Autonomous Carrier Robot

***Project Reference No.:**45S_BE_2487*

***College** : St Joseph Engineering College, Vamanjoor Mangaluru*
***Branch** : Department of Electronics and Communication Engineering*
***Guide(s)** : Dr Binu K G and Ms Deepthi S R*
***Student(S)** : Ms. Sharon Frank*
Mr. Karan B
Mr. Melroy Almeida
Mr. Ashwin K S

Keywords

Robot Operating System (ROS), autonomous navigation, Simultaneous Localization and Mapping (SLAM), path planning.

Introduction

With the advancements in technology robots are used in many sectors of industries. They operate in places that are dangerous and perform operations that are difficult for humans. They also play an essential role in our society such as in hospitals, airports, hotels, etc. This has aided humanity in avoiding several mishaps, as well as saving time and money.

There are times when a large object must be carried across short distances, such as in college campuses or construction sites. Because a worker frequently utilises a vehicle to complete this operation, a significant amount of time, fuel, and labor is spent on a routine task. In light of the above situation the idea of the autonomous carrier robot is brought out in this paper.

The Autonomous Carrier Robot as the name suggests is a navigation robot whose main purpose is to transport heavy objects from one place to another. It has been designed to assist workers within the campus to carry heavy objects therefore reducing manual labor and saving time. The robot utilises technologies like Robot Operating System (ROS) for Simultaneous Localization and Mapping (SLAM), navigation and path planning and PID for smooth movement. For the generation of maps using ROS a sophisticated sensor i.e. the RP LiDAR A2M8 is used.

ROS, or Robot Operating System, is a software development platform for robots. It's a set of tools, libraries, and conventions aimed at making the work of sophisticated programs and reliable robot behavior on a range of robotic systems easier. It establishes a robotics software development standard that may be applied to any robot. ROS provides a set of common robot-specific libraries and tools that may quickly get a robot up and running.

Objective

(a) To integrate Raspberry Pi with ROS.

(b) To visualize the obstacles by integrating ROS and RPLIDAR.

(c) To obtain a map using hector slam and g-mapping.

(d) To collect odometry data for localization and use it for movement of the bot.

(e) Developing a stable structure with the help of the mechanical team.

Methodology:

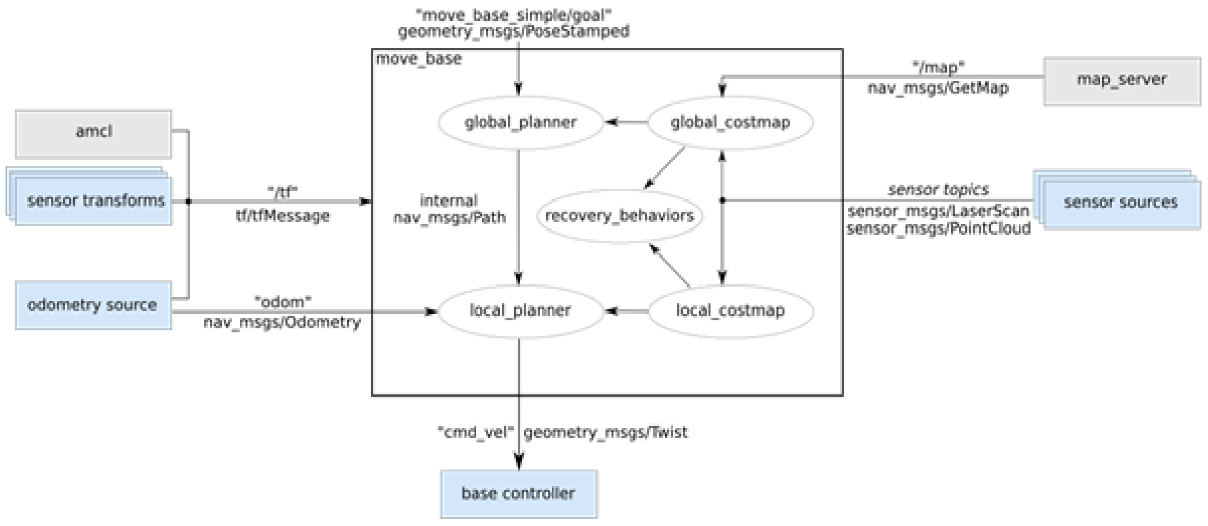

Fig 1: Block Diagram

1)ROS Navigation Stack

The sensor source used for the generation of maps is the RP LiDAR A2M8. Once the global map is generated, it is stored in the map server. The local and the global map are then used by the path planning algorithms named global_planner and local_planner. These algorithms take the inputs of the odometry sources such as the encoders and sensor transformation packages. The output of the path planners are used to control the motor speed and thus the movements of the robot.

