Autonomous MicroMouse Robot

Elman Steve Laguna

University of Texas at San Antonio

November 19, 2025

Outline

Project Overview

- **Goal:** Develop an autonomous robot simulation that can navigate and solve a maze
- **Implementation:** Written in Rust initially but moved simulation to ROS2
- **Features:**
  - Simulated pathfinding algorithms (ROS2 + C++)
  - Used Docker to run ROS2 on my Mac
  - Generated occupancy grids from LiDAR data (Rust + Python)

MicroMouse Challenge

- Classic robotics competition
- Navigate from start to goal
- Find optimal path
- Minimize solve time

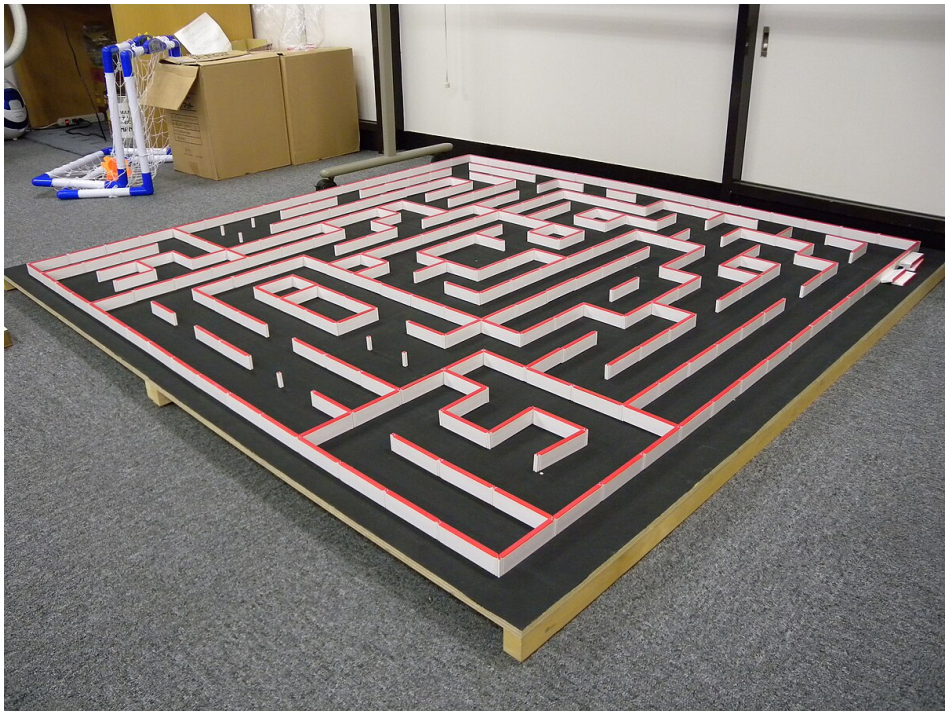


Figure: Competition Maze

Rust System Architecture

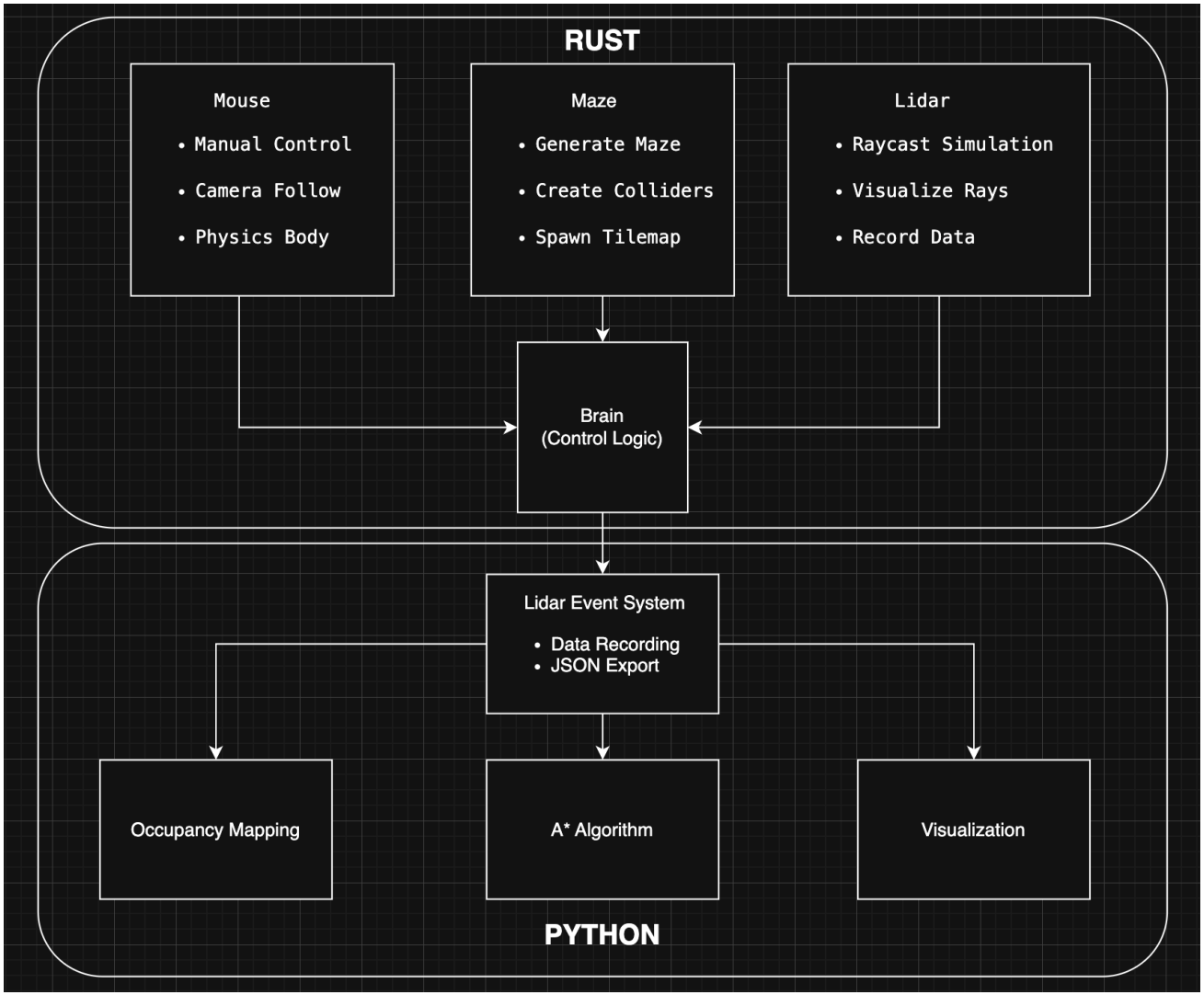


