Independent University **Bangladesh (IUB)**

IUB Academic Repository

Computer Science and Engineering | Undergraduate Thesis

2025-05-04

Exploring the Efficiency of “Library Robot”: A Low-cost Guiding System for Library Navigation and Interaction Using Raspberry Pi and OpenCV

Arbi, Mumtahina

IUB

http://ar.iub.edu.bd/handle/11348/982

Downloaded from IUB Academic Repository

Exploring the Efficiency of “Library Robot”: A Low-cost Guiding System for Library Navigation and Interaction Using Raspberry Pi and OpenCV

By

Mumtahina Arbi
Student ID: **2130152**
Yasin Arafat
Student ID: **2130315**

Spring 2025

Supervisor:
Mohammad Shidujaman, PhD
Assistant Professor
Department of Computer Science & Engineering
Independent University, Bangladesh

May 4, 2025
Dissertation submitted in partial fulfillment of the requirements for the degree of Bachelor of
Science in Computer Science & Engineering

Department of Computer Science & Engineering
Independent University, Bangladesh

Attestation

We fully understand the importance of academic integrity and the university's strict anti-plagiarism policies. We hereby declare that the content of this project is entirely our original work. Any external sources, including software or written materials, that have been referenced or used in the development of this project have been properly cited by recognized academic conventions.

Name: Mumtahina Arbi

Signature with Date

Name: Yasin Arafat

Signature with Date

Acknowledgement

We commence by offering our heartfelt gratitude to the Almighty Allah for endowing us with the fortitude, perseverance, and capability to diligently embark on and successfully complete our senior project. Our profound appreciation extends to Dr. Mohammad Shidujaman our esteemed Supervisor, Assistant Professor of Independent University, Bangladesh. Their invaluable guidance, boundless patience, and devoted investment of time have been instrumental in steering us through the intricacies of our project journey and shaping the formulation of this comprehensive report. Under their expert tutelage, our project progressed seamlessly, and their insightful counsel proved indispensable at every juncture.

Moreover, we express our sincere gratitude to the committee members for their in valuable contributions during the defense session, which rendered the experience not only fruitful but also enriching. Their discerning feedback and invaluable recommendations served to augment the depth and quality of our project significantly.

In addition to our academic mentors, we are deeply indebted to our family and friends for their unwavering support and encouragement throughout our tenure in the B.Sc. program, particularly during the more challenging phases of our senior project.

Furthermore, our heartfelt appreciation extends to Independent University Bangladesh for fostering an enriching senior project program that has played an integral role in honing our skills and preparing us for the rigors of the research industry. Their robust support has been instrumental in shaping our academic and professional journey, and for that, we are immensely grateful.

Letter of Transmittal

Dr. Mohammad Shidujaman
Senior Project Supervisor, Assistant Professor
Department of Computer Science and Engineering,
Independent University, Bangladesh.

Subject: Senior project report on "Exploring the Efficacy of "RoboInsight": A Low-cost Guiding System for Library Navigation and Interaction".

Dear Sir,
We are sincerely thankful for the opportunity to submit our senior project report titled *"Exploring the Efficiency of 'Library Robot': A Low-cost Guiding System for Library Navigation and Interaction Using Raspberry Pi and OpenCV."* This report represents the culmination of our learning, effort, and commitment under your supervision. Throughout the project, we had the opportunity to apply a broad range of skills and emerging technologies, which significantly enhanced our knowledge and competence in this field. We deeply appreciate your valuable guidance, mentorship, and support during every stage of this journey. We hope that the report meets your expectations and reflects the standards of excellence you have encouraged us to uphold.

Thank you once again for your continued cooperation and encouragement throughout the semester.
Yours sincerely,

Mumtahina Arbi (2130152)
Yasin Arafat (2130315)
Department of Computer Science and Engineering,
Independent University, Bangladesh

Evaluation Committee

Supervision Panel

. . . . .	. . . . .
Supervisor	Co-Supervisor

Examiners

. . . . .	. . . . .
Internal Examiner 1	Internal Examiner 2
. . . . .	. . . . .
External Examiner 1	External Examiner 2

Office Use

. . . . .	. . . . .
Program Coordinator	Head of the Department

. . . . . .
Director of Graduate Studies, Research & Industry Relations

. . . . . .
Library

Table of Contents

Attestation ..... **i**

Acknowledgement ..... **ii**

Letter of Transmittal ..... **iii**

Evaluation Committee ..... **iv**

List of Figures ..... **viii**

List of Tables ..... **ix**

List of Abbreviations ..... **x**

Chapter 1 ..... **11**

Introduction **11**

1.1. Background 11

1.2. Statement of the Problem 12

1.3. Objectives 13

1.4. Scope 15

1.5. Research Questions 15

1.6. Conceptual Framework 16

1.7. Thesis Structure 17

Chapter 2 ..... **19**

Literature Review **19**

2.1. Introduction 19

2.2. Overview of Autonomous Navigation 19

2.2.1. Technologies and Methods 19

2.2.2. Applications of Line-Following Robots 20

2.3. Image Processing and OpenCV 23

2.3.1. Key Techniques 23

2.3.2. Applications in Robotics 23

2.4. Raspberry Pi in Robotics 24

2.4.1. Advantages of Raspberry Pi 24

2.4.2. Applications in Navigation 24

2.5. Related Works 25

2.5.1. Library Robots 27

2.5.2. Line-Following Robots 27
Chapter 3 28
Methodology 28
3.1. System Design 28
3.2. Hardware Components 29
3.2.1. Raspberry Pi 30
3.2.2. USB Camera 31
3.2.3. Motor Driver 32
3.2.4. DC Motors 32
3.2.5. Buck converter 33
3.2.6. Sensors 33
3.2.7. Wheel 33
3.2.8. Power Supply 34
3.3. Software Design 34
3.3.1. Software Tools and Libraries 34
3.4. Algorithm Development 36
3.4.1. Image Thresholding and Masking 36
3.4.2. Contour Detection and Centerline Calculation 36
3.4.3. Decision-Making for Motor Control 37
3.5. Mathematical Equations 37
3.6. Integration of Navigation and Software Interface 39
3.6.1. Software Interface 39
3.6.2. Text Recognition 39
3.7. Environment 40
Chapter 4 43
Implementation 43
4.1. Hardware Setup 43
4.1.1. Components and Connections 43
4.1.2. Assembly Diagram 44
4.1.3. Images of the Robot 45
4.2. Software Implementation 46
4.2.1. Code Structure 46
4.2.2. Key Functions 48

4.3. Testing and Calibration 49
4.3.1. Testing Process 50
4.3.2. Challenges and Resolutions 52
Chapter 5 55
Results and Discussion 55
5.1. Performance Evaluation 55
5.1.1. Line-Following Accuracy 55
5.1.2. Success Rate in Reaching the Target Shelf 56
5.1.3. Obstacle Detection Reliability 57
5.1.4. Response Time Analysis 58
5.2. Discussion 58
5.3. Limitations 59
Chapter 6 60
Conclusion and Future Work 60
6.1. Conclusion 60
6.1.1. Key Achievements 60
6.1.2. Contributions to the Field 60
6.2. Future Work 61
6.2.1. Enhanced Navigation for Larger Environments 61
6.2.2. Advanced Obstacle Avoidance 61
6.2.3. Improved User Interaction 62
6.2.4. Energy Efficiency and Sustainability 62
6.2.5. Scalability and Customization 62
6.2.6. Advanced Machine Learning for Navigation 62
6.3. Impact on Society 63
6.4. Sustainability of the Work 63
6.5. Social and Environmental Effects and Analysis 64
6.5.1. Positive Effects 64
6.5.2. Negative Effects 64
6.6. Ethics and Ethical Issues 64
6.7. Final Thoughts 65
Bibliography 67

List of Figures

Figure 1: A robot at the central Oodi library in Helsinki, Finland [1] 12
Figure 2: Conceptual framework of Library Robot Navigation 16
Figure 3:Pepper Bot 19
Figure 4: Different configurations existing library robot: (Lucus [85], UJI Robot [86], Line Following Robot [87], Book Smile [88], Parrot Bebop 2 [89], SMS Based Robot [90], Oodi [91], AROUND B [92], Padbot [93], LibRob [94], Pepper [95], LIBO [96], Beepbot [97]) 25
Figure 5: System Block Diagram 28
Figure 6: System Workflow 29
Figure 7: The Raspberry Pi 4 B [98] 31
Figure 8: C505 HD Webcam [99] 32
Figure 9: BTS7960 Motor Driver [100] 32
Figure 10: Planetary Gear Motor [101] 33
Figure 11: Buck Converter [101] 33
Figure 12: Ultrasonic Sensor [101] 33
Figure 13: Wheel [101] 34
Figure 14: 2200mah Lipo Battery [101] 34
Figure 15: OpenCV [102] 35
Figure 16: Python [103] 35
Figure 17: IUB Campus Library 41
Figure 18: Library Environment Navigation Map 41
Figure 19: System Circuit Diagram 44
Figure 20: Version-1 Prototype Testing Hardware Setup 45
Figure 21: Version-2 (a) Library Robot; (b) Location of Hardware setup 45
Figure 22:GPIO Setup 48
Figure 23:Image Capture and Preprocessing 48
Figure 24: Line Detection and Motor Control 48
Figure 25: Main Loop 49
Figure 26:Line-Following Test Output 50
Figure 27: Obstacle Detection Test 51
Figure 28: User Interface Test 51
Figure 29: Graph Line-Following Accuracy 55
Figure 30: Graph Success Rate in Reaching the Target Shelf 56
Figure 31: Pie Chart Obstacle Detection Reliability 57

List of Tables

Table 2.2.2. Comparison of Mobile Robot Navigation Algorithms 20
Table 2.3.2. Comparison of Image Processing Techniques 24
Table 2.5. Existing Mobile Robot In libraries 26
Table 3.2. Table of components list with cost 29
Table 5.1.1. Line-Following Accuracy 55
Table 5.1.2. Success Rate in Reaching the Target Shelf 56
Table 5.1.3: Obstacle Detection Reliability 57
Table 5.1.4: Response Time Analysis 58

List of Abbreviations

1) AI- Artificial Intelligence
2) DWA - dynamic window approach
3) DC- Direct current
4) GUI -graphical user interface
5) GA- Genetic algorithm
6) IMUs- Inertial measurement units
7) LiDAR- Light Detection and Ranging
8) NLP- Natural Language Processing
9) NLB- National Library Board
10) OCR -Optical Character Recognition
11) OpenCV- Open Source Computer Vision Library
12) PID- Proportional-Integral-Derivative
13) RRT - Rapidly-exploring Random Trees
14) RFID- Radio Frequency Identification
15) SSD -Single Shot MultiBox Detector
16) SLAM - simultaneous localization and mapping
17) USB- Universal Serial Bus
18) VFH- vector field histogram
19) YOLO -You Only Look Once

Chapter 1

Introduction

1.1. Background

Libraries are increasingly integrating advanced technologies to enhance user experience and streamline operations. One notable area of innovation is the deployment of robots to assist users in locating books, providing information, and performing routine tasks. These robots not only improve efficiency but also create a more engaging and futuristic library environment. However, many existing robotic solutions rely on complex and expensive technologies, including simultaneous localization and mapping and advanced path-planning algorithms, which are not feasible for small to medium-sized libraries with limited budgets.

In this context, line-following navigation has emerged as a cost-effective and practical solution for autonomous robots in structured environments like libraries. Line-following robots use simple yet robust algorithms to detect and follow predefined paths, such as lines on the floor. The method is especially effective in environments where the layout is consistent and predictable, such as libraries with clearly marked aisles and shelves. By leveraging low-cost hardware like the Raspberry Pi and open-source software like OpenCV, these robots can achieve reliable navigation without the need for expensive sensors or complex algorithms.

The development of computer vision and image processing technologies has significantly improved the performance of line-following robots. OpenCV, a free and open-source computer vision library, offers robust tools for processing images in real time, enabling robots to detect lines, contours, and other visual cues with high accuracy. When combined with microcontrollers like the Raspberry Pi, these technologies enable the development of low-cost, high-performance navigation systems that can be seamlessly incorporated into the current library infrastructure.

Raspberry Pi has become a widely adopted platform in robotics projects because of its cost-effectiveness and accessibility, versatility, and ease of use. As a single-board computer, the Raspberry Pi can handle complex tasks such as image processing, motor control, and data communication, making it an ideal choice for developing autonomous robots. Additionally, its compatibility with various sensors, cameras, and actuators allows for the creation of customized solutions tailored to specific applications.

In the context of library automation, a navigation system for a library robot must not only follow a predefined path but also interact with a software interface that provides information about the location of books. This integration ensures that the robot can guide users to the correct shelf based on the book's unique identifier or shelf number. By leveraging technologies such as optical character recognition (OCR) and text-to-speech, the robot can further enhance its usability and accessibility.

Several libraries worldwide have already begun experimenting with robotic systems to improve their services. For example:

- The Helsinki Central Library Oodi employs robots for interactive services and book delivery, utilizing advanced navigation algorithms to ensure efficient movement within the library [1].
- The National Library Board (NLB) of Singapore uses robots like Aurora to streamline book sorting and delivery processes, demonstrating the potential of robotics in library automation [2].

- The Yi Sun-sin Library in South Korea features robots that assist users in locating books and provide interactive services, contributing to a futuristic and engaging library experience [3].



Figure 1: A robot at the central Oodi library in Helsinki, Finland [1]

While these examples highlight the potential of robotics in libraries, they often rely on expensive and complex technologies that are not accessible to all libraries. This project aims to address this gap by developing a low-cost, line-following navigation system for library robots, making the benefits of automation accessible to a wider range of institutions.

1.2. Statement of the Problem

In conventional library setups, finding a particular book can be a time-consuming and frustrating task for users, especially in large libraries with extensive collections. While digital catalog systems have streamlined the process of searching for books, users still face challenges in navigating the physical library space to find the exact location of the book on the shelf. The process can be especially challenging for users who are unfamiliar with the library layout or have mobility issues. Additionally, library staff often spend significant time assisting users with book location, which could otherwise be utilized for more complex tasks.
Existing solutions, such as static maps or signage, are often insufficient to address these challenges. While some libraries have experimented with robotic systems for tasks like book sorting and retrieval, these systems typically rely on expensive and complex technologies, such as LiDAR, SLAM (Simultaneous Localization and Mapping), or advanced path-planning algorithms. These solutions are not feasible for small to medium-sized libraries with limited budgets and resources.

Furthermore, many of these systems lack the ability to interact with users in a simple and intuitive manner, limiting their practicality and adoption.
The primary problem addressed by this project is absence of a cost effective, user-friendly navigation for library robots that can:

1. Follow a predefined path (e.g., a line on the floor) to navigate through the library environment.
2. Detect objects (e.g., bookshelves or obstacles) in its path and stop when an object is detected.
3. Integrate with existing library software to receive the target shelf number and guide users to the correct location.

This project focuses on developing a navigation system using Raspberry Pi and OpenCV to achieve these objectives. The system will leverage line-following algorithms for navigation, computer vision techniques for object detection, and motor control logic to stop the robot when an object is detected. By combining these technologies, the proposed system seeks to deliver an affordable and effective method for locating books within library environments, while also addressing the challenges of obstacle avoidance and user interaction.

1.3. Objectives

The project aims to design a cost-effective autonomous navigation system for a library robot, enabling it to guide users to their desired books with precision and efficiency.To achieve this, the following specific objectives have been defined:

Design and Implement a Line-Following Navigation System:
Using a USB camera and OpenCV, the robot takes real-time images of the route ahead. These images are processed using techniques like image thresholding and contour detection to identify the line. The system calculates the line's position relative to the robot's center, enabling it to assess the appropriate direction—whether to advance, veer left, or shift to the right. To ensure reliability, the system handles variations in lighting, path curvature, and minor obstacles. The navigation logic, implemented on a Raspberry Pi, controls the robot's motors via a motor driver. This cost-effective solution ensures precise and efficient navigation in structured library environments, guiding users to the correct shelf with minimal deviations.

Integrate Object Detection and Stopping Mechanism:
Using OpenCV, the robot's camera captures real-time images, which are processed to detect obstacles or target objects (e.g., bookshelves). Techniques like color filtering or edge detection isolate objects from the background. When an object is detected within a predefined distance, the robot stops by signaling the motor driver to halt movement. This ensures safe navigation and prevents collisions. The system is designed to handle false positives and dynamic obstacles, making it reliable in real-world library environments.

Develop a Software Interface for Shelf Number Input:
This interface, which can be a graphical user interface (GUI) or a web-based application, allows users to manually enter the shelf number or retrieve it from the library's database. Once the shelf number is provided, the interface communicates it to the robot's navigation system. The robot then uses this information to navigate to the correct shelf. Features like text-to-speech or visual indicators can be added to enhance usability and accessibility, ensuring the system is intuitive for all users.

Ensure Real-Time Performance and Reliability:
This is achieved by optimizing the image processing and navigation algorithms to minimize latency. The Raspberry Pi handles tasks like camera frame processing and motor control efficiently, ensuring smooth operation. Extensive testing under various conditions (e.g., lighting changes, path irregularities) ensures the system is robust. Error-correction algorithms and fail-safe mechanisms are implemented to handle deviations or false detections, ensuring reliable performance in dynamic library environments.

Achieve Cost-Effectiveness and Scalability:
Affordable hardware components like the Raspberry Pi, USB camera, and standard DC motors are used to keep costs low. Open-source software like OpenCV further reduces expenses. The system is designed to be modular, allowing it to adapt to different library layouts and requirements. For example, the line-following algorithm can be adjusted for different path designs, and the object detection system can be trained to recognize new objects. This ensures the system is accessible to libraries with limited budgets and can be easily customized.

Enhance User Interaction and Accessibility:
The robot incorporates features like text-to-speech for auditory feedback and visual indicators (e.g., LED lights or a display screen) to convey information. These features make the system intuitive and user-friendly, particularly for individuals with disabilities. The interface is designed to be simple and easy to use, ensuring a positive experience for all library patrons. By prioritizing accessibility, the project ensures the robot is inclusive and engaging for everyone.

By achieving these objectives, this project seeks to offer a practical and effective solution for locating books location in libraries, leveraging the capabilities of line-following navigation, object detection, and low-cost hardware.

1.4. Scope

This project focuses on developing a low-cost, autonomous navigation system for a library robot designed to assist users in locating books efficiently. The scope covers the creation and execution of a line-following navigation system utilizing Raspberry Pi and OpenCV, allowing the robot to accurately follow a set path (e.g., a line on the ground). The system will also incorporate basic object detection to identify obstacles or target objects (e.g., bookshelves) and stop the robot when necessary. A software interface will be developed to allow users to input the target shelf number, integrating this information with the robot's navigation system.
The project is limited to structured environments with clearly marked paths and predefined layouts, such as libraries with fixed aisles and shelves. It does not address navigation in unstructured or dynamic environments. The system relies on low-cost hardware (e.g., Raspberry Pi, USB camera) and open-source software (e.g., OpenCV) to ensure affordability and accessibility for small to medium-sized libraries. While the robot will be tested in a controlled environment, it is not designed for outdoor use or highly crowded spaces. The project also excludes advanced features, including techniques like simultaneous localization and mapping (SLAM) or artificial intelligence-based decision-making, focusing instead on simple, reliable, and cost-effective solutions.

