(Optimizing Warehouse Outdoor Security with Autonomous Patrol Robots)

24-25J-053

Design Document

Madhusankha W.V.S.

B.Sc. (Hons) in Information Technology Specializing in
Computer Systems & Network Engineering

Department of Computer Systems Engineering
Sri Lanka Institute of Information Technology
Sri Lanka

December 2024

Declaration

I declare that this is my own work, and this proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Name – Madhusankha W.V.S.
Student ID- IT21252372
Supervisor: Ms. Dinithi Pandithage.

Table of Content

3.2.3 Risk Mitigation Plan with alternative solution identification

3.2.4 Cost Benefit Analysis for the proposed solution

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.5.1 Reliability

3.5.2 Availability

3.5.3 Security

3.5.4 Maintainability

3.6 Other requirements

4. Supporting information

4.1 Table of Contents and Index

4.2 Appendices

1. Introduction

1.1 purpose

This document presents the design and architecture of the patrol robot system, focusing on predefined and dynamic path-planning algorithms. It outlines the approach for creating a simulation to implement core functionalities before hardware integration.

1.2 Scope

Predefined path planning.
Dynamic path adjustments based on simulated sensor data.
Adaptation to obstacles and environment changes.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Path Planning	Algorithms that compute a robot's navigation path.
Dynamic Adjustment	Real-time updates to the planned route
Simulation	A virtual environment replicating real-world robot operations.
LIDER	Light Detection and Ranging.
Python	Programming language is used for simulation.

1.4 Overview

This project aims to design a patrol robot that will be able to follow predefined courses and dynamically adjust its course in response to environmental changes. For now, the functionalities are being simulated with Python.

2. Overall Descriptions

2.1 Product Perspective

The foundation for future hardware implementation is laid by the simulation-based design. This initiative reduces upfront expenses by verifying algorithms in a virtual environment, in contrast to purely hardware-based systems.

2.1.1 System Interfaces

OS Interface: Runs on a Windows/Linux platform using Python.

Simulation Tools: Integrates with visualization using google colab.

2.1.2 User Interfaces

Simulation Output: Visual representation of the robot's movement and path adjustments on a grid.

2.1.3 Hardware Interfaces

placeholder for future integration of LiDAR, PIR sensor, GPS sensor, Night vision camera.

2.1.4 Software Interfaces

uses the NumPy and SciPy Python libraries to do calculations.

2.1.5 Communication Interfaces

None currently. (Will add for hardware integration)

2.1.6 Memory Constraints

Simulations use very little memory (less than 1GB).

2.1.7 Operations

Real-time changes and predetermined path planning in a grid-simulation environment.

2.1.8 Site Adaptation Requirements

Python 3.x and related libraries must be installed.

2.2 Product functions

Navigates a predefined route.
Adjusts path dynamically to avoid obstacles.
Simulates sensor inputs for dynamic decision-making.

2.3 User Characteristics

Target Users:-
Developers who works on navigation system.
Warehouse security staff.
Facility Maintenance Managers.

2.4 Constraints

No physical hardware available yet.
Simulated sensors may not fully replicate real-world behavior.

2.5 Assumptions and Dependencies

Python libraries adequately simulate the environment.
Future versions will include real sensors.

2.6 Apportioning of Requirements

Predefined path simulation.
Dynamic path adjustment in simulation.
Integration of real-world sensors and hardware.

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

The UI will display some visualization of the robot in a warehouse grid; it allows users to visualize the planned and dynamic path of the robot and its interaction with the obstacles.

3.1.2 Hardware Interfaces

The system is in a simulation phase, future hardware interfaces would involve:
LIDAR: For obstacle detection and navigation.

Night Vision Camera: Optional hardware interface for improved vision in low light, enhance obstacle detection in dark.

GPS Sensor: The robot will include a GPS sensor for precise location tracking, ensuring it can follow predefined paths accurately. GPS module will provide real-time geospatial data to the robot's navigation system.

3.1.3 Software interfaces

The system interfaces with the following software :

ROS (Robot Operating System) for simulating robot behavior and integration with hardware.
Google colab for making simulation about guiding robot in predefined and dynamic path.

