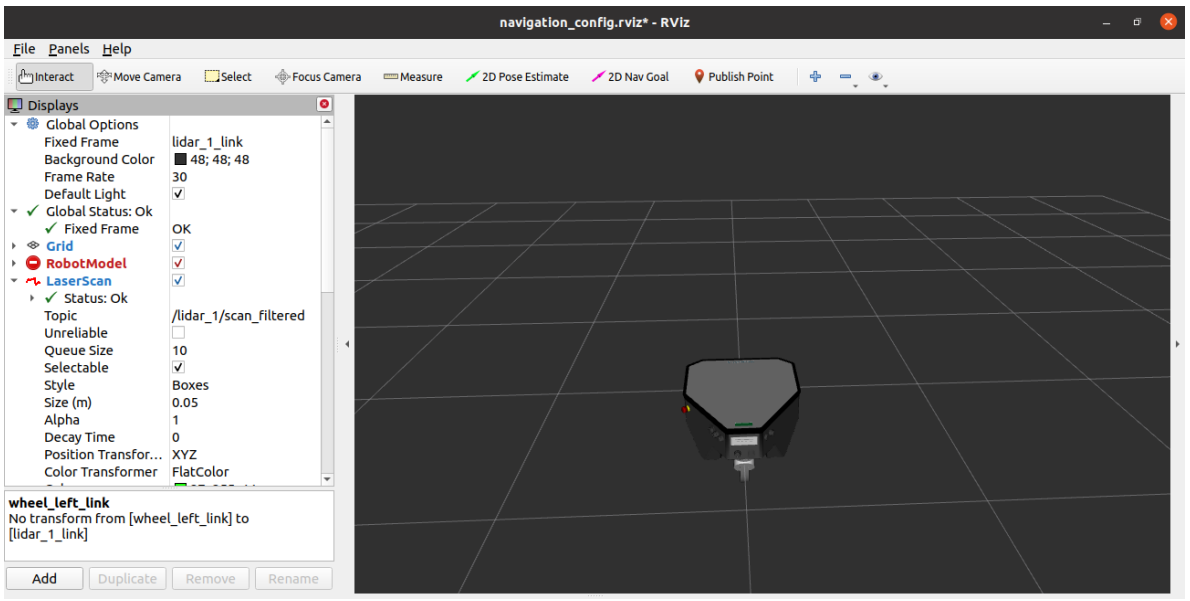


Figure 4.3: robot model representation in Rviz

4.3.3 Moving the robot

In order to move the robot we used two packages:

- Teleop keypad package, a package that enables you to control the robot by keyboard keys by publish /cmd/vel topic to move_base, we used it to control the robot in simulation.
- Joystick launched works with the same concept but with joystick instead, we used it to control the robot in reality.

4.3.4 Creating map

We used Gmapping to build the map by taking readings from lidar, odometry, IMU, GPS (for outdoor only).
First we installed slam_gmapping package by

1. Clone it from github with this command:
   `git clone https://github.com/ros-perception/slam_gmapping.gi/`

2. Install the following dependencies:
   nav_msgs, openslam_gmapping, roscpp, rostest, tf, nodelet.

4.3.5 Navigation

We used 4 packages for navigation

1. AMCL package which is used to localize the robot by observing the environment and comparing observations with the saved map, so it uses lidar readings only, the following command was used to install the package: `sudo apt install ros-noetic-amcl` .
2. was robot_localization, it is built on ekf_localization approach which take readings of odometry, IMU, GPS to and fuse the reading to get a position estimate, installing command:
   `git clone https://github.com/cra-ros-pkg/robot_localization.git`

3. move_base which takes a goal then does the path planning, obstacle avoidance and control robot motion installing command:

`sudo apt install ros-noetic-navigation`.

4. Neo_goal_sequence_driver takes multiple goals and publish them to move_base in the same order, installing command:
   `git clone https://github.com/neobotix/neo_goal_sequence_driver.git` .