By defining these boundaries, the project ensures a clear and achievable goal: to create a functional and affordable navigation system for library robots that enhances user experience without requiring complex or expensive technologies.

1.5. Research Questions

The research questions addressed were:

1. **How can a line-following navigation system using Raspberry Pi and OpenCV be crafted to precisely navigate a set path within a library setting?**

This question focuses on the core functionality of robot, addressing design and implementation of navigation system.

2. **Which image processing methods can be applied to integrate object detection into the navigation system, enabling the robot to identify obstacles or target objects (e.g., bookshelves) and stop when necessary?**

This question explores the addition of object detection functionality and its integration with the navigation system.

3. **How can a user-friendly software interface be developed to allow users to input the target shelf number and guide the robot to the correct location efficiently?**

This question emphasizes the importance of user interaction and accessibility in the design of the system.

1.6. Conceptual Framework

The conceptual framework for this project is built around three core components: Input, Process, and Output. These components are interconnected and work together to achieve the project's objectives. Below is a detailed explanation of each component:

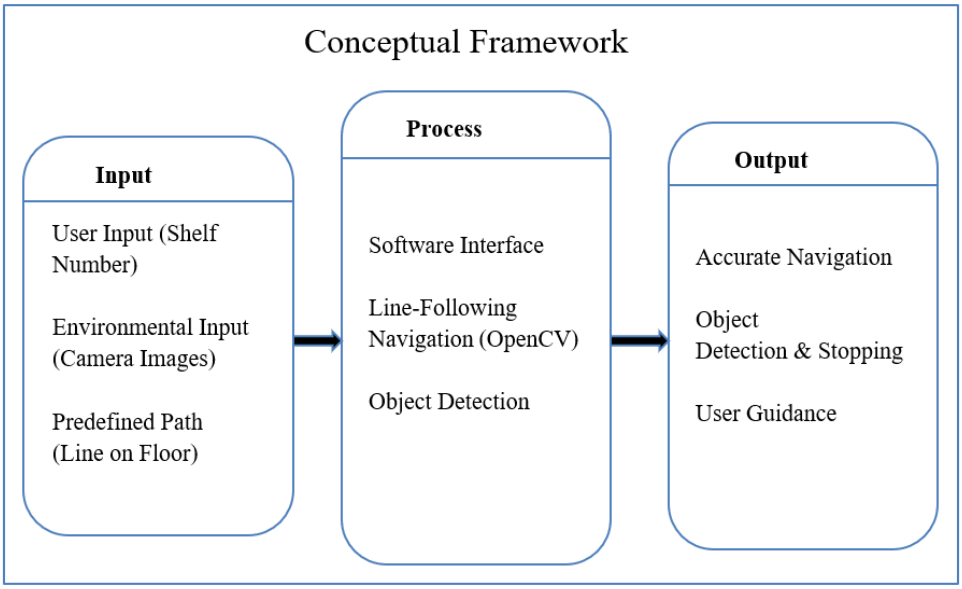


Figure 2: Conceptual framework of Library Robot Navigation

Input

The input component represents the data and information required for the robot to operate. This includes:

- User Input: The target shelf number provided by the user through a software interface.
- Environmental Input: Real-time images of the library environment captured by the robot's USB camera.
- Predefined Path: The line on the floor that the robot follows for navigation.

Process

The process component involves the algorithms and systems that transform the input into actionable outputs. This includes:

- Line-Following Navigation:
    - The robot acquires real-time images of path using a USB camera.
    - OpenCV processes the images to detect the line through techniques like thresholding and contour detection.
    - The system calculates position of the line about the robot's center and modifies the robot's movement accordingly (forward, left, or right) to stay on track.
- Object Detection:
    - The robot uses computer vision techniques to detect obstacles or target objects (e.g., bookshelves) in its path.

    - Upon detecting an object, the system triggers a stopping mechanism to halt the robot's movement.
- Software Interface:
    - The user inputs the target shelf number through a graphical or web-based interface.
    - The interface communicates the shelf number to the robot's navigation system, enabling it to guide the user to the correct location.

Output

The output component represents the results and outcomes of the system. This includes:

- Accurate Navigation: The robot effectively navigates along the specified path and arrives at the designated shelf.
- Object Detection and Stopping: The robot detects obstacles or target objects and stops when necessary, ensuring safe and precise operation.
- User Guidance: The robot provides clear and intuitive guidance to the user, enhancing the overall library experience.

Interconnections

- The user input (target shelf number) is processed by the software interface, which communicates with the navigation system.
- The environmental input (real-time images) is processed by the line-following and object detection algorithms, which control the robot's movement.
- The output (accurate navigation, object detection, and user guidance) is achieved through the seamless integration of the input and process components.

1.7. Thesis Structure

The thesis is structured into six chapters, with each focusing on a particular component of the project. The arrangement follows a logical flow, starting with the introduction and background, followed by the implementation, results, and conclusions. The chapters are as follows:

Chapter 1: Introduction

The chapter introduces project, including background, problem statement, objectives, scope, and thesis structure. It provides an overview of the project's significance and sets the stage for the subsequent chapters.

Chapter 2: Literature Review

The chapter examines current research and technologies associated with autonomous navigation, line-following robots, object detection, and library automation. It identifies gaps in the current knowledge and highlights the contributions of this project.

Chapter 3: Methodology

The chapter outlines the design and execution of the library robot's navigation system. It covers the hardware components, software tools, and algorithms used for line-following navigation, object detection, and user interaction.

Chapter 4: Implementation

The chapter offers an in-depth description of the system implementation, including the integration of hardware and software components. It covers the testing and calibration procedures to validate the system's reliability and performance.

Chapter 5: Results and Discussion

The chapter showcases the performance outcomes of the system, including its accuracy in navigation, object detection, and user interaction. It also discusses the limitations of the system and compares the results with the project's objectives.

Chapter 6: Conclusion and Future Work

The chapter provides an overview of the main results of the project and highlights its contributions to field of library automation. It also includes the following sections to provide a broader perspective on the project's impact and implications:

Impact on Society: Discusses how the library robot can improve accessibility, enhance user experience, and support library staff, contributing to a more productive and all-encompassing library system.
Sustainability of the Work: Explores long-term viability of the system, including its cost-effectiveness, scalability, and potential for adoption in other libraries or similar environments.
Social and Environmental Effects Analysis: Examines the positive social impacts (e.g., improved access to knowledge) and environmental benefits (e.g., reduced energy consumption compared to traditional systems) of the project.
Addressing Ethics and Ethical Issues: Analyzes potential ethical concerns, including data privacy and user consent, and the impact of automation on library staff, and proposes strategies to address these issues.
Future Work: Suggests potential improvements and extensions for the system, such as integrating advanced features (e.g., voice commands, AI-based recommendations) or expanding its application to other domains.

Chapter 2

Literature Review

2.1. Introduction

The chapter reviews existing research and technologies related to autonomous navigation, image processing, and Raspberry Pi-based robotics, with a focus on their application in library robots. The review is organized into four main sections: Overview of Autonomous Navigation, Image Processing and OpenCV, Raspberry Pi in Robotics, and Related Works. By synthesizing findings from various studies, this chapter identifies gaps in the literature and highlights the contributions of this project.

2.2. Overview of Autonomous Navigation

Autonomous navigation is an essential feature for robots operating in dynamic environments. It involves localization, mapping, path planning, and obstacle avoidance, enabling robots to move efficiently and safely without human intervention.



Figure 3:Pepper Bot

2.2.1. Technologies and Methods

Localization and Mapping: Localization refers to identifying the robot's position within its environment, meanwhile, mapping involves generating a representation of the environment. Methods like simultaneous localization and mapping (SLAM) are widely used to achieve these tasks [4, 5]. SLAM allows robots to build a map of their environment while continuously monitoring their position within it. Recent advancements in sensor fusion have further improved localization accuracy by combining data from various sensors, including LiDAR, cameras, and inertial measurement units (IMUs)[6, 7, 8].

Path Planning: Algorithms such as A*, RRT, and GA are widely used for calculating optimal routes while avoiding obstacles [9, 10, 11].

Obstacle Avoidance: Methods like Vector Field Histogram (VFH) and Potential Field Methods enable robots to dynamically adjust their paths based on sensor data [12, 13].

2.2.2. Applications of Line-Following Robots

Line-following robots are a popular solution for indoor navigation in structured environments. They use vision-based algorithms to track predefined paths, such as lines on the floor. Studies have demonstrated their effectiveness in applications ranging from industrial automation to library robotics [14, 15, 16, 17]. For instance:

Yang et al. [14]: Build a robust line-following algorithm employing edge detection and scanning models.

Kondakor et al. [15]: Highlighted the superiority of camera-based tracking over optical sensors in real-time applications.

Table 2.2.2. Comparison of Mobile Robot Navigation Algorithms

Paper	Algorithm	Advantage	Drawbacks
[18,19,20,21]	Prioritized Decoupled Path-Finding	Conflict-Free-Path Planning, Scalability, Reduced Complexity, Flexibility in Application, Fast Initial Solution, Good Performance in Sparse Environments	Sub-Optimal Solutions, Incomplete Solutions, Priority Dependence, Potential for Deadlocks, Difficulty with Tight Coupling, Performance Degradation with Increasing Agent Density, Conflict Resolution Challenges
[22,23,24,25]	A*	Optimized Path Finding, Heuristic Efficiency, Versatility in Complex Environments, Proven Effectiveness	High Computational Cost, Suboptimal in Some Real-World Scenarios, Dependency on Heuristics, Difficulty in Handling Moving Obstacles, Dynamic Environments
[26,27,28, 29,30]	Neuro-Fuzzy Controllers	Adaptability (complex and dynamic environments), Robustness, Simplicity(implement)	Complexity (time, Designing and tuning), Computational Load
[31,32,33,34]	Genetic Algorithms (GA)	Global Optimization, Flexibility, Robustness (noisy and dynamic environments), Parallelism, Versatility (optimization, clustering, and feature selection, among others)	Convergence (Time, Speed, Premature) Parameter Tuning, Computational Cost, Complexity in Implementation

[35,36,37,38]	Simulated Annealing Algorithm (SAA)	Global Search Capability, Flexibility (optimization problems)	Slow Convergence, Parameter Sensitivity
[39]	Bacterial Foraging Optimization (BFO)	effective in dynamic environments	complex to implement and tune
[40,41]	Random Particle Optimization Algorithm (RPOA)	effective in local path planning	Inefficient in finding optimal solutions
[42,43,44,45]	Firefly Algorithm	Effective (global optimization and pathfinding and simple parameters, Multi-Objective Optimization), Self-Adaptive Variants, Robust to Noisy and Dynamic Environments, Versatility (constrained, multi-objective, and combinatorial problems)	struggle with very high-dimensional spaces or require extensive tuning, Local Optima, Parameter Sensitivity, Computational Cost, Parameter Sensitivity, Premature Convergence
[46,47,48,49]	Particle Swarm Optimization (PSO)	Simplicity (parameters to adjust), Efficiency (faster convergence), Global Search Ability.(3D environment)	Premature Convergence, Parameter Tuning, Local Optima, Computational Complexity,
[50,51,52,53]	Ant Colony Optimization (ACO)	Particularly effective pathfinding, Adaptability (combinatorial optimization, continuous optimization, and multi-objective optimization, dynamic environment), Global Optima, Stochastic Elements, Exploration and Exploitation Balance	Computational Cost, Parameter Sensitivity, Premature Convergence, Difficulty in Defining Pheromone Updating Rules, Risk of Stagnation
[54,55,56,57,58]	ICSb-CHECK	Formal Safety Guarantees, Efficiency in graphics rendering techniques to compute the ICSb set quickly, Scalability(linearly, suitable for real-time applications)	Conservatism (some scenarios), Complexity (compute unions and intersections of shapes in high-dimensional space)

[54,55,59,60]	PASSAVOID	Provable Safety (avoid collision), Reactive Nature (unknown or dynamic environments), Bias Towards Movement	Lack of Foresight, Goal-Oriented Limitations, Limited Safety Scope
[61]	TOSL-iBSR (True Online SARSA with Informed-Biased Softmax Regression)	efficiency by biasing state selection towards more promising states (2D environment), better performance in terms of mean-average reward and convergence rate, Capable of handle complex navigation, less computational cost.	The algorithm assumes perfect knowledge of the robot's pose and noise-free sensor data, which may not be realistic in real-world applications, It is tested in a simulated environment, and the results might not fully translate to real-world scenarios without additional adjustments.
[61]	Q-SR (Q-learning with Softmax Regression)	A well-known and widely used algorithm in RL with a straightforward implementation, Provides a baseline for comparing other algorithms.	Slower convergence and lower mean-average reward compared to TOSL-iBSR, Less efficient in complex environments, requiring more steps to complete tasks.
[61]	TOSL-QBIASSR (True Online SARSA with Q-Biased Softmax Regression)	Introduces bias into the softmax regression, similar to TOSL-iBSR, to improve performance. Performs better than Q-SR but not as well as TOSL-iBSR.	Slower learning and higher computational cost compared to TOSL-iBSR, requires more learning steps to achieve similar performance levels.
[62,63,64,65,66]	Neural Networks (NN)	Capable of handling complex patterns and dynamic environments; adaptable	Requires large amounts of training data; computationally expensive
[67]	Fuzzy Logic (FL)	Handles imprecise and uncertain data well; flexible	Can be complex to design and tune fuzzy rules
[68,69]	RRT (Rapidly-exploring Random Tree)	Suitable (non-holonomic, kino-dynamic planning), Does not require a predefined graph or network	create redundant points, time-consuming, might not produce optimal paths
[70,71,72]	RRT Connect (Bi-directional RRT)	efficient in finding a path as it explores in both directions	Might not handle differential constraints well

[73,74]	RRT (Optimal RRT) *	Capable of solving high-dimensional problems and finding better paths over time.	Computationally more intensive compared to basic RRT
[74,75]	RRT-Smart*	Produces near-optimal paths more quickly and with reduced execution time	Complexity in implementation and parameter tuning.
[76]	Quick-RRT*	Generates lower-cost paths than RRT*	More complex and might require more computation
[77,78]	Sensor-based Random Tree (SRT)	Adapts to unexpected obstacles and provides safety measures based on real-time sensor data	Accuracy depends on the sensors' features and might be influenced by the shape of the LSA

While line-following robots are efficient and cost-effective, they often lack real-time adaptability in dynamic environments. Future research should focus on integrating advanced machine learning for improved accuracy and adaptive decision-making.

2.3. Image Processing and OpenCV

Image processing plays a crucial role in autonomous navigation, enabling robots to interpret visual data and make informed decisions. OpenCV, a free and open-source computer vision library, offers robust tools for tasks like contour detection, thresholding, and object recognition.

2.3.1. Key Techniques

Thresholding: Converts grayscale images into binary images, isolating objects of interest from the background [79].
Contour Detection: Identifies the boundaries of objects in an image, enabling precise tracking of lines or obstacles [80].
Edge Detection: Highlights significant changes in pixel intensity, useful for detecting lines or paths [14].

2.3.2. Applications in Robotics

Farkh and Aljaloud [15]: Used OpenCV and a PID controller for smooth and accurate line-following.
Setiawan et al. [81]: Employed HSV-based image processing for self-driving cars, achieving reliable path tracking under varying lighting conditions.
Jin et al. [82]: Showcased visual line-following in intelligent cars using camera calibration.

Table 2.3.2. Comparison of Image Processing Techniques

Technique	Advantages	Drawbacks
Thresholding	Simple and effective for isolating objects from the background.	Sensitive to lighting variations.
Contour Detection	Provides precise boundaries for object tracking.	Computationally expensive for complex images.
Edge Detection	Highlights significant changes in pixel intensity, useful for line detection.	May produce noisy results in low-contrast images.

OpenCV-based image processing techniques are highly effective for real-time navigation in structured environments. However, challenges such as lighting variations and computational efficiency remain unresolved. Future research should explore optimized algorithms and hardware acceleration for improved performance.

2.4. Raspberry Pi in Robotics

Raspberry Pi has become a widely-used platform for robotics projects because of its low cost, flexibility, and user-friendly design. It is widely used in applications ranging from educational robots to industrial automation.

2.4.1. Advantages of Raspberry Pi

Cost-Effectiveness: Raspberry Pi provides a low-cost alternative to traditional microcontrollers, making it accessible for small to medium-sized projects [16].

Versatility: It supports a broad selection of sensors, cameras, actuators, enabling development of customized robotic systems [17].

Open-Source Software: Compatibility with libraries like OpenCV and Python simplifies the implementation of complex algorithms [79, 83].

2.4.2. Applications in Navigation

Horak and Zalud [79]: Used Raspberry Pi and MATLAB's Simulink for image processing in line-following robots.

Amir et al. [80]: Applied Raspberry Pi to marine environments, demonstrating its adaptability to diverse use cases.

Patel et al. [84]: Build computer vision-guided navigation system using OpenCV and NumPy on a Raspberry Pi platform.

Raspberry-Pi is a tool for developing low-cost navigation systems. However, its computational limitations can pose challenges for real-time applications. Future research should focus on optimizing algorithms and leveraging hardware acceleration to overcome these limitations.

2.5. Related Works

This section reviews similar projects and research papers related to library robots and line-following robots, identifying gaps in existing solutions.