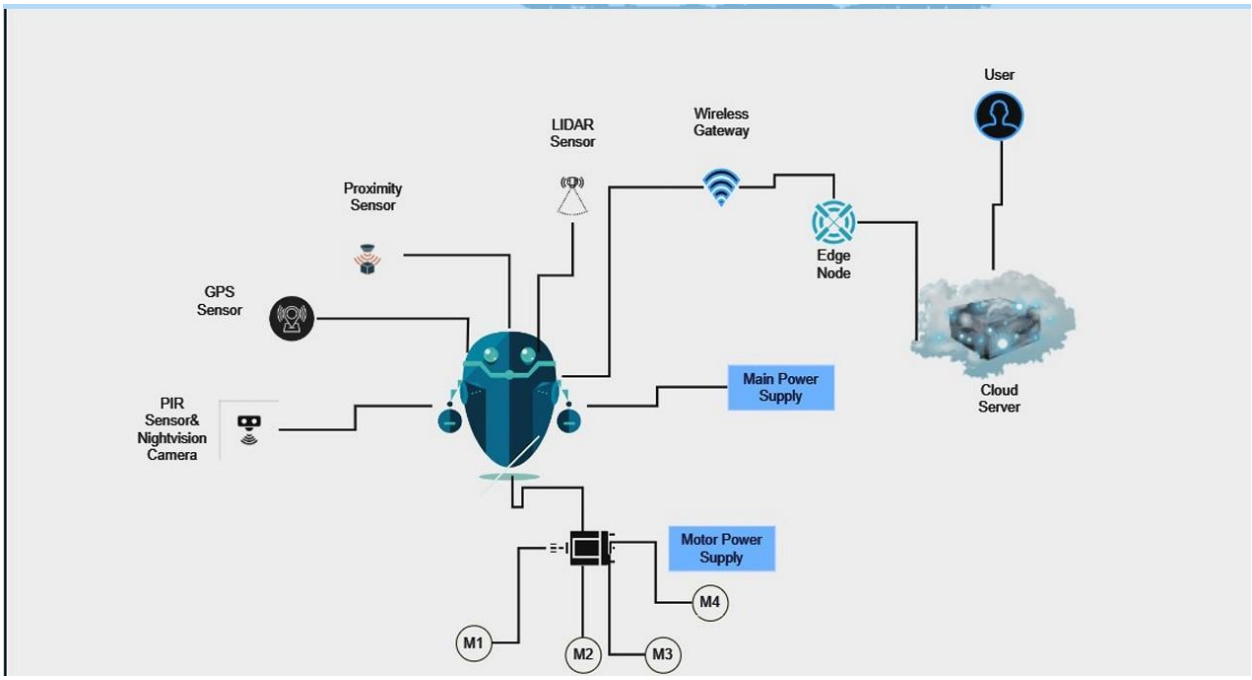
3.1.4 Communication Interfaces

Wi-Fi for communication between the robot and servers, especially for monitoring and control.

Internet connection for remote control and updates (depend on the implementation).

3.2 Architectural Design

3.2.1 High level Architectural Design



3.2.2 Hardware and software requirements with justification

Hardware Requirements:
Raspberry Pi: As the robot's controller for interfacing with sensors and actuators.
LIDER sensor : use for robots to identify and avoid obstacles while performing tasks.
Motors and Actuators: For robot mobility.

Software Requirements:
Python 3.x to implement algorithms.
ROS for robotics simulation and integration of hardware.
Simulation software such as Google colab for robotic movement simulation.

3.2.3 Risk Mitigation Plan with alternative solution identification

Risk: LiDAR and sensor hardware components unavailability.
Mitigation: Software simulation simulates the path and how the robot moves according to the decisions.

Risk: Inaccurate dynamic path, due to sensor imperfections.
Mitigation: Use simulated sensors for development; validate with real-world testing once the hardware is available.

3.2.4 Cost Benefit Analysis for the Proposed Solution

Cost: The initial simulation-based approach significantly reduces the cost as it does not require hardware straight away. Later, costs would be associated with the purchase of LiDAR, sensors, and actuators.
It will benefit from being able to test algorithms in a virtual environment without hardware upfront, thus making the path-planning system iteration and refinement much faster.

3.3 Performance requirements

Speed: The robot should be able to calculate path decisions and navigate the environment in real-time (less than 2 second per decision point).

Memory Usage: It should be RAM-friendly. The usage should be within 2GB for typical path-planning and sensor simulation.

Runtime Efficiency: Dynamic path adjustment algorithms should not introduce delays greater than the robot can handle with respect to its speed capabilities.

3.4 Design Constraints

Real-Time Processing: needs to adjust paths in real time based on sensor input; hence, it needs efficient algorithms.

Simulation Only: Currently limited to software; future hardware integration may require adaptation of existing algorithms.

Data Accuracy: Assume the simulated sensor data to approximate real-world data but may need fine-tuning when actual sensors are used.

3.5 Software system attributes

3.5.1 Reliability

Both static and dynamic path-planning should be flawlessly simulated by the system. Reliability will be verified under real-world circumstances after hardware integration.

3.5.2 Availability

For testing algorithms, the simulation needs to be accessible at all times. The robot system should have less downtime and run continuously in a deployed setting.

3.5.3 Security

Security is primarily about protecting communication between the robot and the monitoring system.

3.5.4 Maintainability

The simulation code shall be modular to allow future improvement in adding new sensors, algorithms, or integration with new hardware.

3.6 Other Requirements

Safety Standard : The path-planning algorithms of the robot should have safety features that meet the requirements for obstacle detection and avoidance, and emergency stop skills in real-world environments.

3.7 References

[1]S. Joy, Richard Lincoln Paulraj, Punith M, Shalini M, Srushti Goudar, and Ruthesh S, "A Raspberry Pi based Smart Security Patrol Robot," Feb. 2023, doi: https://doi.org/10.1109/iccmc56507.2023.10083908.

[2]"An Autonomous Surveillance Robot with IoT based Rescue System Enhancement ," ResearchGate, Mar. 03, 2022.https://www.researchgate.net/publication/359134908_An_Autonomous_Surveillance_Robot_with_IoT_based_Rescue_System_Enhancement

[3]J.-H. Jean and J.-L. Wang, “Development of an indoor patrol robot based on ultrasonic and vision data fusion,” Aug. 2013, doi: https://doi.org/10.1109/icma.2013.6618090.

[4]L. Huang, M. Zhou, K. Hao, and E. Hou, “A survey of multi-robot regular and adversarial patrolling,” IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 4, pp. 894–903, Jul. 2019, doi:https://doi.org/10.1109/jas.2019.1911537.

[5]Chun, W. H., & Papanikolopoulos, N. (2016). Robot surveillance and security. In Springer handbooks (pp. 1605–1626). https://doi.org/10.1007/978-3-319-32552-1_61