

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти магістерський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

WANG QUANQUAN

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Warehouse Material Handling Robot Control System

керівник проєкту (роботи) PhD, associate professor, ZELENKOVA Irina
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) Technical documentation and software tools for the development of the AGVs robot control

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

1) Analysis of multi-platform development optimization approaches;

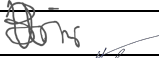
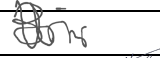
2) Designing a system of automatic code generation;

3) Implementation of the automatic code generation system;

4) Implementation of the system.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ZELENOVA I., associate professor		
нормоконтроль	SHCHERBAK N., senior lecturer		

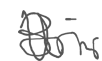
7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Review of analogues of warehouse work management systems and formulation of the development task	10.10.2023 р.	
2	System structure design	25.10.2023 р.	
3	Implementation of system software	10.11.2023 р.	
4	Study of the system	15.11.2023 р.	
5	User manual development	27.11.2023 р.	
6	Registration of the obtained results in the software	02.12.2023 р.	
7	Preparation of supporting material	10.12.2023 р.	

Студент Wang Quanquan
(підпис)WANG QUANQUAN
(ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)Irina ZELENOVA
(ім'я, ПРИЗВИЩЕ)

ABSTRACT

EN: 81 p., 14 pictures, 10 references, 1 appendix

AUTOMATED WAREHOUSE ROBOTS, CONTROL SYSTEMS, INERTIAL NAVIGATION, WIRELESS COMMUNICATION, QR CODE

This thesis delves into the analysis, design, and testing of a comprehensive storage and handling robot control system. The research begins with an exploration of warehouse robot control systems, examining their fundamental aspects and functional requirements. The system scheme design encompasses the overall structure, navigation and positioning mode selection, and communication mode selection.

The focus then shifts to the software design of the warehouse handling robot control system, elucidating the overall architecture and workflow. The System Initialization Module and Data Acquisition Module designs are detailed, emphasizing their crucial roles in system functionality. The subsequent section unfolds the hardware debugging process, encompassing tests for the main control module, motion control module, and navigation and positioning module commissioning. The control system software debugging phase is thoroughly discussed, covering joint commissioning efforts to ensure seamless system integration.

This thesis serves as a comprehensive guide to the development and testing of a warehouse logistics handling robot control system, providing insights into the intricacies of system design, software architecture, and debugging processes. The culmination of this research contributes to the advancement of efficient and reliable control systems for handling robots in dynamic warehouse environments. This work incorporates a demonstrator program developed in the Python programming language, specializing in determining the trajectory of a robot in warehouse conditions. The program addresses the tasks of finding the optimal path for the robot, avoiding obstacles, and defining the coordinates of the start and end points.

CONTENTS

Introduction	6
1 Analysis and overall design of the storage and handling robot control system	8
1.1 Overview of warehouse robot control systems	10
1.2 Analysis of System Functional Requirements	15
1.3 System scheme design	18
1.3.1 Overall structure of the system	18
1.3.2 Navigation and positioning mode selection	32
1.3.3 Communication mode selection	34
2 Design of warehouse handling robot control system software	35
2.1 Overall architecture design of system software	37
2.1.1 The overall workflow of the warehouse logistics robot software system	39
2.1.2 System Initialization Module Design	43
2.2 Software Design for Motion Control Module	44
3 Warehouse logistics handling robot control system test	49
3.1 Hardware debugging of control system	49
3.1.1 Main control module test	49
3.1.2 Motion control module test	52
3.1.3 Commissioning a navigation and positioning module	54
3.2 Control system software debugging	56
Conclusion	75
References	77
Appendix A	78

INTRODUCTION

In the traditional logistics business model, manual handling is time-consuming, with high work intensity and relatively low efficiency and high error rates in sorting by personnel. It cannot meet the characteristics of multi-category and small-batch electronic logistics operations. In addition, emerging labor is generally uninterested in monotonous and arduous screening work, and with the deepening of aging in China's population, labor costs continue to rise, making logistics companies urgently need intelligent logistics equipment and the use of automation technology to improve the sorting and handling capabilities of e-commerce logistics.

Automated Guided Vehicles (AGVs) are a type of handling robot used for equipment handling and automated assembly. Traditional AGVs are mostly traction-type or forklift-type, with large volume and requiring wider aisles, making them unsuitable for the current logistics warehouse environment.

Traditional handling robots require ground installation of guidance belts and are difficult to achieve corresponding performance requirements at once when storage environment or storage procedures change. Modern logistics systems are based on traditional logistics systems, supported by advanced scientific theory and modern scientific and technological capabilities. They require handling robots to obtain real-time and continuous dynamic decision-making through perception of surrounding environmental information, as well as the ability to select and control paths during travel, and have the characteristics of long-distance transportation and transportation to any location. In summary, the use of warehouse handling robots is beneficial for reducing the cost of logistics sorting and handling, reducing personnel input, improving logistics management, reducing the probability of goods handling damage, and improving the efficiency of modern logistics, promoting the development of the logistics industry.

Additionally, the related technology of warehouse handling robots can be used in other fields, including material handling, hotel freight handling, military, and

dangerous sites. Therefore, designing and implementing a warehouse handling robot based on the logistics industry has high research and practical value and good social benefits. This Master's Thesis focuses on the design and implementation of a control system for a warehouse handling robot, as a research subject, based on the actual needs of the electronic logistics industry.

Firstly, the overall scheme design of the control system for the warehouse handling robot is completed according to the system requirements, and the selection of key components is determined.

Then, the hardware is designed based on the overall scheme of the control system, followed by the completion of the embedded software design for the control system.

Finally, the software and hardware of the control system for the warehouse handling robot are tested to ensure the feasibility and correctness of the work presented in this Master's Thesis.

1 ANALYSIS AND OVERALL DESIGN OF THE STORAGE AND HANDLING ROBOT CONTROL SYSTEM

This chapter first focuses on the characteristics of the working environment of the warehouse handling robot, determines the navigation and communication methods, briefly introduces the mechanical structure and sensor layout of the robot, and finally, based on the functional requirements analysis and combined with the performance indicators of the robot, selects the core components of the robot.



Figure 1.1 – Image of a robotic warehouse

The implementation of robotics in warehouse material handling systems marks a significant evolution in the logistics and supply chain domain. Understanding the historical context and the importance of efficient material handling sets the stage for comprehending the necessity and impact of advanced control systems for warehouse robots.

Historically, warehouse operations have undergone a transformative journey

from manual handling to semi-automated processes and, more recently, the integration of sophisticated robotic systems. The evolution encompasses the shift from traditional forklifts and conveyors to the adoption of automated guided vehicles (AGVs) and, ultimately, intelligent robotic systems capable of autonomous decision-making. Exploring this evolutionary timeline provides insights into the driving forces behind the adoption of robotic solutions in material handling within warehouse environments.



Figure 1.2 – Robotic forklift

Efficient material handling is a cornerstone of successful warehouse management, directly influencing overall operational productivity and customer satisfaction. Timely and accurate movement of goods, coupled with minimized errors, not only enhances the internal workflow but also optimizes resource utilization. The significance of streamlining material handling processes becomes particularly pronounced in the context of the modern demand for just-in-time

inventory management and the growing complexities of supply chain networks. This section delves into the critical role that efficient material handling plays in ensuring the competitiveness and sustainability of warehouses in a dynamic business landscape.

Understanding the historical progression and recognizing the pivotal role of efficient material handling sets the groundwork for exploring the intricacies of implementing a sophisticated control system for warehouse handling robots. This chapter aims to provide a comprehensive overview that facilitates a deeper understanding of the challenges and opportunities associated with the integration of robotics in contemporary warehouse operations.

1.1 Overview of warehouse robot control systems

This section provides a comprehensive overview of warehouse robot control systems, encompassing their evolution, significance, types, key components, integration with navigation technologies, challenges, case studies, and future trends. It serves as a foundational understanding for subsequent chapters, laying the groundwork for in-depth exploration and analysis.

Warehouse robot control systems represent the technological backbone that orchestrates the actions and responses of robotic entities operating within complex warehouse environments. Robot control systems refer to the set of algorithms, software, and hardware components designed to regulate and govern the behavior of robotic platforms. These systems play a pivotal role in enabling robots to perceive their surroundings, make informed decisions, and execute precise actions, thereby facilitating efficient material handling within warehouses.

The primary purpose of robot control systems is to imbue robots with a level of autonomy that enables them to navigate, manipulate objects, and interact with their environment intelligently. These systems are designed to enhance operational

efficiency, reduce human intervention, and ensure the seamless integration of robots into warehouse workflows. By defining the rules and parameters governing a robot's behavior, control systems contribute to the overall reliability and productivity of robotic applications in warehousing.

The evolution of control systems in robotics has been a dynamic process, closely intertwined with advancements in computing power, sensor technologies, and artificial intelligence. Understanding this evolution provides insights into the sophistication and capabilities of contemporary control systems.

Early Control Mechanisms: In the early stages of robotics, control mechanisms were simplistic and primarily reactive. Robots were tethered to pre-programmed routines, limiting their adaptability to dynamic environments.

Emergence of Closed-Loop Systems: With the advent of closed-loop control systems, robots gained the ability to receive feedback from sensors and adjust their actions in real-time. This marked a significant shift towards more adaptive and responsive robotic behavior.

Integration of Artificial Intelligence: The incorporation of artificial intelligence (AI) techniques, such as machine learning and neural networks, has propelled control systems to new heights. Modern warehouse robots can learn from experience, optimize their decision-making processes, and adapt to changing conditions.

Advancements in Human-Robot Interaction: Recent developments focus on enhancing human-robot collaboration by incorporating intuitive control interfaces. This evolution aims to create a seamless interaction between human operators and robotic systems, fostering a more user-friendly and productive working environment.

Warehouse robot control systems are integral to the seamless integration of robotic technologies into warehouse operations. This section delves into the significance of these systems, emphasizing their crucial role in enhancing robot autonomy and contributing to overall operational efficiency.

Robot autonomy refers to the ability of robotic systems to operate and make decisions independently, with minimal human intervention. The significance of control systems in this context lies in their capacity to empower robots with a high

degree of autonomy.

Navigation and Obstacle Avoidance: Control systems enable robots to navigate through the warehouse environment autonomously. By processing data from sensors and making real-time decisions, robots can avoid obstacles, choose optimal paths, and adapt to changes in their surroundings.

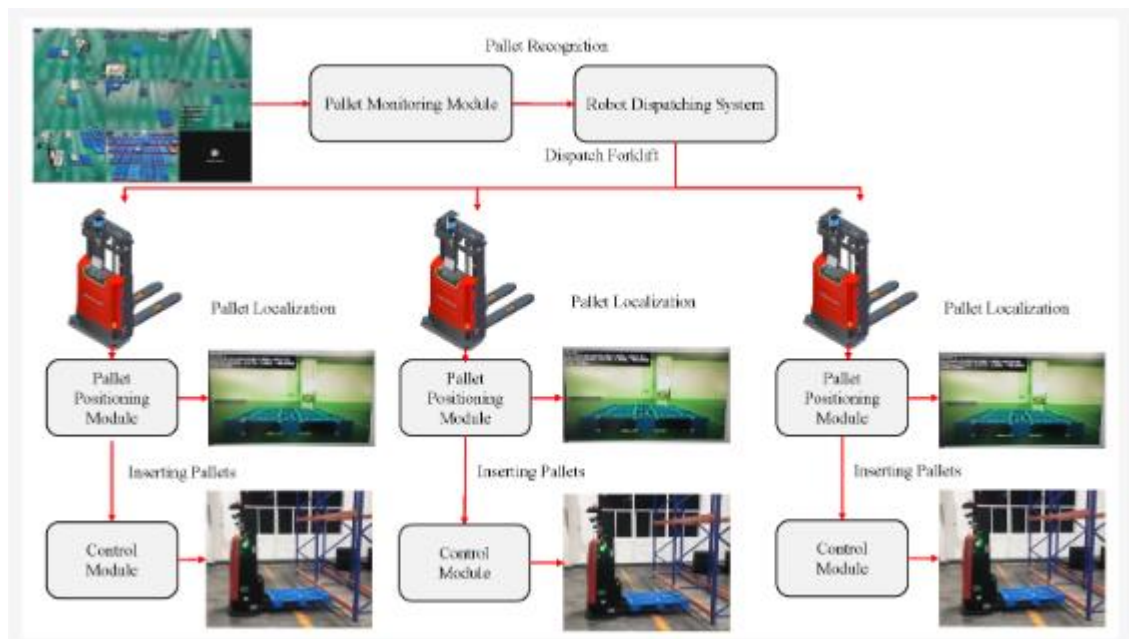


Figure 1.3 – Forklift Robot Data Processing from Sensors

Adaptive Decision-Making: Sophisticated control algorithms allow robots to make adaptive decisions based on the information gathered from their sensors. This adaptability is crucial in handling dynamic warehouse environments where conditions may change unpredictably.

Task Execution without Human Intervention: Control systems play a pivotal role in enabling robots to execute tasks without continuous human oversight. Whether it's picking items from shelves, transporting goods, or interacting with other machines, the autonomy provided by control systems contributes to operational efficiency and reduces the need for constant human guidance.

Operational efficiency is a key metric for the success of warehouse management, and control systems significantly contribute to achieving and optimizing this efficiency.

Precision and Accuracy in Operations: Control systems ensure precise control over robotic movements, leading to accurate execution of tasks. This precision is crucial for tasks such as picking and placing items, inventory management, and other material handling processes.

Reduction of Processing Time: Efficient control systems streamline robotic processes, reducing the time required to complete tasks. This reduction in processing time directly contributes to faster and more responsive warehouse operations, positively impacting order fulfillment and delivery timelines.

Optimization of Resource Utilization: Control systems aid in optimizing the use of resources, including energy and time. By providing robots with intelligent decision-making capabilities, these systems contribute to minimizing resource wastage and maximizing overall operational output.

In summary, the significance of control systems in warehouse robotics is underscored by their role in enhancing robot autonomy and contributing to operational efficiency. As warehouses increasingly embrace automation, understanding and optimizing these control systems become imperative for achieving a competitive edge in the modern logistics landscape.