4.3.6 Rosserial

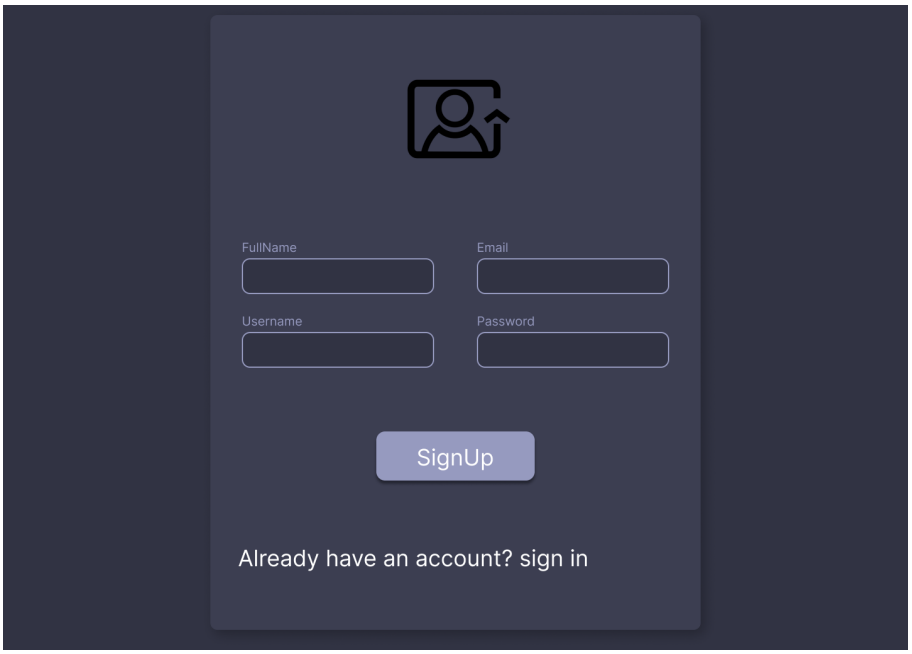
rosserial is a ROS package that enables communication between a host computer and microcontrollers for embedded systems. It provides a lightweight communication protocol that allows ROS nodes on the host computer to exchange messages and data with nodes running on microcontrollers.
We used the package to read from usb ports (ACM0 for IMU, ACM1 for GPS) and publish the data in specific topics.
Package install commands: `sudo apt install ros-noetic-rosserial-arduino`
`sudo apt install ros-noetic-rosserial-python`

4.3.7 Simulation

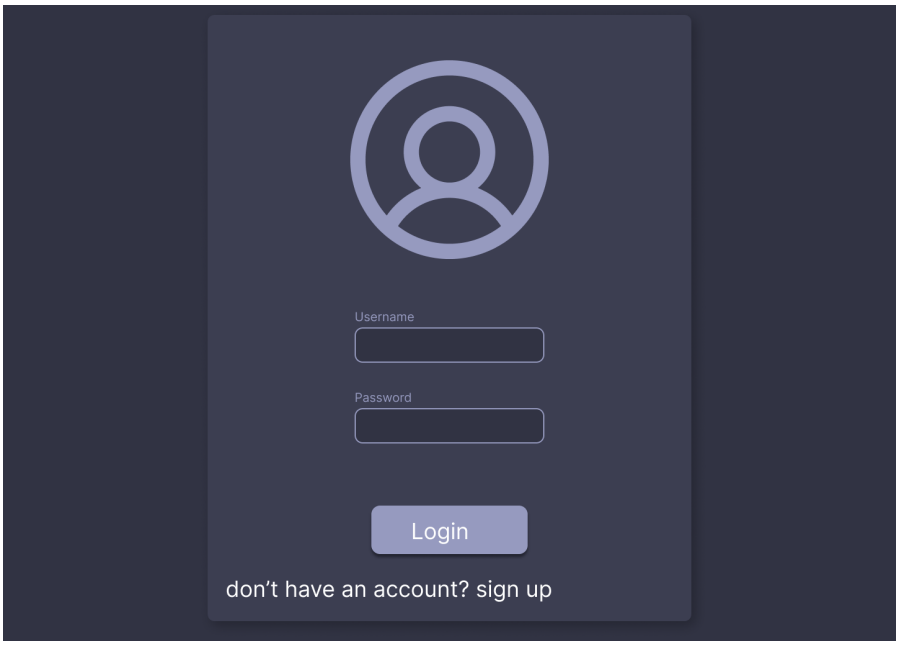
Gazebo is a widely used open-source 3D simulation environment for robotics and autonomous systems. It provides a realistic physics engine and a visually appealing virtual environment where you can simulate and test robots, sensors, and various other components[33].
We used gazebo in order to test and validate every component of the system before applying it to the physical model.