Figure: Rust Implementation

ROS2 Control Flow

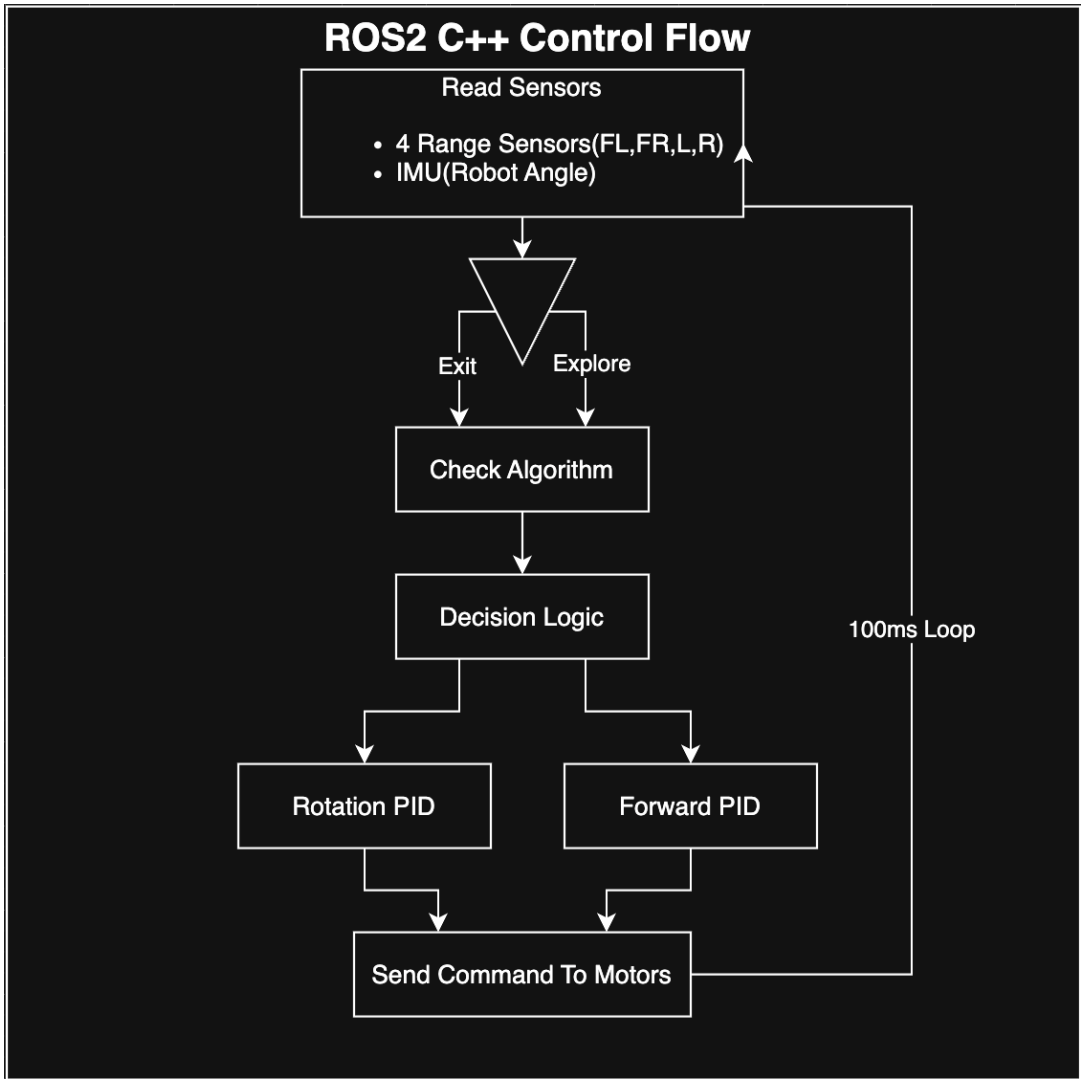


Figure: ROS2 Control Flow

Pathfinding Implementation

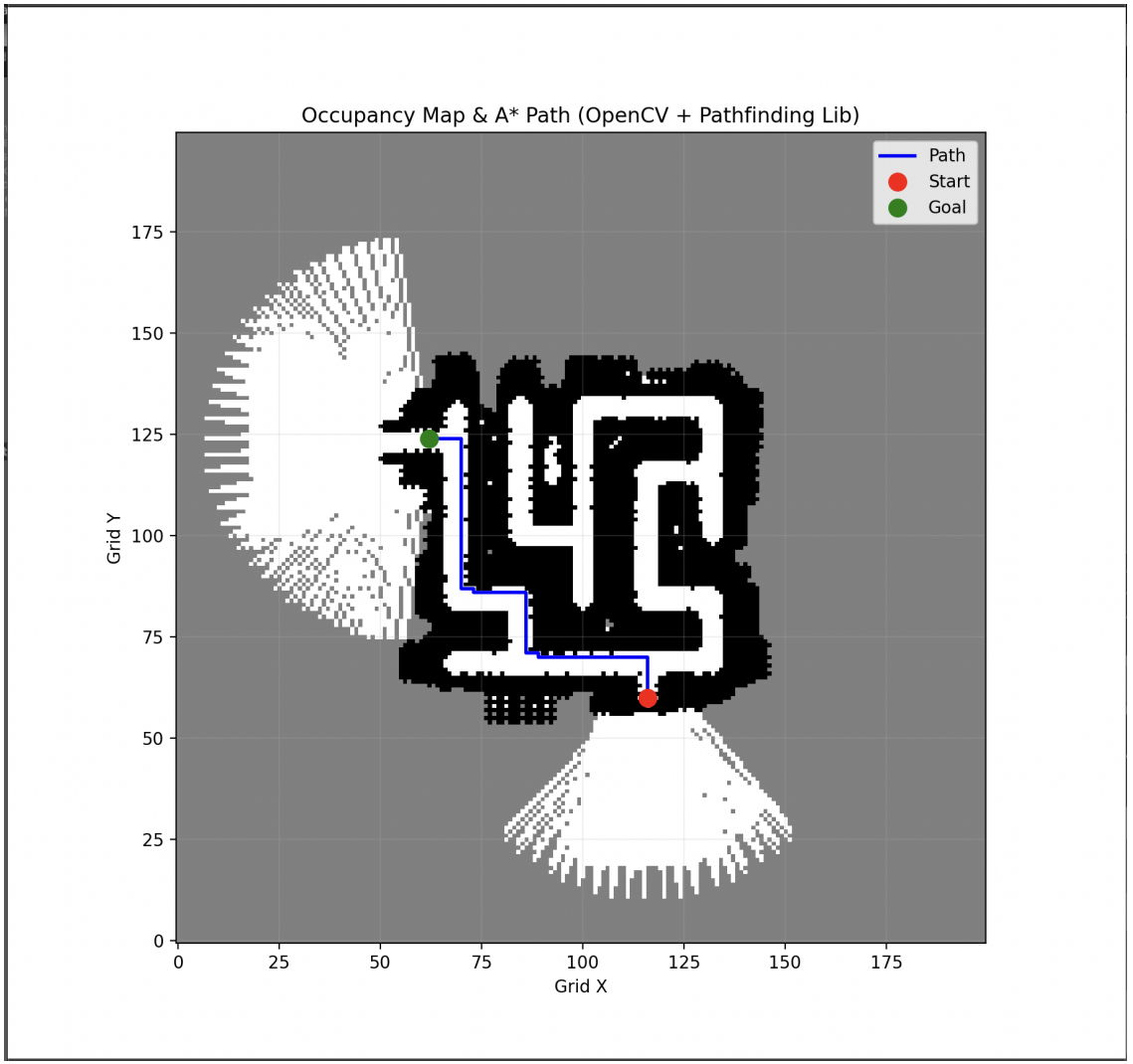
Algorithms Used:

- Manual(Rust)
- Right-Hand Rule(Rust WIP)
- A* Search(In Post-Processing)
- A* Search(ROS2)

```
// Example Python code
finder = AStarFinder(
    diagonal_movement=
        DiagonalMovement.
        never
)
}
```

Maze Mapping Strategy

- **Exploration Phase:** Robot discovers maze structure
- **Mapping:** Build internal representation using a 2D grid
- **No Diagonal Movement:** Restrict to orthogonal moves
- **Optimization:** Calculate shortest path after full exploration



Grid-Based Pathfinding

How It Works:

- Map = 2D grid of cells
