Autonomous Delivery System for Hospitals and Hotels

Akshat, Yash Agrawal, Rahul Gupta, Abhishek Kumar, Vivek Ashok Bohara
Indraprastha Institute of Information Technology Delhi (IIIT-Delhi)

Abstract

- Implementation of a prototype of a delivery robot using ROS (Robot Operating System) framework that maps its surroundings and autonomously navigates to the destination provided.
- Algorithms like Hector SLAM and AMCL are integrated for mapping and pose identification of the mobile robot, and Dynamic-Window Approach (DWA) Planner helps calculate the shortest path in the space available for movement.
- Implementation of *differential drive* odometry.
- The working is demonstrated through mapping and localization in a physical lab space.

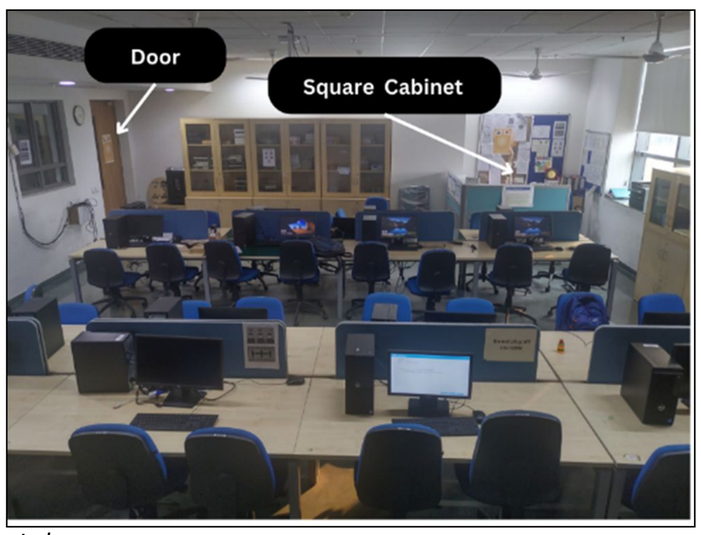
Introduction

- Recent works show the usage of different sensors and hardware for calculating odometry data, such as wheel odometry and Inertial Measurement Unit (IMU) for localization.
- In this project, Wheel Odometry is used to keep track of the ground truth for the mobile robot.
- In the proposed method, Hector SLAM, AMCL, and a wheel odometry node are integrated and modified to obtain the accurate location and orientation of the robot.
- These mobile robots are useful in places where there is a need of repeated tasks like Hotels and Hospitals

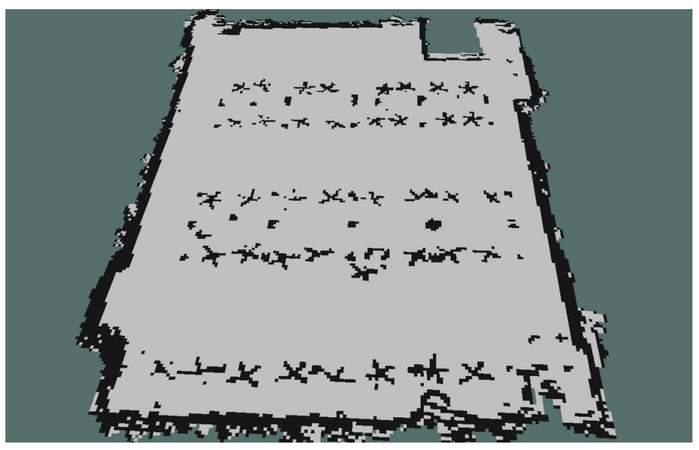
Methodology

The functionality of the mobile robot can be understood in two sections:

- Section 1: Mapping
  - The robot collects the data about the surrounding environment using laser scans.
  - Hector SLAM is used in this process, its robustness and accuracy provides a precise map for localization and navigation.
  - This map will be used throughout the navigation process.