4.4 Web application

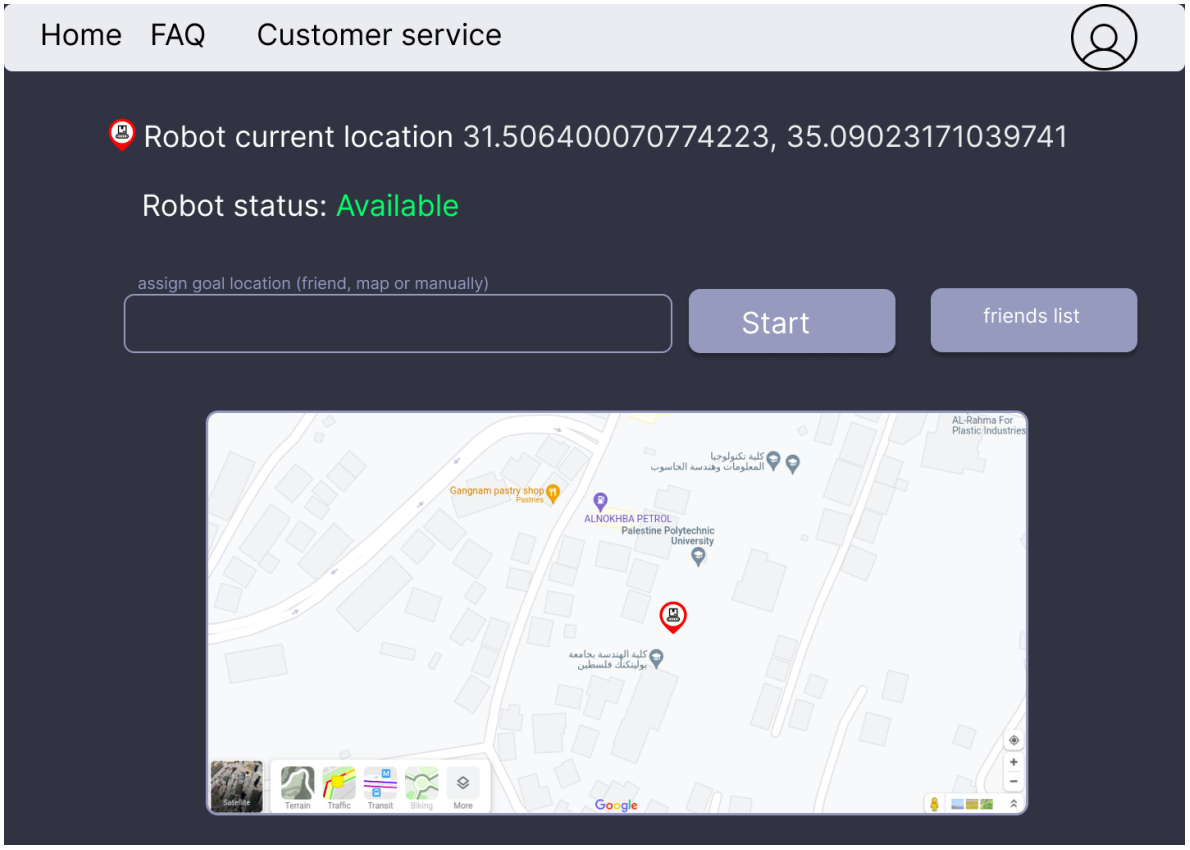
We utilized Node.js to develop a web application, leveraging the React JavaScript library for designing the user interface. React facilitated the creation of a user-friendly interface by utilizing JSX files, which combine HTML and JavaScript seamlessly. For the back-end, we employed the Express JavaScript library to establish endpoints that receive and process user requests. As a NoSQL database, we utilized MongoDB to store user information efficiently, figure 4.4 illustrates the user interface.


FullName
Email
Username
Password
SignUp
Already have an account? sign in

(a)


Username
Password
Login
don’t have an account? sign up

(b)



(c)

Figure 4.4: (a) sign up page, (b) sign in page, (c) Home page

4.5 implementation challenges

- We had a problem of integrating additional components (IMU and GPS) with ROS environment because arduino board couldn't handle to run ROS nodes on its memory; so we connected the components directly to usb ports.
- The low price and inaccuracy of the GPS module led to errors in the map, unfortunately we didn't have any solution to this problem, the only solution was to replace the module with RTK GPS which cost 250$ on average which we couldn't handle.
- Map building process needs the campus to be empty of people and vehicles, we didn't have this chance, the urban environment led to noise in the map at the end.

Chapter 5: Testing and Validation

5.1 Preface

This chapter elucidates the methodology employed for component testing in the project and showcases the outcomes of the system implementation.