This methodology was proposed to be implemented on a test robot first for evaluating the responses of ROS after which the same was to be implemented on the final carrier robot which can carry heavy objects and perform navigation and mapping at the same time.

2)Hardware Architecture

The Raspberry Pi (Rpi) 4b is the main controller used in the system. It is used along with a 64 GB SD card so that it can support Ubuntu 20.04 along with ROS Noetic. The main sensor used is the RPLiDAR A2M8 which is a 360 degree 2D laser scanner.

It is used for the creation of maps which can be later used for autonomous navigation. The Rpi communicates with the Arduino uno using rosserial protocol. The Arduino, which is the lower level microcontroller, controls the Quad Encoder Geared DC Motor via motor drivers. An SMPS is used as the supply for the motors and the LiDAR and Rpi are powered by a two output MI power bank. The Rpi can be controlled remotely using a laptop with the help of Secure Shell (SSH) and Virtual Network Computing (VNC).

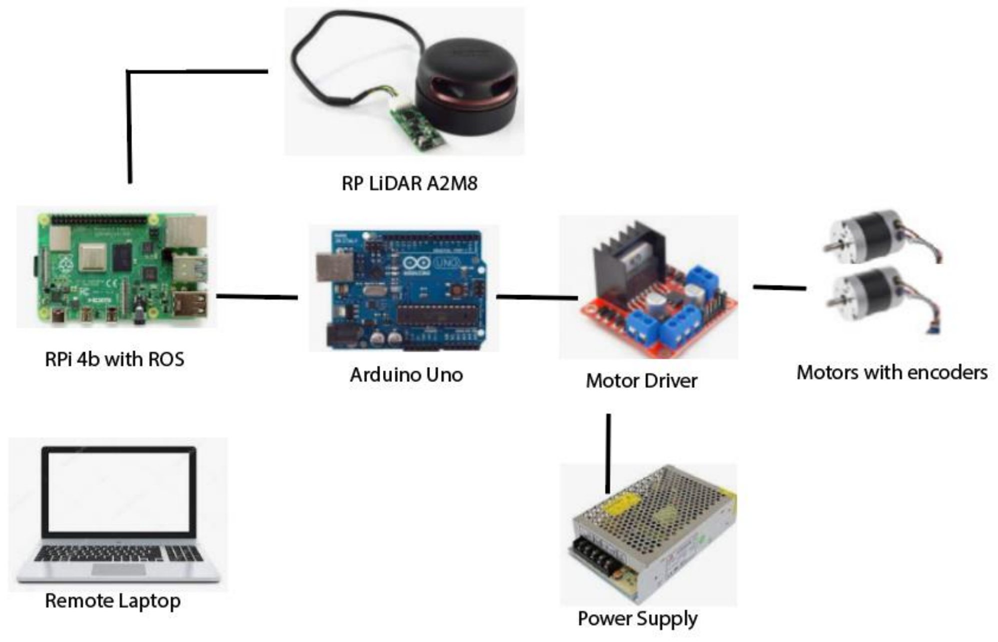

Fig 2: Hardware Architecture

Results and Conclusion:

Small Test Robot

The ability to control a robot's velocity from a distance is important for a variety of tasks, including mapping, exploring unknown terrain, and accessing hard-to-reach locations.

Encoders are used to obtain the odometry of the robot. When the motors of the test robot rotate the built-in encoders generate pulses which are measured in the form of ticks. These ticks can be used to measure the distance traveled by the robot and thus locate it. The ticks are published using the topics /left_ticks and /right_ticks for the left and right wheel respectively. The number of ticks calculated per revolution was 272.

The motors were then controlled with rqt_robot_steering interface and teleop_twist_keyboard

A room was mapped and a global cost map was created using lidar while moving the bot remotely using a teleop twist keyboard. The map generated was stored in the map server as a PGM andYAML file.

