Maze Solving Robot (MAZE)

Project Lead: David Boosi
Members: Nishyanth Arimanda, Alejandro Garza, Hadeed Khan, Ben Kumar, Jeremy Leicht, Aayush Sharma,

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 we focused on developing navigation software, integrating SLAM using ROS2, and verifying results in Gazebo.

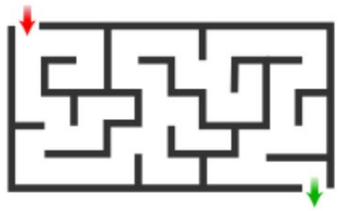

Figure 1: *2D layout of a conceptualized maze*

ROS2 Nodes

The `rplidar_ros` node (SlamTec) publishes raw 360° LiDAR scans to `/scan`. Google's Cartographer performs SLAM using LiDAR and wheel odometry, generating an occupancy grid at `/map`. The `wall_finder` node reformats this into a uniform processed grid, the `wavefront` node applies the frontier exploration algorithm to produce step commands, and `step_mover` converts those into velocity commands executed by the Arduino.

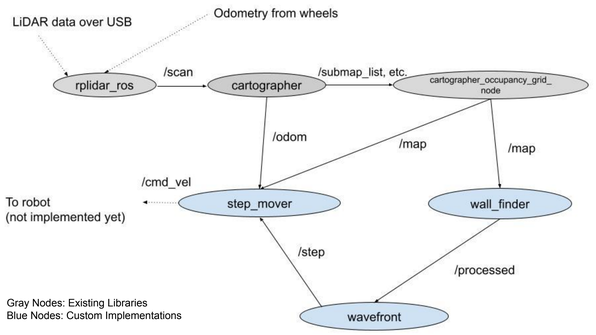

Figure 2: *ROS2 Graph with node interactions and topics*

Gazebo Simulation

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.

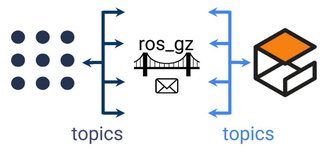

Figure 3: *Topic bridging between ROS2 and Gazebo*

Navigation and Pathfinding

Navigation uses two graph-based algorithms operating on an occupancy grid. The frontier detection algorithm identifies boundaries between free and unexplored space, selecting the nearest frontier cell using a Manhattan distance heuristic and planning a route via A*. Once the maze is fully mapped, A* re-runs from the entrance to compute the globally optimal shortest path to the exit.

Algorithm 2 Frontier Detection & Path Planning — `plan_path`

Require: map // *occupancy grid: UNKNOWN* $= -1$, *FREE* $= 0$, *WALL* $= 1$
Require: $pose = (r_x, r_y, r_\theta)$ // *current robot pose*
if path is not empty **then**
  return ▷ already have an active plan; do not replan
end if
frontiers $\leftarrow \emptyset$
for each cell (r, c) in map, excluding border cells **do**
  if map$[r][c]$ = FREE **then**
    for all neighbour $(n_r, n_c) \in \text{adj}_4(r, c)$ **do**
      if map$[n_r][n_c]$ = UNKNOWN **then**
        append (r, c) to frontiers
        break ▷ one unknown neighbour is sufficient
      end if
    end for
  end if
end for
if frontiers $= \emptyset$ **then**
  finished $\leftarrow$ True
  return ▷ entire reachable area has been explored
end if
start $\leftarrow (\lfloor r_y/\delta \rfloor, \lfloor r_x/\delta \rfloor)$
sort frontiers by $|r - \text{start}_r| + |c - \text{start}_c|$ ▷ Manhattan distance heuristic
goal $\leftarrow$ frontiers[0] ▷ select nearest frontier
path $\leftarrow$ A*(start, goal, map)

Figure 4: *Frontier exploration algorithm*

A* prioritizes cells by f-score, which is the sum of traveled distance plus Manhattan distance to the goal. Unknown cells are treated as traversable, and the final path is reconstructed by backtracking through stored parent pointers.