5.2 Hardware Testing

5.2.1 Lidar testing

The lidar scanner comes as a configured part built in the robot so we had to make sure it's reading right values; so we echo the /lidar_1 topic which containes the lidar readings, we got those results for one sequence:

seq: 29815
stamp:
secs: 1684317975
nsecs: 683482985
frame_id: "lidar_1_link"
angle_min: -2.3561944496154785
angle_max: 2.3561944496154785
angle_increment: 0.008726646192371845
time_increment: 4.629629620467312e-05
scan_time: 0.039999999105930333
range_min: 0.009999999776482582

And the ranges vector was valid and pretty accurate.

5.2.2 GPS testing

First we run experimental code on the arduino to send the GPS readings to serial screen. It took the module about four minutes to lock on a satellite when it runs under a clear sky, unfortunately the module doesn't work at all inside buildings.

5.2.3 IMU testing

After configuration We had run the MPU on Rviz and it did give us an accurate clear data of its movement and orientation, figure 5.1 shows the test result on Rviz.

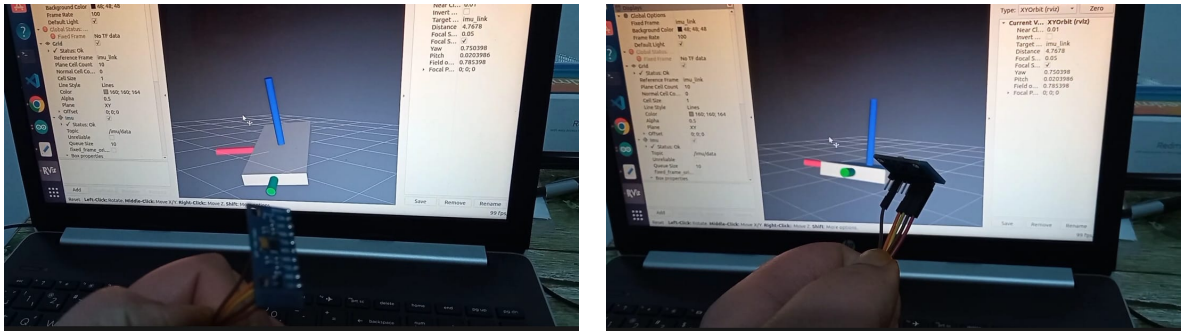


Figure 5.1: illustration of IMU working

5.3 Software testing

5.3.1 Robot launch

We were able to run the main default terminal in the robot and control the robot using a joystick. Everything works as expected including the emergency stop, figure 5.2 shows the main terminal.