Control systems in warehouse robotics are diverse, catering to various operational needs and preferences. This section explores the three main types of control systems: centralized, decentralized, and hybrid, highlighting their characteristics and applications.

Centralized control systems in warehouse robotics involve a singular, central authority that oversees and directs the actions of all robotic entities within the system.

Characteristics:

- **Unified Decision-Making.** All robots receive commands and make decisions based on a centralized controller, ensuring a synchronized and unified approach to warehouse tasks;
- **Streamlined Communication.** Communication between robots is facilitated through the central control unit, which can optimize task allocation and coordination;

- **Complex Task Planning.** Well-suited for scenarios where complex task planning and coordination are required, such as in large-scale warehouses with a high volume of simultaneous tasks.

Centralized control systems are commonly employed in environments where a high level of coordination is essential, and tasks demand real-time adjustments based on a centralized decision-making process.

Decentralized control systems distribute decision-making processes across individual robotic entities, allowing each robot to operate autonomously within predefined parameters.

Characteristics:

- **Autonomous Decision-Making.** Each robot has its own decision-making capabilities, enabling it to adapt to local conditions without constant input from a central controller;

- **Flexibility and Scalability.** Decentralized systems are often more flexible and scalable, as additional robots can be integrated without significantly affecting the overall system architecture;

- **Redundancy and Fault Tolerance.** If one robot encounters a fault or disruption, others can continue to operate independently, contributing to system robustness.

Decentralized control systems find applications in environments where flexibility and adaptability are crucial, and tasks can be distributed among robots without constant central oversight.

Hybrid control systems combine elements of both centralized and decentralized approaches, creating a flexible and adaptive solution that leverages the strengths of each.

Characteristics:

- **Combination of Centralized and Decentralized Elements.** Hybrid systems may have a central controller for high-level task planning while allowing individual robots to make local decisions within their designated areas;

- Optimization of Coordination. By blending centralized and decentralized elements, hybrid systems seek to optimize coordination and resource utilization;
- Adaptability to Varied Task Requirements. Well-suited for environments where some tasks benefit from centralized coordination, while others require the autonomy of decentralized control.

Hybrid control systems are applied in warehouses with diverse operational requirements, allowing for a tailored approach that balances centralized oversight and decentralized adaptability.

1.2 Analysis of System Functional Requirements

The work environment of warehousing and logistics handling robots in e-commerce logistics distribution is briefly laid out. Several shelves are neatly placed in the work area, and the warehousing and logistics handling robots wait for instructions from the personnel in the dispatch area. Upon receiving the instructions, the robots move to the bottom of the shelf and use a lifting mechanism to transport the shelf to the sorting area for processing before returning it to its original position. In order to maximize the use of the warehousing space, the shelves are placed as closely together as possible and have a multi-layered structure. When arranging the shelves, a space the width of one shelf is left for the robot to pass through when transporting the shelves. Taking into account the requirements and characteristics of domestic and international warehousing handling robots, the control system of the designed warehousing handling robot should be able to achieve the following functions.

Handling function. The warehousing handling robot needs to achieve the movement function. In the working space, the robot needs to be able to move in a straight line, make turns in place, and transport shelves through narrow passages to complete the handling task.



Figure 1.4 – Unmanned Handling Forklift Automated Guidance Vehicle

Navigation and positioning. The robot needs precise guidance and positioning methods to shuttle between the work area and the shelves to complete the handling task. The robot uses sensors to receive environmental information and combines with motion control algorithms to accurately move to the target position.

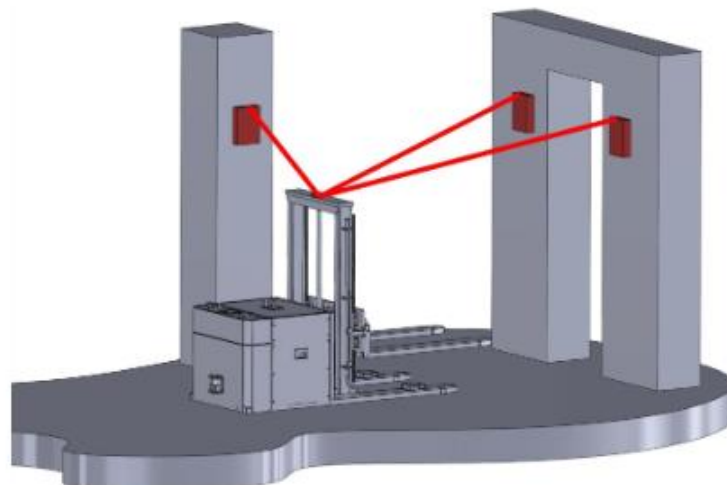


Figure 1.5 – Warehouse Material Handling Robot Laser Guided Navigation System

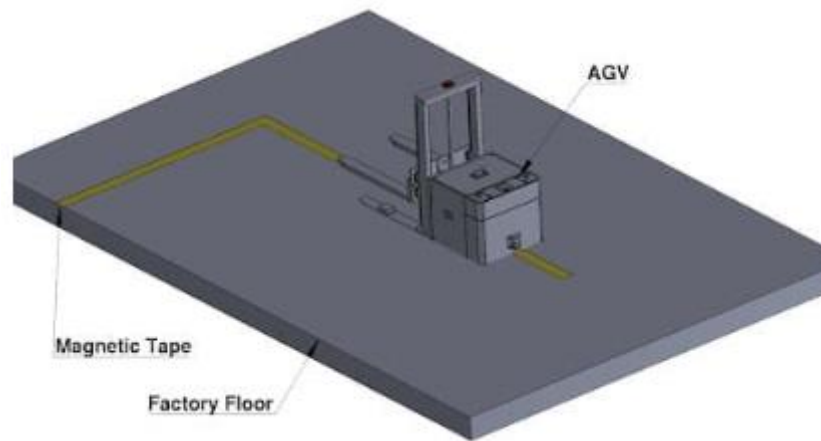


Figure 1.6 – Warehouse Material Handling Robot Magnetic Tape Navigation System

Wireless communication function. As the task instructions received by the robot come from warehouse management, it is necessary to implement wireless communication function. At the same time, the robot needs to provide real-time feedback on position, battery status, work status, etc. to warehouse management.



Figure 1.7 – WI-FI Communication

Safety and remote monitoring function. The robot needs to have active obstacle avoidance function during the walking process. When encountering obstacles, the robot should take evasive actions to ensure safe operation.



Figure 1.8 – Warehouse Security Areas

Autonomous charging function: Warehousing handling robots need to overcome the time limitation of work to improve the transportation efficiency of logistics goods. The implementation of autonomous charging technology can effectively improve the operating efficiency of robots.

1.3 System scheme design

1.3.1 Overall structure of the system

Starting from the functional requirements and performance parameters in the previous section. The control system is divided into six sub-modules for design.

Power supply and management module: The power supply is the source of power for the robot, and in order to avoid damaging the battery due to over-discharge during long-term use, a voltage monitoring circuit is added to monitor the battery voltage and ensure that it stays above the reference value.

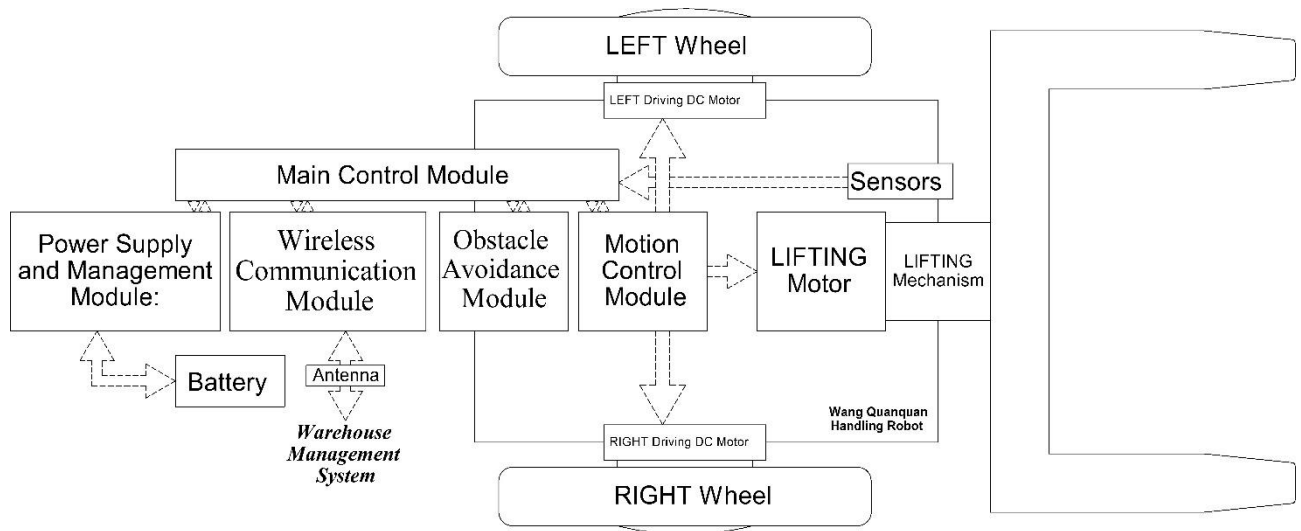


Figure 1.9 – Structure of the System

Main control module. This is the "brain" of the control system and plays a coordinating role among various modules. The main control module can receive feedback information from each module and make effective decisions and processing for control of relevant modules.

Motion control module. This module controls the movement of the robot, including linear walking, turning, and control of the two brushless DC motors for driving, as well as the lifting motor in the lifting mechanism. Motor speed and direction control is required during robot movement.

Obstacle avoidance module. This module is used to detect obstacles during robot movement to prevent collision and damage to the robot. The robot makes corresponding evasive actions when encountering obstacles and enters a stop/waiting state.

Navigation and positioning module. This module is used for navigation and positioning of the robot in the warehouse handling environment. The control system obtains position information of the robot in the environment through external sensors to provide security for robot handling.

Wireless communication module. This module is the connection link between the robot and the warehouse management system, used for data transmission and receiving task instructions and feedback information such as robot position and status

from the warehouse management system.

The Main Control Module serves as the central intelligence and coordination hub within the control system of a warehouse material handling robot. Often referred to as the "brain" of the robot, this module plays a crucial role in orchestrating and synchronizing the actions of various subsystems, ensuring efficient and effective operation.

Purpose and Functionality: The primary purpose of the Main Control Module is to receive, process, and act upon information from different modules within the robot. It serves as the decision-making center, interpreting feedback from sensors and other modules to generate commands and instructions for the robot's operation. The coordination facilitated by the main control module is essential for achieving seamless and autonomous material handling tasks within a warehouse setting.

Key Functions:

- **Feedback Processing.** The main control module receives feedback information from various sensors and subsystems, including the motion control module, navigation module, and positioning system. This feedback provides real-time data on the robot's environment, status, and performance;
- **Decision-Making.** Based on the received feedback, the main control module employs decision-making algorithms to determine the next course of action for the robot. This includes tasks such as path planning, obstacle avoidance, and coordination with other robots or systems in the warehouse;
- **Command Generation.** The module generates commands for the robot's actuators and effectors, instructing the motion control module on how to execute specific movements or tasks. It also communicates with other subsystems to trigger relevant actions, such as picking up or placing items;
- **Error Handling.** The main control module is equipped with error handling mechanisms to detect and respond to anomalies or disruptions in the robot's operation. This ensures robustness and resilience in the face of unexpected events.

Types of Main Control Modules:

- Centralized Control. In centralized control systems, a single main control module oversees the entire fleet of robots within the warehouse. This approach is effective for tasks requiring centralized coordination and decision-making;
- Distributed Control. In distributed control systems, each robot has its own main control module, allowing for more autonomous decision-making at the robot level. This is suitable for scenarios where robots need to adapt to local conditions independently.

Technologies Utilized:

- Advanced Processing Units. Main control modules leverage powerful processors capable of handling complex algorithms and decision-making tasks in real-time;
- Communication Protocols. To facilitate communication with other modules, the main control module employs various communication protocols, such as CAN (Controller Area Network) or Ethernet;
- Artificial Intelligence (AI). AI technologies, including machine learning algorithms, may be integrated into the main control module for adaptive decision-making and continuous improvement based on experience;
- Sensor Integration. The main control module interfaces with a variety of sensors, including Lidar, cameras, and proximity sensors, to gather comprehensive information about the robot's surroundings.

In essence, the Main Control Module acts as the nerve center of the warehouse material handling robot, enabling it to navigate, manipulate objects, and interact with the environment autonomously and efficiently. Its capabilities and technologies employed are pivotal in ensuring the successful integration of robotics into warehouse logistics.

Artificial Intelligence (AI) is a transformative technology that holds immense potential for optimizing the performance and capabilities of warehouse material handling robots. When integrated into the main control module, AI, particularly machine learning algorithms, enables adaptive decision-making and continuous improvement based on the robot's experience and the evolving dynamics of the

warehouse environment.

Purpose and Functionality: The integration of AI into the main control module serves several essential purposes:

- **Adaptive Decision-Making:** AI algorithms empower the robot to make adaptive decisions based on real-time data and changing conditions. This enhances the robot's ability to respond dynamically to various situations, improving overall efficiency and flexibility;
- **Continuous Improvement:** AI enables the robot to learn from its experiences and interactions within the warehouse environment. Through continuous learning, the robot can refine its decision-making processes, adapt to evolving scenarios, and optimize its performance over time;
- **Autonomous Problem-Solving:** AI equips the robot with the capability to autonomously identify and address challenges. This includes navigating around obstacles, optimizing path planning, and dynamically adjusting to variations in the warehouse layout.

Types of AI in Warehouse Material Handling:

- **Machine Learning (ML).** ML algorithms enable the robot to learn patterns and correlations from data without explicit programming. This includes supervised learning for task-specific training and unsupervised learning for discovering hidden patterns within the data;
- **Reinforcement Learning.** Reinforcement learning allows the robot to learn optimal decision-making strategies by receiving feedback in the form of rewards or penalties based on its actions. This is particularly valuable for dynamic environments where the robot must adapt and learn from experience;
- **Computer Vision.** Computer vision, a subset of AI, enables the robot to interpret and understand visual information from cameras and sensors. This is critical for tasks such as object recognition, navigation, and obstacle avoidance.

