Shamazon Warehouse Robot

Pratik Sunil Bhujbal
UID: 117555295

Maitreya Kulkarni
UID: 117506075

Mohammed Maaruf Vazifdar
UID: 117509717

Project Overview:

As it stands today, robots in the workforce are proficient mostly at specific, repeatable tasks for which they are precisely programmed. Delivery robots are autonomous robots, and as the name suggests, are used for delivery tasks in a dynamic environment. The following project will employ a delivery robot with autonomous capabilities with static and dynamic obstacle avoidance. The robot will receive a package from a conveyor belt and will navigate through a map to deliver it between departments inside a warehouse with multiple levels, where the robot uses an elevator gazebo plugin to communicate and know the state and location of the elevator and upon reaching the desired level/ floor, update the map. Upon reaching the goal the package will be delivered by the robot's conveyor belt.

Assumptions:

1. Package collection and delivery locations are known.
2. Map has been generated and is known to the robot.
3. Robot drive configuration - 4 wheel differential drive.
4. Robot carries only 1 package at a time.
5. Package dimensions are fixed.

Project Technologies:

1. Operating System: Ubuntu 18.04
2. Build tool: catkin_make
3. Version control: Github
4. Build Check: Travis
5. Code coverage: Coveralls
6. Framework: ROS
7. Simulator: Gazebo (version 9+)
8. Programming language: C++
9. Open-source libraries/ packages: ROS Navigation stack[1], gazebo_ros_pkgs[2], pgm_map_creator[3].
10. License: MIT License.

Development Process:

This project involves three programmers, following the Agile Iterative Process and programming in-pairs. To ensure product quality roles of driver, navigator and design-keeper will be switched amongst the team regularly. To prevent the occurrence of bugs and failure of sub-modules the team will adapt to the Test Driven Development philosophy. All programming will be done using modern C++, following Google Style Sheets and abiding to the practices of OOPS.

Build/Look for a gazebo world file and modify it to add a conveyor, and elevator in a warehouse environment. Modify robot's urdf to add conveyor on top, add sensors-plugins, configure navigation stack, and test autonomous navigation and obstacle avoidance.

Write C++ scripts for controlling conveyor to provide packages, provide navigation goals to the robot, establish robot-elevator communication and update the map on level transitions. Finally, control the robot's conveyor to deliver the package to the goal locations based on the user's input.

Project risks & mitigations:

1. Odometry error: Can use sensor fusion algorithms from multiple sensors for odometry data.
2. A package might fall off due to sudden jerks, so robot velocity and acceleration will be constrained.

Deliverables:

Robot will autonomously navigate in the warehouse while avoiding static/ dynamic obstacles and collect the package from a conveyor belt (pickup location). Then navigate to the elevator, update the map, proceed to the next level, and deliver the package to goal locations based on the user's input. The sensors data, navigation goals, local and global plans, local costmap

updation for dynamic obstacles, and global costmap for level transitions will be visualized in RViz.

Initial Test Cases:

1. Tests for Class member functions (Level 1).
2. Map server updation test (Level 2).
3. Elevator ROS service test (Level 2).

Fallback Plan:

Performing the task autonomously of collecting the package from the conveyor to delivering it to the goal location on the same level/floor with static/dynamic obstacle avoidance.

References:

Links:

1. https://spectrum.ieee.org/amazon-introduces-two-new-warehouse-robots

Papers:

1. N. Kimura et al., "Mobile dual-arm robot for automated order picking system in warehouse containing various kinds of products," 2015 IEEE/SICE International Symposium on System Integration (SII), 2015, pp. 332-338, doi: 10.1109/SII.2015.7404942.