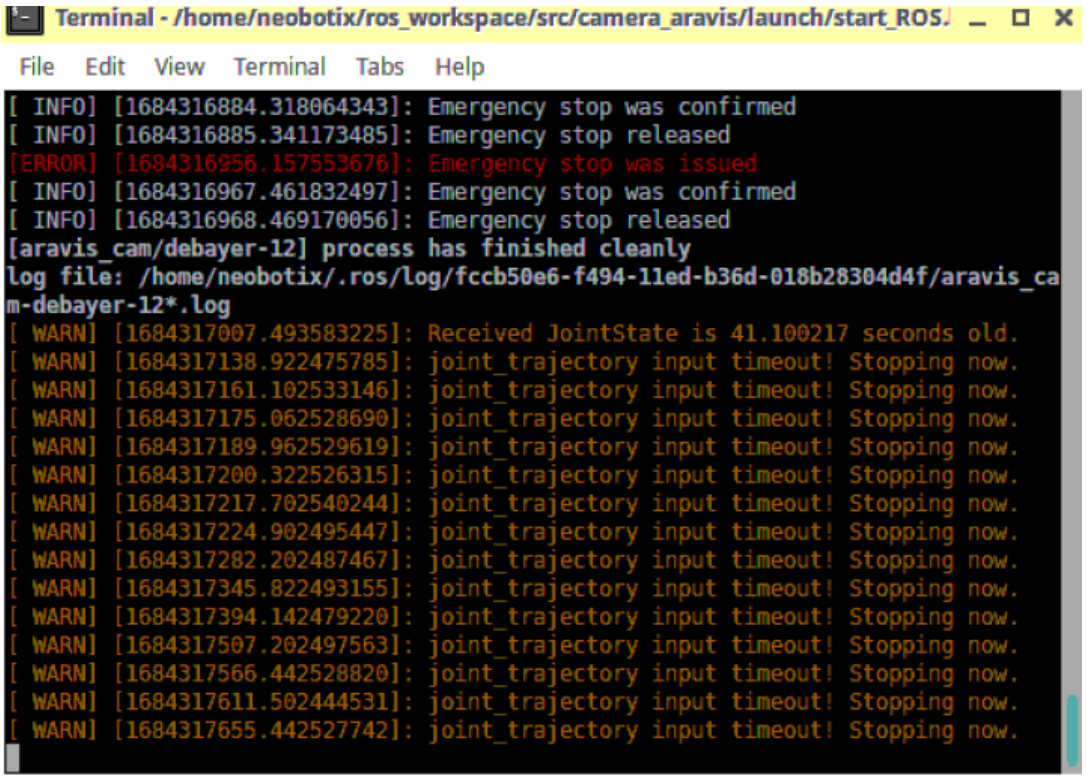


Figure 5.2: main robots terminal

5.3.2 Load robot model to Rviz

We were able to run the robot model to Rviz and visualize the received sensors data, figure 5.3 shows the robot loaded into Rviz.

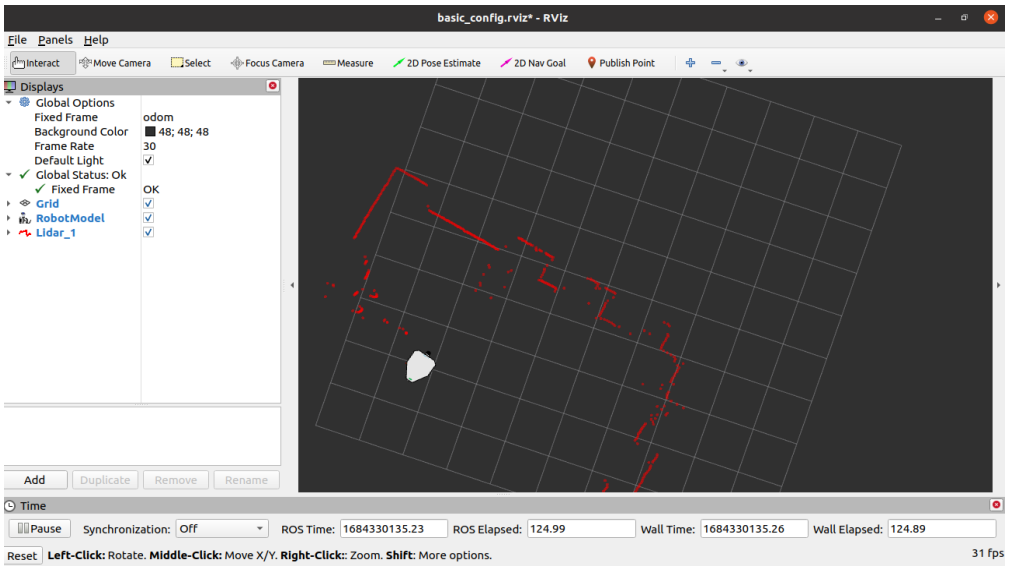
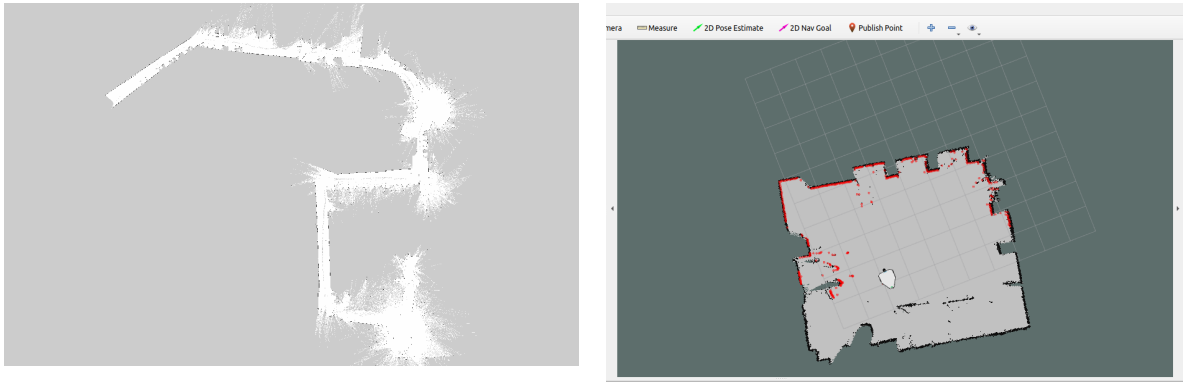


Figure 5.3: mp-500 model inside Rviz

5.3.3 Map testing

We constructed both an indoor and outdoor map utilizing the Gmapping package. The indoor map was created with precision, offering clear and accurate information. However, the outdoor map posed challenges due to the dynamic environment and unreliable GPS readings, resulting in noise and less precise mapping, figure 5.4 shows the result of mapping indoors and outdoors.