Technologies Utilized:

- Neural Networks. Neural networks, inspired by the human brain, are foundational to many AI applications. Deep learning, a subset of neural networks, allows the robot to process complex data and extract meaningful insights;
- Natural Language Processing (NLP). In scenarios where human-robot interaction involves language, NLP technologies can be employed. This allows the robot to understand and respond to verbal or written commands;
- Predictive Analytics. AI algorithms can analyze historical data to make predictions about future events or trends within the warehouse. This is valuable for anticipating demand patterns and optimizing material handling strategies;
- Cluster Analysis. For tasks such as inventory management, cluster analysis can be employed to categorize and group similar items, helping the robot make informed decisions about storage and retrieval.

Integration with Robot Control System: The integration of AI into the main control module requires seamless collaboration with other control modules, such as motion control, obstacle avoidance, and navigation. The AI algorithms influence decision-making processes, task prioritization, and overall robot behavior.

Real-world Applications:

- Dynamic Path Planning. AI enables the robot to dynamically adjust its path planning based on real-time data, ensuring optimal routes and avoiding congested areas;
- Optimized Material Handling. AI algorithms can optimize the robot's approach to material handling tasks, including efficient picking and placing of items based on historical data and current demand;
- Predictive Maintenance. AI can predict potential failures or maintenance needs by analyzing sensor data, allowing for proactive maintenance scheduling and minimizing downtime;
- Adaptive Response to Environmental Changes. The robot can adapt to changes in the warehouse layout, item placements, or operating conditions, ensuring robust performance in dynamic environments.

In summary, the integration of AI, particularly machine learning algorithms,

into the main control module enhances the warehouse material handling robot's adaptability, decision-making capabilities, and overall performance. This technology empowers the robot to learn, evolve, and continuously improve its efficiency and effectiveness in diverse warehouse scenarios.

The Motion Control Module is a critical component of the warehouse material handling robot, responsible for governing the robot's movements with precision and efficiency. It manages various aspects of motion, including linear walking, turning, and control of the motors associated with driving and lifting mechanisms.

Purpose and Functionality: The primary purpose of the Motion Control Module is to ensure controlled and coordinated movement of the robot within the warehouse environment. It translates high-level commands generated by the main control module into specific instructions for the robot's actuators and motors. The key functionalities of the motion control module include:

- **Linear Walking and Turning.** The module coordinates the robot's linear movements and turning actions. This involves controlling the speed and direction of the robot as it navigates through the warehouse, ensuring smooth and precise motion;
- **Motor Control.** The motion control module manages the operation of key motors within the robot, including the brushless DC motors used for driving and the lifting motor in the lifting mechanism. It regulates the speed, direction, and acceleration of these motors to execute different tasks;
- **Lifting Mechanism Control.** In robots equipped with lifting mechanisms, the motion control module is responsible for controlling the lifting motor. This allows the robot to raise and lower its lifting mechanism as needed for tasks like picking up or placing items;
- **Speed and Direction Control.** The module ensures accurate control of motor speed and direction during robot movement. This involves dynamically adjusting these parameters based on the robot's operational requirements and the environment.

Types of Motion Control Modules:

- Closed-Loop Motion Control. In closed-loop systems, feedback from sensors, encoders, or other sources is used to continuously adjust and optimize the robot's motion. This approach enhances precision and adaptability, particularly in dynamic environments;
- Open-Loop Motion Control. Open-loop systems operate based on pre-programmed commands without continuous feedback. While simpler, they may be suitable for tasks with well-defined and predictable motion paths.

Technologies Utilized:

- Motor Controllers. Motion control modules integrate motor controllers to regulate the speed, torque, and direction of motors accurately. These controllers may use Pulse Width Modulation (PWM) techniques for fine-tuned control;
- Encoder Feedback Systems. Encoders provide feedback on the position and speed of motors, allowing the motion control module to adjust motor parameters in real-time for precise control;
- Inertial Measurement Units (IMUs). IMUs, including accelerometers and gyroscopes, may be employed to measure the robot's orientation and acceleration, aiding in motion control and stability;
- Proximity Sensors. Proximity sensors help detect obstacles or changes in the environment, allowing the motion control module to adjust the robot's path and avoid collisions;
- PID Control Algorithms. Proportional-Integral-Derivative (PID) control algorithms are often used for motor speed and position control, offering a robust method for maintaining desired motion characteristics.

In summary, the Motion Control Module is instrumental in orchestrating the robot's movements, translating high-level commands into precise control of motors and actuators. The choice of motion control technologies influences the robot's agility, responsiveness, and overall performance in warehouse material handling tasks.

The Obstacle Avoidance Module is a vital component integrated into the warehouse material handling robot's control system. Its primary function is to detect

obstacles during the robot's movement and implement measures to prevent collisions and potential damage. The module enables the robot to take evasive actions, ensuring safe and efficient navigation within the warehouse environment.

Purpose and Functionality: The primary purpose of the Obstacle Avoidance Module is to enhance the robot's safety and prevent unintended collisions with obstacles in its path. This module is designed to:

- **Obstacle Detection.** Utilize various sensors to identify the presence of obstacles in the robot's proximity. Common sensors include ultrasonic sensors, infrared sensors, Lidar, or cameras;
- **Collision Prevention.** Implement real-time analysis of obstacle data to determine potential collision risks. The module assesses the spatial relationship between the robot and detected obstacles to prevent physical contact;
- **Evasive Actions.** Initiate appropriate evasive actions when obstacles are detected. These actions may include adjusting the robot's path, slowing down, stopping, or temporarily entering a waiting state until the obstacle is cleared;
- **Safe Navigation.** Enable the robot to navigate around obstacles, choosing alternative paths or routes to reach its destination without compromising safety or efficiency.

Types of Obstacle Avoidance Modules:

- **Reactive Obstacle Avoidance.** In reactive systems, the module responds to immediate obstacle detection by altering the robot's trajectory or stopping its movement. This approach is effective for real-time avoidance but may not optimize navigation efficiency;
- **Predictive Obstacle Avoidance.** Predictive systems anticipate potential obstacles based on historical data or predictive algorithms. This approach allows the robot to proactively plan its path, optimizing for both safety and efficiency.

Technologies Utilized:

- **Ultrasonic Sensors.** Ultrasonic sensors emit high-frequency sound waves and measure the time it takes for the waves to bounce back. Changes in the return time indicate the presence of obstacles;

- Infrared Sensors. Infrared sensors detect obstacles by measuring the reflection of infrared light. They are commonly used for short to medium-range obstacle detection;
- Lidar (Light Detection and Ranging). Lidar sensors use laser beams to create a detailed, three-dimensional map of the surroundings, allowing precise detection of obstacles and accurate distance measurement;
- Cameras. Vision-based systems use cameras to capture images and employ image processing algorithms to identify obstacles. Cameras provide rich visual data for obstacle recognition;
- Collision Avoidance Algorithms. Advanced algorithms, such as potential field algorithms or machine learning models, may be employed to analyze sensor data and determine optimal evasive actions.

Integration with Robot Control System: The Obstacle Avoidance Module is seamlessly integrated into the overall robot control system, collaborating with the Motion Control Module and the Main Control Module. When an obstacle is detected, the Obstacle Avoidance Module communicates with these modules to execute evasive actions and ensure the robot's safe navigation.

In summary, the Obstacle Avoidance Module is a critical safety feature that enhances the warehouse material handling robot's ability to navigate complex environments by detecting obstacles and taking proactive measures to prevent collisions. The selection of sensors and algorithms plays a pivotal role in the effectiveness of this module, contributing to the overall safety and efficiency of the robotic system.

The Navigation and Positioning Module is a fundamental component within the control system of a warehouse material handling robot. Its primary role is to facilitate the precise navigation and positioning of the robot within the handling environment. By leveraging external sensors and advanced technologies, this module ensures accurate location awareness, contributing to both the efficiency and safety of the robot's operations.

Purpose and Functionality: The Navigation and Positioning Module is

specifically designed to address the following objectives:

- Navigation. Provide the robot with the capability to autonomously navigate through the warehouse environment, avoiding obstacles, and reaching predefined destinations;
- Positioning. Accurately determine and communicate the robot's position within the warehouse, allowing for precise control and coordination of material handling tasks;
- Security. Enhance the security of robot handling by ensuring that the robot operates within predefined zones and follows safe paths, reducing the risk of collisions or unintended interactions.

Types of Navigation and Positioning Modules:

- Simultaneous Localization and Mapping (SLAM). SLAM-based modules create a real-time map of the environment while simultaneously determining the robot's position within that map. This approach is particularly useful for scenarios with dynamic environments;
- Beacon-Based Navigation. Utilizes fixed beacons or markers within the environment that emit signals. The robot uses these signals to triangulate its position and navigate accordingly;
- Visual Odometry. Relies on visual data from cameras to estimate the robot's motion and position. Visual odometry is effective for indoor navigation where distinct visual features can be identified;
- GPS-Based Navigation. In outdoor or large warehouse environments, GPS signals can be used to determine the robot's global position. However, this approach may have limitations in indoor or obstructed settings.

Technologies Utilized:

- Lidar (Light Detection and Ranging). Lidar sensors provide detailed 3D maps of the surroundings, enabling accurate navigation and obstacle avoidance;
- Cameras. Vision-based systems use cameras for visual odometry and identification of visual landmarks, contributing to precise positioning;

- Ultrasonic Sensors. Ultrasonic sensors help detect proximity to walls or obstacles, supporting navigation and preventing collisions;
- Inertial Measurement Units (IMUs). IMUs, including accelerometers and gyroscopes, aid in tracking the robot's orientation and movement, contributing to overall positioning accuracy;
- Radio Frequency Identification (RFID). RFID tags can be strategically placed within the warehouse, and the robot uses RFID readers to identify its position relative to these tags.

Integration with Robot Control System: The Navigation and Positioning Module seamlessly integrates with the overall robot control system, collaborating with the Main Control Module and Motion Control Module. It provides essential data on the robot's position, which influences decision-making and motor control for precise navigation.

Security Measures: The module often includes security features to enforce safe handling, such as geofencing to define permissible areas and alert systems to notify operators or other robots in case of deviations from the planned path.

In conclusion, the Navigation and Positioning Module is pivotal for ensuring the warehouse material handling robot's ability to navigate accurately and securely within its environment. By combining various sensor technologies and navigation strategies, this module contributes to the overall efficiency and safety of material handling operations.

The Wireless Communication Module is a crucial component of the warehouse material handling robot's control system, serving as the communication link between the robot and the warehouse management system (WMS). Its primary role is to facilitate seamless data transmission, enabling the exchange of task instructions and real-time feedback information, including the robot's position and status.

Purpose and Functionality: The Wireless Communication Module is designed to fulfill several key functions within the context of warehouse material handling:

- Data Transmission. Establish a reliable and high-speed communication link between the robot and the warehouse management system for the exchange of data;

- Task Instructions. Receive task instructions from the warehouse management system, including information about material handling tasks, destinations, and priorities;
- Feedback Information. Transmit real-time feedback information back to the warehouse management system. This includes the robot's current position, operational status, and any relevant data collected during the execution of tasks.

Types of Wireless Communication Modules:

- Wi-Fi (Wireless Fidelity). Utilizes standard Wi-Fi protocols for wireless communication. Wi-Fi modules offer high data transfer rates and are suitable for environments with established Wi-Fi networks;
- Bluetooth. Bluetooth modules are commonly used for short-range communication. They are suitable for scenarios where the robot needs to communicate with nearby devices, such as sensors or other robots;
- RFID (Radio Frequency Identification). RFID technology, often used for identification and tracking, can also serve as a communication module. RFID tags on the robot can be read by RFID readers in the warehouse;
- 5G/4G (Mobile Networks). In situations where a robot needs to operate over a larger area or outdoors, it can use 5G or 4G networks for wireless communication. This is particularly beneficial for real-time data transmission in expansive warehouse environments.

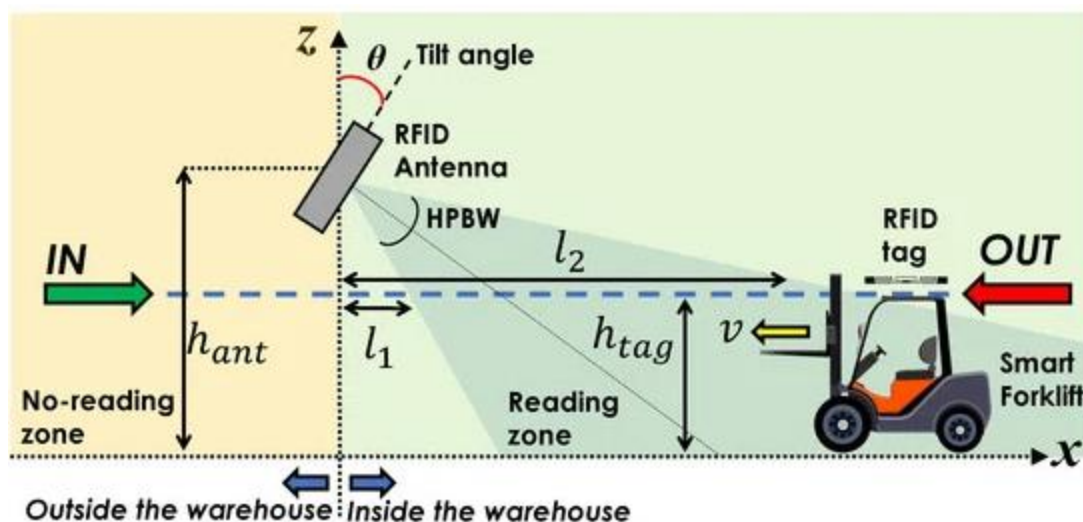


Figure 1.10 – RFID technology in the warehouse

Technologies Utilized:

- Wireless Protocols. The module employs various wireless communication protocols, such as TCP/IP or MQTT, to establish reliable connections and facilitate data exchange;
- Antenna Systems. Antennas are employed to enhance signal strength and coverage, ensuring stable communication between the robot and the warehouse management system;
- Encryption and Security Protocols. To protect sensitive data during transmission, encryption protocols and security measures, such as SSL/TLS, may be implemented to safeguard communication against unauthorized access;
- Quality of Service (QoS) Mechanisms. QoS mechanisms may be utilized to prioritize data packets, ensuring that critical information, such as task instructions or emergency signals, is transmitted with higher priority and minimal latency.