- Each cell: **0** = free, **255** = wall
- Neighbors = adjacent cells
- Check neighbors to find path

Example:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

1 = walkable, **0** = blocked

Implementation:

- Occupancy grid from LiDAR data
- Convert to walkability matrix
- A* searches adjacent cells
- No explicit connection storage

Benefits:

- Memory efficient: O(N)
- Fast neighbor lookup: O(1)
- Perfect for grid-based mazes
- Simple to implement

Issues Remaining

- Get a deeper understanding of ROS2
- Improve understanding of pathfinding algorithms
- Get a deeper understanding of how the ROS2 Control Flow works
- Implement more pathfinding algorithms in Rust
- Improve robot control in Rust simulation

Lessons Learned

- ROS2 simulates motors, sensors, physics, and control flow
- Rust is good if you want to simulate something such as pathfinding algorithms
- Bevy is a good ECS framework for simulations
- ROS2 on Mac is difficult
- Python is good for data analysis and visualization
- What is the problem you want to focus on?
- Algorithms, sensors, control - just a small piece in a big system?
- Solving the bigger problem and offloading the details to existing tools is sometimes better
- Depends on your goals

Video Demonstration

Youtube Demo:

https://youtu.be/U2DLx_qEK7s

Questions?

Thank you!

Project Repository: `https://github.com/salsasteve/mouse_maze`

Credits: ROS2 implementation by `https://github.com/delipl/ros-micromouse`