Maze Solving Robot (MAZE)

Project Lead: David Boosi
Members: Nishyanth Arimanda, Joshua Dantas, Nishyanth Gollamudi, Ben Kumar, Jeremy Leicht, Vencel Villanueva

TEXAS A&M UNIVERSITY Engineering
TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Problem Statement

Given an unknown maze, the robot should be capable of navigating from the entrance to the exit. The maze will be represented by a 3D network of corridors with exactly one entrance, one exit, and the possibility of multiple solutions. Navigation and obstacle avoidance will be conducted solely using LiDAR input over vision-based systems. This semester focused on developing rudimentary navigation software using ROS2, and verifying results in Gazebo.

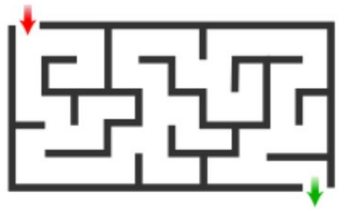

Figure 1: *2D layout of a conceptualized maze*

ROS2 Overview

Robot Operating System (ROS2) is a middleware complete with open-source libraries, firmware, and algorithms that standardize robotics software development. Through ROS2, operational processes, such as driving motors and reading sensor data, are modularized as nodes which communicate to one another via topics, services, actions, and parameters which follow publisher/subscriber and request/reply methodologies. By abstracting software from hardware this way, ROS2 permits the testing and simulation of maze navigation algorithms in a virtual environment without the need for physical components. This project used Python software implemented in ROS2 Humble within Ubuntu 22.04.

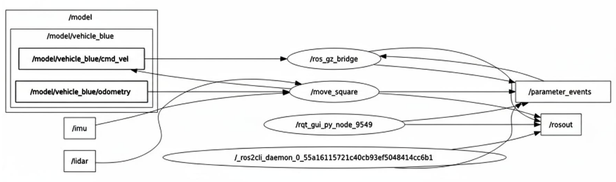

Figure 2: *ROS2 RQT Graph with node interactions for maze navigation software*

Gazebo Implementation

Gazebo Fortress is leveraged to emulate navigation in a virtual 3D environment in the absence of a physical robot. Through Gazebo, ROS2 software simulates the robot moving through a maze while receiving and processing LiDAR input to avoid walls. LiDAR data is then piped into accompanying RViz2 software to visualize point clouds of the environment.

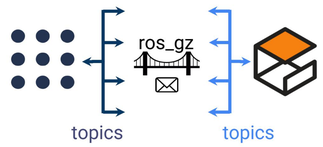

Figure 3: *Topic bridging between ROS2 and Gazebo*

Navigation Algorithm

The current navigation algorithm is based on the left-hand rule. The robot follows the wall and turns left whenever there is a open path to the left or when it reaches a dead end. The algorithm uses the LiDAR data to locate path openings and walls. To maintain the same distance to the wall while moving, two measurements from the LIDAR are used to calculate the angle of the robot relative to the wall.

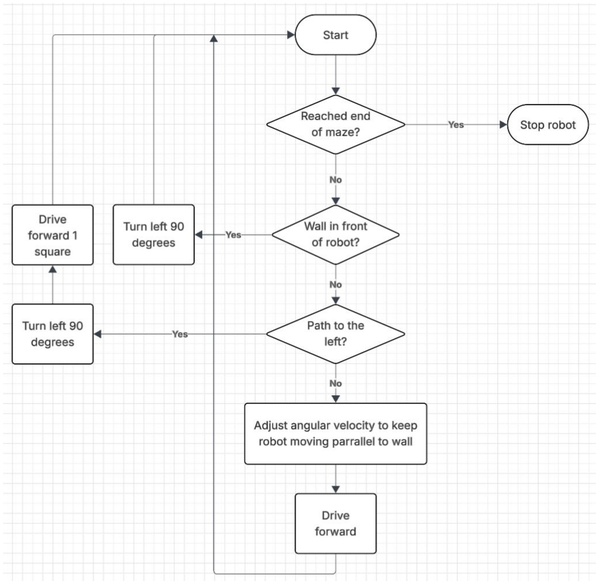

Figure 4: *Maze navigation algorithm flow chart*

LiDAR Integration

LiDAR (Light Detection and Ranging) technology uses a device that emits lasers and records the time it takes for a beam to strike an object. The consistent ability to detect physical obstacles, along with a 360-degree ranger, makes LiDAR more appealing over computer vision for navigating maze corridors.

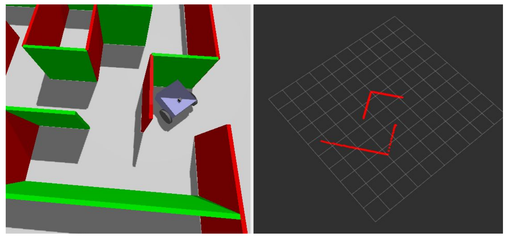

Figure 5: *LiDAR readings of robot surroundings*

Physical Design

The design of the robot was overhauled from a spherical shape to tiered rectangular plates for ease of assembly and optimized area usage. Among the changes made were:

- Only one caster wheel instead of two. The new single rear caster wheel acts as a support for two forward motor driven wheels.
- No outer shell in the chassis for ease of assembly
- Slots for individual electrical components.
- Stilted design based on individual standoffs connecting layers.

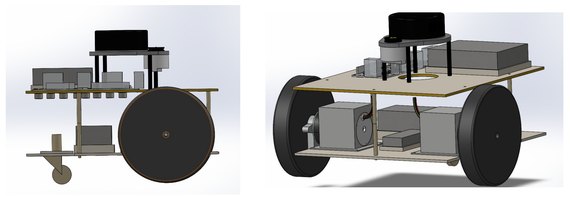

Figure 6: *Profile and Isometric CAD model views*

Accomplishments

Following are the goals accomplished this semester:

- **Maze Generation:** Sample mazes were generated for Gszebo testing using SDF (Simulation Description Format), an XML-like markup language.
- **Robot Modeling:** A simulation robot reflective of the CAD design was modeled in SDF with virtual LiDAR and IMU hardware.
- **Algorithm Testing:** Gazebo was leveraged to test wall-following algorithms written in Python with ROS2 Humble middleware.
- **Docker Containerization:** The robot's ROS2 workspace was simulated within a Docker container to permit cross-compilation across various architectures.

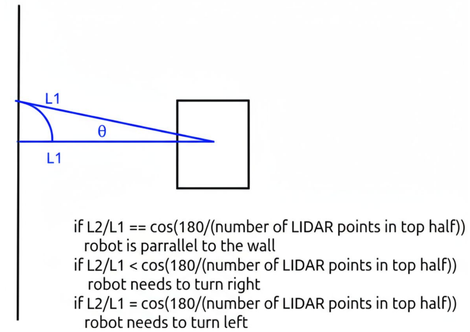

Figure 7: *Wall distance calculation logic*

Future Work

- **Hardware**
  - 3D print platforms for the robot chassis and assemble working prototype.
  - Interface hardware components together such that Jetson Nano can process LiDAR data.
- **Software**
  - Develop graph-based algorithms for recording the maze layout and computing the shortest path from entrance to exit.
  - Integrate the Google Cartographer library for SLAM (Simultaneous Localization and Mapping).