Integration with Robot Control System: The Wireless Communication Module seamlessly integrates with the overall robot control system, working in tandem with the Main Control Module to ensure that task instructions are received, and feedback information is transmitted accurately and promptly.

Real-time Monitoring and Control: The module enables real-time monitoring and control of the robot's activities by providing the warehouse management system with continuous updates on the robot's position, status, and task execution progress.

Fault Tolerance and Redundancy: In robust systems, redundancy and fault tolerance mechanisms may be implemented within the Wireless Communication Module to ensure continuity of communication even in the presence of network disruptions or hardware failures.

In summary, the Wireless Communication Module plays a pivotal role in establishing a reliable and efficient communication link between the warehouse material handling robot and the warehouse management system. Its capabilities influence the overall responsiveness, coordination, and effectiveness of the robot's operations within the warehouse environment.

1.3.2 Navigation and positioning mode selection

Navigation and positioning technology is the primary concern in the development of mainstream automated guided vehicles (AGVs). Currently, there are six main navigation methods for AGVs: Electromagnetic induction guidance: This involves embedding guide cables into the ground and installing induction coils on the AGV. When the coils are energized, they produce electromagnetic induction with the guide cables to achieve guidance. Advantages include mature technology, low cost, hidden guide cables, resistance to damage, and stable magnetic signal retention. However, it is difficult to change or maintain the cables, making it unsuitable for narrow storage environments. Magnetic tape guidance: This method involves laying magnetic tapes or nails on the ground to replace buried metal wires. The AGV is equipped with a magnetic sensor to receive the magnetic induction signals on the ground and detect the intensity of the magnetic field of the tapes, comparing and analyzing the deviation to control the AGV's direction for guidance. Advantages include low manufacturing costs, simple tape laying, flexible path modification, and independence from light interference. However, it can damage the warehouse's original working environment and is susceptible to mechanical damage from hard objects on the ground. Long-term exposure to dusty environments can also affect guidance precision.

Optical guidance: This method involves laying colored tapes or other colored substances on the ground and using an optical sensor to collect the colored band's information to recognize the path's color deviation, thus achieving AGV guidance.

Advantages include flexible use, low cost, easy path modification and maintenance, and only requiring surface application. However, it may also be susceptible to environmental damage and affects precision in dusty environments. It is sensitive to the stain and mechanical wear of the ribbon, so the degree of reliance of this method is low. When the reflective tape is stained. It will lead to poor reliability and low precision, and the external reflective light band is highly influenced by the light shadow, which is a reference to the storage environment machine. There are too many robots, and the image processing effect is poor due to

the frequency and complexity of robot motion.

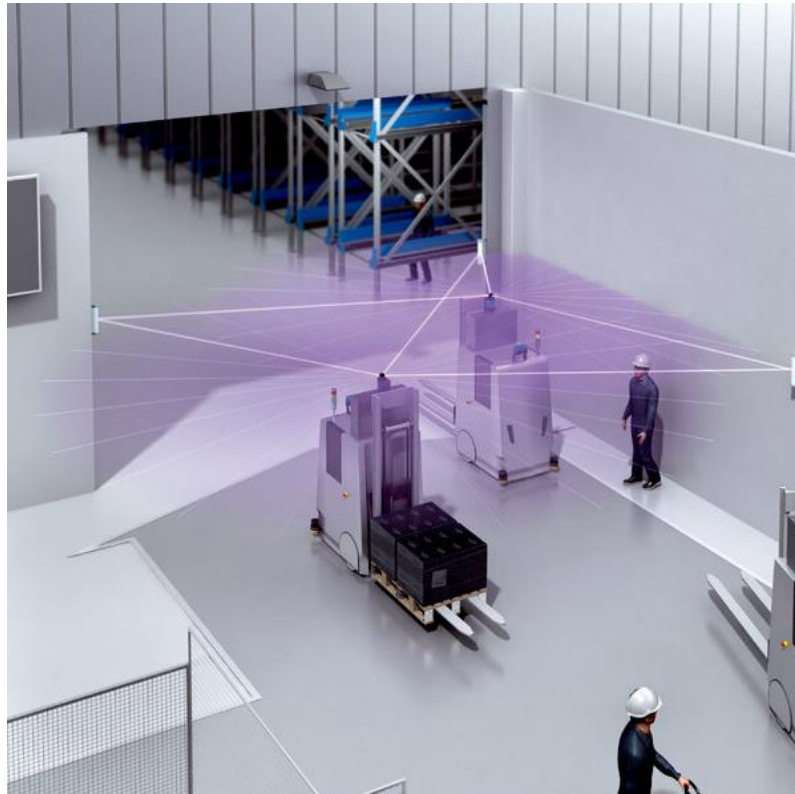


Figure 1.11 – Optical AGV Guidance

Laser guide is through continuous geometric operations to achieve the location of the determination. The robot transmits senses through lasers. The device emits a continuous laser beam and collects the laser beam of different angles reflected back by the reflector plate to determine the machine. The driver should be the front position, navigation direction W and the route to be traveled. Advantages: technology to advance, positioning precision is higher, yes. The ground will not cause a shadow, the path is easier to change and maintenance, do not need to lay on the ground receiving material; Missing point: make use of Bengao, the difficulty of opening, the sensor and the reflection plate can not exist in the obstacle.

Visual guidance is to process the collected environmental image information through the camera and extract the feature information. The guide is completed by the difference of the map, which is more advanced than the optical guide with color bands. Advantage: Setup and installation. The matching ratio is simple, W can be

used as a road sign with different shape plan positioning, road sign is simple, do not need to be on the ground. Laying routes on the upper berth; Missing point: the recognition technology is immature and sensitive to light, which is needed for the high-definition camera head at the same time The cost is higher.

Inertial navigation is to use the touron and speed meter to calculate the position information and speed information, from and determined Navigation parameters of robot in environment. After the robot is installed with the inertial component, the deviation signal is matched by the gyroscope. In order to determine the position and direction of the body of the machine, The final result is the guide t W to the robot. Advantages: the technology is advanced, the guide square is precise, the use of spirit, suitable for the use of fan. The circumference is wide. Missing point: vibration on the coder and spiral instrument and other inertial components of the shadow ring is very large, long distance from the use of cumulative error. It is mentioned in this text one.

The navigation method combining inertial navigation, two-dimensional code and gray phase is used for robots in the logistics storage environment. In use. In the driving process, the robot first collects two-dimensional code information through twodimensional code reader. the initial positioning information is obtained. Then, based on the W inertial device, the yaw Angle information during the robot's driving process is collected, and the encoder can get the speed of the robot, the knot and push people get machine movement. Be due to the yaw Angle information obtained by the robot is stored in the cumulative cumulative error over a long period of time, because of which the deviation needs to be corrected. Positive, so as to lead into the gray information line correction. Body method is: the first to obtain two - dimensional code information line initial positioning, detection. The landmark gray value is modified to correct the running error, and then inertial devices are used to cooperate with the coder to conduct navigation in a short time.

1.3.3 Communication mode selection

In practical logistics and warehousing, the environment where robots operate is a warehouse with a relatively large floor area, requiring multiple robots to collectively perform handling tasks. Operators use terminal devices in the warehouse

logistics to issue instructions and adjust and control the handling tasks of each robot. At the same time, the robots also need to send relevant status information to the upper computer. Therefore, wireless communication technology is used, mainly including ZigBee communication, Bluetooth communication, and Wi-Fi communication:

- ZigBee. ZigBee is a local area network protocol specified based on the IEEE 802.15.4 standard, mainly used for short-range transmission environments, and is a low-speed wireless communication technology. The advantage of the technology is that it consumes low power when used and has a low cost. It is widely used in intelligent home control, medical device control, and safety and fire protection;
- Bluetooth. Bluetooth is a wireless technology suitable for short-range data transmission. It is mainly used in communication devices such as mobile phones, hands-free headsets, and smart bracelets;
- Wi-Fi. Wi-Fi (Wireless Fidelity) is a wireless technology suitable for medium to long-distance data transmission. It carries information quite fast, covers a wide range, and can expand the usage area by adding relay products. Therefore, in this article, Wi-Fi communication is selected as the communication method between robots and upper computer terminals.

2 DESIGN OF WAREHOUSE HANDLING ROBOT CONTROL SYSTEM SOFTWARE

The efficient control and management of handling robots within a warehouse environment are contingent upon the efficacy of the accompanying software system. This subsection elucidates the paramount significance of a well-designed software system, focusing on key aspects that contribute to the overall effectiveness of handling robot operations.

A well-crafted software system serves as the linchpin for orchestrating the intricate movements and actions of handling robots within a warehouse. Its

significance can be distilled into several key facets:

Enhanced Operational Efficiency: A well-designed software system acts as the backbone, streamlining robot operations and minimizing processing times. Efficient algorithms and logical workflows contribute to increased overall efficiency.

Optimized Path Planning: Strategic path planning capabilities are pivotal in reducing travel distances and time. The software's ability to generate optimal paths enhances sustainability and cost-effectiveness by minimizing energy consumption.

Real-Time Decision-Making: The software's capacity for real-time data processing enables swift and informed decision-making. Instantaneous adjustments to robot routes in response to dynamic warehouse conditions ensure adaptability.

Collision Avoidance and Safety: Embedded collision avoidance mechanisms prevent unintended collisions between handling robots and obstacles, ensuring a safe working environment for both warehouse personnel and the robotic fleet.

User-Friendly Interface: An intuitive graphical user interface (GUI) facilitates seamless interaction between warehouse operators and the handling robot control system. Users can input coordinates, monitor robot movements, and respond promptly to system alerts.

Scalability and Adaptability: A robust software architecture allows for system scalability to accommodate changes in warehouse layouts and increasing robot fleets. Adaptive features ensure the system evolves alongside technological advancements.

Data Security and Integrity: Incorporating security measures is imperative to safeguard sensitive warehouse data. Ensuring data integrity and confidentiality is crucial for maintaining the integrity of warehouse operations.

Cost-Efficiency and Resource Optimization: Efficient control mechanisms contribute to resource optimization, reducing operational costs. Well-managed robot fleets minimize idle times, energy consumption, and maintenance expenses.

Integration with Overall Warehouse Management: Seamless integration with broader warehouse management systems enhances cohesion and synchronization across various operational aspects. The handling robot control system acts as a synergistic component within the larger warehouse ecosystem.

In summation, the design of the software system is a pivotal determinant in the success of handling robots in a warehouse setting. Its role in path planning, real-time decision-making, user interaction, and overall system integration underscores its critical importance in ensuring optimal control and management of handling robots.

2.1 Overall architecture design of system software

The warehouse handling robot control system software is designed with well-defined objectives aimed at optimizing warehouse operations, increasing efficiency, and ensuring effective handling robot management. The system focuses on key functionalities to meet specific goals:

Path Planning: The objective is to optimize robot trajectories, minimizing travel distances and time. This enhances operational efficiency and reduces energy consumption.

Obstacle Avoidance: The system aims to implement robust mechanisms for collision avoidance between handling robots and obstacles, ensuring safety, preventing damage to goods, and minimizing downtime.

Real-Time Control: Enabling real-time monitoring and control of handling robots facilitates prompt decision-making and responsiveness to dynamic warehouse conditions.

User-Friendly Interface: Providing an intuitive graphical user interface enhances user experience, streamlining interaction between warehouse operators and the control system.

Scalability and Adaptability: The software is designed to be scalable, accommodating changes in warehouse layouts and an expanding fleet of handling robots, future-proofing the system.

Data Security and Integrity: Robust security measures are implemented to ensure the confidentiality and integrity of warehouse data, safeguarding sensitive

information from unauthorized access or breaches.

Cost-Efficiency and Resource Optimization: The objective is to optimize resource allocation, minimize operational costs, and reduce idle times and energy consumption, contributing to overall cost-effectiveness.

Integration with Warehouse Management: Seamless integration with broader warehouse management systems enhances coordination and synchronization across various operational aspects.

These system objectives collectively contribute to the development of handling robot control system software that is robust, adaptable, and geared towards achieving optimal warehouse management and control.

The functional requirements of the warehouse handling robot control system software encompass a comprehensive set of features that are essential for effective system operation. These functionalities address core aspects crucial to the successful management and control of handling robots within the warehouse environment:

- **Robot Initialization.** The software should facilitate the initialization of handling robots, ensuring proper system recognition and communication;
- **Coordinate Input.** The system must provide user-friendly interfaces for inputting coordinates, allowing precise definition of robot start and finish positions;
- **Obstacle Handling.** Robust mechanisms for obstacle handling are required, including the ability to input obstacle coordinates, visualize obstacles on the interface, and implement algorithms for obstacle avoidance during path planning;
- **Path Visualization.** The software should incorporate features for visualizing calculated paths between robot start and finish positions. This includes real-time display and updates of the planned trajectory, highlighting areas with potential obstacles;
- **Coordinate Validation.** The system should perform validation checks on input coordinates to ensure they fall within specified ranges, preventing errors and enhancing user experience;

- Real-Time Monitoring. Real-time monitoring capabilities are essential, allowing operators to observe the current status and location of handling robots, as well as dynamically changing warehouse conditions;
- Trajectory Calculation. The software must implement algorithms for efficient trajectory calculation, considering factors such as path optimization, obstacle avoidance, and adherence to specified coordinates;
- User Notifications. The system should incorporate a notification mechanism to alert users in case of invalid coordinates, obstacles, or other critical events that may impact system performance;
- Integration with External Systems. Seamless integration with external systems, such as warehouse management software, is a requirement for enhanced coordination and data exchange;
- User Authentication and Authorization. Security features, including user authentication and authorization mechanisms, should be implemented to control access and safeguard sensitive system functionalities.

2.1.1 The overall workflow of the warehouse logistics robot software system

This section provides a comprehensive exploration of the warehouse logistics robot software system's workflow, covering initialization, user inputs, path planning, real-time monitoring, and adaptive functionalities. It aims to offer readers a clear understanding of how the software orchestrates the movement of handling robots within a warehouse setting.