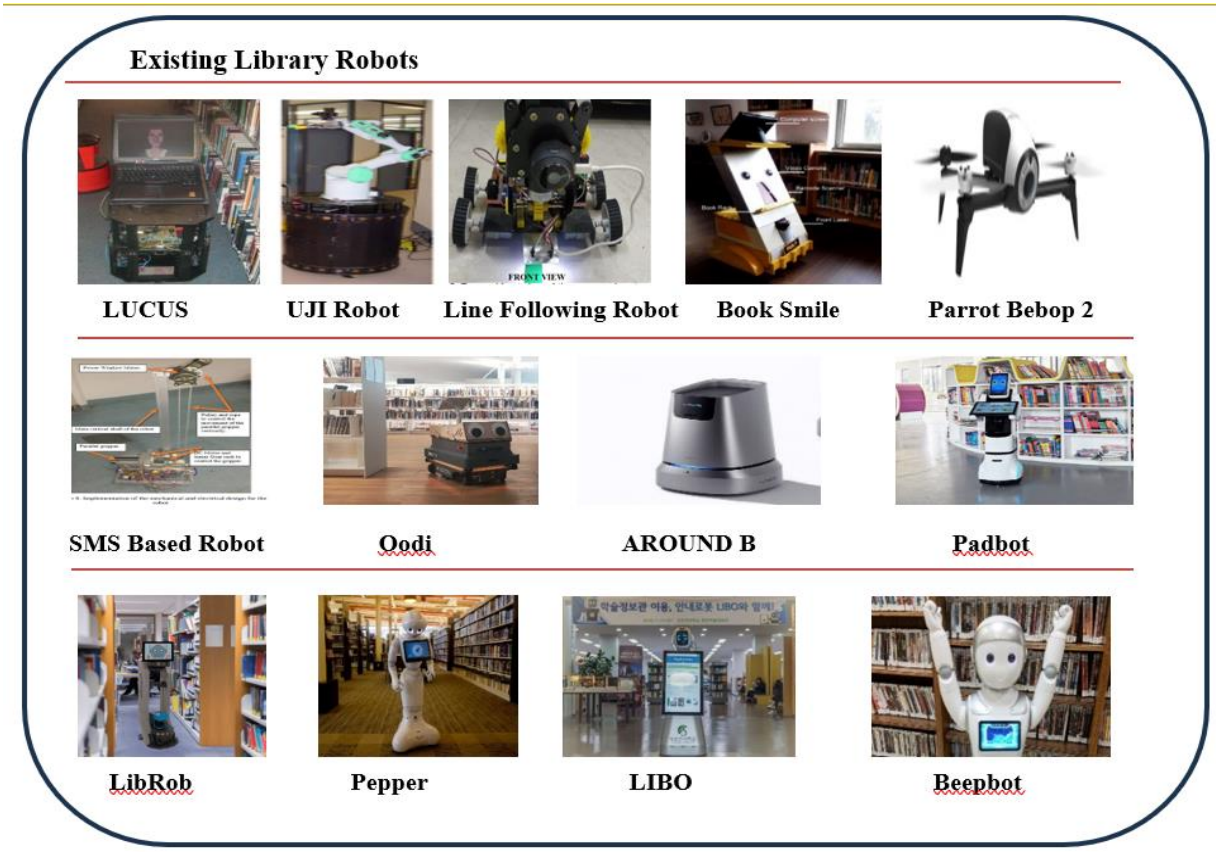


Figure 4: Different configurations existing library robot: (Lucus [85], UJI Robot [86], Line Following Robot [87], Book Smile [88], Parrot Bebop 2 [89], SMS Based Robot [90], Oodi [91], AROUND B [92], Padbot [93], LibRob [94], Pepper [95], LIBO [96], Beepbot [97])

Table 2.5 below highlights mobile robots used in library environments, their core applications, and the algorithms they employ.

Table 2.5. Existing Mobile Robot In libraries

Robot	Application	Algorithm	Reference
LUCUS	-Guiding Users (elderly, disabled, or unfamiliar with the library's layout) to specific book locations or sections -Assisting with Inventory -Obstacle Detection -Interaction (User Guidance) -Localization (combination of odometry, sonar data, and vision-based algorithms like the Extended Kalman Filter)	Localization System, User Guidance, Extended Kalman Filter	[85]
UJI Librarian Robot	- Autonomous searching, locating, and retrieving specific books - Book Identification (OCR) - Grasping Mechanism - Navigation with pre-acquired map	Book Identification, Grasping Mechanism, Optical Character Recognition (OCR)	[86]
Line Following Library Robot	- Automated Book Retrieval and Return - Library Management - Inventory Management	Path Navigation, Book Identification, Voice Command Recognition	[87]
Book Smile	Guide child patrons in a library	Localization System, Human Sensing, Context-Aware Navigation, Speech and Visual Guidance	[88]
Parrot Bebop 2 Quadcopter	Automate book inventory and localization in a library	ArUco Marker Detection, Waypoint-Based Path Planning, Transition Processing, OCR	[89]
SMS Based Pick and Place Bot	For library management	Path Planning, Book Identification, Object Manipulation, SMS Command Handling	[90]
Oodi	Guide customers to books and book categories	Navigation Algorithm, Interaction Algorithm, Emotion System	[91]
AROUND B	Assists customers in bookstores	Navigation, Object Detection, Path Planning	[92]
Padbot	Guide Library, Education, Delivery	Navigation, Communication	[93]
LibRob	Assist library users	Navigation and Localization (SLAM), NLP, Multi-	[94]

		language GUI, Motion Control, Database Adapter	
Pepper	Retail Assistance, Hospitality, Education, Research, Public Spaces	NLP, Facial Recognition, Path Planning and Navigation	[95]
LIBO	Enhance the academic and campus experience	NLP, Path Planning, Machine Learning	[96]
Beepbot	Library experience by engaging visitors interactively	Navigation and Mapping (SLAM), Speech Recognition, AI and Machine Learning	[97]

2.5.1. Library Robots

Helsinki Central Library Oodi [1]: Utilizes robots equipped with SLAM and path planning algorithms for book delivery and user assistance.

National Library Board (NLB) of Singapore [2]: Employs robots like Aurora for book sorting and retrieval, leveraging advanced navigation and object detection systems.

Yi Sun-sin Library, South Korea [3]: Features robots that provide interactive services and book delivery, enhancing user engagement.

2.5.2. Line-Following Robots

Farkh and Aljaloud [15]: Demonstrated the effectiveness of PID controllers for smooth line-following.
Yang et al. [16]: Developed a robust algorithm using edge detection and scanning models.
Setiawan et al. [81]: Employed HSV-based image processing for self-driving cars.

While existing solutions demonstrate the potential of library and line-following robots, they often rely on expensive and complex technologies, limiting their accessibility for small to medium-sized libraries. This project addresses these gaps by developing a low-cost, line-following navigation system using Raspberry Pi and OpenCV.

Chapter 3

Methodology

3.1. System Design

Library robot's navigation system is designed to autonomously follow a predefined path, detect obstacles, and guide users to the correct shelf based on the target shelf number. The system architecture consists of hardware components (e.g., Raspberry Pi, USB camera, motors) and software modules (e.g., OpenCV, Python) that work together to achieve these tasks.

Block Diagram

Below is a block diagram illustrating the system architecture:

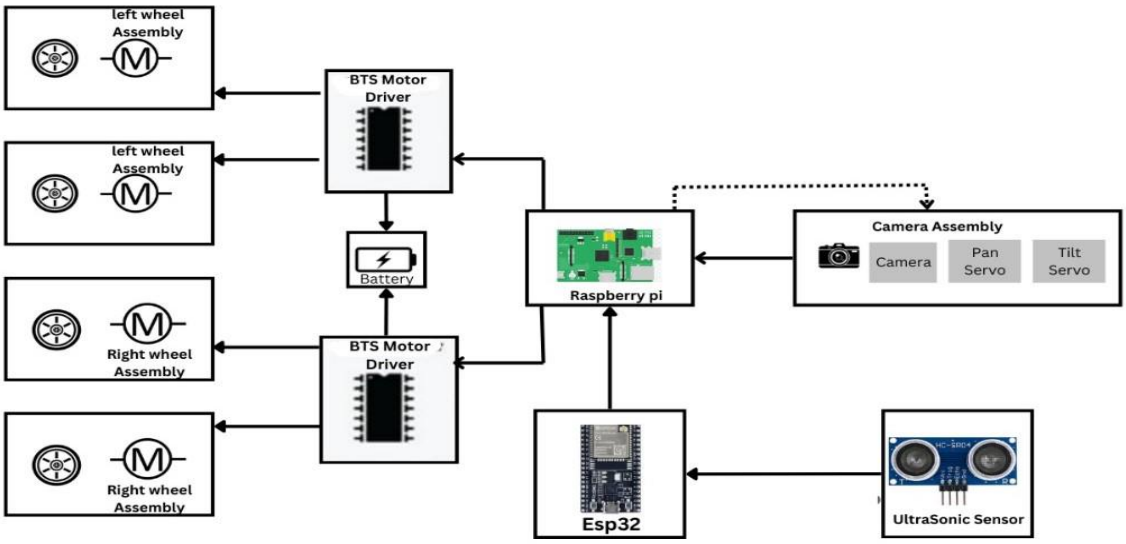


Figure 5: Block Diagram of the System

The system's block diagram methodology illustrates how the Raspberry Pi, ESP32, motor drivers, sensors, and camera are interconnected. Two motor drivers are directly linked to the power supply, with each motor driver controlling two motors. The Raspberry Pi is connected to both motor drivers and the camera, forming a cohesive system for effective navigation.

Ultrasonic sensor is connected to the ESP32, which is responsible for detecting obstacles in the path. The ESP32 communicates with the Raspberry Pi, sending real-time obstacle detection data to assist in navigation.

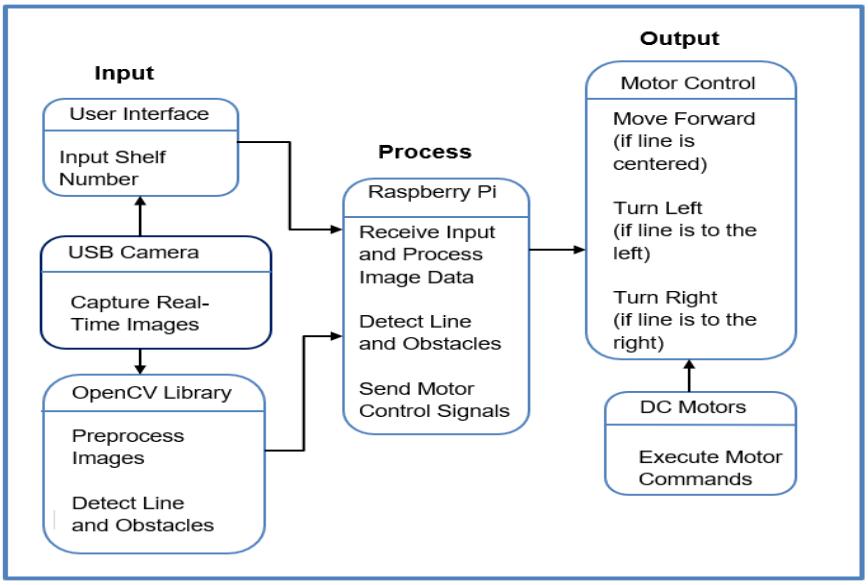


Figure 6: System Workflow

System Workflow

1. The user inputs the target shelf number through a software interface.
2. The USB camera captures real-time images of the predefined path (e.g., a line on the floor).
3. The Raspberry Pi processes the images using OpenCV to detect the line and any obstacles.
4. Based on the processed data, the system controls the motors to navigate the robot along the path.
5. When the robot reaches the target shelf, it stops and provides user guidance (e.g., visual or auditory feedback).

3.2. Hardware Components

The hardware components of the library robot are selected to ensure cost-effectiveness, reliability, and ease of integration.

Table 3.2. Table of components list with cost

Component	Specification	Function	Price (Taka)	Price (USD)
Raspberry Pi 4B	Supports 2.4GHz and 5GHz Wi-Fi, powered by a 64-bit quad-core ARMv8 processor with 1GB of RAM	Main processing unit	16500	$135.25

Logitech C505 HD webcam	HD webcam offering 720p resolution with a long-range microphone		2300	$18.85
Motor driver (2x)	BTS motor driver	Enables movement and physical actions	900	$7.38
Sensors	ultrasonic sensor	Environmental interaction and data collection	100	$0.82
Motor (4x)	4.37gb gear motor	Enables movement and physical actions	3000	$24.59
Wheels(4x)	movement or transportation	Supporting loads in machines	1400	$11.48
Power Supply (Lithium Battery 12V 100Ah)	Rechargeable battery	Powers the robot consistently	2450	$20.08
Wire & Mounting Hardware	For connecting and remove damage	Connecting power & remove body damage	5000	$40.98
3D Printed Body	Custom enclosure for the robot	Protective enclosure for components	35000	$286.89
Buck converter			100	$0.82
Esp32			500	$4.10
Total Cost			**67,250**	**$546.32**

3.2.1. Raspberry Pi

Raspberry Pi is a family of compact single-board computers (SBCs) created by the Raspberry Pi Foundation in collaboration with Broadcom, based in the United Kingdom. Initially, the project aimed to encourage the teaching of basic computer science in schools. However, the original model exceeded expectations in popularity, expanding beyond its educational focus. It gained traction in various fields such as robotics, home and industrial automation, and among computer and electronics enthusiasts, thanks to its affordability, modular design, open architecture, and support for HDMI and USB standards.



Figure 7: The Raspberry Pi 4 B [98]

The Raspberry Pi 4 Models A and B were introduced in June 2019, featuring a 1.5 GHz 64-bit quad-core ARM Cortex-A72 processor, integrated 802.11ac Wi-Fi, Bluetooth 5, and full gigabit Ethernet with no throughput limitations. It includes two USB 2.0 ports, two USB 3.0 ports, and options for 1, 2, 4, or 8 GB of RAM. Additionally, it supports dual-monitor setups through two micro HDMI (Type D) ports, allowing up to 4K resolution. The 1 GB RAM variant has been discontinued, and the 2 GB version's price has been lowered. The 8 GB variant features an updated circuit board. The Raspberry Pi 4 is powered through a USB-C port, which can also supply extra power to connected peripherals when using the appropriate power supply unit (PSU). However, it operates on 5 volts, unlike some other mini computers that support 9 or 12 volts. Initially, the Raspberry Pi 4 had a design flaw that prevented third-party e-marked USB cables, like those used with MacBooks, from providing power. Testing by Tom's Hardware on 14 cables showed that 11 worked correctly, and this issue was addressed in revision 1.2, released in late 2019. By mid-2021, Pi 4 B models were released with the improved Broadcom BCM2711C0 chip, which is now used in both the Pi 4 B and Pi 400, although the factory clock frequency for the Pi 4 B remained unchanged.

The Raspberry Pi 4 is equipped with a Broadcom BCM2711 System on Chip (SoC), featuring a 1.5 GHz (upgraded to 1.8 GHz in later models) 64-bit quad-core ARM Cortex-A72 processor, accompanied by a 1 MB shared L2 cache. In contrast to earlier versions, which employed a custom interrupt controller not ideal for virtualization, the BCM2711 integrates an ARM Generic Interrupt Controller (GIC) 2.0, enabling efficient interrupt distribution for ARM-based virtualization. Additionally, the Raspberry Pi 4 replaces the VideoCore IV GPU of previous models with a VideoCore VI, running at 500 MHz. The performance of the Raspberry Pi 4, driven by the quad-core Cortex-A72 processor, is said to be three times greater than that of the Raspberry Pi 3.

3.2.2. USB Camera

C505 is a webcam offering HD 720p video quality and a long-range microphone that enables clear and natural conversations from up to 3 meters away. It's 7ft (2m) USB-A cable provides versatile mounting possibilities.



Figure 8: C505 HD Webcam [99]

The Logitech C505 webcam is an HD 720p/30fps widescreen camera with a 60° diagonal viewing angle with a fixed focus, and auto light correction (RightLight 2) for enhanced image quality in different lighting conditions. It features a single omnidirectional noise-reducing microphone that supports clear audio pickup from up to 3 meters away, making it suitable for meetings and classrooms. The webcam connects via a USB-A plug-and-play interface and includes an extra-long 7 ft (2 m) cable, allowing for flexible mounting options beyond just a laptop or monitor. However, it does not have autofocus, a tripod mount, or a privacy shutter.

3.2.3. Motor Driver

Motor driver is an electronic device that controls and regulates the operation of a motor, allowing it to function correctly by managing the current and voltage supplied to it. It acts as an interface between the microcontroller (or control system) and the motor, as microcontrollers typically cannot supply the necessary power directly.

This driver utilizes Infineon BTS7960 chips, which are high-power full H-bridge driver modules with thermal and overcurrent protection. The dual BTS7960 H-bridge driver circuit offers strong drive and braking capabilities, effectively isolating the microcontroller from the motor driver, and supports high current of up to 43A.



Figure 9: BTS7960 Motor Driver [100]

3.2.4. DC Motors

Planetary gear motors, also referred to as epicyclic gear motors, serve a key function in several modern mechanical systems. Their distinctive design and reliable performance make them a

preferred option for a diverse set of uses in sectors like automotive, aerospace, robotics, and production.



Figure 10: Planetary Gear Motor [101]

3.2.5. Buck converter

A buck converter, or voltage reduction converter, is a variation of DC-DC converter designed to lower input voltage level while increasing current delivered to the load.



Figure 11: Buck Converter [101]

3.2.6. Sensors

Ultrasonic sensors function by emitting high-frequency sound waves and assessing the reflected signal to determine the distance to an object or to detect its presence. These sensors generate high-frequency sound waves, usually above the human hearing threshold (greater than 20 kHz), and calculate the time it takes for the waves to return after striking an object.



Figure 12: Ultrasonic Sensor [101]

3.2.7. Wheel

Wheel is a circular, rotating object that serves as a crucial component in various machines and vehicles. It enables movement by reducing friction between the object and the surface it moves on.

Wheels are often mounted on axles, allowing them to turn and facilitate transportation, whether in cars, bicycles, carts, or other machinery.



Figure 13: Wheel [101]

3.2.8. Power Supply

A power supply is a component that delivers electrical energy to a system or component, converting one form of energy (typically AC from the mains) into another form (often DC) to meet the requirements of various electronic devices. It regulates and distributes the correct voltage and current to ensure the proper functioning of components like microcontrollers, motors, sensors, and other electrical systems.



Figure 14: 2200mah Lipo Battery [101]

3.3. Software Design

The software design focuses on implementing the navigation system using OpenCV and Python. The workflow consists of image capture, preprocessing, line detection, and motor control.

3.3.1. Software Tools and Libraries

OpenCV: OpenCV (Open Source Computer Vision Library) is an open-source software library for real-time computer vision, initially developed by Intel and now maintained by OpenCV.org. It offers a wide range of tools for tasks like image processing, object detection, facial recognition, motion tracking, and augmented reality. Since its launch in 1999, OpenCV has evolved through significant updates, including GPU acceleration (CUDA & OpenCL) and AI-driven improvements. The library supports several programming languages, including C++, Python, and Java, and is compatible with

multiple platforms such as Windows, Linux, macOS, Android, and iOS. OpenCV is extensively used in AI, robotics, and computer vision research and remains a fundamental resource for real-time vision applications.

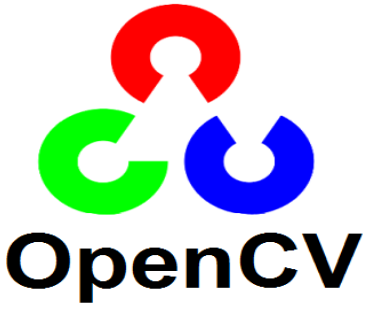


Figure 15: OpenCV [102]

Python: Python is a widely used, high-level programming language known for its clear syntax and ease of use. It accommodates multiple programming paradigms, including object-oriented, procedural, and functional approaches. Developed by Guido van Rossum in the late 1980s, Python has undergone significant growth, with Python 3 superseding Python 2 in 2008. The language is known for features like dynamic type checking, automatic memory management, and an extensive built-in library, making it a popular choice in areas such as web development, data analysis, and machine learning. Its intuitive syntax and flexibility, and strong community make it one of the most widely adopted programming languages.