(a) *(b)*

Figure 5.4: (a) outdoor map, (b) indoor map

5.3.4 Localization testing

The AMCL package performed well, but it required an initial pose estimate from us. Without the initial estimate, it would provide random location estimations. This limitation was due to AMCL relying solely on lidar readings. However, the robot_localization package excelled in providing precise robot location estimates. Unlike AMCL, it didn't require an initial position estimate. It achieved high accuracy by fusing odometry data with IMU and GPS readings. Figure 5.5 visually demonstrates the clear position estimate produced by the robot_localization package.

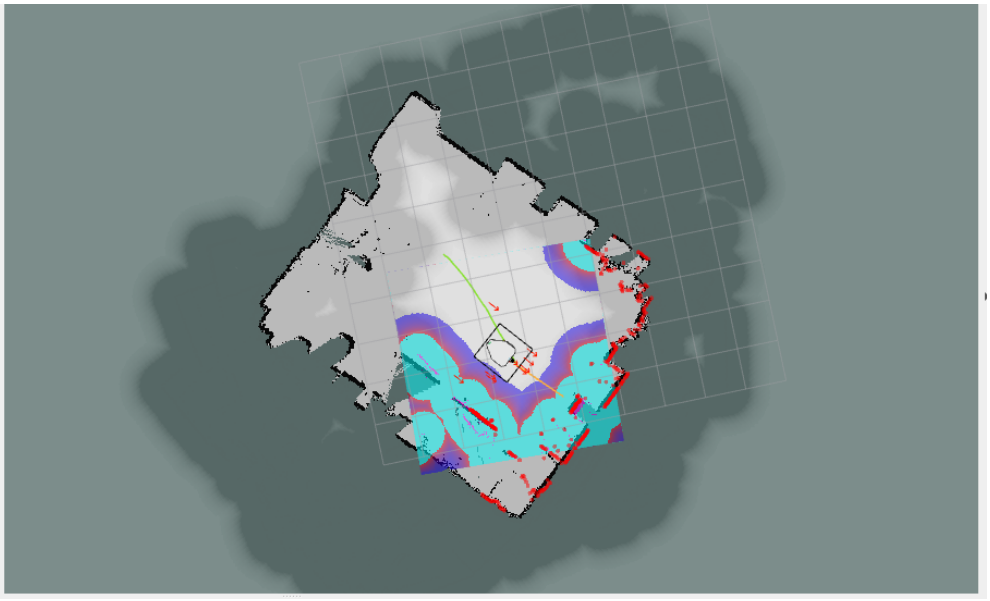


Figure 5.5: illustration of robot_localization working in Rviz

5.3.5 Navigation testing

The move_base package works perfectly as expected once it has a clear map and clear goal coordinates (x,y and theta) it controls robot motion to get a clear view if it wasn't clear then it draws a global path from current to goal locations and it start moving toward the goal, if any new object close the road it will go to reset state then a new local path will be generated to avoid the object safely, also it increase and decrease the speed based on surrounding environment.

5.3.6 Goal sequence testing

Goal sequence package works as expected, it take a goals array that contains objects of goals (x,y and theta), then the goals keep sent in order to move_base, if move_base fails to reach any goal the operation falls, figure 5.6 shows visualization of goal sequence in Rviz.