Repositories:

1. https://github.com/ros-planning/navigation/tree/melodic-devel
2. https://github.com/ros-simulation/gazebo_ros_pkgs/tree/melodic-devel
3. https://github.com/hyfan1116/pgm_map_creator
4. https://github.com/pal-robotics/aruco_ros/tree/melodic-devel

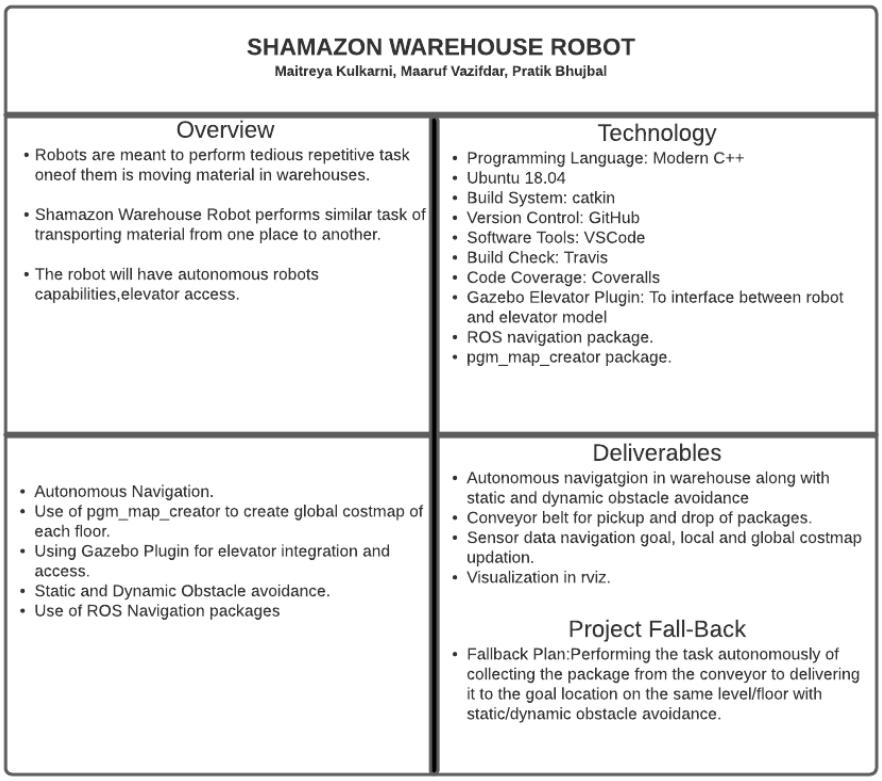

Fig1. Quad Chart

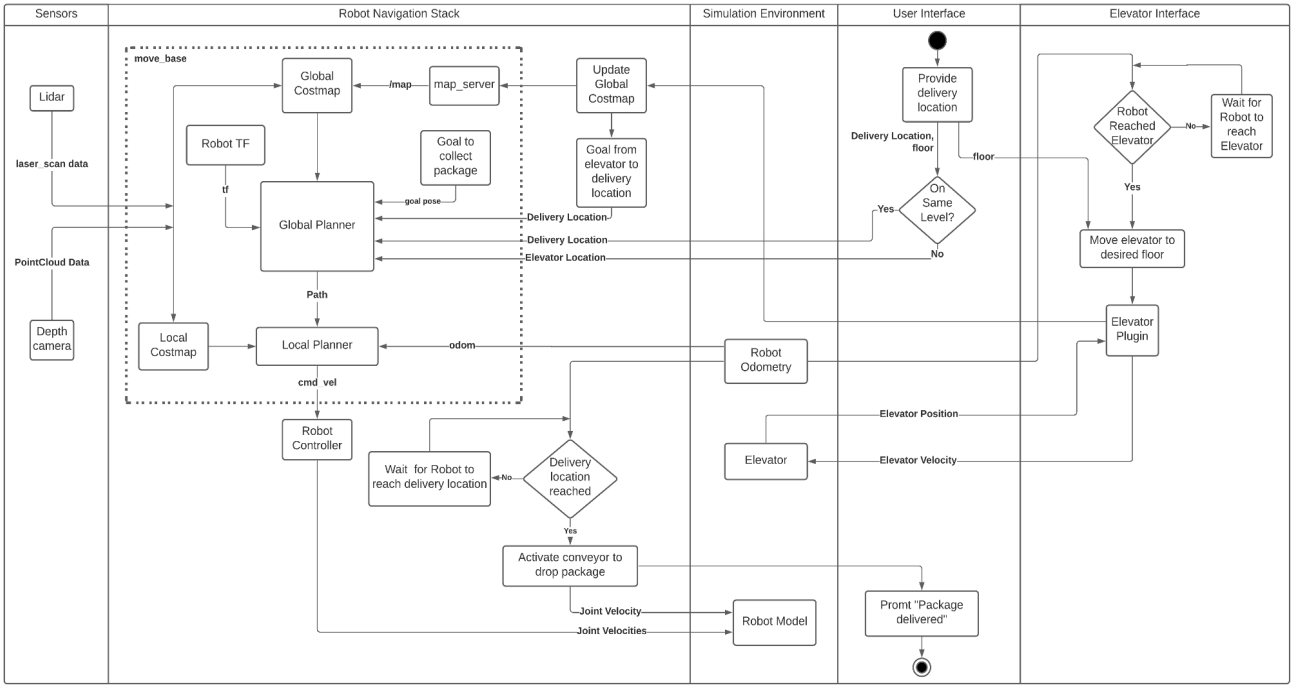

Fig2. Activity Diagram

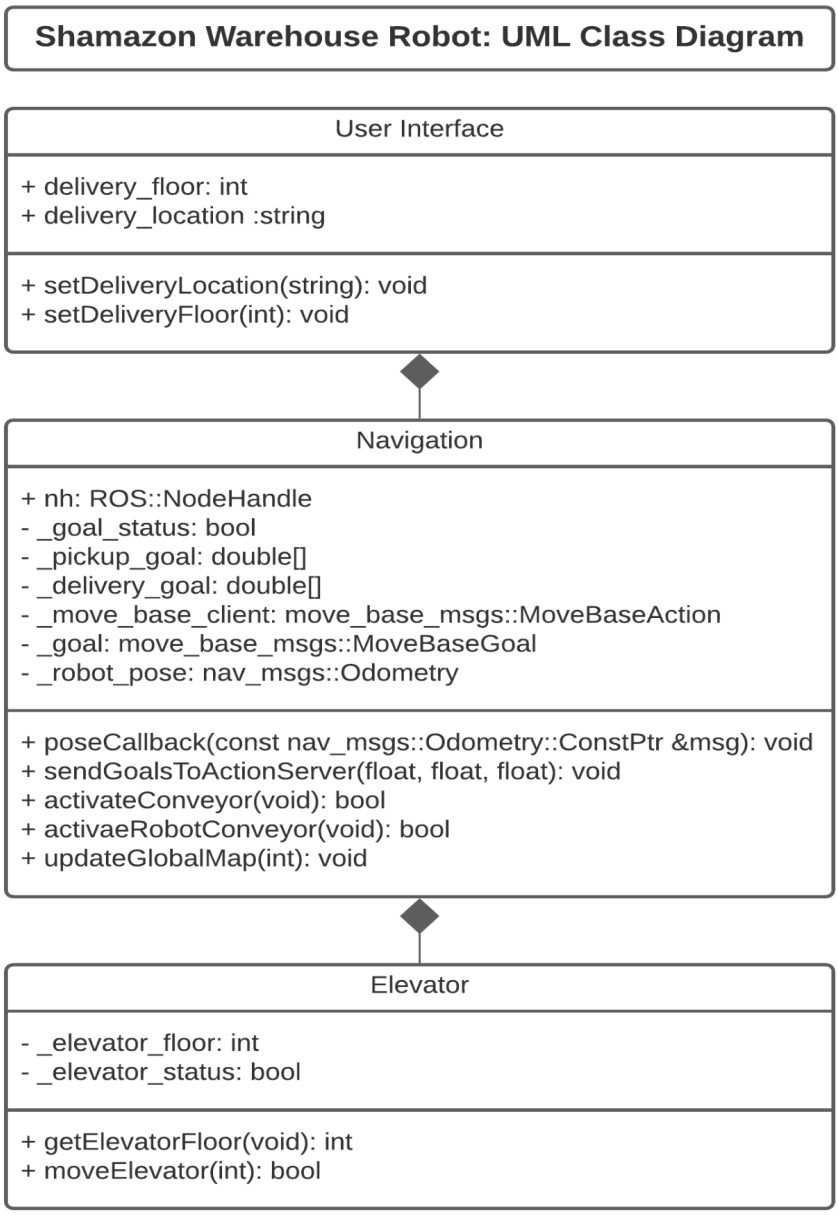

Fig.3 Class Diagram