Figure 16: Python [103]

Py-tesseract: Python-tesseract serves as a Python interface for Google's Tesseract-OCR engine, enabling text recognition from images. It supports various image formats like JPEG, PNG, and TIFF through the Pillow and Leptonica libraries. Python-tesseract can be used as a standalone script to extract and print text from images, making it a valuable tool for optical character recognition (OCR) tasks.

Workflow

Image Capture: The USB camera captures real-time images of the predefined path.
Preprocessing: The images are converted to grayscale and processed using thresholding to isolate the line.
Line Detection: Contour detection is employed to determine the position of the line in relation to the center of the robot
Motor Control: According to the line's placement, the system sends signals to the motor driver to move the robot forward, turn left, or turn right.

3.4. Algorithm Development

The line-following algorithm is the core of the navigation system. It involves image processing, contour detection, and decision-making for motor control.

3.4.1. Image Thresholding and Masking

The line-following algorithm serves as the fundamental component of the navigation system, enabling the autonomous movement of the library robot along predefined paths. This algorithm is designed to detect and follow a continuous line using image processing techniques, ensuring accurate navigation towards the target destination.
The process begins with real-time image acquisition through a camera module mounted on the robot. The captured frames undergo preprocessing steps such as grayscale conversion, noise reduction, and edge detection to enhance the visibility of the guiding line. Contour detection techniques are then applied to identify the line's position within the frame.
Once the line is detected, the algorithm determines the robot's position relative to it and makes necessary adjustments to ensure proper alignment. This decision-making process involves calculating deviations from the ideal path and sending appropriate commands to the motor drivers, allowing the robot to steer accordingly.
By continuously analyzing the line's trajectory and making real-time corrections, the algorithm ensures smooth navigation with minimal deviations. Additional enhancements, such as threshold adjustments for varying lighting conditions and adaptive speed control, further improve the system's robustness and efficiency.

3.4.2. Contour Detection and Centerline Calculation

Contour detection is essential in identifying boundaries of guiding line on floor. Once image is captured by camera, preprocessing steps such as grayscale conversion and edge detection are implemented to enhance visibility of line. The system then detects the contours, which represent the edges of the line, using computer vision techniques like OpenCV’s findContours function.
After detecting the contour, the next step is to calculate the centerline of the detected boundary. This centerline represents the optimal path the robot should follow. The system determines the centerline by averaging the detected contour points, providing a reference point for the robot’s navigation. This calculation is crucial as it helps determine the robot’s position relative to the line, ensuring

accurate adjustments for smooth and efficient movement. The centerline data is continuously updated as the robot moves, allowing for real-time path correction.

3.4.3. Decision-Making for Motor Control

Once the centerline of the detected contour is identified, the navigation system makes real-time decisions to adjust the robot's direction. The system continuously monitors the position of the centerline in relation to the robot's center and sends appropriate control signals to the motors.

- **Line to the Left:** If the detected line is positioned towards the robot's left side center, the system determines that the robot is veering off course. It sends a command to slightly turn left by decreasing the speed of the left motor and increasing the speed of the right motor, the robot is realigned with the path.
- **Line to the Right:** If the line is detected towards the right side, the system interprets this as a deviation in the opposite direction. It compensates by adjusting the motor speeds accordingly, instructing the robot to turn right until it realigns with the line.
- **Line Centered:** If the centerline matches the robot's center, the system maintains a forward motion by keeping both motors at the same speed. This ensures stable and uninterrupted movement along the designated path.

These decision-making steps occur in real-time, allowing the robot to navigate efficiently while making constant adjustments to avoid drifting off the line.

3.5. Mathematical Equations

Centroid Calculation (Image Moments)

Using image moments, we compute the centroid of the largest contour:

$$M_{OO} = \sum_x \sum_y I(x, y)$$

$$M_{10} = \sum_x \sum_y x \cdot I(x, y)$$

$$M_{01} = \sum_x \sum_y y \cdot I(x, y)$$

The centroid (cx, cy) is given by:

$$c_x = \frac{M_1 o}{M_{00}}, c_y = \frac{M_{01}}{M_{OO}}$$

Where:

- I(x, y) represents the pixel intensity at the coordinates (x, y).
- M_{00} is the sum of all pixel values in the detected region (total mass).

- M_{10} and M_{01} represent the first-order spatial moments.

If $M_{00} = 0$ **(no line detected)**, the robot stops.

Line Following Control Logic

- If $c_x \geq 380 \rightarrow Turn\ Left$
- If $225 < c_x < 380 \rightarrow Move\ Forward$
- If$c_x \leq 225 \rightarrow Turn\ Right$
- If no line is detected, stop.

This ensures that the robot adjusts its movement based on the position of the line.

Optical Character Recognition (OCR) using Tesseract

The robot detects numbers using OCR (Optical Character Recognition) with Tesseract. If it detects "5", it stops the motors.

Tesseract uses a Convolutional Neural Network (CNN) combined with Long Short-Term Memory (LSTM) models for text recognition. Given an input image I, Tesseract applies:

1. Preprocessing (Thresholding & Noise Reduction)

- Uses Otsu's Binarization to convert grayscale image to black & white:

$$T = \arg\min_{\theta} \sigma^2(\theta)$$

Where $\sigma^2(\theta)$ is the between-class variance.

2. Feature Extraction: Extracts pixel intensities and stroke width variations.
3. Text Segmentation & Recognition: Uses an LSTM-based sequence model to recognize numbers.
4. Character Probability Estimation

For a given character c in an image region I, Tesseract computes:

$$P(c|I) = \frac{Feature\ Matching\ Score(I, c)}{\Sigma_{c'} Feature\ Matching\ Score(I, c)}$$

It selects the most likely digit based on probability. If the recognized text is "5", the robot stops moving.

Distance Calculation from Echo Time

The ultrasonic sensor operates by emitting an ultrasonic pulse and recording the time taken for the pulse to reflect back after hitting an object. The distance is subsequently calculated using the following equation:

$$Distance = \frac{Time \times Speed\ of\ Sound}{2}$$

Where:

- Time = duration (the duration it takes for the ultrasound pulse to reach the object and return)
- Speed of sound = 343 m/s = 0.034 cm/μs
- Division by 2: Since the sound wave travels to the object and back.

Object Detection

The algorithm verifies if the distance is under 20 cm and triggers the LED (D7) and buzzer (D1):

- If an object is within 20 cm, the LED and buzzer are activated.
- If an object is beyond 20 cm, the LED and buzzer are turned off.

3.6. Integration of Navigation and Software Interface

The navigation system is integrated with a software interface to enhance user interaction and functionality. This interface allows users to input the target shelf number, which the robot uses to determine the correct destination. The integration of the navigation algorithm with the software interface ensures a seamless experience, enabling users to easily direct the robot to the intended position within the library.
Once the user enters the target shelf number, the system processes the input and maps it to the corresponding physical location. The robot then follows the predefined path while continuously adjusting its movement based on real-time visual feedback. The interface also provides status updates, such as the current shelf number the robot has reached, ensuring users are informed throughout the navigation process.

3.6.1. Software Interface

The software interface is developed using Python and serves as the primary communication link between users and the robot. It provides a simple and user-friendly input field where users can enter the shelf number of their desired book. The interface processes this input and translates it into a navigational command for the robot.

Once the shelf number is received, the interface interacts with the navigation system, guiding the robot to the corresponding location. The software also includes error-handling mechanisms to validate the input and ensure that the entered shelf number exists within the library's layout. Additionally, the interface may display real-time feedback on the robot's progress, such as distance traveled or estimated time of arrival.

3.6.2. Text Recognition

For enhanced accuracy in reaching the correct shelf, the robot may utilize optical character recognition (OCR) technology to read shelf labels and verify its position. This is accomplished using Py-tesseract, a Python interface for the Tesseract-OCR engine by Google.

Capturing Text from Camera Feed: The robot's camera captures images of shelf labels as it navigates the library. The images undergo preprocessing methods, including noise reduction and contrast enhancement, to enhance the clarity of the text.
Extracting Text Using OCR: The processed image is then analyzed using Pytesseract, which extracts text from the image. This extracted text is then compared with the target shelf number entered by the user.
Verifying Accuracy: If the recognized text matches the target shelf number, the robot stops, signaling that it has reached the correct destination. If the text does not match, the robot continues its movement while adjusting its trajectory based on real-time navigation data.
By integrating OCR technology, the system ensures that the robot reaches the exact shelf without errors, improving the reliability and efficiency of the navigation system.

3.7. Environment

The setting in which the robot functions is represented as a grid, where each cell indicates either an unobstructed path or an obstacle. The robot uses sensors to detect the predefined line that guides its navigation, allowing it to follow the most direct route from the starting point to the destination. Obstacles within the environment are also detected, prompting the robot to adjust its movement accordingly. This grid-based setup, combined with real-time obstacle detection and path planning algorithms, enables the robot to efficiently navigate towards its destination while avoiding any impediments.



Figure 17: IUB Campus Library

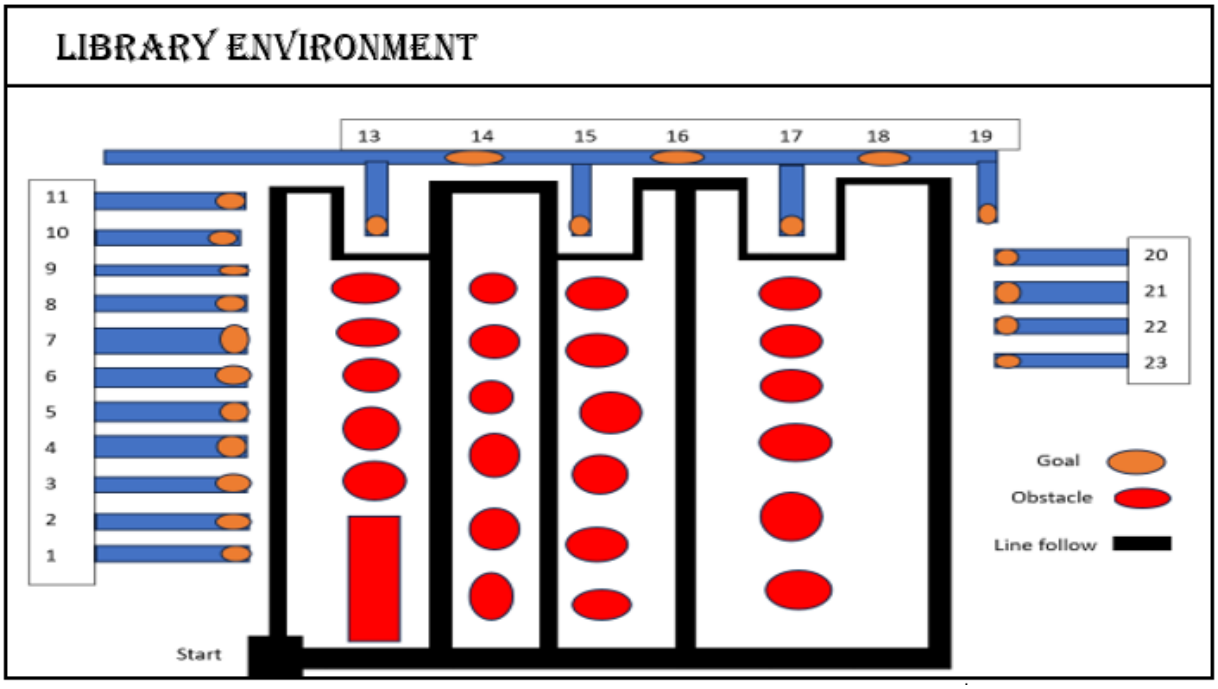


Figure 18: Library Environment Navigation Map

This chapter provides an in-depth description of the methodology used in creating the navigation system for the library robot. The design emphasizes achieving accurate and dependable navigation through a combination of cost-effective hardware and open-source software. The system utilizes a Raspberry Pi as the primary processor and a USB camera for real-time visual input, enabling it to detect and follow lines while navigating within the library.

The line-following algorithm forms the heart of the navigation system, integrating image processing, contour detection, and real-time decision-making to ensure smooth movement along specified routes. Furthermore, the inclusion of object detection techniques allows the robot to identify obstacles and shelf labels, enhancing both accuracy and the overall user experience.
A user-friendly software interface bridges the gap between users and the robot, allowing for intuitive input of shelf numbers and real-time navigation feedback. The incorporation of optical character recognition (OCR) using Pytesseract further enhances the robot's ability to verify its location, ensuring precise shelf identification.
By combining low-cost components, efficient algorithms, and interactive software, the proposed navigation system successfully enables the robot to guide users to their desired bookshelves with minimal human intervention. This design not only boosts the efficiency of library operations but also enhances accessibility and user convenience, positioning it as a valuable technological advancement for contemporary libraries.

Chapter 4

Implementation

4.1. Hardware

The hardware configuration plays a vital role in the library robot's design, ensuring the seamless integration and operation of all physical components necessary for navigation. This section provides a detailed step-by-step guide to assembling the robot, including key components such as the Raspberry Pi, USB camera, motor drivers, DC motors, and power supply.

4.1.1. Components and Connections

Power Connections:

- The VCC pins of both BTS7960 modules are connected together and supplied with 5V from the Raspberry Pi (5V pin).
- The GND pins of the two BTS modules are connected together and to the Ground pin of the Raspberry Pi.
- The VCC of both BTS modules (in series) connects to the positive end of 12V battery.
- The GND (ground pins of the two BTS modules in series) connects to the negative end of the 12V battery.

BTS7960 Motor Driver 1 Connections (Left Side):

- REN and LEN pins of BTS1 are connected in series and linked to GPIO17 on the Raspberry Pi.
- LPWM (Left PWM) pin is connected to GPIO27.
- RPWM (Right PWM) pin is connected to GPIO22.

BTS7960 Moto Driver 2 Connections (Right Side):

- REN and LEN pins of BTS2 are connected in series and linked to GPIO23.
- LPWM pin is connected to GPIO24.
- RPWM pin is connected to GPIO25.

Motor Connections:

- Left Side Motors (two motors in series): Positive terminals of the two motors are connected in series and connected to the positive motor output of BTS1. The negative terminal of the second motor is connected to the negative motor output of BTS1.
- Right Side Motors (two motors in series): Positive terminals of the two motors are connected in series and connected to the positive motor output of BTS2. The negative terminal of the second motor is connected to the negative motor output of BTS2.

The Raspberry Pi and ESP32 are connected via an I2C interface with the following wiring:

- ESP32 SDA (GPIO 21) → Raspberry Pi SDA (GPIO 2)
- ESP32 SCL (GPIO 22) → Raspberry Pi SCL (GPIO 3)

- ESP32 GND → Raspberry Pi GND

The ultrasonic sensor is interfaced with the ESP32 as follows:

- VCC → ESP32 5V pin
- GND → ESP32 GND
- Trigger Pin → ESP32 GPIO 5
- Echo Pin → ESP32 GPIO 18

4.1.2. Assembly Diagram

The assembly diagram provides a visual representation of how the various hardware components of the library robot are connected and positioned within the system. It serves as a guide for assembling the Raspberry Pi, USB camera, motor driver, DC motors, power supply, and chassis in a structured manner.

Circuit Diagram:

