**Project Report: Scaling the Virtual Robot Controller**

**Introduction**

The initial version of the Virtual Bot Controller was a minimal prototype that allowed a robot to move around a 2D canvas using keyboard keys or basic command inputs. While functional, it was limited in scope and did not simulate a real-world environment effectively.

Over the course of the project, we scaled and restructured the application into a more feature-complete simulation with the ability to navigate across multiple floors, detect obstacles, and perform delivery actions. This document outlines the improvements made, the current structure, and pending areas for further development.

**Original Application Overview**

The original version of the project included:

* A single canvas representing one generic space.
* A virtual robot that could move forward, backward, left, and right using buttons or typed commands.
* Basic command parsing for simple directions.
* Reset functionality to return the robot to the center.
* Optional state save/load functionality using JSON.

**Limitations**

* Only one environment (no multi-floor capability).
* No obstacles or room layouts.
* No contextual delivery logic (e.g., pickup, drop-off).
* Static interface with minimal visual guidance.
* No real-world simulation value.

**Upgraded System Features**

The current version of the system adds architectural, visual, and functional depth to the project. Key additions include:

**1. Multi-Floor Navigation**

* Introduced support for **eight separate floors**, each represented by its own canvas.
* Users can switch between floors using floor buttons.
* Each floor maintains its own robot instance, position, path, and layout.

Note: While all eight floors are structurally present, some of the floor layouts (e.g., Floors 6 through 8) are placeholders and still need full room/obstacle designs.

**2. Visual Layouts and Obstacles**

* Each floor includes labeled spaces such as reception, meeting rooms, cafeterias, labs, etc.
* Rooms and objects are drawn directly on the canvas and stored as obstacles.
* Collision detection prevents the robot from moving through walls or blocked areas.

**3. Collision Detection and Boundaries**

* Collision logic checks for overlap with defined obstacles and canvas boundaries.
* Robot movement is blocked automatically when it would result in a collision.
* Applies to all movement types including diagonal and Bezier-based paths.

**4. Per-Floor State Management**

* Each floor remembers the robot’s last position, path history, and color.
* When switching floors, the robot state is saved and restored without data loss.
* State saving/loading now includes the full multi-floor context, making it persistent between sessions.

**5. Delivery Simulation**

* Added pickup and delivery buttons to simulate basic package handling.
* "Return to base" functionality resets the robot and brings it back to Floor 1. Delivery status is shown at the top of the UI for user reference.

**6. AI-Powered Command Parsing**

* The parser module handles user input commands like:
  + "forward 100"
  + "backward-left 50"
  + "bezier 100 150 200 250 300 350"
  + "reset"
* Ensures consistent handling and makes it easier to extend to natural language input in the future.

**7. UI and Usability Enhancements**

* Cleaner, more modern interface with consistent styling across widgets.
* Activity log for tracking movement and command history.
* Live error messages when invalid actions are attempted.
* Easy-to-use buttons with directional arrows and clear labeling.

**Design Improvements**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Feature** | **Original Version** | **Current Version** | | --- | --- | --- | | Canvas/Floor Support | Single canvas | Eight independent floors | | Obstacle Handling | None | Full collision detection | | Movement Types | Basic directions only | Diagonal and Bezier curves supported | | Delivery System | Not present | Pickup, drop-off, and return to base | | State Save/Load | Basic (single floor) | Full multi-floor state management | | UI Design | Minimal Tkinter layout | Structured, styled interface | | Activity Logging | None | Full command history view | |

**Pending Enhancements**

While the core framework for all floors is in place, some areas are still pending or open for future expansion:

* Floors 6 to 8 currently contain placeholder layouts and need detailed zone/obstacle designs.
* Obstacle and room layouts could be moved to external JSON or map files for easier editing.
* Integration with real-time sensor data or robot SDKs (e.g., ROS) can make it suitable for real robots.
* Pathfinding algorithms (A\*, Dijkstra) could be added to handle automated delivery routing.

**Summary**

This project evolved from a basic single-canvas movement simulator into a more structured, scalable, and realistic delivery robot simulation tool. By introducing multi-floor architecture, collision logic, per-floor state persistence, and command parsing, we’ve laid a strong foundation for both educational and practical robotics use cases.

The current implementation simulates key behaviors expected from an indoor autonomous delivery robot and provides room for future expansion in complexity and interactivity.