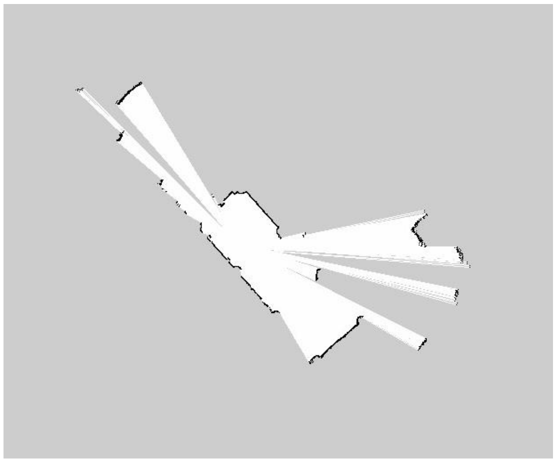

Fig. 3 Map generated.

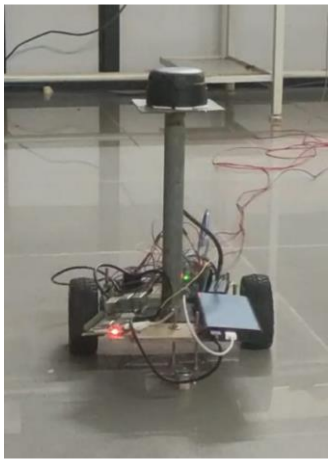

Fig. 4 Small Test Robot

To achieve autonomous navigation using ROS a launch file was written to start multiple nodes required for the movement of the robot. roslaunch is a tool for easily launching multiple ROS nodes locally and remotely via Secure Shell (SSH), as well as setting parameters on the Parameter Server. It includes options to automatically respawn processes that have already died. roslaunch takes in one or more extensible markup language (XML) configuration files (with the .launch extension) that specify the parameters to set and nodes to launch, as well as the machines that they should be run on

The map from the map server is loaded and a 2D pose estimate is given to help AMCL localize the robot's initial position on the map. The 2D nav goal tool of RVIZ is used to send the destination. rqt_graph provides a GUI plugin for visualizing the ROS computation graph.Its components are made generic so that other packages where graph representation has to be achieved can depend upon this package

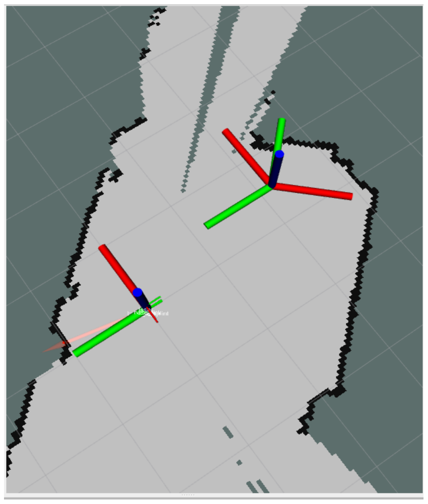

Fig. 5 Initial and final pose on the map

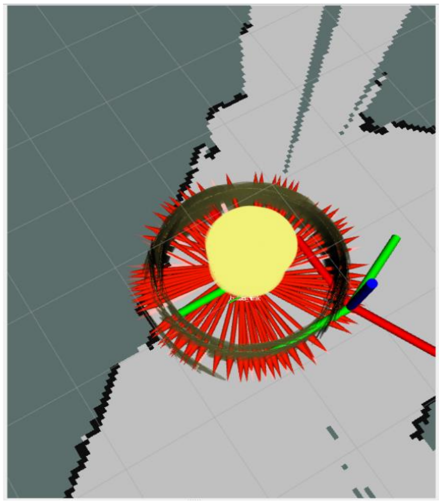

Fig. 6 Robot Trajectory

The following rqt_graph was generated on executing the autonomous navigation launch file.