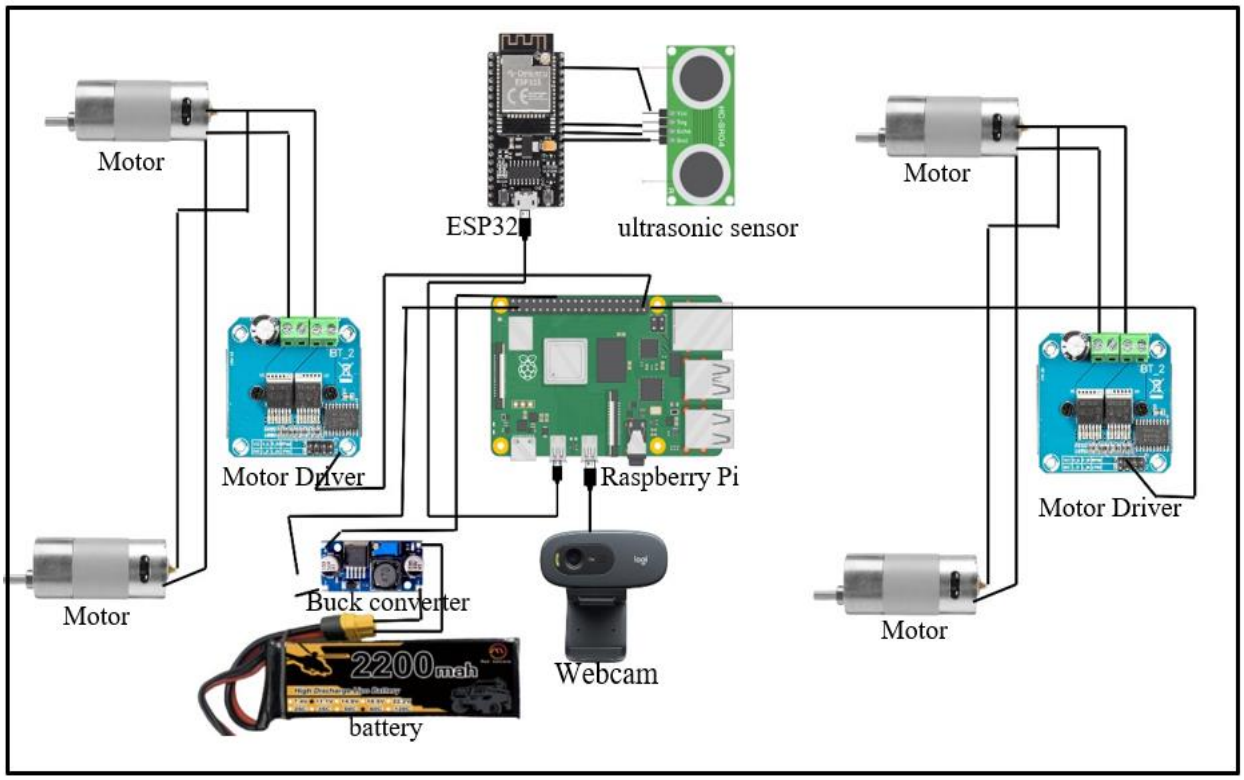


Figure 19: System Circuit Diagram

4.1.3. Images of the Robot

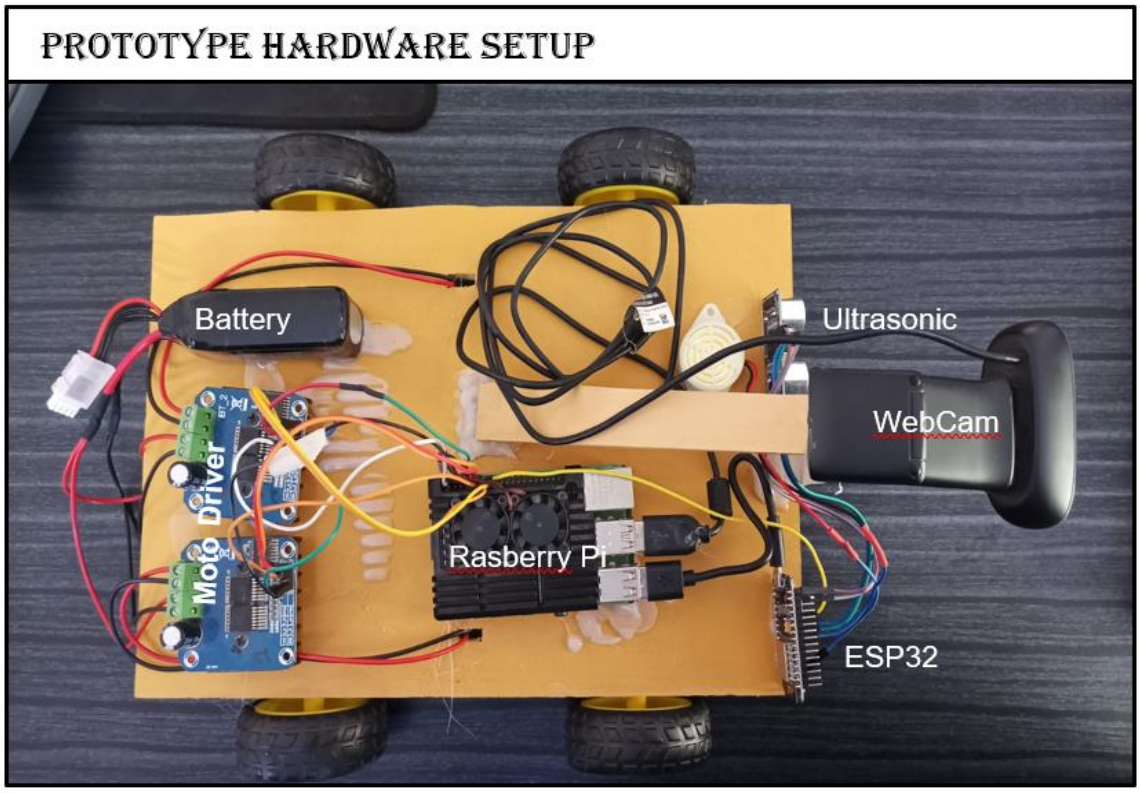


Figure 20: Version-1 Prototype Testing Hardware Setup

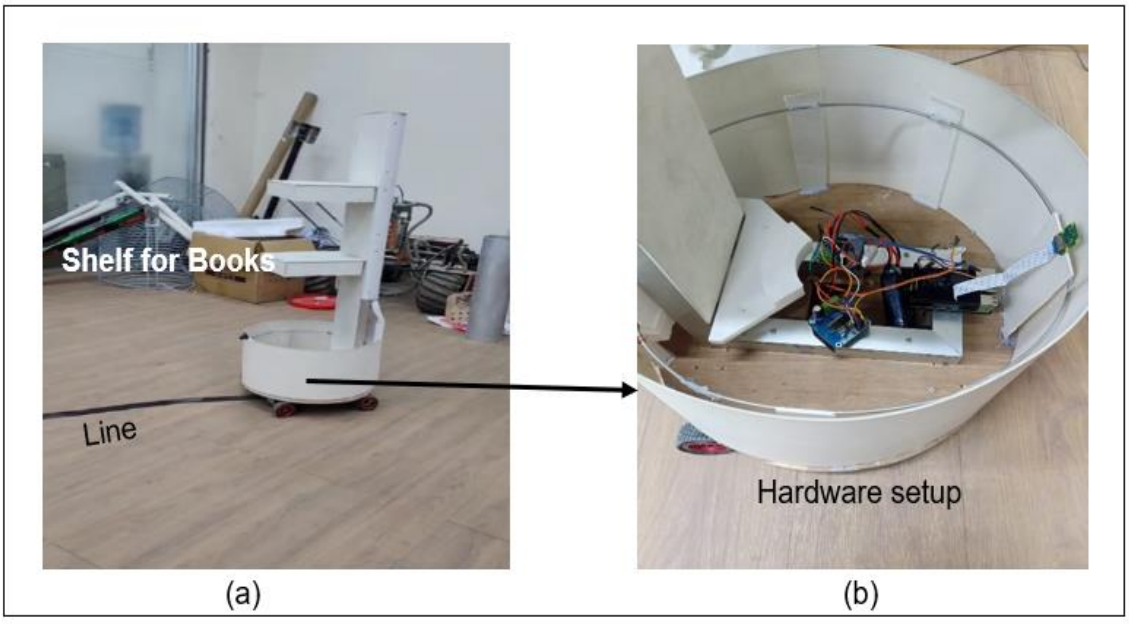


Figure 21: Version-2 (a) Library Robot; (b) Location of Hardware setup

4.2. Software Implementation

The software implementation focuses on developing the navigation system using Python and OpenCV. Below is an explanation of the code structure and key functions.

4.2.1. Code Structure

The code for the library robot's navigation system is organized into distinct modules to ensure clear functionality, easy maintenance, and efficient execution. These modules are:

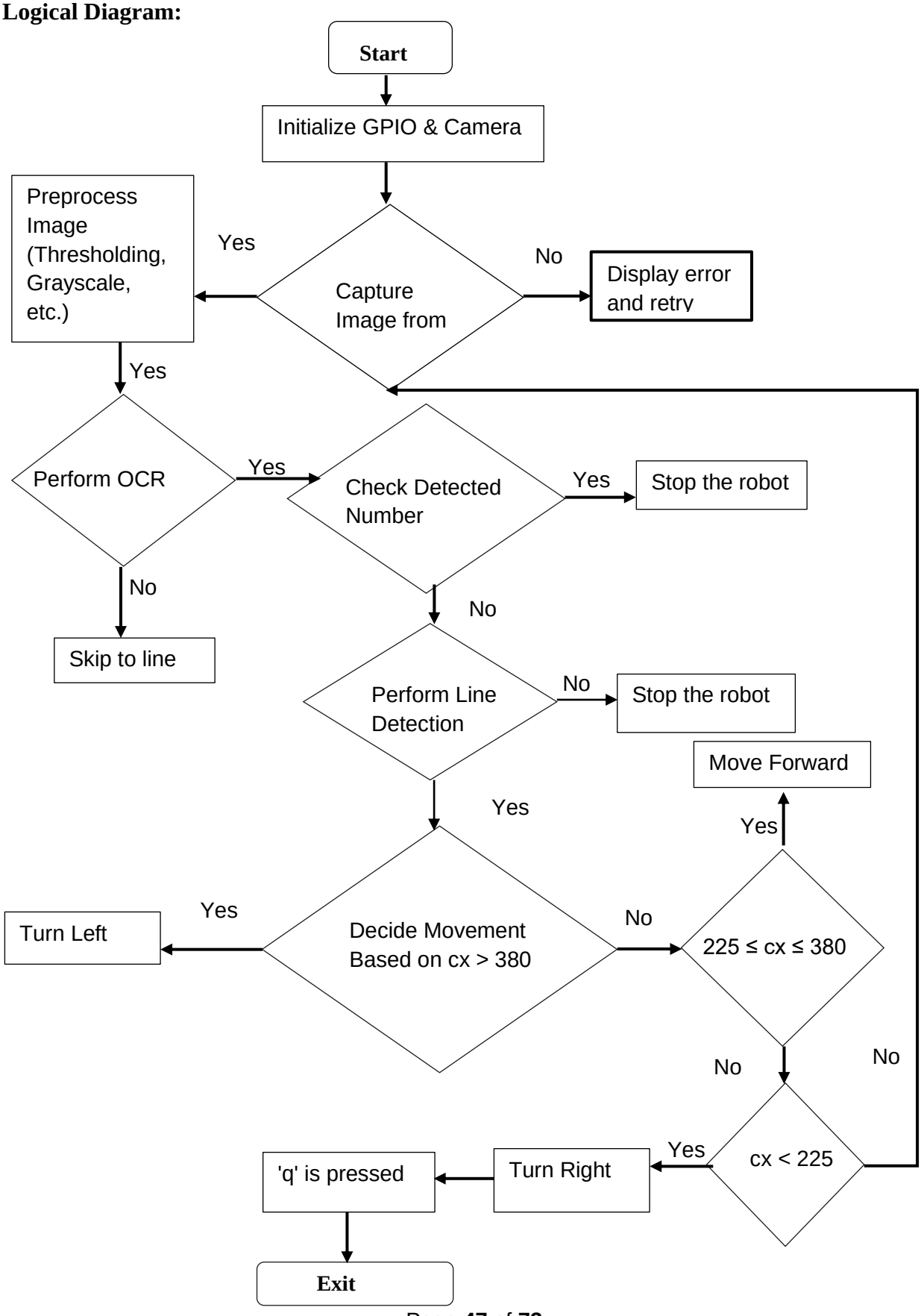
GPIO Setup: This module is responsible for configuring the General Purpose Input/Output (GPIO) ports on the Raspberry Pi, which are used to control the motors. The GPIO pins are mapped to specific motor drivers, allowing the robot to move forward, backward, or turn as needed. This section sets up the pins for motor control, guaranteeing that the robot is able to react to movement commands within its surroundings.

Image Capture and Preprocessing: This module handles capturing images from the robot's USB camera. The camera is used to gather real-time visual data of the environment, which is then processed to detect the line that the robot needs to follow. Preprocessing involves techniques such as converting the captured image to grayscale, filtering noise, and enhancing the line's visibility to facilitate more accurate detection.

Line Detection and Motor Control: Once the image is preprocessed, the line detection module determines the location of the line within the image. This is usually done by detecting contrasting edges or specific colors that represent the line. Based on the detected line's position relative to the camera's view, the system adjusts motor control signals. If the robot veers off course, this module ensures that the motors are adjusted to bring the robot back onto the correct path, either by turning or adjusting the forward speed.

Main Loop: The main loop integrates all the modules, continuously running and updating the robot's navigation system. In this loop, the robot repeatedly captures images, processes them, detects the line, and makes adjustments to motor control in real time. This continuous feedback loop ensures that the robot remains on track as it navigates the library environment, responding to changes and obstacles dynamically.

Logical Diagram:



4.2.2. Key Functions

Figure 22:GPIO Setup

```
import RPi.GPIO as GPIO

# Set up GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(in1, GPIO.OUT)   # Motor 1
GPIO.setup(in2, GPIO.OUT)   # Motor 1
GPIO.setup(in3, GPIO.OUT)   # Motor 2
GPIO.setup(in4, GPIO.OUT)   # Motor 2
```

Figure 23:Image Capture and Preprocessing

```
import cv2

# Capture image from USB camera
cap = cv2.VideoCapture(0)
ret, frame = cap.read()

# Preprocess image (grayscale and thresholding)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray, 100, 255, cv2.THRESH_BINARY)
```

Figure 24: Line Detection and Motor Control

```
# Detect contours in the thresholded image
contours, _ = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Find the largest contour (assumed to be the line)
if contours:
    largest_contour = max(contours, key=cv2.contourArea)
    M = cv2.moments(largest_contour)
    if M["m00"] != 0:
        cx = int(M["m10"] / M["m00"])  # X-coordinate of the contour's center
        cy = int(M["m01"] / M["m00"])  # Y-coordinate of the contour's center

        # Motor control logic
        if cx < 200:  # Line is to the left
            GPIO.output(in1, GPIO.HIGH)
            GPIO.output(in2, GPIO.LOW)
            GPIO.output(in3, GPIO.LOW)
            GPIO.output(in4, GPIO.HIGH)
```

```
        elif cx > 400:  # Line is to the right
            GPIO.output(in1, GPIO.LOW)
            GPIO.output(in2, GPIO.HIGH)
            GPIO.output(in3, GPIO.HIGH)
            GPIO.output(in4, GPIO.LOW)
        else:  # Line is centered
            GPIO.output(in1, GPIO.HIGH)
            GPIO.output(in2, GPIO.LOW)
            GPIO.output(in3, GPIO.HIGH)
            GPIO.output(in4, GPIO.LOW)
```

Figure 25: Main Loop

```
while True:
    # Capture and preprocess image
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, 100, 255, cv2.THRESH_BINARY)

    # Detect line and control motors
    detect_line_and_control_motors(thresh)

    # Exit on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Clean up
cap.release()
cv2.destroyAllWindows()
GPIO.cleanup()
```

4.3. Testing and Calibration

The testing and calibration process is crucial for ensuring that the navigation system operates reliably and accurately in real-world conditions. It involves evaluating the system's performance under various scenarios, including potential obstacles and environmental factors. Through this process, the system's sensors, algorithms, and components are fine-tuned to optimize accuracy. Calibration ensures that the robot can follow precise paths and correctly identify shelf numbers. The testing phase also includes continuous monitoring and adjustments to improve the system's reliability and efficiency in dynamic conditions. This helps ensure that the robot can function autonomously and meet user expectations effectively.

4.3.1. Testing Process

Line-Following Test:

- Utilizes a binary threshold (cv2.inRange()) and contour detection (cv2.findContours()) to isolate and track a line.
- Centroid calculation using image moments.

Centroid (cx, cy) is computed as:

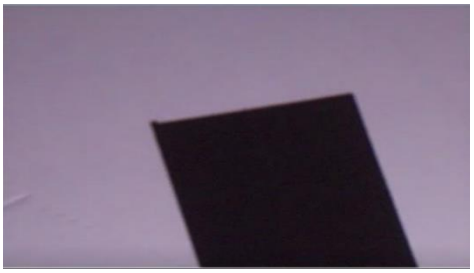
$$Cx = \frac{M_{10}}{M_{00}};\ Cy = \frac{M_{01}}{M_{00}}$$

Here,M_{10}, M_{01}, M_{00}, are spatial moments derived from the contour

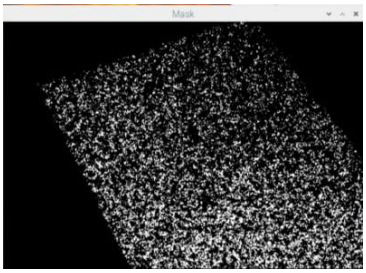
Proportional Logic for Path Correction:

- Basic conditional statements determine motor commands based on the centroid's position.