The overall workflow of the warehouse logistics robot software system is as follows:

- power on and reset the system;
- initialize each module of the system;
- enter a dead loop and wait for a triggered interrupt;
- if the interrupt from serial port 0 is triggered, set the communication interrupt flag to 1, receive the control command sent by the upper computer, and then clear the communication interrupt flag;

- if the interrupt from serial port 1 is triggered, set the QR code interrupt flag to 1, receive the QR code information, and feedback the position information, then clear the QR code interrupt flag;
- if the interrupt from serial port 2 is triggered, set the inertial navigation interrupt flag to 1, receive the inertial navigation information, feedback the yaw information, and then clear the inertial navigation interrupt flag;
- if the infrared interrupt is triggered, set the infrared interrupt flag to 1, apply an emergency brake to the control motor, and then clear the infrared interrupt flag;
- if the time interrupt is triggered, execute the speed acquisition, speed control function, grayscale sensor information acquisition, direction control function, and state information sending function, and clear the time flag variable;
- during the motion control process, set the control period through the timer, execute the PWM total control to adjust the speed, and continue the loop. The interrupt functions of the timer, infrared, encoder, wireless communication module, QR code landmark, and inertial navigation are involved in the process.



QRCode



RFID Tag

Figure 2.1 – QR code and RFID Tag

As a practical development of the software part of the robot control system, I

chose to develop a program for determining the robot's trajectory from the initial point of its position to the final destination.

The software system serves as the backbone for coordinating the movement of handling robots within the dynamic environment of a warehouse. By ensuring a streamlined workflow, it significantly contributes to the optimization of warehouse logistics operations.

System Initialization: Upon launch, the software system undergoes a robust initialization process. This involves loading essential configurations, establishing connections with the hardware components of the handling robots, and preparing a solid foundation for subsequent operations.



Figure 2.2 – QR Floor Codes

User Interface Overview: The graphical user interface (GUI) serves as the primary means of interaction for users. It provides a visual representation of the warehouse space, incorporating key elements such as input fields, buttons, and graphical displays essential for user engagement.

Robot Start Coordinates Input: Users interact with the system to input the

initial coordinates of the handling robot. This input is subjected to validation mechanisms, ensuring accuracy and feasibility for subsequent path planning.

Robot Finish Coordinates Input: Similar to the start coordinates, users input the finishing coordinates of the handling robot. Stringent validation mechanisms are in place to guarantee precise and viable input, setting the stage for path planning.

Obstacle Coordinates Input: Users define obstacle coordinates within the warehouse space, contributing to a dynamic environment. To avoid conflicts with robot start and finish coordinates, the system incorporates checks and balances, ensuring a conflict-free path.

Path Planning and Visualization: The algorithmic process of calculating the optimal path from the start to finish coordinates is a pivotal aspect. The software system visualizes the calculated path, considering the presence of obstacles and adapting to warehouse constraints.

Real-Time Monitoring: Real-time monitoring is integrated to provide users with instant feedback on the robot's position, the presence of obstacles, and the ongoing path. This enhances situational awareness for effective control and decision-making.

Confirming Coordinates and Path Execution: Upon confirming the input coordinates, the system executes the calculated path with precision. This step ensures the seamless orchestration of the handling robot's movement based on the planned trajectory.

Path Adjustment and Replanning: The software system exhibits responsiveness to changes in coordinates during runtime. It dynamically adjusts the path to accommodate real-time alterations, showcasing its adaptability to evolving warehouse conditions.

In conclusion, the software system's overall workflow integrates user inputs, path planning, and real-time monitoring. This comprehensive orchestration ensures the optimal movement of handling robots, contributing significantly to the efficiency and effectiveness of warehouse logistics operations.

2.1.2 System Initialization Module Design

The System Initialization module within the warehouse logistics robot software system plays a pivotal role in setting the stage for efficient and error-free operations. This section delves into the design aspects and functionalities of the System Initialization module, elucidating its crucial contributions to the overall workflow.

The System Initialization module serves as the software's launchpad, orchestrating the necessary tasks to prepare the system for subsequent operations. It encompasses a series of steps that culminate in establishing a stable foundation for seamless user interactions.

Upon initiation, the System Initialization module takes charge of loading essential configurations. These configurations encompass parameters such as warehouse layout, permissible coordinates, and robot specifications. The module ensures that the software aligns with the specific requirements of the warehouse environment.

Integral to the initialization process is the establishment of robust connections with the hardware components of handling robots. This includes communication interfaces, sensor networks, and actuators.

To mitigate the risk of errors and enhance the reliability of subsequent operations, the System Initialization module incorporates stringent data validation mechanisms. It checks the consistency and correctness of loaded configurations, ensuring that the software operates within predefined constraints.

The warehouse environment is dynamic, and the System Initialization module addresses this by calibrating the software to adapt to varying conditions. This involves assessing factors such as lighting conditions, obstacle placement, and any changes in the overall warehouse layout. The module optimizes the system's parameters based on this calibration, enhancing adaptability.

Recognizing the potential for unforeseen circumstances during initialization, the module integrates robust error handling mechanisms. In the event of an error, the system provides informative and actionable reports to assist users or administrators in resolving issues promptly.

To facilitate troubleshooting and system analysis, the System Initialization module incorporates logging and audit trail features. It records key events, configurations, and interactions during the initialization process. This log serves as a valuable resource for diagnosing issues and tracking system performance.

The design of the System Initialization module ensures a methodical and reliable preparation of the warehouse logistics robot software system. By addressing key elements such as configuration loading, hardware connectivity, data validation, environment calibration, error handling, and logging, this module lays the groundwork for subsequent stages, fostering a robust and resilient software ecosystem.

2.2 Software Design for Motion Control Module

The Wang Quanquan Handling Robot Control System is a sophisticated software application designed to manage and control handling robots within a warehouse environment. Tailored to meet the specific needs of efficient material handling, this program serves as a crucial component in the broader system for orchestrating robot movements and optimizing warehouse operations.

Purpose and Functionality: The primary purpose of the Wang Quanquan Handling Robot Control System is to streamline the movement and coordination of handling robots within a warehouse setting. It provides an intuitive and feature-rich interface for operators to input and manage key parameters, ensuring precise control over the robots' paths and interactions with obstacles.

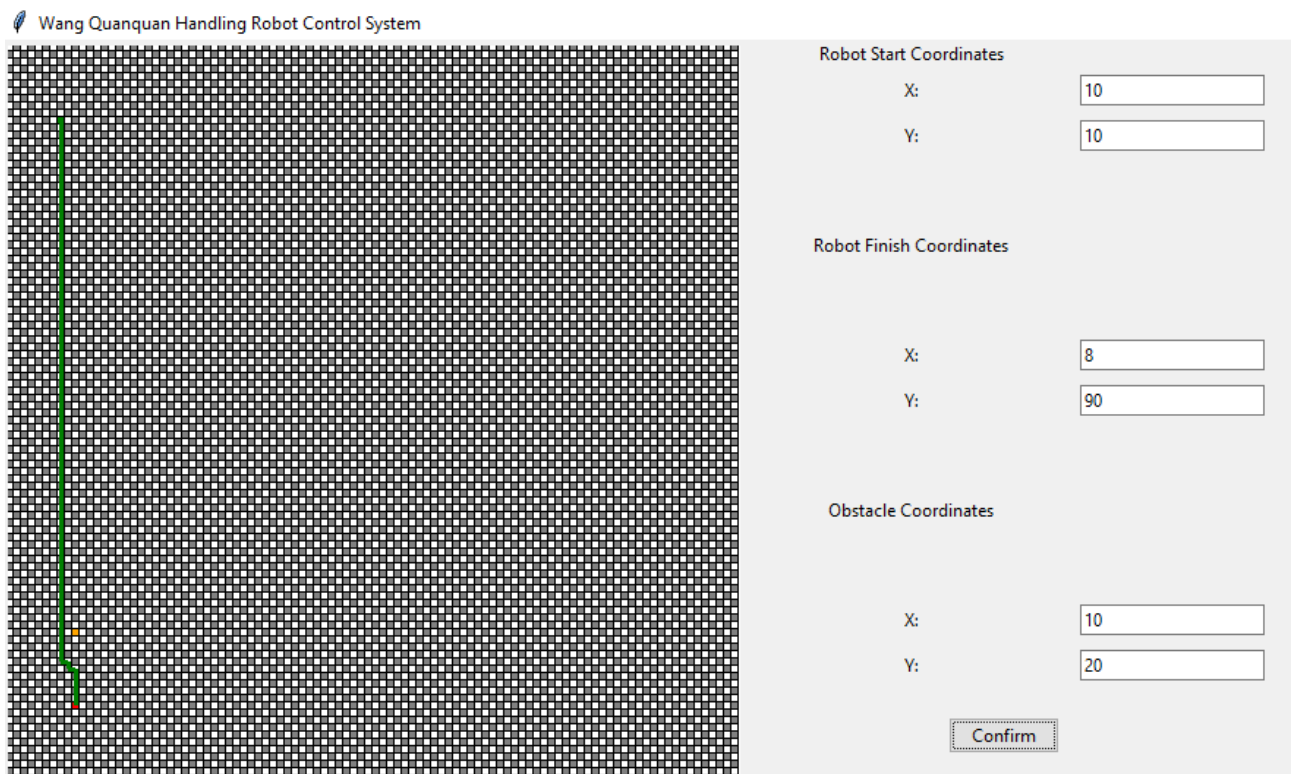


Figure 2.3 – User Interface of Wang Quanquan Handling Robot Control System

Key Features:

- Robot Initialization. The system enables the seamless initialization of handling robots, establishing communication and recognition within the warehouse network;
- Coordinate Input. Users can effortlessly input start and finish coordinates for handling robots, facilitating precise positioning for various tasks;
- Obstacle Handling. The program incorporates robust mechanisms for obstacle management, allowing users to input obstacle coordinates and implementing advanced algorithms for obstacle avoidance during path planning;
- Path Visualization. Real-time visualization of calculated paths between start and finish positions enhances situational awareness. The interface displays trajectories, highlighting potential obstacles and dynamically updating as conditions change;
- Coordinate Validation. The system performs thorough validation checks on input coordinates, ensuring they fall within specified ranges to prevent errors and optimize robot movements;

- Real-Time Monitoring. Operators benefit from real-time monitoring capabilities, enabling them to track handling robots' current status, locations, and respond promptly to changing warehouse conditions;
- Trajectory Calculation. Efficient trajectory calculation algorithms optimize paths, considering factors like path length, obstacle avoidance, and adherence to specified coordinates;
- User Notifications. The system includes a notification mechanism to alert users to critical events, such as invalid coordinates or obstacles, enhancing overall system reliability;
- Integration with External Systems. Seamless integration with external warehouse management systems ensures a cohesive flow of data and coordination between the Handling Robot Control System and broader warehouse operations;
- User Authentication and Authorization. Security measures, including user authentication and authorization, safeguard system functionalities and control access to critical features.

Overall Workflow:

- Initialization. Operators initialize handling robots within the system;
- Coordinate Input. Precise start and finish coordinates are input for robot tasks;
- Obstacle Management. Users can define obstacle coordinates and implement obstacle avoidance strategies;
- Path Planning. The system calculates optimal trajectories considering obstacles and specified coordinates;
- Real-Time Monitoring. Continuous monitoring of robot status and path execution;
- Notifications. Alerts for critical events and potential issues;
- Integration. Seamless integration with broader warehouse management systems.

The Wang Quanquan Handling Robot Control System thus stands as a comprehensive solution, enhancing the efficiency and precision of handling robot

operations within a warehouse environment.

The decision to utilize Python for the development of the Wang Quanquan Handling Robot Control System was rooted in several considerations, taking into account the unique requirements of the application and the broader context of warehouse management. Below are the reasons for choosing Python, along with an overview of the libraries, classes, commands, and other components employed in the development process:

Advantages of Python:

- Readability and Simplicity. Python's clean and readable syntax significantly contributed to the program's clarity, making it easier to understand, maintain, and modify. This simplicity aligns with the need for an intuitive interface for operators;
- Rich Ecosystem of Libraries. Python boasts an extensive ecosystem of libraries that expedite development. For the Wang Quanquan Handling Robot Control System, libraries such as Tkinter, a standard GUI toolkit, and networkx for graph-based algorithms played pivotal roles;
- Cross-Platform Compatibility. Python's cross-platform compatibility ensures that the Handling Robot Control System can be deployed on various operating systems without major modifications, enhancing its versatility and accessibility;
- Community Support. Python's large and active community provides a wealth of resources, documentation, and community-driven support. This was particularly beneficial for troubleshooting and refining the system;
- Integration Capabilities. Python facilitates seamless integration with other languages and systems. This feature is crucial for ensuring the Wang Quanquan Handling Robot Control System can communicate effectively with external warehouse management systems.

Utilized Components and Commands:

- Tkinter Library. Tkinter was employed for the graphical user interface (GUI). Its simplicity and integration with Python make it an ideal choice for creating interactive interfaces;
- Networkx Library. The Networkx library was utilized for graph-based algorithms, especially for calculating optimal paths considering obstacles. This library provides robust graph structures and algorithms;
- Core Classes. The program is structured around core classes representing the robot, obstacles, and the overall system. This object-oriented approach enhances modularity and facilitates code organization;
- Input Validation. Input validation commands were implemented to ensure that user-provided coordinates fall within the specified ranges, minimizing the risk of errors and enhancing the system's reliability;
- Real-Time Monitoring Commands. Real-time monitoring of robot status and path execution involved commands for continuous data updates and visualization;
- Security Commands. User authentication and authorization were implemented using relevant Python commands to enhance the overall security of the system.

Challenges and Considerations:

While Python presented numerous advantages, it's essential to acknowledge potential challenges, including:

- Performance Limitations. Python, being an interpreted language, may exhibit performance limitations compared to compiled languages. However, for the Handling Robot Control System, where real-time requirements are moderate, this was deemed acceptable;
- Global Interpreter Lock (GIL). Python's GIL could potentially impact multithreading performance. Mitigation strategies, such as using multiprocessing for parallelism, were considered where necessary;

- Library Dependencies. Managing dependencies, especially with external libraries, requires careful consideration to ensure compatibility and minimize potential issues during deployment.