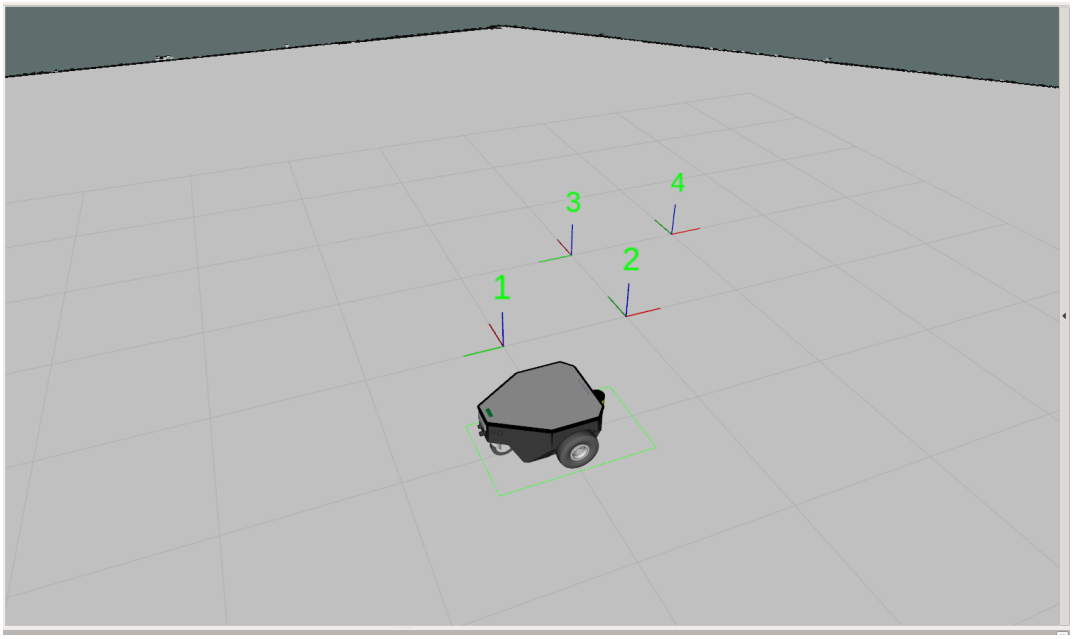


Figure 5.6: Goals array in Rviz

5.4 System validation

Once we conducted individual tests on each component of the system, we proceeded to integrate and run the entire project. During this phase, the system demonstrated satisfactory performance. It successfully acquired sensor readings and processed them to construct a map. Moreover, it accomplished self-localization within the map and effectively navigated to multiple goals, ensuring safe traversal by avoiding collisions.

Chapter 6: Conclusion and Future work

6.1 Conclusion

In this thesis, we proposed a solution for autonomous outdoor package delivery. Through the utilization of fundamental hardware and software components, we successfully developed an autonomous and mobile working prototype. Furthermore, we designed an application that facilitated user interaction with the robot, enabling them to communicate and assign missions. Despite encountering various challenges along the way, the system passed all assigned tests. However, it is important to note that the prototype's performance was hindered by the incorporation of inexpensive components, which served as a significant drawback.

6.2 Future work

Some future works are suggested and recommended to improve the project:

1. Attach additional sensors like camera for example to enhance the overall performance.
2. Add new features to the system like turning back to the charging station autonomously.
3. Add multiple robots to the system and make them work as one group.group

References

[1] *Wikipedia contributors. (2022b, November 8). Simultaneous localization and mapping. Wikipedia. https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping*

[1.1] *Kümmerle, Rainer & Pfaff, Patrick & Triebel, Rudolph & Burgard, Wolfram. (2007). Active Monte Carlo Localization in Outdoor Terrains Using Multi-level Surface Maps. Autonome Mobile Systeme 2007. 29-35. 10.1007/978-3-540-74764-2_5.*

[2] *Se, S., Lowe, D., & Little, J. (2001, October). Local and global localization for mobile robots using visual landmarks. In Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180) (Vol. 1, pp. 414-420). IEEE.*

[3] *Mao, L. (2019, March 21). Introduction to Bayesian Filter. Lei Mao's Log Book. https://leimao.github.io/article/Introduction-to-Bayesian-Filter/*

[4] *Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999, May). Monte carlo localization for mobile robots. In Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C) (Vol. 2, pp. 1322-1328). IEEE.*

[5] *Welch, G. F. (2020). Kalman filter. Computer Vision: A Reference Guide, 1-3.*

[6] *Path Planning. (n.d.). MATLAB & Simulink (May, 2020). https://www.mathworks.com/discovery/path-planning.html?s_tid=srchtitle_Design%2C+simulate%2C+and+deploy+path+planning+algorithms_1*

[7] *Véras, L. G. D., Medeiros, F. L., & Guimarães, L. N. (2019). Systematic literature review of sampling process in rapidly-exploring random trees. IEEE Access, 7, 50933-50953.*

[8] *Wikipedia contributors. (2022a, October 2). A* search algorithm. Wikipedia. https://en.wikipedia.org/wiki/A*_search_algorithm*

[9] *Wikipedia contributors. (2022c, October 22). D*. Wikipedia. https://en.wikipedia.org/wiki/D**

[10] *El-Rabbany, A. (2002). Introduction to GPS: the global positioning system. Artech house.*

[11] *Treffers, C., & van Wietmarschen, L. (2016). Position and orientation determination of a probe with use of the IMU MPU9250 and a ATmega328 microcontroller.*