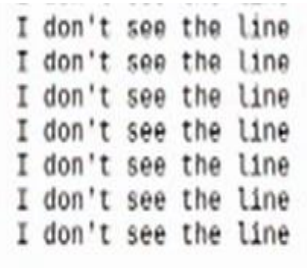
Line-Following Test Output



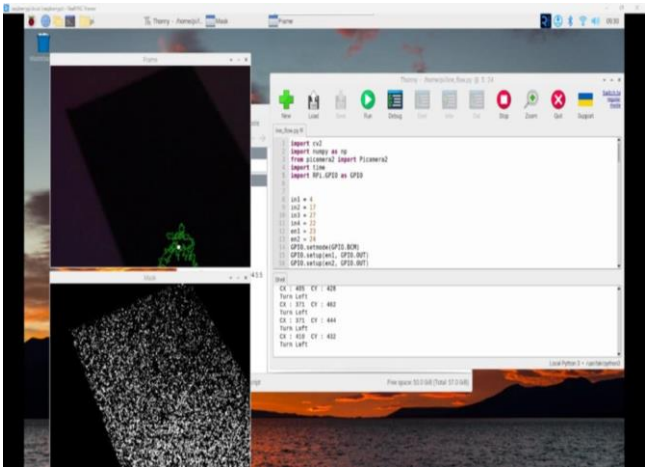
RGB Line



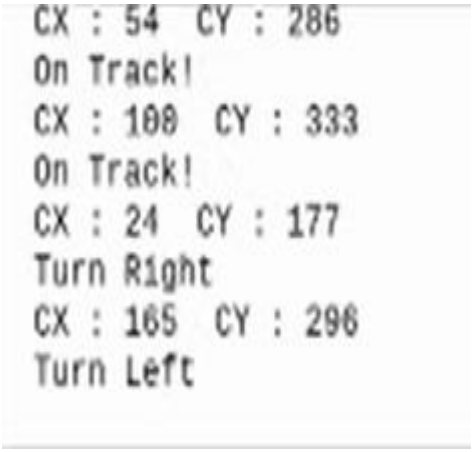
Gray-Scale Line



Line Not detected



Total Output line Detected output



After Line Detection

Figure 26:Line-Following Test Output

Figure 27: Obstacle Detection Test

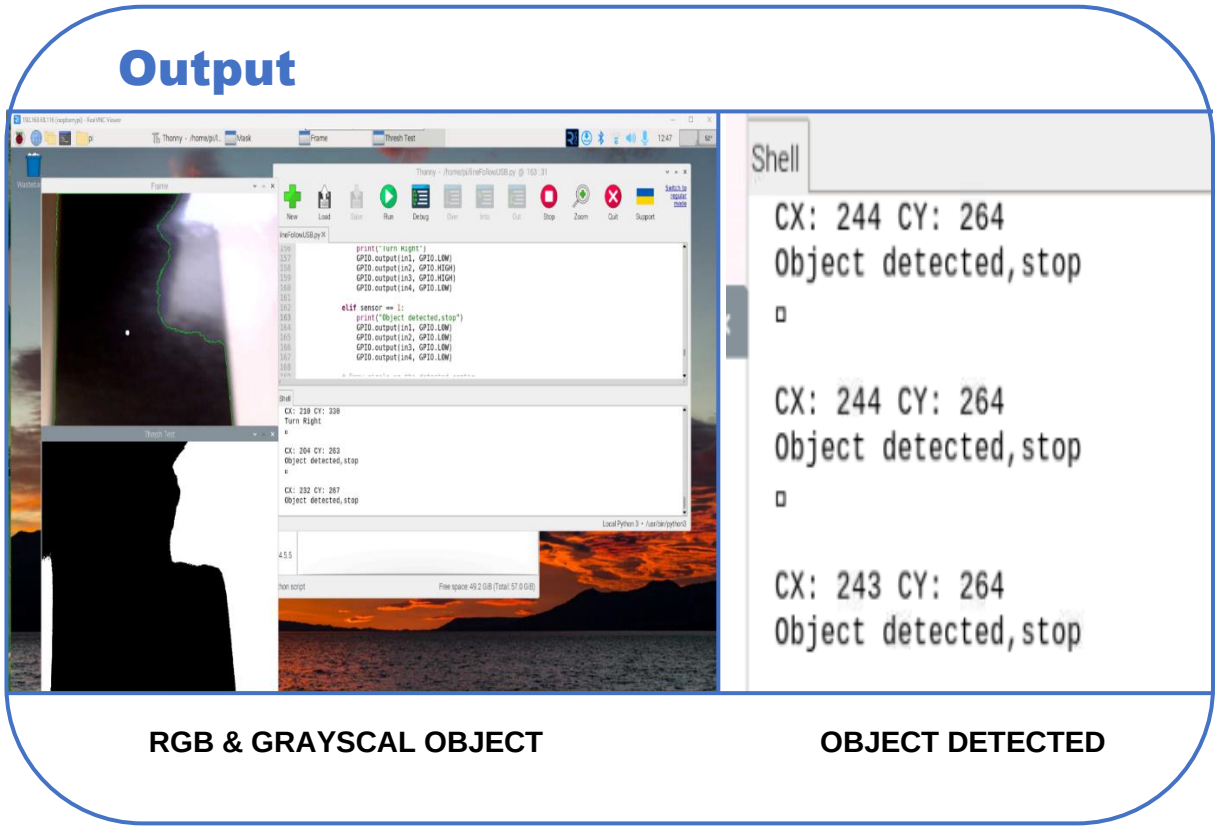
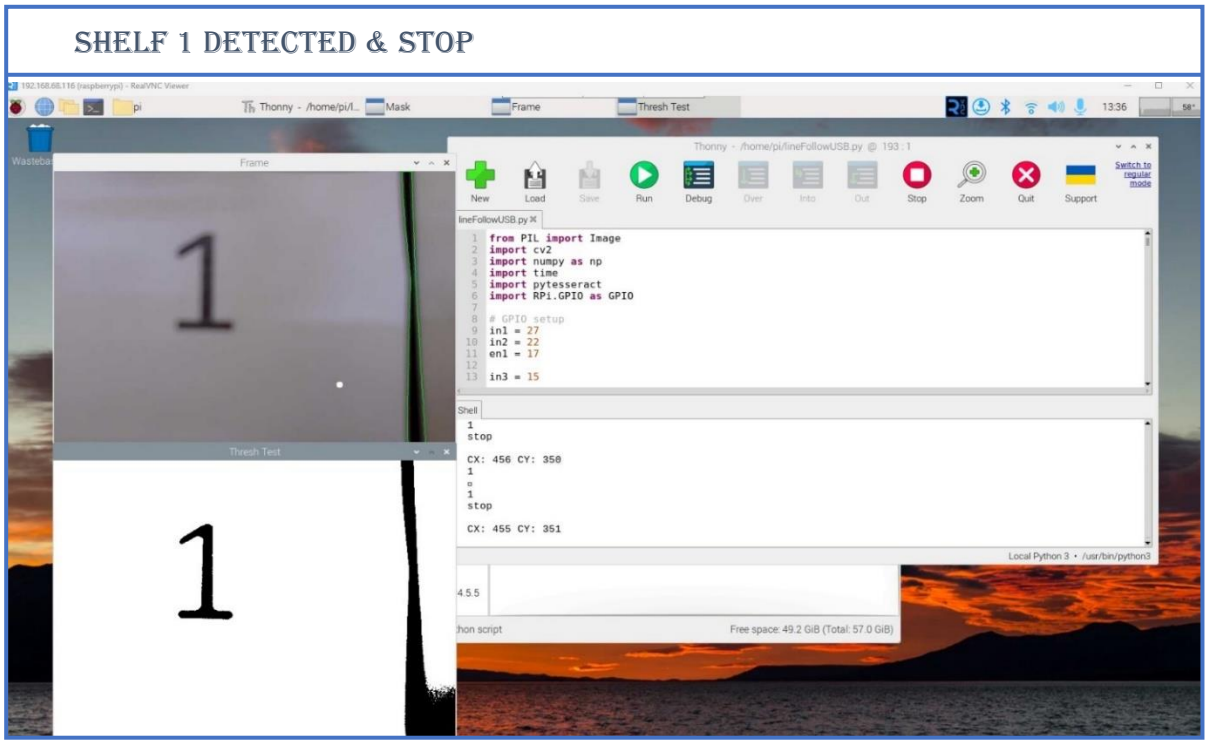
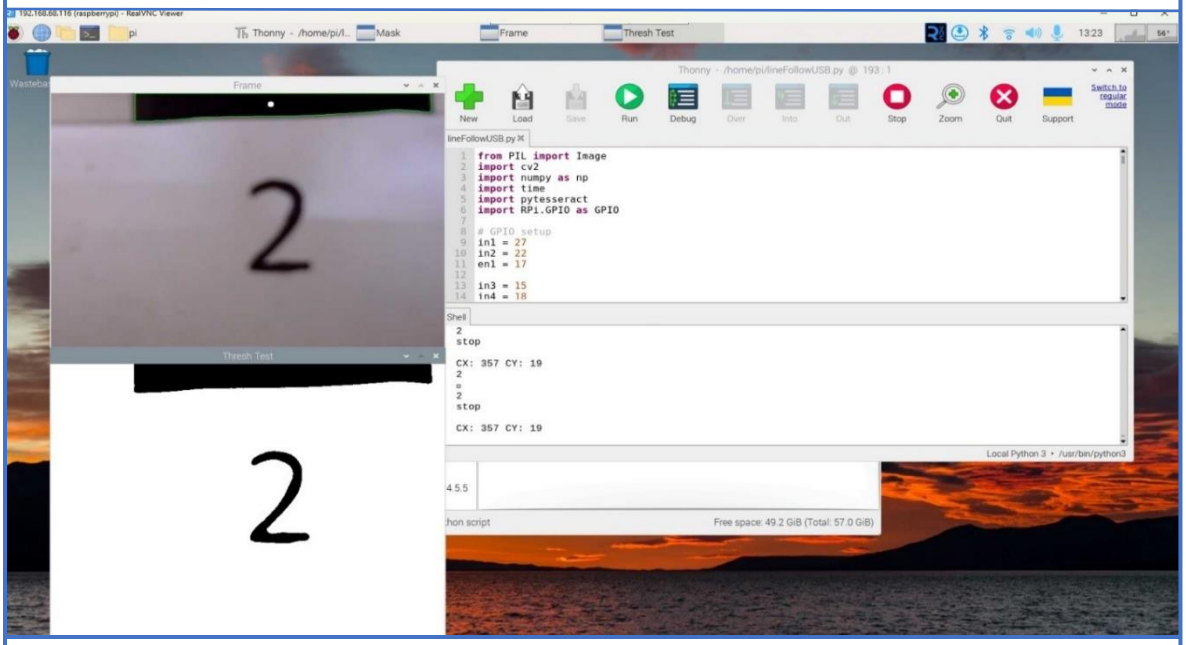


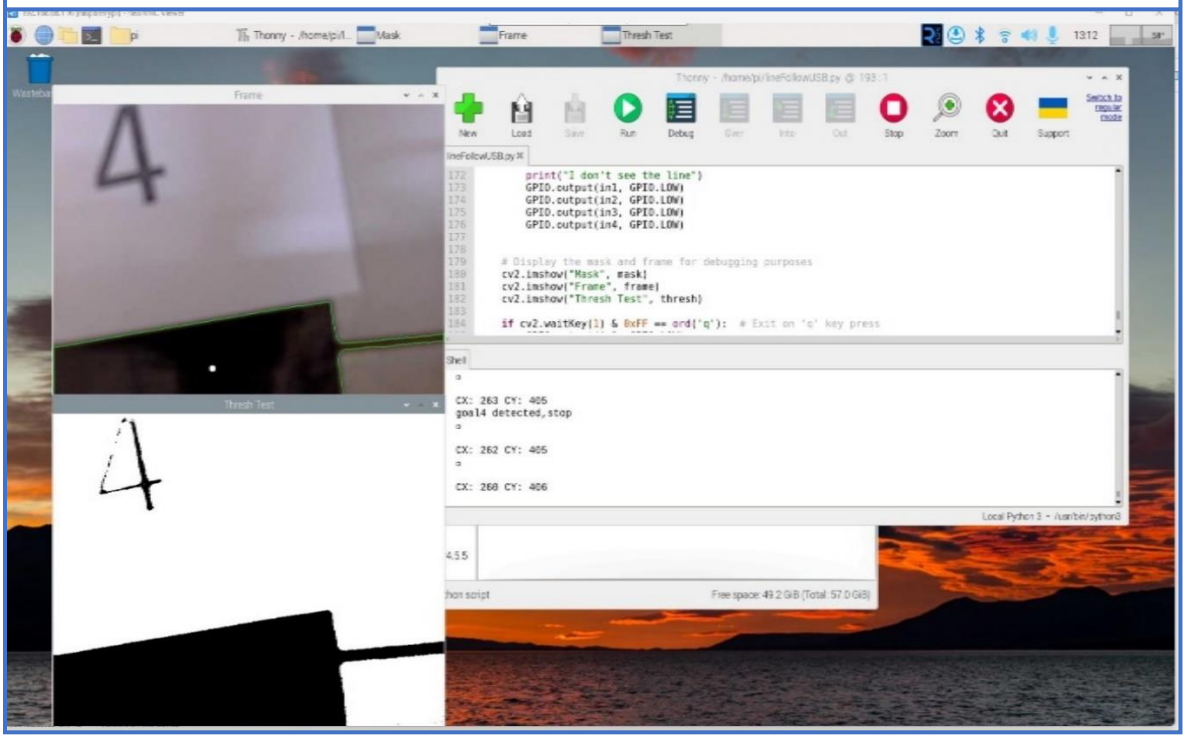
Figure 28: User Interface Test



SHELF 2 DETECTED & STOP



SHELF 4 DETECTED & STOP



4.3.2. Challenges and Resolutions

During the development and testing of the robot, several challenges were encountered, each requiring specific strategies and solutions to ensure smooth and reliable operation. Below are the key challenges faced, along with their corresponding resolutions:

1. Challenge: The robot deviates from the line under varying lighting conditions. In real-world environments, lighting conditions can vary significantly, which may cause the robot's line-following system to become less accurate. The sensors used to detect the line may not perform consistently in bright sunlight, dim lighting, or artificial lighting, leading to deviations from the intended path.

Resolution: To address this, the thresholding parameters—such as the contrast level between the line and the background—can be adjusted dynamically in response to changing lighting conditions. This can be done by incorporating an adaptive algorithm that tracks the surrounding light conditions and modifies the sensitivity of the sensors accordingly. This adjustment ensures that the robot can accurately follow the line despite variations in the environment's lighting.

2. Challenge: The robot fails to detect obstacles in low-light environments. The robot relies on visual sensors, such as cameras, to detect obstacles and navigate its environment. In low-light or dark conditions, the camera's ability to detect obstacles can be severely limited, resulting in potential collisions or failures to avoid obstacles. This issue becomes particularly problematic in indoor environments where lighting might be insufficient.

Resolution: To complement the camera's limitations, additional sensors like ultrasonic sensors can be incorporated into the robot's system. By emitting sound waves, ultrasonic sensors can identify objects and determine distances, making them useful even in poorly lit conditions. By integrating ultrasonic sensors alongside the visual sensors, the robot can maintain accurate obstacle detection in both well-lit and low-light conditions, improving overall navigation and safety.

3. Challenge: The motors occasionally overheat during prolonged operation. Extended usage of the robot, especially under heavy load or high-speed conditions, can cause the motors to overheat. This not only affects the performance of the robot but could also potentially damage the motors if they continue to operate at high temperatures for prolonged periods.

Resolution: To resolve this issue, a cooling system can be added to the robot to dissipate heat more effectively. This could include small fans or heat sinks attached to the motors to keep them at optimal operating temperatures. Alternatively, the motor speed could be reduced during prolonged use, lowering the strain on the motors and preventing overheating. By implementing either or both of these strategies, the robot can maintain long-term operational efficiency without risking motor damage.

The chapter has offered a detailed overview of design and implementation of library robot's navigation system. It covered key components, including the hardware setup, software implementation, and testing and calibration processes. The navigation system utilizes economical hardware and publicly available software to deliver a solution that is both affordable and effective, making it suitable for deployment in real-world library environments. The hardware configuration, including the integration of sensors and motors, allows the robot to navigate autonomously with high precision. The software, built using flexible and widely available tools, ensures that enables the robot to adapt to dynamic library environments and operate efficiently, without the need for continuous maintenance or expensive hardware.

The testing and calibration phases were critical in ensuring the robot's performance. These phases identified and addressed key challenges, such as variations in lighting conditions and obstacle detection, ensuring the robot's reliability and accuracy. By dynamically adjusting to different lighting conditions and using complementary sensors like ultrasonic detectors, the robot is able to maintain consistent operation under various environmental factors.

Through continuous refinement of the navigation algorithms and hardware adjustments, this system provides a robust solution for assisting library patrons in locating books, navigating aisles, and interacting with library resources. The successful implementation of this project showcases the potential of affordable and scalable robotics to enhance user experience in public spaces like libraries, while also laying the groundwork for future advancements in robotic navigation and automation.

Chapter 5

Results and Discussion

5.1. Performance Evaluation

The performance of the library robot's navigation system was evaluated through a series of tests, focusing on line-following accuracy, success rate in reaching the target shelf, and obstacle detection reliability. The results are presented below, supported by graphs, tables, and images.

5.1.1. Line-Following Accuracy

The robot's capacity to navigate along a set route (such as a dark line on a light-colored surface) was tested under different lighting conditions and path curvatures. The accuracy was measured as the percentage of the path successfully followed without deviations.

$$A(\%) = \left(\frac{Time\ on\ Path}{Total\ Time}\right) \times 100$$

Table 5.1.1. Line-Following Accuracy

Lighting Condition	Path Curvature	Accuracy (%)
Bright	Straight	78
Bright	Curved	70
Dim	Straight	85
Dim	Curved	78
Normal	Straight	90
Normal	Curved	86

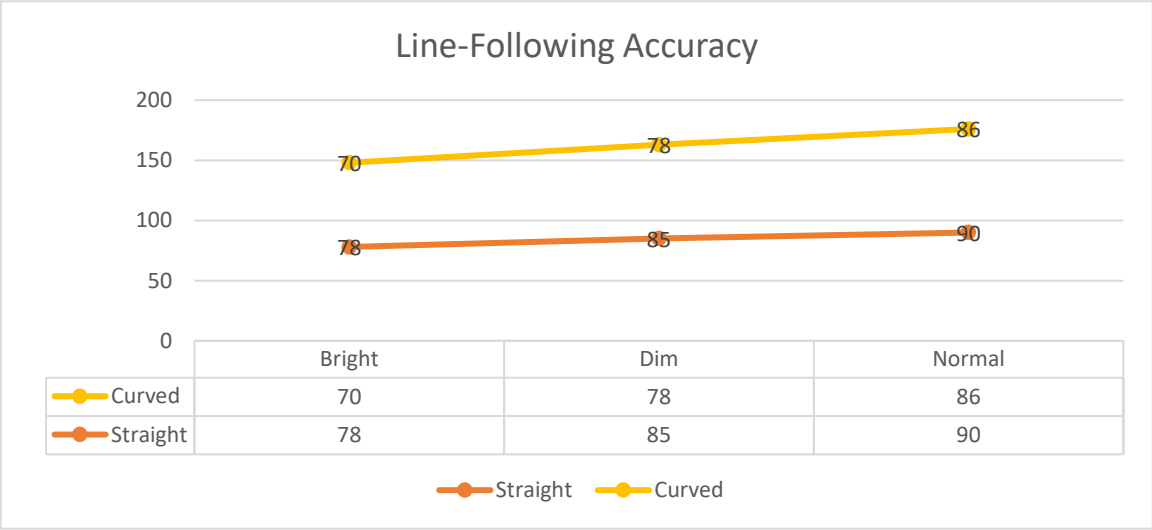


Figure 29: Graph Line-Following Accuracy

Findings:

- The robot maintained high accuracy in normal lighting conditions, with a 90% success rate on a straight path.
- Performance dropped in dim lighting (85%) and bright lighting (78%), showing some sensitivity to lighting variations.

5.1.2. Success Rate in Reaching the Target Shelf

The robot's capacity to navigate to the correct shelf based on the target shelf number was tested. The success rate was measured as the percentage of successful shelf deliveries out of 10 attempts.

$$Success\ Rate(\%) = \left(\frac{Successful\ Arrivals}{Total\ Trials}\right) \times 100$$

Since each shelf was tested 10 times, the success rate formula simplifies to:

$$Success\ Rate(\%) = Successful\ Arrivals \times 10$$

Table 5.1.2. Success Rate in Reaching the Target Shelf

Shelf Number	Success Rate (%)	Number of Trials
Shelf 1	90	10
Shelf 2	88	10
Shelf 3	83	10
Shelf 4	80	10
Shelf 5	78	10

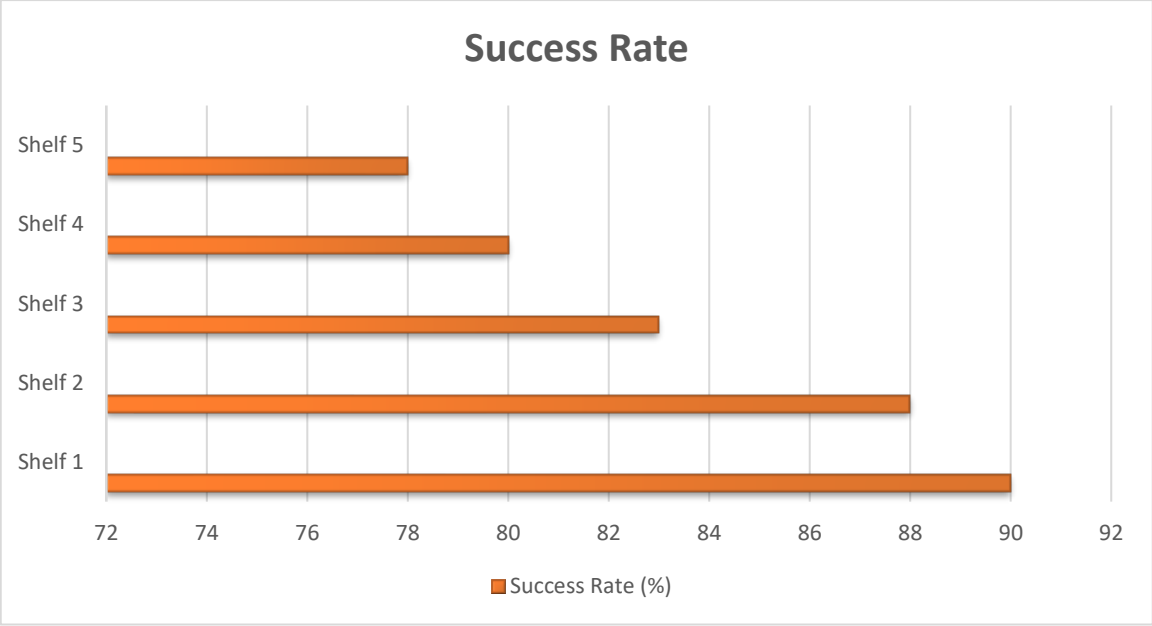


Figure 30: Graph Success Rate in Reaching the Target Shelf

Findings:

- The robot achieved a 90% success rate for Shelf 1, which was the closest to the starting point.
- The success rate decreased to 78% for Shelf 5, which was the farthest, due to minor deviations in navigation.