In summary, Python's readability, extensive library support, and versatility made it a suitable choice for the Wang Quanquan Handling Robot Control System. The employed libraries, classes, and commands were carefully selected to address specific functionalities, ensuring a robust and efficient implementation of the control system.

3 WAREHOUSE LOGISTICS HANDLING ROBOT CONTROL SYSTEM TEST

This section outlines the comprehensive testing procedures for the hardware components of the warehouse logistics handling robot control system. It includes tests for the main control module, motion control module, and the critical navigation and positioning module. The breakdown into specific tests ensures a systematic approach to debugging and validating the functionality of each component, contributing to the overall reliability of the control system.

3.1 Hardware debugging of control system

3.1.1 Main control module test

The Main Control Module serves as the central intelligence of the warehouse logistics handling robot, governing decision-making and overseeing the coordination of various subsystems. This section outlines the specific tests conducted to verify the functionality of the Main Control Unit.

Power-On Self-Test (POST).

Objective: ensure the Main Control Unit initializes correctly and performs a comprehensive self-check upon power-up.

Test Steps:

- power on the robot's main control unit;
- monitor system boot sequence for any error messages or abnormal behavior;
- verify the initialization of key components, including processors, memory, and internal communication buses;
- confirm that the control unit transitions to an operational state without errors;
- criteria for Success. The Main Control Unit successfully completes the power-on self-test without encountering critical errors, and the system transitions to an operational state ready for further testing.

Communication Protocols Check.

Objective: validate the communication interfaces and protocols utilized by the Main Control Unit for seamless interaction with other modules and external systems.

Test Steps:

- check the integrity of wired and wireless communication interfaces (e.g., Ethernet, Wi-Fi, Bluetooth);
- verify the correct configuration of communication protocols such as MQTT, ROS (Robot Operating System), or any proprietary protocols;
- transmit test messages between the Main Control Unit and peripheral modules to ensure bidirectional communication;
- evaluate error handling mechanisms for communication disruptions.

Criteria for Success. The Main Control Unit demonstrates reliable communication with peripheral modules, and error-handling mechanisms effectively manage communication disruptions.

The Main Control Module Test ensures the robustness and reliability of the central intelligence governing the warehouse logistics handling robot. Through the

Power-On Self-Test and Communication Protocols Check, this test segment aims to confirm that the Main Control Unit initializes correctly, communicates effectively with other components, and is prepared for subsequent testing phases.

The evaluation of centralized decision-making is a critical aspect of testing the Main Control Module. This process assesses the module's ability to receive, process, and act upon commands effectively, as well as its capability to handle errors gracefully.

Command Reception and Processing.

Objective: verify the Main Control Module's competence in receiving and processing commands, ensuring that it interprets directives accurately and initiates appropriate actions.

Test Steps:

- simulate various command inputs representing common operational scenarios;
- monitor the Main Control Module's response to each command, observing the initiation of corresponding actions;
- evaluate the speed and accuracy of command interpretation and execution;
- introduce variations in command sequences to test the system's adaptability to dynamic instructions.

Criteria for Success. The Main Control Module consistently interprets commands correctly, executes corresponding actions promptly, and demonstrates adaptability to dynamic scenarios.

Objective: assess the Main Control Module's ability to detect, report, and manage errors effectively, ensuring the overall robustness and reliability of the control system.

Test Steps:

- introduce intentional errors in command inputs or simulate fault scenarios;
- observe the Main Control Module's response to error conditions, including error messages, alerts, and any automatic corrective actions taken;

- evaluate the system's ability to recover from errors and resume normal operation;
- test the logging and reporting mechanisms for error events.

Criteria for Success. The Main Control Module promptly detects and appropriately responds to errors, providing clear and informative error messages, implementing corrective actions when possible, and maintaining the ability to recover from fault conditions.

The Evaluation of Centralized Decision-Making focuses on ensuring that the Main Control Module effectively processes commands, initiates actions accurately, and exhibits robust error handling mechanisms. These tests collectively contribute to the reliability and resilience of the centralized decision-making processes within the warehouse logistics handling robot control system.

3.1.2 Motion control module test

The Motion Control Module is responsible for the precise control of the robot's movements. This section outlines the tests conducted to validate the functionality of the Motion Control System, including motor calibration and path planning and execution.

Validation of Motion Control System.

Motor Calibration.

Objective: ensure the accurate calibration of motors to achieve consistent and precise robot movements.

Test Steps:

- initiate the motor calibration process;
- monitor the calibration procedure for each motor, including feedback from encoders or sensors;
- verify that each motor is calibrated to the specified parameters;
- test the robot's movements in controlled conditions to assess the accuracy of calibrated motors.

Criteria for Success. The Motion Control System successfully calibrates all motors, resulting in accurate and consistent movements within specified tolerances.

Path Planning and Execution.

Objective: validate the ability of the Motion Control System to plan and execute robot movements along predefined paths.

Test Steps:

- input predefined paths or trajectories for the robot;
- monitor the system's generation of motion plans based on the input paths;
- execute the robot's movements along the specified paths, considering factors such as speed, acceleration, and precision;
- introduce variations in the input paths to assess the system's adaptability to dynamic scenarios.

Criteria for Success: The Motion Control System successfully plans and executes robot movements along predefined paths, demonstrating accurate control, adaptability, and responsiveness to dynamic input.

The Motion Control Module Test focuses on validating the Motion Control System's capability to achieve accurate motor calibration and execute precise robot movements along predefined paths. These tests contribute to the overall reliability and performance of the warehouse logistics handling robot, ensuring that its motion control functions operate seamlessly in diverse operational scenarios.

Testing Algorithms for Map Interpretation.

Validate the effectiveness of navigation algorithms in interpreting and utilizing environmental maps for autonomous navigation.

Test Steps:

- provide the system with predefined maps of the warehouse environment;
- evaluate the algorithm's ability to interpret map data and generate accurate representations of the surroundings;
- test the system's response to variations in map complexity, including different warehouse layouts, obstacles, and landmarks;
- assess the algorithm's adaptability to changes in the environment, such as the addition or removal of obstacles.

Criteria for Success. The Navigation and Positioning Module demonstrates

robust map interpretation algorithms, accurately representing the warehouse environment and adapting to dynamic changes.

Evaluating Localization Accuracy.

Objective: assess the accuracy of the system in determining the robot's position within the warehouse environment.

Test Steps:

- initiate the robot in various locations within the warehouse;
- monitor the system's localization algorithms as the robot navigates through the environment;
- validate the accuracy of the system's estimates of the robot's position in real-time;
- introduce challenges such as varying lighting conditions or reflective surfaces to test the resilience of localization algorithms.

Criteria for Success. The Navigation and Positioning Module achieves high localization accuracy, providing real-time and reliable information about the robot's position within the warehouse.

3.1.3 Commissioning a navigation and positioning module

Commissioning the Navigation and Positioning Module involves thorough tests to validate navigation algorithms, map interpretation, and localization accuracy. These tests ensure that the robot can autonomously navigate the warehouse environment with precision and reliability, laying the foundation for effective and efficient material handling operations.

Positioning System Calibration.

The accuracy of the positioning system is critical for the successful navigation and operation of the warehouse logistics handling robot. This section outlines the calibration tests for the various sensors and the integration of the Positioning System with external positioning technologies.

Calibration of Sensors (e.g., Lidar, Cameras).

Objective: calibrate individual sensors to ensure accurate and synchronized data acquisition for precise robot positioning.

Test Steps:

- initiate calibration processes for each sensor, including Lidar and cameras;
- validate the alignment and synchronization of data streams from different sensors;
- test the accuracy of distance measurements for Lidar sensors and the quality of image capture for cameras;
- assess the impact of calibration on the overall accuracy of the robot's perceived environment.

Criteria for Success. Sensor calibration results in accurate and synchronized data, contributing to the precision and reliability of the robot's positioning system.

Integration with GPS or Indoor Positioning Systems.

Objective: integrate the robot's Positioning System with external technologies such as GPS or Indoor Positioning Systems to enhance overall localization accuracy.

Test Steps:

- integrate GPS receivers or Indoor Positioning System modules into the robot's control system;
- validate the communication and data exchange between the Positioning System and external positioning technologies;
- test the robot's ability to seamlessly transition between indoor and outdoor environments, if applicable;
- evaluate the impact of external positioning data on the overall accuracy of the robot's global position.

Criteria for Success. The integration with external positioning technologies enhances the robot's overall localization accuracy and allows for reliable operation in varied environments.

Positioning System calibration tests focus on ensuring the accuracy and synchronization of sensor data, and successful integration with external positioning technologies. These tests contribute to the overall reliability of the robot's autonomous navigation and positioning capabilities within the warehouse environment.

3.2 Control system software debugging

Software debugging is a critical phase in the development process, aimed at identifying and resolving defects or issues within the codebase. This intricate process plays a pivotal role in ensuring the reliability, functionality, and performance of the warehouse logistics handling robot control system software. The following subsections provide an overview of the background that necessitates debugging activities and the fundamental purposes it serves:

In the context of developing complex software systems such as the warehouse logistics handling robot control system, errors and defects inevitably arise during the coding phase. These issues may range from syntactical errors and logical flaws to runtime bugs that impact the system's stability and functionality. The warehouse setting introduces additional complexities, such as the need for precise navigation, obstacle avoidance, and real-time decision-making, making debugging an indispensable part of the software development lifecycle.

Purpose of Software Debugging.

The primary purpose of software debugging is to ensure the creation of a reliable and error-free control system for warehouse handling robots. The debugging process serves the following crucial objectives:

- Defect Identification and Localization. Debugging helps pinpoint the exact location and nature of defects within the source code. This is essential for efficient troubleshooting and resolution;
- Error Resolution. Once defects are identified, the debugging process facilitates the correction of errors, ensuring that the software operates according to its specifications;
- Enhancing Software Reliability. By systematically addressing bugs and glitches, debugging contributes to the overall reliability and robustness of the software, reducing the likelihood of malfunctions;

- Optimizing Performance. Debugging activities also focus on optimizing the performance of the control system. This involves identifying and rectifying bottlenecks or inefficiencies that may impact the system's responsiveness;
- Minimizing Development Delays. Timely debugging helps minimize delays in the development timeline. Rapid identification and resolution of issues contribute to meeting project deadlines;
- Ensuring User Satisfaction. Ultimately, the purpose of debugging is to deliver a software system that meets or exceeds user expectations. A thoroughly debugged control system enhances user satisfaction by providing a seamless and error-free robotic handling experience in a warehouse environment.

Unit testing and Test-Driven Development (TDD) are integral components of the software development process, playing a crucial role in ensuring the quality, reliability, and maintainability of the warehouse logistics handling robot control system software.

Importance of Unit Testing.

Unit testing involves the examination of individual units or components of the software to validate that each unit functions as intended. The importance of unit testing within the context of the control system includes:

- Error Detection at an Early Stage. Unit testing allows for the early detection of errors or defects within specific units of code. Identifying issues at this stage enables swift corrective actions and prevents the accumulation of errors in the overall system;
- Isolation of Defects. By testing individual units in isolation, unit testing facilitates the isolation of defects to specific components. This makes it easier to pinpoint and rectify issues without affecting the entire system;
- Enhanced Code Quality. Rigorous unit testing contributes to improved code quality by ensuring that each unit performs its intended function accurately. This results in a more reliable and cohesive software system;

- Facilitation of Refactoring. Unit tests provide a safety net for developers when refactoring or modifying existing code. Developers can confidently make changes, knowing that unit tests will quickly detect regressions if introduced.

TDD Methodology.

Test-Driven Development is a development approach that emphasizes writing tests before implementing the actual code. The TDD methodology involves the following key steps:

- Write a Test. Developers begin by writing a test that defines a new function or improvement of an existing function;
- Run the Test. The test is executed, and it is expected to fail initially since the corresponding code has not been implemented;
- Write the Code. Developers then write the minimum amount of code necessary to pass the test;
- Run All Tests. All tests, including the newly added one, are run to ensure that the new code did not break existing functionality;
- Refactor Code. If needed, developers refactor the code to improve its structure and maintainability while ensuring that all tests still pass.

Integration of Unit Testing into Development Workflow.

The integration of unit testing into the development workflow involves several key considerations:

- Continuous Integration. Unit tests are seamlessly integrated into the continuous integration (CI) process, ensuring that tests are automatically executed whenever changes are made to the codebase. This facilitates early detection of integration issues;
- Automation. Automated testing tools and frameworks are employed to streamline the execution of unit tests, allowing for frequent and consistent testing throughout the development cycle;
- Code Coverage. Monitoring code coverage helps assess the effectiveness of unit testing by indicating the percentage of code that is exercised by tests. Adequate code coverage ensures a comprehensive validation of the software;

- Collaboration. Unit testing is a collaborative effort involving developers, testers, and other stakeholders. Clear documentation of tests and their outcomes fosters effective collaboration and knowledge sharing.

By emphasizing unit testing and adopting the principles of Test-Driven Development, the control system's software development process ensures a robust, well-tested codebase that meets the required specifications and user expectations.

Logging and Tracing.

The implementation of logging and tracing mechanisms is a critical aspect of the warehouse logistics handling robot control system software, contributing to effective debugging, system monitoring, and bug identification.

Implementation of Logging Mechanisms.

Logging is the process of recording relevant information about the system's operation, events, and errors. The control system software incorporates robust logging mechanisms to capture valuable data for analysis and debugging. Key elements of the logging implementation include:

- Event Logging. Recording significant events, such as user inputs, system responses, and critical operations, provides a chronological record of the software's behavior;
- Error Logging. Logging error messages, stack traces, and exception details aids in identifying and addressing issues promptly. Error logs provide essential information for debugging and troubleshooting;
- Level-Based Logging. Implementing different logging levels (e.g., INFO, DEBUG, WARNING, ERROR) allows developers to control the granularity of logged information. This ensures that detailed logs are available during debugging while minimizing noise in production environments;
- Structured Logging. Employing a structured logging approach, such as using JSON or key-value pairs, enhances log readability and facilitates automated log analysis;

- Log Rotation and Retention. To manage log file sizes and prevent resource exhaustion, the system incorporates log rotation and retention policies. This ensures that logs are appropriately managed over time.