Lab space



Map of the lab space viewed in RViz

- Section 2: Navigation
  - Odometry data extraction
    - An Odometry publisher ros node is programmed in a way to convert the incoming wheel tick data to odometry data.
    - The robot's movement is in differential wheeled form, hence the difference in rotation of wheels gives the updated data.

$$d_{\text{left}} = \frac{ticks_{\text{left}} - ticks_{\text{prev\_left}}}{\text{ticks per meter}} \qquad d_{\text{avg}} = \frac{d_{\text{left}} + d_{\text{right}}}{2}$$

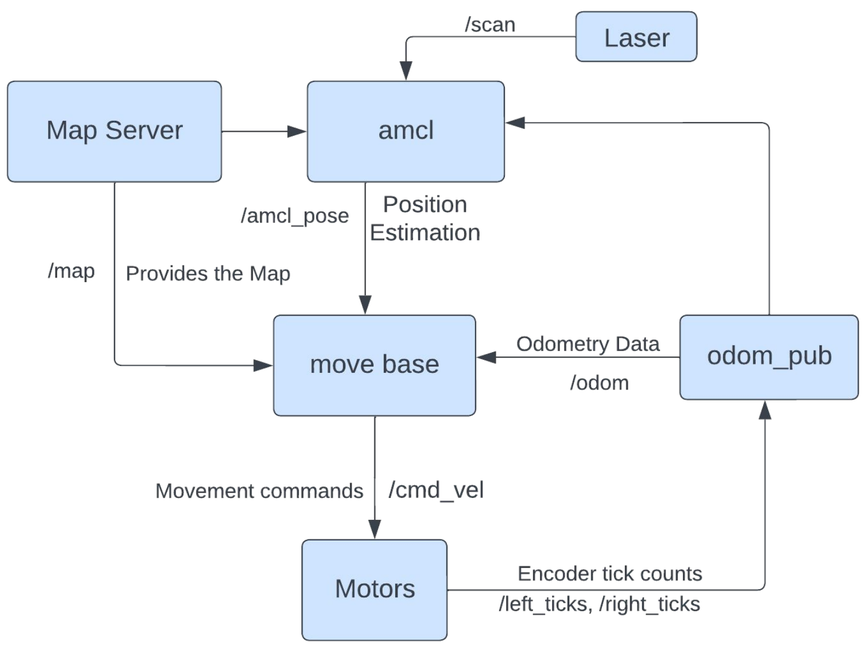
$$d_{\text{right}} = \frac{ticks_{\text{right}} - ticks_{\text{prev\_right}}}{\text{ticks per meter}} \qquad \theta = \frac{d_{\text{left}} - d_{\text{right}}}{\text{base width}}$$

$$x = d_{\text{avg}} \cos\theta \qquad y = -d_{\text{avg}} \sin\theta$$

$$\Delta x = x\cos\theta - y\sin\theta$$

$$\Delta y = x\sin\theta + y\cos\theta$$

  - Localization:
    - The extracted odometry data gives the relative position update whenever the robot displaces from its current position.
    - Laser scans provide us the distance of various obstacles with respect to the robot's position.
    - AMCL node combines the odometry data with alignment of incoming laser scans along the **pre-built map** to estimate current position of the robot in the mapped world.
  - Path Planning
    - On the pre-built map, the move_base package creates layers to create costmaps.
    - These costmaps provide a base for path-planning algorithms. (TEB, DWA etc)



Results and Conclusion

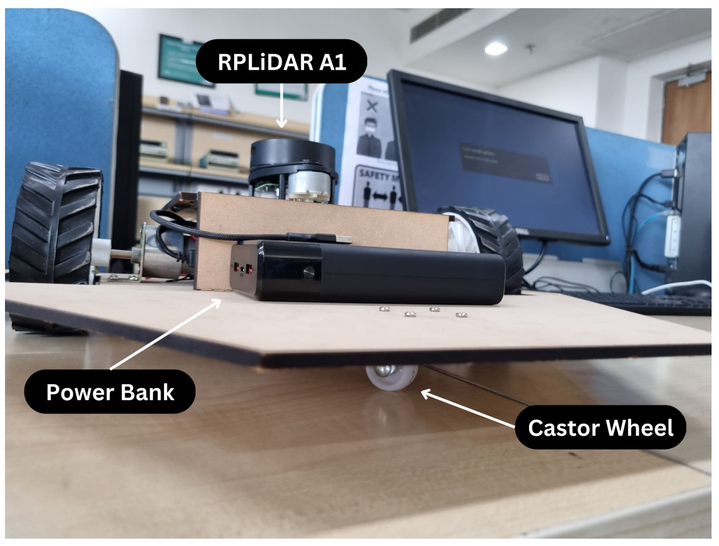
Different errors calculated while keeping tolerance of reaching the goal provided within 15 cm.