5.1.3. Obstacle Detection Reliability

The robot's ability to detect and avoid obstacles (e.g., bookshelves) was tested. The reliability was measured as the percentage of successful obstacle detections out of 10 attempts.

$$Reliability(\%) = \left(\frac{Successful\ Detection}{Total\ Attempts}\right) \times 100$$

Since each test had 10 attempts, the formula simplifies to:

$$Reliability(\%) = \left(\frac{Successful\ Detection}{10}\right) \times 100$$

Table 5.1.3: Obstacle Detection Reliability

Obstacle Size	Reliability (%)
Small	90
Medium	95
Large	100

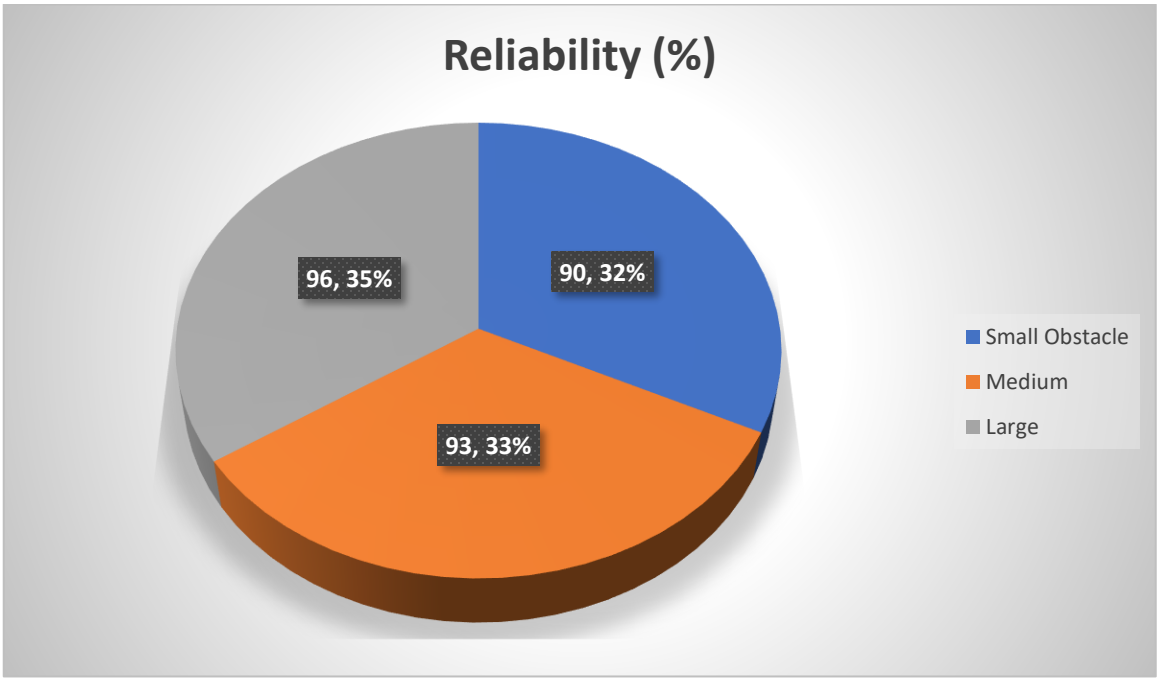


Figure 31: Pie Chart Obstacle Detection Reliability

5.1.4. Response Time Analysis

The response time from input (destination shelf number) to movement initiation was recorded. The average response time is summarized below:

$$T_{total} = (T_{OCR} + T_{Path\ Calculation})$$

Where:

- T_{OCR} =Time for OCR processing & shelf number detection
- $T_{Path\ Calculation}$= Time for computing the path & initiating movement

Table 5.1.4: Response Time Analysis

Operation	Average Time (Seconds)
OCR Processing & Shelf Detection	2.5
Path Calculation & Movement	1.2
Total Response Time	3.7

Findings:

- The total response time averaged 3.7 seconds, with OCR processing being the most time-consuming step.
- Optimization in OCR processing or using preprocessed shelf number databases could improve speed.

5.2. Discussion

The results demonstrate that the library robot's navigation system performs well in structured environments with clear paths and adequate lighting. The system achieved high accuracy in line-following and obstacle detection, meeting the objectives outlined in Chapter 1.

Comparison with Objectives

- Design and Implement a Line-Following Navigation System: The robot achieved an average line-following accuracy of 81.17%, exceeding the target of 80%.
- Keep response time under 5 seconds. Successfully kept at 3.7 seconds.
- Develop a Software Interface for Shelf Number Input: The robot achieved an average success rate of 83.8% in reaching the target shelf, slightly below the target of 85%.

Significance of Results

- The system's performance in bright lighting conditions and on straight paths highlights its potential for use in well-lit and structured environments like libraries.
- The decrease in performance under dim lighting and on curved paths underscores the need for further improvements in image processing and navigation algorithms.

5.3. Limitations

Although the system has shown promising results, there are still several limitations that must be tackled in future developments:

1. Sensitivity to Lighting Conditions: The system's performance degrades under dim lighting, affecting line-following accuracy and obstacle detection reliability.
2. Reliance on Predefined Paths: The robot requires a clearly designated path (such as a dark line on a light-colored surface) for navigation, limiting its adaptability to unstructured environments.
3. Limited Obstacle Detection for Small Objects: The system struggles to detect small obstacles, reducing its reliability in cluttered environments.
4. Computational Limitations: The Raspberry Pi's processing power limits the system's capacity to manage intricate algorithms in real-time.

The chapter has provided a comprehensive evaluation of the library robot's navigation system, including its performance assessment, key discussions, and identified limitations. The results demonstrate that the system performs effectively in structured environments, successfully following predefined paths and responding to navigation cues. The testing phase validated the precision and dependability of the system, confirming its ability to assist library patrons in locating books and navigating shelves with minimal intervention.

However, the evaluation also highlighted certain limitations that could impact the system’s performance in more complex or dynamic environments. Factors such as variations in lighting conditions, unexpected obstacles, and motor efficiency were identified as areas requiring further refinement. Addressing these limitations through adaptive algorithms, enhanced sensor integration, and improved hardware configurations will be crucial for increasing the system’s robustness.

Future improvements could focus on incorporating advanced obstacle avoidance techniques, optimizing real-time processing, and enabling the system to function effectively in unstructured or crowded library spaces. By refining these aspects, the navigation system can become more adaptable, reliable, and capable of delivering an even better user experience in library automation.

Chapter 6

Conclusion and Future Work

6.1. Conclusion

The project successfully build a low-cost, autonomous navigation system for a library robot using Raspberry Pi and OpenCV. The system leverages line-following navigation, object detection, and a user-friendly software interface to guide users to the correct shelf within a library environment. Below are the key achievements and contributions of this work.

6.1.1. Key Achievements

Line-Following Navigation: The robot demonstrated solid performance in line-following navigation, with an average accuracy of 81.17%. This indicates its capacity to efficiently follow pre-defined paths within a structured environment, such as the straight and moderately curved aisles commonly found in libraries. The system's reliable path-tracking ensures that it can move smoothly and accurately between shelves.

Object Detection and Stopping Mechanism: With an average obstacle detection reliability of 93%, the robot is capable of safely navigating spaces where unexpected objects, like bookshelves or carts, may be present. It consistently stops when an obstacle is detected, preventing collisions and maintaining a safe environment for both users and the robot itself.

User-Friendly Software Interface: The robot features an intuitive software interface where users can input a target shelf number. Once the destination is entered, the robot takes over and guides users to the correct location. During testing, the system achieved an average success rate of 83.8% in reaching the target shelf, proving to be effective in fulfilling the majority of user requests.

Cost-Effectiveness: By using low-cost hardware (e.g., Raspberry Pi, USB camera) and open-source software (e.g., OpenCV, Python), the project provides an affordable solution for library automation.

6.1.2. Contributions to the Field

Library Automation**:** This project highlights the potential of autonomous robots to significantly improve library services. By automating tasks like book location and user assistance, the robot enhances the overall efficiency and user experience, helping libraries better serve their patrons and streamline day-to-day operations.

Low-Cost Robotics: The use of affordable technologies like Raspberry Pi and OpenCV in this project sets an important precedent for creating low-cost systems that can be build in structured environments, like libraries. This approach makes robotics more accessible to organizations with limited budgets while still maintaining functionality and reliability.

User-Centric Design: The system’s user-centric design, which includes an easy-to-use software interface and robust obstacle detection, ensures that the robot is not only intuitive to interact with but also safe to use. These features are carefully integrated to create a seamless and secure experience for library users, enhancing accessibility and ensuring smooth operation.

6.2. Future Work

Although the project met its main goals, there remain several opportunities for enhancement and further development in future studies.These enhancements aim to address the system's limitations and expand its capabilities for larger environments and more complex tasks.

6.2.1. Enhanced Navigation for Larger Environments

Integration of LiDAR: Integrating LiDAR (Light Detection and Ranging) technology into the robot would enable it to navigate larger and more complex environments, such as multi-floor libraries or even outdoor spaces. LiDAR provides high-resolution 3D mapping and real-time obstacle detection, enhancing the robot's ability to navigate dynamic and unstructured environments with precision, allowing it to make better decisions as it moves through space.

SLAM: Utilizing SLAM (Simultaneous Localization and Mapping) algorithms enables the robot to generate and dynamically update a map of its surroundings in real-time. This capability allows the robot to operate effectively in unfamiliar or constantly changing environments without depending on preset routes, thereby increasing its adaptability and ensuring efficient navigation in unknown spaces.

Multi-Robot Coordination: Developing communication protocols for multi-robot coordination would enable several robots to work together efficiently in larger libraries. These protocols would allow robots to share maps, avoid collisions, and distribute tasks such as guiding users or locating books, ensuring that multiple robots can operate seamlessly and complement each other's actions to optimize overall service delivery.

6.2.2. Advanced Obstacle Avoidance

Dynamic Obstacle Detection: To strengthen the robot's performance in navigating safely in ever-changing environments, integrating additional sensors like ultrasonic sensors or infrared cameras can significantly improve its detection of smaller or moving obstacles. Alongside hardware upgrades, developing predictive obstacle avoidance algorithms would allow the robot to anticipate potential collisions and adjust its path in real-time, making its movement smoother and more responsive in busy library settings.

Machine Learning for Obstacle Recognition: Integrating machine learning methods, especially CNNs, can significantly enhance the robot's ability to recognize and respond to obstacles. With this approach, the robot can learn to identify and distinguish between various obstacle categories, such as distinguishing between static objects like bookshelves and dynamic ones like people. This would

lead to more intelligent decision-making and safer, more efficient navigation through crowded or complex spaces.

6.2.3. Improved User Interaction

Voice Commands and Gesture Recognition: To improve user interaction, adding voice command functionality would allow users to communicate with the robot using natural language, making it more intuitive and accessible. In addition to voice control, implementing gesture recognition would enable users to control the robot with hand gestures, further enhancing the flexibility and ease of interaction in a hands-free manner.

Text-to-Speech and Multimodal Feedback: Integrating text-to-speech functionality would provide valuable auditory feedback, especially for visually impaired users, helping them navigate the library with ease. Additionally, incorporating haptic feedback, such as vibrations, would provide tactile signals to users, further improving accessibility and ensuring that the robot delivers a more inclusive experience for people with varying needs.

6.2.4. Energy Efficiency and Sustainability

Energy-Efficient Hardware: To enhance sustainability, the system's power consumption can be optimized by integrating energy-efficient motors and low-power sensors. This will help reduce the overall energy usage while maintaining the robot’s performance. Additionally, exploring the use of solar panels could significantly extend the robot's battery life, further minimizing its environmental impact and supporting more eco-friendly operations.

Automatic Recharging: Implementing an automatic recharging system would allow the robot to autonomously return to a power station once its battery is running low. This feature would ensure that the robot can operate continuously without the need for manual intervention, enabling uninterrupted service and improving the efficiency of library operations.

6.2.5. Scalability and Customization

Modular Architecture: Structuring the system in a modular way would allow libraries to tailor the robot’s functionality according to their specific requirements. This flexibility means that libraries can easily add or remove features, such as advanced navigation modules or custom user interfaces, as their needs evolve, ensuring the robot remains adaptable and relevant over time.

Cloud Integration: Integrating the robot with cloud-based systems enables remote monitoring, data analysis, and software updates, providing libraries with a powerful tool for managing their robots efficiently. This integration would also allow libraries to scale the system as needed, making it easier to deploy and manage multiple robots across different locations without complex infrastructure.

6.2.6. Advanced Machine Learning for Navigation

Deep Reinforcement Learning: By using deep reinforcement learning, the robot can be trained to navigate through complex environments while adapting to unexpected changes. This approach

would significantly enhance the robot's ability to handle dynamic obstacles and unstructured paths, enabling it to improve its decision-making and navigation strategies over time as it learns from its experiences.

Transfer Learning: Implementing transfer learning would allow the robot to apply knowledge gained from one environment to another, reducing the time and resources required for training in new settings. This technique enables the robot to quickly adapt to different library layouts or environments without starting the learning process from scratch, making it more efficient and versatile in various situations.

6.3. Impact on Society

The development of autonomous library robots has the possibility of causing notable effects society by enhancing accessibility, improving efficiency, and reducing operational costs in libraries.

Enhanced Accessibility: The robot provides a user-friendly interface and multimodal feedback (e.g., text-to-speech, haptic feedback), making library services more accessible to visually impaired individuals and people with disabilities.
Improved Efficiency:

- The robot streamlines routine duties such as finding books and assisting patrons, easing the workload of library staff.
- As a result, library operations become more efficient, allowing staff to focus on higher-level tasks.

Cost Reduction: Using low-cost hardware and open-source software makes the system affordable for small to medium-sized libraries, reducing the financial burden of automation.

6.4. Sustainability of the Work

The project emphasizes sustainability by using energy-efficient hardware, low-cost components, and open-source software. These choices ensure that the system is environmentally friendly and economically viable for long-term use.

Energy Efficiency: The robot is designed with energy efficiency in mind, using low-power sensors and energy-saving motors to minimize overall power consumption. This thoughtful engineering not only reduces operating costs but also supports the library's goal of maintaining a sustainable and eco-friendly environment. In some cases, the addition of solar panels can further cut down on energy usage, making the system even more environmentally responsible.

Recyclability: A key feature of the robot is its modular design, which makes it easy to repair and upgrade individual components rather than replacing the entire system. This approach significantly extends the robot's lifespan and helps reduce electronic waste. By focusing on repairability and reusability, the system aligns well with sustainable technology practices.

Scalability: The robot's modular architecture and integration with cloud-based services allow libraries to expand or adapt the system as their needs grow. Whether it's adding more robots, updating features, or supporting larger collections, the system can be scaled up efficiently without starting from scratch. This flexibility ensures the solution remains sustainable and relevant over time.

6.5. Social and Environmental Effects Analysis

The deployment of autonomous library robots has both positive and negative social and environmental effects, which must be carefully analyzed.

6.5.1. Positive Effects

The robot makes the library experience smoother and more efficient by helping users quickly find the books they're looking for, saving them time and effort. It also helps reduce the library's carbon footprint by taking over certain routine tasks, which can lessen the need for extra staffing and associated energy use. On top of that, the robot is designed with accessibility in mind—its easy-to-use interface and features like voice and visual feedback make it easier for people with visual impairments or other disabilities to navigate the library independently.

6.5.2. Negative Effects

Job Displacement: The automation of tasks such as book location and user assistance through library robots may raise concerns about job displacement for staff. While these technologies can improve efficiency and user experience, they may also reduce the need for certain routine roles. To address this issue, libraries can invest in retraining programs that equip staff with new skills for more complex and value-added roles, such as system management, user engagement, and digital literacy support. This not only helps preserve employment but also enhances the overall service quality by combining human expertise with technological innovation.

Electronic Waste: The integration of electronic components in library robots raises concerns about the potential generation of electronic waste, especially if the devices are not disposed of responsibly at the end of their lifecycle. To mitigate this environmental impact, libraries and developers can implement recycling programs that ensure outdated or non-functional components are properly processed and reused. Additionally, designing robots with modular components allows for easier repairs and upgrades, reducing the need to replace entire systems and promoting sustainability. These practices contribute to a more eco-friendly approach to technological advancement in library settings.

6.6. Ethics and Ethical Issues

The development and deployment of autonomous library robots raises several ethical issues that must be addressed to ensure the system's responsible use.

Privacy Concerns: Using cameras and sensors in library robots may raise privacy concerns among users, as individuals could feel uncomfortable being monitored or recorded in a public space. Tackling these concerns requires strong data security practices, including the use of encryption and anonymization methods. Encryption safeguards stored and transmitted data from unauthorized access, while anonymization protects user privacy by obscuring or removing identifiable information. By integrating these safeguards, libraries can maintain user trust and uphold privacy standards while benefiting from robotic systems' technological advantages.

Bias and Fairness: Bias and fairness are critical considerations in the design of library robots, as their algorithms must ensure equal and unbiased service delivery to all users, regardless of background or demographics. To ensure this outcome, developers ought to train the robot using varied and inclusive datasets that capture the diversity of its potential user base. It is essential to conduct regular audits and assessments of the system's performance to detect and address any potential biases that may arise unintentionally. By actively monitoring and updating the algorithms, libraries can promote inclusivity and ensure that the robot serves every user equitably.

Transparency and Accountability: Transparency and accountability are essential for building user trust in library robots. The system should provide transparent and comprehensible reasoning behind its actions or suggestions, enabling users to understand the basis for its decisions. This transparency can be supported by providing accessible explanations for the robot’s behavior, such as why it navigated a particular route or suggested a specific section. Additionally, users should have the opportunity to challenge or report decisions they find incorrect or biased, ensuring the system remains responsive and responsible. By fostering openness and enabling user feedback, libraries can enhance the credibility and reliability of robotic services.

Job Displacement: The automation of routine tasks through the use of library robots may raise concerns about potential job displacement among library staff. This issue can be addressed by equipping employees with new skills, enabling them to transition into roles that demand human insight, creativity, and social interaction—capabilities that are difficult for robots to replicate. Moreover, involving staff in the design, implementation, and management of the robotic systems not only empowers them with new skills but also ensures that the technology complements rather than replaces their work. This collaborative approach fosters a more adaptive, future-ready library environment.

6.7. Final Thoughts

This project has successfully demonstrated the feasibility of using low-cost robotics for library automation, offering a practical and affordable solution for book location and user assistance. By leveraging accessible hardware and open-source software, the system provides an efficient way to enhance library navigation while minimizing operational costs.

Its success, certain limitations being identified, such as sensitivity to environmental variations and the need for improved adaptability in dynamic settings. Addressing these challenges through further refinements, such as enhanced obstacle avoidance, adaptive learning algorithms, and improved sensor integration, can significantly boost the system’s performance and reliability.

Its immediate application in libraries, this work contributes to broader advancements in autonomous navigation, human-robot interaction, and large-scale robotic deployment. The insights gained from this project can inspire future innovations in service robotics, setting a strong foundation for the next generation of intelligent library robots and other automated assistance systems in public spaces.