Effective Use of Tracing for Bug Identification.

Tracing involves capturing detailed information about the execution flow of the software. Tracing mechanisms are strategically employed to facilitate bug identification and resolution:

- Instrumentation. Code instrumentation is applied to selectively trace specific sections of the codebase. This allows developers to focus on relevant areas during debugging;

- Conditional Tracing. Developers can enable or disable tracing dynamically based on specific conditions, reducing overhead when detailed tracing is unnecessary;

- Integration with Debugging Tools. Tracing information is integrated with debugging tools, providing developers with comprehensive insights into the system's execution flow;

- Timestamps and Contextual Information. Tracing entries include timestamps and contextual information to establish the sequence of events and aid in correlating traces with specific actions or inputs;

- Trace Visualization. Visualization tools are utilized to present traces in a user-friendly manner, enabling developers to comprehend complex execution flows and identify irregularities.

By implementing robust logging mechanisms and leveraging effective tracing strategies, the control system software ensures that developers have access to detailed information for diagnosing issues, identifying bugs, and optimizing the overall system performance. These practices contribute to the creation of a resilient and maintainable software solution.

Dynamic Analysis Tools.

In the context of the warehouse logistics handling robot control system software, dynamic analysis tools play a pivotal role in assessing and optimizing system performance. This section delves into the implementation and utilization of

profilers and memory debugging tools.

Profilers and Performance Analyzers.

Profilers are instrumental in gauging the runtime behavior of the software, allowing developers to identify bottlenecks and areas for optimization. The control system software integrates the following profilers and performance analyzers:

- CPU Profilers. These tools monitor the execution time of different functions and methods, providing insights into CPU utilization. By identifying functions with high execution times, developers can optimize critical paths in the code;
- Memory Profilers. Memory usage is a crucial aspect of system performance. Memory profilers are employed to analyze memory allocation, detect memory leaks, and optimize memory-intensive operations;
- Code Coverage Tools. Evaluating code coverage ensures that all parts of the codebase are exercised during testing. Code coverage tools help identify areas that lack test coverage, aiding in the creation of comprehensive test suites;
- Execution Profilers. Profilers that capture the order and frequency of function executions assist developers in understanding the software's runtime behavior. This information is valuable for identifying redundant or inefficient code.

Memory Debugging Tools.

Effective memory management is paramount for the stability and reliability of the control system software. Memory debugging tools are utilized to identify and rectify memory-related issues:

- Memory Leak Detection. Tools that detect memory leaks by identifying allocated memory blocks that are not deallocated. This ensures that the software maintains efficient memory usage over extended periods;
- Heap Analysis Tools. Analyzing the heap memory allows developers to understand memory allocation patterns, detect fragmentation issues, and optimize memory allocation strategies;

- Static and Dynamic Analysis. Combining static and dynamic analysis tools helps identify potential memory-related vulnerabilities at both compile-time and runtime, enhancing the overall robustness of the software.

By incorporating profilers and memory debugging tools, the control system software undergoes thorough dynamic analysis, enabling developers to address performance bottlenecks, optimize memory utilization, and enhance the overall efficiency of the warehouse logistics handling robot control system. These tools contribute to the creation of a software solution that meets stringent performance requirements and delivers a seamless user experience.

Collaboration Tools.

Collaboration tools play a crucial role in the development and debugging process of the warehouse logistics handling robot control system software. This section explores the integration and utilization of version control systems, issue tracking systems, and collaborative debugging platforms.

Version Control Systems.

Version control systems are fundamental for managing and tracking changes to the source code throughout the software development lifecycle. The control system software incorporates the following version control practices:

- Git Integration. Git is employed as the primary version control system, enabling developers to track changes, collaborate seamlessly, and maintain a comprehensive history of code modifications;
- Branching Strategies. Leveraging branching strategies such as feature branching and release branching facilitates concurrent development of distinct features and ensures a stable codebase for deployment;
- Code Reviews. Implementing a systematic code review process through pull requests enhances code quality. Code reviews involve collaborative assessments of proposed changes, ensuring that modifications align with coding standards and best practices.

Issue Tracking Systems.

Efficient issue tracking is vital for managing and resolving software-related

concerns. The control system software incorporates the use of an issue tracking system for:

- Bug Tracking. The identification and tracking of software bugs are streamlined through an issue tracking system. This facilitates systematic debugging and resolution of reported issues;
- Feature Requests. Users and developers can submit feature requests through the issue tracking system, enabling the development team to prioritize and implement new functionalities based on user needs.

Collaborative Debugging Platforms.

Collaborative debugging platforms enhance communication and cooperation among developers, fostering a cohesive development environment. The control system software leverages the following collaborative debugging practices:

- Real-time Communication. Integration with real-time communication tools ensures swift and effective collaboration among developers. Chat platforms, video conferencing, and collaborative coding environments contribute to efficient issue resolution;
- Shared Debugging Sessions. Utilizing collaborative debugging tools allows developers to share debugging sessions. This facilitates joint problem-solving, with multiple developers examining and diagnosing issues concurrently;
- Knowledge Sharing. Collaborative platforms serve as repositories for knowledge sharing. Developers document debugging processes, solutions, and best practices, fostering a collective understanding of the software's intricacies.

By seamlessly integrating version control, issue tracking, and collaborative debugging tools, the control system software benefits from enhanced code management, efficient issue resolution, and a collaborative development environment. These tools collectively contribute to the creation of a robust and maintainable warehouse logistics handling robot control system.

Common Software Bugs in Control Systems.

Effective debugging is essential to identify and rectify common software bugs that may arise during the development of the warehouse logistics handling robot

control system. This section outlines prevalent software bugs and strategies for addressing them.

Syntax Errors and Typos:

- Identification. Syntax errors and typos are typically detected by the software compiler during the compilation phase. These errors often result from incorrect language syntax or misspelled keywords;
- Prevention and Resolution. Developers can minimize syntax errors through meticulous code writing and employing integrated development environments (IDEs) with syntax highlighting and error checking. Regular code reviews and testing further aid in identifying and resolving syntax-related issues.

Logical Errors in Algorithm Implementation:

- Identification. Logical errors occur when the algorithmic logic within the code is flawed, leading to incorrect program behavior. These errors may not be apparent during the compilation phase and often surface during runtime;
- Prevention and Resolution. Code reviews, unit testing, and employing debugging tools are effective strategies for identifying and addressing logical errors. The use of test cases helps validate algorithmic correctness, ensuring that the implemented logic aligns with the intended functionality.

Memory Leaks and Resource Management:

- Identification. Memory leaks arise when a program fails to release allocated memory, leading to gradual memory consumption. Proper resource management, including file closures and deallocation of dynamic memory, is crucial;
- Prevention and Resolution. Automated tools, such as memory profilers, assist in detecting memory leaks during development. Adhering to best practices for resource allocation and deallocation, coupled with systematic testing, helps prevent and address memory-related issues.

Concurrency and Multithreading Issues:

- Identification. Concurrency issues, such as race conditions and deadlocks, may occur when multiple threads access shared resources concurrently. These issues can result in unpredictable behavior and system instability;

- Prevention and Resolution. Thorough testing, including stress testing under varying load conditions, helps uncover concurrency issues. Implementing thread-safe programming practices, using synchronization mechanisms, and carefully managing shared resources mitigate the risk of multithreading problems.

By understanding and actively addressing common software bugs, developers enhance the reliability and robustness of the control system software for warehouse logistics handling robots. The combination of preventive measures, systematic testing, and efficient debugging practices contributes to the creation of a stable and high-performing software system.

Strategies for Efficient Bug Identification.

Efficient bug identification is paramount in ensuring the reliability and functionality of the warehouse logistics handling robot control system software. This section delves into strategies that facilitate the rapid and accurate detection of bugs during the development process.

Regression Testing for Bug Reproduction:

- Objective: regression testing involves re-executing previously conducted test cases to ensure that recent code modifications do not introduce new bugs or adversely impact existing functionalities;
- Implementation: developers perform regression testing by systematically running a suite of test cases that cover critical functionalities. This strategy aids in identifying unintended consequences of code changes and ensuring the system's overall stability.

Isolation of Bug-Prone Components:

- Objective: identifying bug-prone components involves singling out modules or sections of code that have historically demonstrated a higher likelihood of containing bugs;
- Implementation: by utilizing version control data, bug tracking reports, and historical project information, developers can identify components that frequently undergo modifications or encounter issues. Focused testing and code review efforts can then be directed toward these areas, improving bug detection efficiency.

Impact Analysis and Code Refactoring:

- Objective: impact analysis involves evaluating the potential repercussions of a bug fix or code modification on the overall system. Code refactoring focuses on restructuring existing code to enhance clarity, maintainability, and bug resilience;
- Implementation: developers conduct impact analyses to gauge the effects of bug fixes on interconnected components. This ensures that modifications do not inadvertently introduce new issues. Concurrently, code refactoring minimizes the occurrence of bugs by improving code quality and readability.

By employing these strategies, developers can streamline the bug identification process, fostering a more robust and resilient warehouse logistics handling robot control system software. The combination of regression testing, targeted component isolation, and thoughtful impact analysis contributes to the creation of reliable software that meets the demands of warehouse automation and material handling.

Real-world Debugging Scenarios.

Navigating real-world debugging scenarios is an integral part of refining the warehouse logistics handling robot control system software. This section delves into specific instances of debugging challenges, the identification of software bugs, and the valuable lessons learned throughout the debugging process.

Identification and Resolution of Specific Software Bugs:

- a) Scenario 1: Path Planning Anomalies:
 - bug Description: deviations observed in the robot's path planning algorithm, leading to suboptimal routes;
 - resolution: through extensive logging and dynamic analysis, developers identified inaccuracies in the cost function used for path planning. Adjustments were made to enhance the algorithm's efficiency;
- b) Scenario 2: Memory Leaks in Motion Control Module:
 - bug Description: gradual depletion of memory resources over prolonged system operation;

- resolution: employing memory debugging tools, developers identified and rectified memory leaks in the motion control module. Systematic checks for resource deallocation were implemented.

Lessons Learned from Debugging Challenges:

- Adopting Comprehensive Logging Practices. Lesson: the implementation of detailed logging mechanisms proved crucial in tracing the execution flow and identifying the root causes of anomalies. Enhancing logging practices became a priority for effective debugging;

- Proactive Unit Testing for New Features. Lesson: introduction of Test-Driven Development (TDD) practices for newly introduced features reduced the likelihood of undetected bugs. Early and continuous testing during development emerged as a proactive strategy;

- Utilizing Collaborative Debugging Platforms. Lesson: integration of collaborative debugging tools facilitated efficient communication among team members. Quick identification and resolution of bugs were achieved through shared insights and collaborative problem-solving;

- Implementing Dynamic Analysis for Performance Optimization. Lesson: profilers and performance analyzers aided in uncovering bottlenecks and inefficiencies in critical modules. This led to targeted optimizations for enhanced real-time control.

In navigating these real-world debugging scenarios, the development team gained valuable insights into improving code quality, implementing proactive testing methodologies, and fostering collaborative debugging practices. These lessons learned contribute to an iterative enhancement process, ensuring the continuous improvement of the warehouse logistics handling robot control system software.

Application of Debugging Tools.

Implementing effective debugging tools is a critical aspect of the warehouse logistics handling robot control system software development. This section explores the successful deployment of debugging tools in practice, emphasizing the importance of tool selection for achieving optimal results.

Successful Deployment of Debugging Tools in Practice:

a) Version Control Systems (VCS):

- role: VCS, such as Git, played a pivotal role in maintaining code version history and facilitating collaborative development;
- success: efficient tracking of code changes, quick identification of introduced bugs, and streamlined collaboration among developers contributed to successful codebase management;

b) Issue Tracking Systems:

- role: systems like Jira were employed for systematic bug tracking, task management, and project coordination;
- success: clear categorization of issues, assignment of tasks, and prioritization streamlined the debugging process, allowing the team to address critical issues promptly;

c) Collaborative Debugging Platforms:

- role: collaborative platforms like GitHub provided a shared space for debugging discussions, code reviews, and collaborative issue resolution;
- success: seamless communication, exchange of insights, and joint debugging efforts accelerated the identification and resolution of complex bugs.

Achieving Optimal Results through Tool Selection:

a) Profilers and Performance Analyzers:

- selection criteria: tools like cProfile and Py-Spy were chosen for profiling critical sections of the code to identify performance bottlenecks;
- optimal results: profiling aided in pinpointing areas requiring optimization, resulting in enhanced real-time control and overall system responsiveness;

b) Memory Debugging Tools:

- selection criteria: memory debugging tools like Valgrind were selected to detect memory leaks and manage resource utilization effectively;
- optimal results: Identification and resolution of memory-related issues ensured stable, long-term system operation without resource depletion;

c) Dynamic Analysis Tools:

- selection criteria: incorporation of dynamic analysis tools like PyCharm's debugger for real-time code analysis;
- optimal results: real-time insights into code execution allowed for immediate bug identification and contributed to continuous improvement in system stability.

In summary, the strategic deployment of version control systems, issue tracking platforms, collaborative debugging tools, and specialized debugging instruments significantly contributed to the successful development and continuous enhancement of the warehouse logistics handling robot control system software. The judicious selection and application of these tools played a vital role in achieving optimal results throughout the debugging process.

Importance of Documentation in Debugging.

Documentation serves as a cornerstone in the debugging phase of the warehouse logistics handling robot control system software. This section delves into the crucial role of documentation, emphasizing the creation of detailed debugging reports and the maintenance of an updated knowledge base.

Creating Detailed Debugging Reports:

a) Bug Reports:

- contents: each bug report includes comprehensive information such as error messages, steps to reproduce the issue, and specific circumstances leading to the bug;
- purpose: detailed bug reports streamline the debugging process by providing developers with clear insights into the nature and context of the issues;

b) Code Annotations:

- contents: code annotations are added to the source code, explaining complex algorithms, critical decision points, and potential pitfalls;
- purpose: annotated code serves as a reference for developers, facilitating a deeper understanding of the codebase and aiding in the identification of logic errors;

c) Debugging Sessions Documentation:

- contents: documentation of debugging sessions includes recorded steps, encountered challenges, and solutions implemented during the debugging process;
- purpose: capturing the debugging journey ensures that lessons learned and successful debugging strategies are documented for future reference.