Electronics and Hardware

- **Jetson Nano**: Primary processor running ROS2
- **Arduino**: Deterministic real-time motor control
- **RPLiDAR**: 360° laser scanning for detection & mapping
- **IMU**: Supplementing heading and SLAM accuracy data
- **DC Motors + L298N Driver**: Differential drive actuation
- **3S LiPo + Buck Converter**: Primary power; regulated down to 5V for Jetson Nano

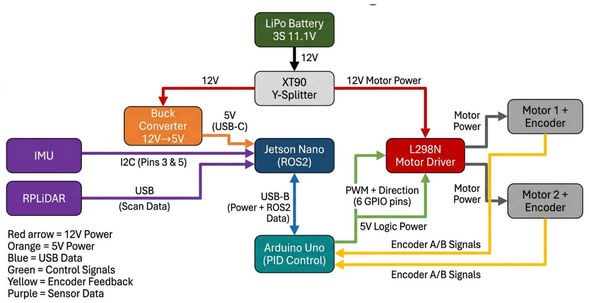

Figure 5: *Hardware block diagram*

Physical Design

The robot chassis was redesigned from a 3-tier to a 2-tier layout, reducing its footprint to fit within maze corridors. Platforms are constructed from 3D printed plates with rubber wheels for reliable ground contact. These driven wheels are supplemented with two free spinning ball bearing rollers for horizontal support. The physical maze is assembled from cardboard sheets secured by 3D-printed joints, ensuring precise 90-degree corners and minimizing gaps that could disrupt LiDAR scans.

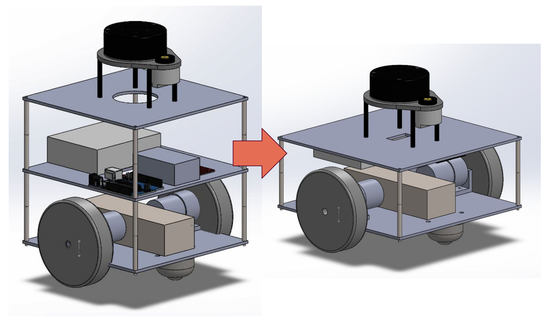

Figure 6: *CAD Model Updates*

Accomplishments

Following are the goals accomplished this semester:

- **Algorithm Research**: Frontier exploration and A* pathfinding designed in independent pygame simulation
- **Docker Containerization**: ROS2 Jazzy custom image cross-compiled for ARM64 (Jetson) via Docker buildx
- **Robot Prototype**: single-tier chassis with Arduino motor control bench-tested with encoder feedback
- **Maze Prototype**: Cardboard maze mock-up with 3D-printed alignment joints for LiDAR-compatible walls

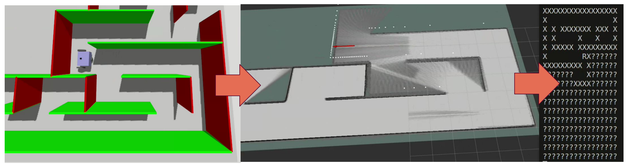

Figure 7: *Transformation from LiDAR data to map grid*

Future Work

- **Hardware Integration**: Test complete system power draw under simultaneous motor and compute load with all hardware
- **Sensor Dataset Collection**: Deploy LiDAR and IMU to collect real-world sensor data and validate Cartographer SLAM mapping outside of simulation
- **Software Validation**: Transition `cartographer` and `step_mover` nodes from Gazebo to physical hardware; characterize timing constraints between Arduino's real-time encoder loop and Jetson's non-real-time SLAM

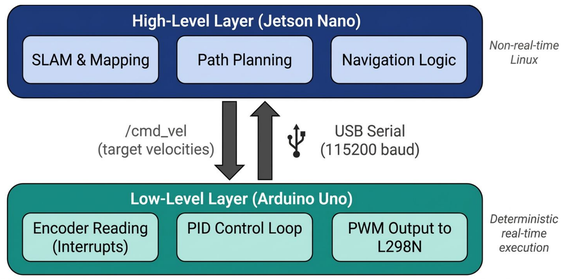

Figure 8: *Control architecture diagram*