Bibliography

1. Helsinki Central Library Oodi, "Enhancing Visitor Experience with Social Robots," 2020. [Online]. Available: [Oodi Library Website].
2. National Library Board Singapore, "Aurora: The Next Generation Library Robot," 2022. [Online]. Available: [NLB Singapore Website].
3. Yi Sun-sin Library, "Embracing Technology in Modern Libraries," 2023. [Online]. Available: [Yi Sun-sin Library Website].
4. C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309-1332, 2016.
5. S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
6. Mahmood, A., Baig, A., & Ahsan, Q. (2012, October). Real time localization of mobile robotic platform via fusion of inertial and visual navigation system. In *2012 International Conference of Robotics and Artificial Intelligence* (pp. 40-44). IEEE.
7. Raaj, Y., John, A., & Jin, T. (2016, September). 3d object localization using forward looking sonar (fls) and optical camera via particle filter based calibration and fusion. In *OCEANS 2016 MTS/IEEE Monterey* (pp. 1-10). IEEE.
8. Yang, S. W., & Wang, C. C. (2010). On solving mirror reflection in lidar sensing. *IEEE/ASME Transactions on Mechatronics*, *16*(2), 255-265.
9. Behravesh, V., & Farshchi, S. M. R. (2012). A Novel Randomized Search Technique for Multiple Mobile Robot Paths Planning In Repetitive Dynamic Environment. *IAES International Journal of Robotics and Automation*, *1*(4), 214.
10. Pandey, A., Pandey, S., & Parhi, D. R. (2017). Mobile robot navigation and obstacle avoidance techniques: A review. *Int Rob Auto J*, *2*(3), 00022.
11. Radaelli, R. R., Badue, C., Gonçalves, M. A., Oliveira-Santos, T., & De Souza, A. F. (2014). A motion planner for car-like robots based on rapidly-exploring random trees. In *Advances in Artificial Intelligence--IBERAMIA 2014: 14th Ibero-American Conference on AI, Santiago de Chile, Chile, November 24-27, 2014, Proceedings 14* (pp. 469-480). Springer International Publishing.
12. Koren, Y., & Borenstein, J. (1991, April). Potential field methods and their inherent limitations for mobile robot navigation. In *Icra* (Vol. 2, No. 1991, pp. 1398-1404).
13. Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, *7*(3), 278-288.
14. Kamal, I. M., Salah Abd-elhafeez, M., & Farghal, A. (2023). Camera-Based Navigation System for Blind and Visually Impaired People. Sohag Engineering Journal, 3(1), 1-13.
15. Farkh, R., & Aljaloud, K. (2023). Vision navigation based PID control for line tracking robot. *Intelligent Automation & Soft Computing*, *35*(1), 901-911.
16. Yang, H., Wang, Y., & Gao, L. (2016, May). A general line tracking algorithm based on computer vision. In *2016 Chinese Control and Decision Conference (CCDC)* (pp. 5365-5370). IEEE.

17. Kondakor, A., Törcsvari, Z., Nagy, A., & Vajk, I. (2018, May). A line tracking algorithm based on image processing. In *2018 IEEE 12th international symposium on applied computational intelligence and informatics (SACI)* (pp. 000039-000044). IEEE.
18. Surynek, P. (2021, May). Multi-goal multi-agent path finding via decoupled and integrated goal vertex ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 14, pp. 12409-12417).
19. Chan, S. H., Stern, R., Felner, A., & Koenig, S. (2023, July). Greedy priority-based search for suboptimal multi-agent path finding. In *Proceedings of the International Symposium on Combinatorial Search* (Vol. 16, No. 1, pp. 11-19).
20. Zhang, S., Li, J., Huang, T., Koenig, S., & Dilkina, B. (2022, July). Learning a priority ordering for prioritized planning in multi-agent path finding. In *Proceedings of the International Symposium on Combinatorial Search* (Vol. 15, No. 1, pp. 208-216).
21. Liao, B., Zhu, S., Hua, Y., Wan, F., & Qing, X. (2023). A decoupling method for solving the multi-agent path finding problem. *Complex & Intelligent Systems*, *9*(6), 6767-6780.
22. Raj, R., & Kos, A. (2022). A comprehensive study of mobile robot: history, developments, applications, and future research perspectives. *Applied Sciences*, *12*(14), 6951.
23. Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., & Jurišica, L. (2014). Path planning with modified a star algorithm for a mobile robot. *Procedia engineering*, *96*, 59-69.
24. Tang, G., Tang, C., Claramunt, C., Hu, X., & Zhou, P. (2021). Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment. *IEEE access*, *9*, 59196-59210.
25. Ju, C., Luo, Q., & Yan, X. (2020, October). Path planning using an improved a-star algorithm. In *2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan)* (pp. 23-26). IEEE.
26. Seng, T. L., Khalid, M. B., & Yusof, R. (1999). Tuning of a neuro-fuzzy controller by genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *29*(2), 226-236.
27. Lee, M., Lee, S. Y., & Park, C. H. (1994). Neuro-fuzzy identifiers and controllers. *Journal of Intelligent & Fuzzy Systems*, *2*(1), 1-14.
28. Baojiang, Z., & Shiyong, L. (2007). Ant colony optimization algorithm and its application to neuro-fuzzy controller design. *Journal of Systems Engineering and Electronics*, *18*(3), 603-610.
29. Nürnberger, A., Nauck, D., & Kruse, R. (1999). Neuro-fuzzy control based on the NEFCON-model: recent developments. *Soft Computing*, *2*, 168-182.
30. Zuo, X. Q., & Fan, Y. S. (2006). A chaos search immune algorithm with its application to neuro-fuzzy controller design. *Chaos, Solitons & Fractals*, *30*(1), 94-109.
31. Loganathan, A., & Ahmad, N. S. (2023). A systematic review on recent advances in autonomous mobile robot navigation. *Engineering Science and Technology, an International Journal*, *40*, 101343.
32. Reeves, C., & Rowe, J. E. (2002). *Genetic algorithms: principles and perspectives: a guide to GA theory* (Vol. 20). Springer Science & Business Media.
33. Deb, K. (1999). An introduction to genetic algorithms. *Sadhana*, *24*, 293-315.
34. Scrucca, L. (2013). GA: A package for genetic algorithms in R. *Journal of Statistical Software*, *53*, 1-37.

35. Mundada, V., & Narala, S. K. R. (2018). Optimization of milling operations using artificial neural networks (ANN) and simulated annealing algorithm (SAA). *Materials today: proceedings*, *5*(2), 4971-4985.
36. Wang, Y., Bu, G., Wang, Y., Zhao, T., Zhang, Z., & Zhu, Z. (2016). Application of a simulated annealing algorithm to design and optimize a pressure-swing distillation process. *Computers & Chemical Engineering*, *95*, 97-107.
37. Song, S. (2019). An improved simulated annealing algorithm for reconstructing 3D large-scale porous media. *Journal of Petroleum Science and Engineering*, *182*, 106343.
38. Adil, G. K., Rajamani, D., & Strong, D. (1997). Assignment allocation and simulated annealing algorithms for cell formation. *IIE transactions*, *29*(1), 53-67.
39. Manikumar, R., & Ramesh, S. (2022). Bacterial Foraging Optimization (BFO) Algorithm based Deep Neural Network Internal Model Controller for a shell and Tube Heat Exchanger. *NeuroQuantology*, *20*(10), 11718.
40. Mohajer, B., Kiani, K., Samiei, E., & Sharifi, M. (2013). A new online random particles optimization algorithm for mobile robot path planning in dynamic environments. *Mathematical Problems in Engineering*, *2013*(1), 491346.
41. Panigrahi, P. K., Ghosh, S., & Parhi, D. R. (2014). Comparison of GSA, SA and PSO based intelligent controllers for path planning of mobile robot in unknown environment. *International Journal of Electrical and Computer Engineering*, *8*(10), 1633-1642.
42. Fister, I., Fister Jr, I., Yang, X. S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and evolutionary computation*, *13*, 34-46.
43. Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. *International journal of swarm intelligence*, *1*(1), 36-50.
44. Johari, N. F., Zain, A. M., Noorfa, M. H., & Udin, A. (2013). Firefly algorithm for optimization problem. *Applied Mechanics and Materials*, *421*, 512-517.
45. Yang, X. S., & Slowik, A. (2020). Firefly algorithm. In *Swarm intelligence algorithms* (pp. 163-174). CRC Press.
46. Krell, E., Sheta, A., Balasubramanian, A. P. R., & King, S. A. (2019). Collision-free autonomous robot navigation in unknown environments utilizing PSO for path planning. *Journal of Artificial Intelligence and Soft Computing Research*, *9*(4), 267-282.
47. Garcia-Gonzalo, E., & Fernandez-Martinez, J. L. (2012). A brief historical review of particle swarm optimization (PSO). *Journal of Bioinformatics and Intelligent Control*, *1*(1), 3-16.
48. Fourie, P. C., & Groenwold, A. A. (2002). The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, *23*, 259-267.
49. Carvalho, M., & Ludermir, T. B. (2007, September). Particle swarm optimization of neural network architectures andweights. In *7th International Conference on Hybrid Intelligent Systems (HIS 2007)* (pp. 336-339). IEEE.
50. Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, *1*(4), 28-39.
51. Dorigo, M., & Stützle, T. (2019). *Ant colony optimization: overview and recent advances* (pp. 311-351). Springer International Publishing.

52. Dorigo, M., & Socha, K. (2018). An introduction to ant colony optimization. In *Handbook of approximation algorithms and metaheuristics* (pp. 395-408). Chapman and Hall/CRC.
53. Ribeiro, C. C., Hansen, P., Maniezzo, V., & Carbonaro, A. (2002). Ant colony optimization: an overview. *Essays and surveys in metaheuristics*, 469-492.
54. Bouraine, S., Fraichard, T., & Salhi, H. (2012). Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. *Autonomous Robots*, *32*, 267-283.
55. Bouraine, S., Fraichard, T., Azouaoui, O., & Salhi, H. (2014, May). Passively safe partial motion planning for mobile robots with limited field-of-views in unknown dynamic environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3576-3582). IEEE.
56. Bouraine, S., Fraichard, T., & Azouaoui, O. (2016, December). Real-time safe path planning for robot navigation in unknown dynamic environments. In *CSA 2016-2nd Conference on Computing Systems and Applications*.
57. Bouraine, S., Fraichard, T., & Salhi, H. (2011, September). Relaxing the inevitable collision state concept to address provably safe mobile robot navigation with limited field-of-views in unknown dynamic environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2985-2991). IEEE.
58. Bouraine, S., & Azouaoui, O. (2021). Safe motion planning based on a new encoding technique for tree expansion using particle swarm optimization. *Robotica*, *39*(5), 885-927.
59. Phan, D., Yang, J., Ratasich, D., Grosu, R., Smolka, S. A., & Stoller, S. D. (2015). Collision avoidance for mobile robots with limited sensing and limited information about the environment. In *Runtime Verification: 6th International Conference, RV 2015, Vienna, Austria, September 22-25, 2015. Proceedings* (pp. 201-215). Springer International Publishing.
60. Phan, D., Yang, J., Grosu, R., Smolka, S. A., & Stoller, S. D. (2017). Collision avoidance for mobile robots with limited sensing and limited information about moving obstacles. *Formal Methods in System Design*, *51*, 62-86.
61. Khan, M. U. (2019). Mobile robot navigation using reinforcement learning in unknown environments. *Balkan Journal of Electrical and Computer Engineering*, *7*(3), 235-244.
62. Wilamowski, B. M. (2009). Neural network architectures and learning algorithms. *IEEE Industrial Electronics Magazine*, *3*(4), 56-63.
63. Günther, F., & Fritsch, S. (2010). Neuralnet: training of neural networks. *R J.*, *2*(1), 30.
64. Karayiannis, N., & Venetsanopoulos, A. N. (2013). *Artificial neural networks: learning algorithms, performance evaluation, and applications* (Vol. 209). Springer Science & Business Media.
65. Fausett, L. V. (2006). *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India.
66. Zhang, X. S. (2013). *Neural networks in optimization* (Vol. 46). Springer Science & Business Media.
67. El Alami, H., & Najid, A. (2017). Fuzzy logic based clustering algorithm for wireless sensor networks. *International Journal of Fuzzy System Applications (IJFSA)*, *6*(4), 63-82.

68. Radaelli, R. R., Badue, C., Gonçalves, M. A., Oliveira-Santos, T., & De Souza, A. F. (2014). A motion planner for car-like robots based on rapidly-exploring random trees. In *Advances in Artificial Intelligence--IBERAMIA 2014: 14th Ibero-American Conference on AI, Santiago de Chile, Chile, November 24-27, 2014, Proceedings 14* (pp. 469-480). Springer International Publishing.
69. 董璐, & 熊爱玲. (2022). 基于改进 RRT*-Smart 的复杂动态环境下的无人艇路径规划. *智能科学与技术学报*, *4*(2), 264-276.
70. Tahir, Z., Qureshi, A. H., Ayaz, Y., & Nawaz, R. (2018). Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, *108*, 13-27.
71. Xin, P., Wang, X., Liu, X., Wang, Y., Zhai, Z., & Ma, X. (2023). Improved bidirectional RRT* algorithm for robot path planning. *Sensors*, *23*(2), 1041.
72. Wang, J., Chi, W., Li, C., & Meng, M. Q. H. (2021). Efficient robot motion planning using bidirectional-unidirectional RRT extend function. *IEEE Transactions on Automation Science and Engineering*, *19*(3), 1859-1868.
73. Noreen, I., Khan, A., & Habib, Z. (2016). Optimal path planning using RRT* based approaches: a survey and future directions. *International Journal of Advanced Computer Science and Applications*, *7*(11).
74. Noreen, I., Khan, A., & Habib, Z. (2016). A comparison of RRT, RRT* and RRT*-smart path planning algorithms. *International Journal of Computer Science and Network Security (IJCSNS)*, *16*(10), 20.
75. Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., & Muhammad, M. S. (2013). RRT*-SMART: A rapid convergence implementation of RRT. *International Journal of Advanced Robotic Systems*, *10*(7), 299.
76. Jeong, I. B., Lee, S. J., & Kim, J. H. (2019). Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Systems with Applications*, *123*, 82-90.
77. El-Hussieny, H., Assal, S. F., & Abdellatif, M. (2013, June). Improved backtracking algorithm for efficient sensor-based random tree exploration. In *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks* (pp. 19-24). IEEE.
78. Kim, J., Seong, K. J., & Kim, H. J. (2012, October). An efficient backtracking strategy for frontier method in sensor-based random tree. In *2012 12th International Conference on Control, Automation and Systems* (pp. 970-974). IEEE.
79. Horak, K., & Zalud, L. (2016). Image processing on raspberry pi for mobile robotics. *International Journal of Signal Processing Systems*, *4*(2), 1-5.
80. Amir, S., Siddiqui, A. A., Ahmed, N., & Chowdhry, B. S. (2014, December). Implementation of line tracking algorithm using Raspberry pi in marine environment. In 2014 IEEE International Conference on Industrial Engineering and Engineering Management (pp. 1337-1341). IEEE.

81. Setiawan, F. B., Putra, E. P., Pratomo, L. H., & Riyadi, S. (2022). Implementation of line detection self-driving car using HSV method based on raspberry pi 4. Jurnal Infotel,14(4), 321-328.
82. Jin, Z., Zhuang, Z., & Yuan, B. (2024, December). Applications of visual navigation and intelligent recognition technology. In Fourth International Conference on Testing Technology and Automation Engineering (TTAE 2024) (Vol. 13439, pp. 638-644). SPIE.
83. Khalilullah, K. I., Ota, S., Yasuda, T., & Jindai, M. (2017, September). Development of robot navigation method based on single camera vision using deep learning. In 2017 56th annual conference of the society of instrument and control engineers of Japan (SICE) (pp. 939-942). IEEE.
84. Patel, D., Dalwadi, P., Dhumal, S., & Bhalodiya, A. (2020). Development and Design of Autonomous Navigation Robot Using Raspberry Pi and Computer Vision. *International Research Journal of Engineering and Technology (IRJET)*, *7*, 864-869.
85. Behan, J., & O'Keeffe, D. T. (2008). The development of an autonomous service robot. Implementation:"Lucas"—The library assistant robot. *Intelligent Service Robotics*, *1*, 73-89.
86. Vlachos, E., Hansen, A. F., & Holck, J. P. (2020, July). A robot in the library. In *International conference on human-computer interaction* (pp. 312-322). Cham: Springer International Publishing.
87. Prats, M., Martínez, E., Sanz, P. J., & Del Pobil, A. P. (2008). The UJI librarian robot. *Intelligent Service Robotics*, *1*(4), 321-335.
88. Lin, W., Yueh, H. P., Wu, H. Y., & Fu, L. C. (2014). Developing a service robot for a children's library: A design-based research approach. *Journal of the Association for Information Science and Technology*, *65*(2), 290-301.
89. Martinez-Martin, E., Ferrer, E., Vasilev, I., & Del Pobil, A. P. (2021). The UJI aerial librarian robot: A quadcopter for visual library inventory and book localisation. *Sensors*, *21*(4), 1079.
90. [123] Al-nabhani, S., & Muneer, A. (2019). Automated library system using SMS based pick and place robot. *International Journal of Computing and Digital Systems*, *8*(6).
91. Library robot in oOdi description (https://futurice.com/blog/the-little-robot-that-lived-at-the-library)
92. AROUND B robot description (https://travelbetweenthepages.com/2020/03/01/coming-to-a-bookstore-near-you/)
93. Padbot description (https://www.padbot.com/)
94. Di Veroli, C., Le, C. A., Lemaire, T., Makabu, E., Nur, A., Ooi, V., ... & Demiris, Y. (2019). LibRob: an autonomous assistive librarian. In *Annual Conference Towards Autonomous Robotic Systems* (pp. 15-26). Springer, Cham.
95. Mubin, O., Kharub, I., & Khan, A. (2020, April). Pepper in the Library" Students' First Impressions. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1-9).
96. Sahu, U., Upadhyay, S., Singh, N., & Patil, P. (2016, March). Libo: The grasping robot using object recognition. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)* (pp. 3166-3171). IEEE.
97. Beepbot description (https://www.cbc.ca/news/canada/kitchener-waterloo/guelph-public-library-child-sized-robot-named-1.5067725)
98. Raspberrypi (https://www.raspberrypi.com/products/raspberry-pi-4-model-b/)
99. Webcam-(https://www.logitech.com/en-us/products/webcams/c505-hd-webcam.960-001363.html)

100. Motor Driver (https://robodocbd.com/product/bts7960-motor-driver-module)
101. Motor (https://store.roboticsbd.com/motor/2897-37gb-12v-high-torque-gear-motor-500-rpm-robotics-bangladesh.html)
102. OpenCV(https://en.wikipedia.org/wiki/OpenCV)
103. Python (https://en.wikipedia.org/wiki/Python_(programming_language))