[11.1] *Inertial measurement unit - Wikipedia. (2022, September 2). Inertial Measurement Unit - Wikipedia. https://en.wikipedia.org/wiki/Inertial_measurement_unit#/media/File:Apollo_Inertial_Measurement_Unit.png*

[12] *Behroozpour, B., Sandborn, P. A., Wu, M. C., & Boser, B. E. (2017). Lidar system architectures and circuits. IEEE Communications Magazine, 55(10), 135-142.*

[13] *Kelemen, M., Virgala, I., Kelemenová, T., Mikova, L., Frankovský, P., Lipták, T., & Lörinc, M. (2015). Distance measurement via using of ultrasonic sensor. Journal of Automation and Control, 3(3), 71-74.*

[14] Lee, Daegyu, et al. "Assistive Delivery Robot Application for Real-World Postal Services." *IEEE Access* 9 (2021): 141981-141998.

[15]Kao, I-Hsi, Jian-An Su, and Jau-Woei Perng. "Design of Outdoor Autonomous Moble Robot." *arXiv preprint arXiv:2201.12605* (2022).

[16]Buckeridge, Sophie, et al. "Autonomous social robot navigation in unknown urban environments using semantic segmentation." *arXiv preprint arXiv:2208.11903* (2022).

[17]Buermeyer, Lucas. *Campus Delivery Robot*. Diss. WORCESTER POLYTECHNIC INSTITUTE, 2022.

[18]Chethan, D. S., et al. "Development of Reduced Human Intervention Smart Delivery Robot for Covid-19 Pandemic." (2021).

[19]Gujarathi, Aniket, et al. "Design and Development of Autonomous Delivery Robot." *arXiv preprint arXiv:2103.09229* (2021).

[20] Wikipedia contributors. (2022, October 8). Robot navigation. Wikipedia. https://en.wikipedia.org/wiki/Robot_navigation

[21] rtabmap (2019, January, 10)- ROS Wiki. http://wiki.ros.org/rtabmap

[22] gmapping - ROS Wiki.. Gmapping - ROS Wiki. Retrieved May 20, 2023, from http://wiki.ros.org/gmapping

[23] navsat_transform_node — robot_localization 2.6.12 documentation. (May 2020.). http://docs.ros.org/en/melodic/api/robot_localization/html/navsat_transform_node.html

[24] move_base - ROS Wiki. (n.d.). Move_Base - ROS Wiki. Retrieved May 20, 2023, from http://wiki.ros.org/move_base

[24.1] MongoDB Atlas: Cloud Document Database. (n.d.). MongoDB. Retrieved May 20, 2023, from https://www.mongodb.com/cloud/atlas/lp/try4

[25] Kuitunen, M. (2019). *Cross-Platform Mobile Application Development with React Native* (Bachelor's thesis).

[26] *Neobotix: Mobile Robot MP-500. (2019). https://www.neobotix-robots.com/products/mobile-robots/mobile-robot-mp-500*

[27] *Neobotix: Mobile Robot MP-400. (2019). https://www.neobotix-robots.com/products/mobile-robots/mobile-robot-mp-400*

[28] *Balasuriya, B. L. E. A., Chathuranga, B. A. H., Jayasundara, B. H. M. D., Napagoda, N. R. A. C., Kumarawadu, S. P., Chandima, D. P., & Jayasekara, A. G. B. P. (2016, April). Outdoor robot navigation using Gmapping based SLAM algorithm. In 2016 moratuwa engineering research conference (mercon) (pp. 403-408). IEEE.*

[29] *Frese, U. (2005, April). A proof for the approximate sparsity of SLAM information matrices. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (pp. 329-335). IEEE.*

[30] *Search-Based Planning and Replanning in Robotics and Autonomous Systems,(September 2018).* https://www.researchgate.net/publication/327985957_Search-Based_Planning_and_Replanning_in_Robotics_and_Autonomous_Systems

[31] *Orozco-Rosas, U., Montiel, O., & Sepúlveda, R. (2019). Mobile robot path planning using membrane evolutionary artificial potential field. Applied Soft Computing, 77, 236–251.* https://doi.org/10.1016/j.asoc.2019.01.036

[32] *MPU-6050 | TDK InvenSense. (2021). TDK InvenSense. Retrieved May 21, 2023, from* https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

[33] *Gazebo simulator - Wikipedia. (2019, March 24). Gazebo Simulator - Wikipedia.* https://en.wikipedia.org/wiki/Gazebo_simulator