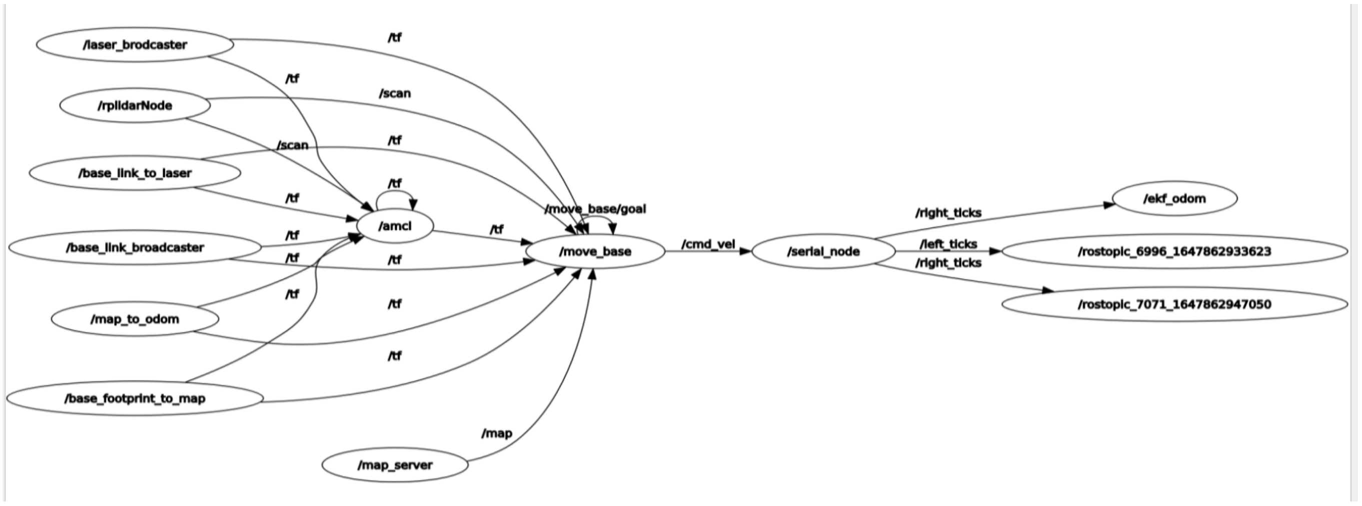

Fig. 7 Rqt graph

1) Autonomous Carrier Robot

The code developed in the test robot was used in a bigger structure. This structure was able to withstand a weight of more than 50 kgs. It had two wheels powered by two MY1016ZL 24V 250W DC Motor, two free wheels. The battery used was 24V, 12Ah. The Lidar and Arduino mega were connected to the Raspberry pi which computed all the inputs and sent the required pwm values to the motors. Mapping and Navigation were performed on this robot.

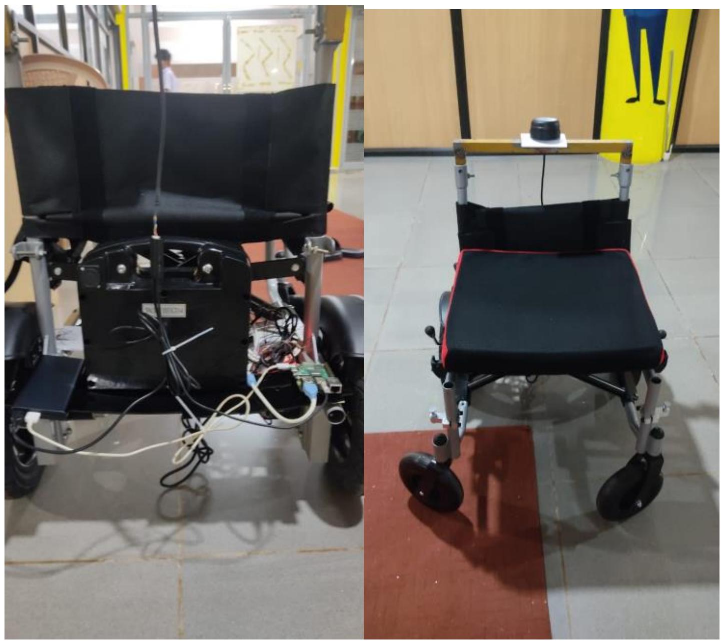

Fig. 8 Back and Front view of the autonomous carrier robot

Scope for future work:

The future scope of this project involves the integration of Collision Prediction based Social Force Model (CP-SFM). This model enables the robot to be socially aware, that is, the robot will respect the personal space of the human beings in its environment and navigate through a human friendly path. This will allow the smooth deployment of the autonomous carrier robot in public places which have less density of population. In case of people rich environments, deep inforcement learning can be incorporated with ROS as it makes use of Markov Decision Process. Instead of using RaspberryPi, Jetson Nano can be used as it has more processing speed and AI/ML capabilities.ROS 2 has can be brought into the design instead of ROS. ROS 2 provides more real time results as ros core is eliminated in its architecture . Further a GUI can be developed and designed to make the entire system user friendly.