Iterations	Translation given (in m)	Actual translation(in m)	Error %age
1	1.9911	2.1347	6.72694
2	2.0128	2.0325	0.96925
3	1.9976	2.0218	1.19695

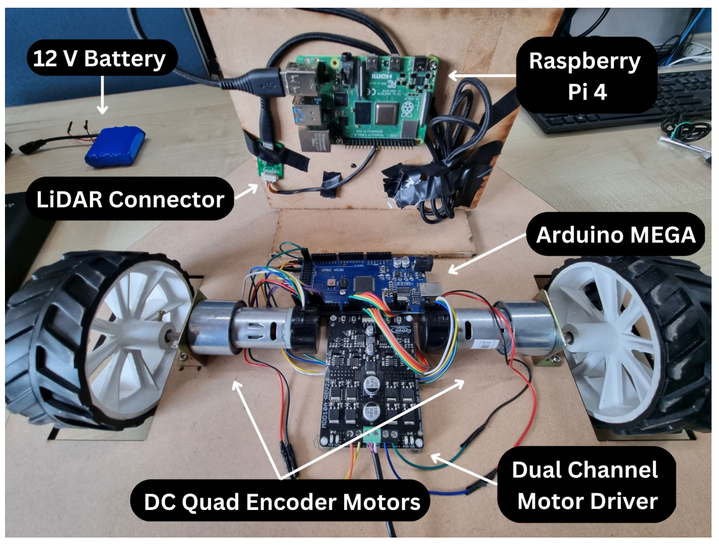
Iterations	Rotation given (in Deg)	Actual rotation (in Deg)	Error %age
1	94.93629166	87.95706887	7.93480
2	179.518714	174.5663241	2.83697
3	90.96935218	96.98776666	6.20533

The robot reaches its desired goal autonomously with minimal error

Hardware Prototype



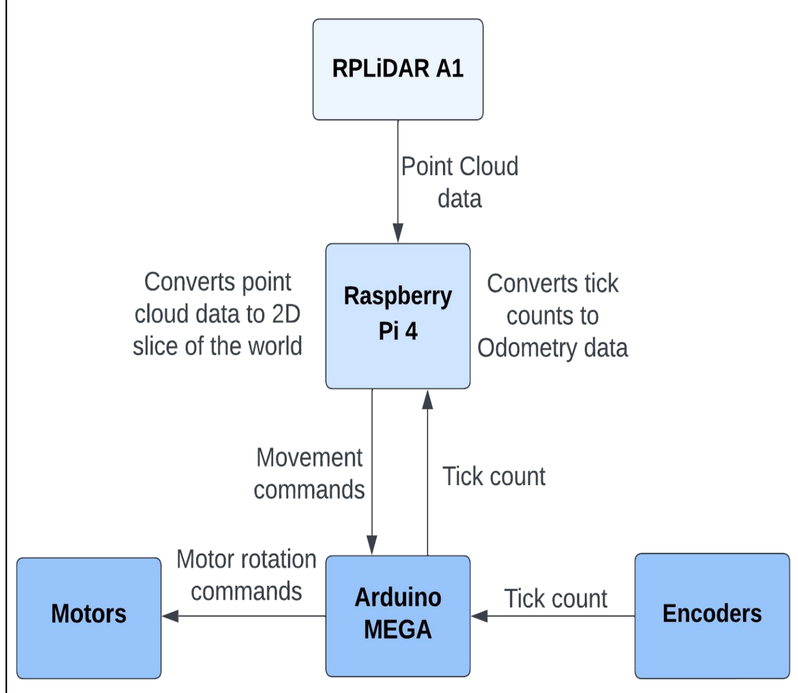
Front View of the robot



Interior of the robot

The hardware prototype is built in several stages and parts.

- *Stage 1- The outer body:* The body of the robot is made using 5mm wooden sheet. The design of the sheet was made separately and then laser cutted for the body.
- *Stage 2 - The Internal Circuitry:*
  - The main brain is the Raspberry Pi 4 (RPi) with Ubuntu Server 20.04 and ROS Noetic installed. RPi runs all the systems and applications related to ROS and Navigation.
  - An Arduino MEGA (MEGA) drives the motors via the Cytron Dual Channel Motor driver. This driver helps to control high current motors.
- *Stage 3 - Sensors:* Two sensors are used.
  - RPLiDAR A1: gives the laser scan
  - Encoded Motors: provides with the precise ticks for odometry calculations.



Hardware Architecture

SCAN QR to meet the Project Supervisor