Maintaining an Updated Knowledge Base:

a) Common Issues Repository:

- contents: a repository containing documented solutions to frequently encountered issues and their resolutions;
- purpose: the knowledge base acts as a quick reference for developers facing similar problems, reducing resolution time for recurrent issues;

b) Best Practices Guides:

- contents: guides documenting best practices, coding conventions, and effective debugging strategies;
- purpose: a centralized resource for developers to adhere to established practices, fostering consistency and minimizing common pitfalls;

c) Collaborative Documentation Platforms:

- contents: shared platforms for collaborative documentation, enabling input from multiple team members;
- purpose: collective knowledge sharing through collaborative documentation ensures that the entire team benefits from the insights and experiences of individual members.

In conclusion, the importance of documentation in the debugging phase cannot be overstated. The creation of detailed debugging reports and the maintenance of an updated knowledge base contribute significantly to efficient issue resolution, knowledge sharing, and the continuous improvement of the warehouse logistics handling robot control system software.

Adoption of Best Practices.

This section delves into the significance of adopting best practices in the debugging phase of the warehouse logistics handling robot control system software.

It emphasizes the establishment of code review guidelines and the implementation of standardized debugging processes.

Code Review Guidelines:

a) Structured Code Reviews:

- process: instituting a systematic code review process involving peer evaluations of code changes before integration;
- purpose: structured code reviews facilitate early bug detection, ensure code quality, and promote knowledge sharing among team members;

b) Code Review Checklists:

- components: comprehensive checklists covering coding standards, error handling, and adherence to established practices;
- purpose: checklists serve as a guide for reviewers, ensuring that critical aspects of the code are thoroughly examined during the review process;

c) Knowledge Transfer:

- approach: Code reviews as an opportunity for knowledge transfer, where experienced developers share insights with team members;
- purpose: fostering a collaborative environment through code reviews enhances team-wide understanding of the codebase and debugging strategies.

Standardized Debugging Processes:

a) Issue Tracking Systems:

- implementation: integration of robust issue tracking systems to log, prioritize, and assign debugging tasks;
- purpose: issue tracking centralizes debugging efforts, provides visibility into the status of identified issues, and ensures accountability within the development team;

b) Debugging Workflows:

- establishment: defining standardized workflows for identifying, reproducing, and resolving software bugs;
- purpose: standardized debugging workflows streamline the debugging process, reducing ambiguity and facilitating consistent and efficient issue resolution;

c) Root Cause Analysis:

- approach: emphasizing a thorough root cause analysis for every identified bug;
- purpose: determining the underlying cause of issues ensures that debugging efforts address the core problem, preventing the recurrence of similar bugs.

In conclusion, the adoption of best practices during the debugging phase is paramount for ensuring the reliability and maintainability of the warehouse logistics handling robot control system software. Code review guidelines and standardized debugging processes contribute to a collaborative and efficient development environment, ultimately enhancing the overall quality of the software.

Feedback Loops and Continuous Monitoring.

This section explores the importance of feedback loops and continuous monitoring in the debugging process of the warehouse logistics handling robot control system software. It highlights the implementation of continuous integration and deployment practices, along with regular code reviews and feedback sessions.

Implementing Continuous Integration and Deployment:

a) Continuous Integration (CI):

- integration frequency: regularly integrating code changes into a shared repository;
- automated testing: incorporating automated testing processes as part of the CI pipeline to detect integration issues promptly;
- purpose: continuous integration ensures that code changes are regularly validated, reducing the likelihood of introducing bugs during the integration phase;

b) Continuous Deployment (CD):

- automated deployment: automating the deployment process to swiftly deliver validated code changes into production;
- rollback mechanism: implementing a rollback mechanism to quickly revert to a stable version in case of deployment issues;

- purpose: continuous deployment streamlines the release process, allowing for rapid and reliable software updates.

Regular Code Reviews and Feedback Sessions:

a) Scheduled Code Reviews:

- frequency: establishing regular intervals for comprehensive code reviews;
- feedback mechanism: providing constructive feedback to developers during code reviews;
- purpose: scheduled code reviews contribute to ongoing improvement by addressing code quality and identifying potential issues early in the development cycle;

b) Feedback Sessions:

- interactive discussions: facilitating interactive discussions to address specific concerns, share insights, and exchange feedback;
- actionable feedback: ensuring that feedback is actionable, promoting continuous learning and improvement;
- purpose: regular feedback sessions create a collaborative environment, fostering a culture of continuous improvement within the development team.

Team Training and Skill Enhancement.

This section delves into the significance of team training and skill enhancement initiatives in the context of debugging for the warehouse logistics handling robot control system software.

Periodic Training Programs for Debugging Techniques:

a) Targeted Training Modules:

- content: developing training modules focused on debugging techniques, tools, and best practices;
- delivery: conducting periodic training sessions delivered by experienced developers or external experts;
- purpose: periodic training programs enhance the debugging skills of team members, ensuring they stay abreast of the latest methodologies and tools;

b) Hands-On Workshops:

- practical exercises: including hands-on debugging exercises in workshops to reinforce theoretical knowledge;
- collaborative learning: encouraging collaboration and knowledge exchange during practical sessions;
- purpose: hands-on workshops provide a practical understanding of debugging concepts, fostering skill development among team members.

Knowledge Sharing Sessions within the Development Team:

a) Knowledge Forums:

- structured sessions: organizing regular forums for developers to share insights, experiences, and debugging tips;
- cross-functional sharing: encouraging developers from different teams to share their expertise;
- purpose: knowledge sharing sessions promote a culture of continuous learning and collaboration within the development team;

b) Documentation Initiatives:

- knowledge repositories: establishing repositories for documenting debugging challenges, solutions, and best practices;
- accessibility: ensuring documentation is easily accessible and regularly updated;
- purpose: documentation initiatives serve as valuable references for team members, aiding in the resolution of similar challenges in the future.

CONCLUSION

This Master's Thesis focuses on the development of warehouse handling robots in the contemporary logistics industry. Starting from the functional requirements of warehouse handling robots and taking into account the characteristics of the robot usage environment, a control system for warehouse handling robots was designed and implemented. The system uses QR code readers, grayscale sensors, inertial navigation modules, infrared ranging sensors, and wireless communication modules to complete handling tasks sent by a terminal. Completed the overall design of the control system for warehouse handling robots. Based on the actual needs of the system and the performance parameters of the robots, a control system solution for handling robots in warehouse environments was designed. After analyzing the advantages and disadvantages, the type of core components was determined, and a mechanical structure design plan for the warehouse handling robots was completed using a covert design approach.

The Wang Quanquan Handling Robot Control System stands as a sophisticated software solution designed to orchestrate the movement of handling robots within a warehouse environment. This comprehensive system integrates key functionalities such as path planning, obstacle avoidance, and real-time control, offering a streamlined approach to warehouse logistics. The software's modular architecture ensures efficient coordination of tasks, from robot initialization to precise coordination input and adaptive functionalities.

Noteworthy features include the ability to dynamically visualize and adapt to the environment, enhancing the system's responsiveness to real-world challenges. The incorporation of advanced algorithms, including the A* pathfinding algorithm, ensures optimal route planning while considering dynamic obstacles. The software also excels in providing a user-friendly interface for inputting robot start and finish coordinates, facilitating seamless interaction.

The system's robustness is further emphasized by its capability to handle obstacles, allowing for dynamic re-routing and efficient navigation. The integration

of collaborative debugging tools and continuous monitoring practices ensures the system's reliability, contributing to the overall efficiency of warehouse logistics.

In summary, the Wang Quanquan Handling Robot Control System emerges as a versatile and reliable software solution, addressing the complexities of warehouse handling robot control. Its thoughtful design, encompassing various modules and functionalities, positions it as a valuable asset for optimizing logistics operations in a dynamic warehouse setting.

REFERENCES

- 1) Smith, J. (2020). *Intelligent Robots: Their Role in Our Future*. Springer International Publishing.
- 2) Wang, X., & Zhang, Y. (2019). A Survey on Intelligent Robots: Progress and Challenges. *Robotics*, 8(2), 38.
- 3) Hassabis, D. (2017). Future of AI and its impact on society. *Nature*, 550(7676), 324-327.
- 4) Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484-489.
- 5) Kubricht, J., & Janschek, K. (2019). Collaboration between Humans and Intelligent Robots: A Systematic Review of the State of the Art. *International Journal of Social Robotics*, 11(4), 571-584.
- 6) Mahajan, S., Kumar, S., & Saini, J. S. (2020). An Overview of Intelligent Robots and Their Applications. *Robotics and Automation Letters*, 5(2), 2392-2399.
- 7) Lee, J., Kim, Y. J., & Cho, K. (2018). Deep Learning in Medical Robotics: A Review. *IEEE Access*, 6, 8906-8924.
- 8) Zhang, L., Yang, J., & Feng, G. (2019). Design of an intelligent robot system based on the Internet of Things. *Cluster Computing*, 22(2), 3307-3319.
- 9) Hussain, M., Muhammad, K., & Lee, Y. K. (2019). A Review of Artificial Intelligence Based Intelligent Robots. *Journal of Intelligent and Robotic Systems*, 94(3- 4), 319-345.
- 10) Park, H., Jeon, H. G., Lee, J., & Kim, K. H. (2018). Development of an intelligent robot for food service using computer vision and deep learning. *Journal of Intelligent Manufacturing*, 29(7), 1639-1650

APPENDIX A

Listing A.1

```

import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

class WangQuanquanHandlingRobotControlSystem:
    def __init__(self, master):
        self.master = master
        self.master.title("Wang Quanquan Handling Robot Control
System")
        self.master.geometry("600x600")

        self.create_widgets()

    def create_widgets(self):
        self.canvas = tk.Canvas(self.master, width=500,
height=500, background="white")
        self.canvas.grid(row=0, column=0, rowspan=10)

        ttk.Label(self.master, text="Robot Start
Coordinates").grid(row=0, column=1, padx=50)
        ttk.Label(self.master, text="X:").grid(row=1, column=1,
padx=25)
        self.start_x_entry = ttk.Entry(self.master)
        self.start_x_entry.grid(row=1, column=2, pady=5)
        ttk.Label(self.master, text="Y:").grid(row=2, column=1,
padx=25)
        self.start_y_entry = ttk.Entry(self.master)
        self.start_y_entry.grid(row=2, column=2, pady=5)

        ttk.Label(self.master, text="Robot Finish
Coordinates").grid(row=3, column=1, pady=50)
        ttk.Label(self.master, text="X:").grid(row=4, column=1,
padx=25)
        self.finish_x_entry = ttk.Entry(self.master)
        self.finish_x_entry.grid(row=4, column=2, pady=5)
        ttk.Label(self.master, text="Y:").grid(row=5, column=1,
padx=25)
        self.finish_y_entry = ttk.Entry(self.master)
        self.finish_y_entry.grid(row=5, column=2, pady=5)

        ttk.Label(self.master, text="Obstacle
Coordinates").grid(row=6, column=1, pady=50)
        ttk.Label(self.master, text="X:").grid(row=7, column=1,
padx=25)
        self.obstacle_x_entry = ttk.Entry(self.master)
        self.obstacle_x_entry.grid(row=7, column=2, pady=5)

```

```

        ttk.Label(self.master, text="Y:").grid(row=8, column=1,
padx=25)
        self.obstacle_y_entry = ttk.Entry(self.master)
        self.obstacle_y_entry.grid(row=8, column=2, pady=5)

        ttk.Button(self.master, text="Confirm",
command=self.confirm_coordinates).grid(row=9, column=1,
columnspan=2, pady=20)

    def draw_chessboard(self):
        self.canvas.delete("all")
        for i in range(0, 500, 5):
            for j in range(0, 500, 5):
                color = "white" if (i // 5 + j // 5) % 2 == 0 else
"gray"
                self.canvas.create_rectangle(i, j, i + 5, j + 5,
fill=color)

    def draw_cell(self, x, y, color):
        self.canvas.create_rectangle(x * 5, 500 - (y + 1) * 5, (x
+ 1) * 5, 500 - y * 5, fill=color)

    def confirm_coordinates(self):
        start_x = int(self.start_x_entry.get())
        start_y = int(self.start_y_entry.get())
        finish_x = int(self.finish_x_entry.get())
        finish_y = int(self.finish_y_entry.get())
        obstacle_x = int(self.obstacle_x_entry.get())
        obstacle_y = int(self.obstacle_y_entry.get())

        if not (1 <= start_x <= 100) or not (1 <= start_y <= 100)
or not (1 <= finish_x <= 100) or not (1 <= finish_y <= 100) or not
(1 <= obstacle_x <= 100) or not (1 <= obstacle_y <= 100):
            messagebox.showwarning("Warning", "Coordinates must be
in the range from 1 to 100.")
            return

        self.draw_chessboard()
        self.draw_cell(start_x - 1, start_y - 1, "red")
        self.draw_cell(finish_x - 1, finish_y - 1, "green")
        self.draw_cell(obstacle_x - 1, obstacle_y - 1, "orange")

        path = self.calculate_path((start_x - 1, start_y - 1),
(finish_x - 1, finish_y - 1), (obstacle_x - 1, obstacle_y - 1))
        if path:
            self.draw_path(path)
        else:
            messagebox.showinfo("Info", "No path available.")

    def draw_path(self, path):
        for i in range(len(path) - 1):
            x1, y1 = path[i]
            x2, y2 = path[i + 1]

```

```

        self.canvas.create_line(x1 * 5 + 2.5, 500 - y1 * 5 -
2.5, x2 * 5 + 2.5, 500 - y2 * 5 - 2.5, width=3, fill="green")

    def calculate_path(self, start, finish, obstacle):
        chessboard_graph = {(i, j): [] for i in range(100) for j
in range(100)}
        for i in range(100):
            for j in range(100):
                if (i + 1, j) in chessboard_graph:
                    chessboard_graph[(i, j)].append((i + 1, j))
                if (i, j + 1) in chessboard_graph:
                    chessboard_graph[(i, j)].append((i, j + 1))
                if (i - 1, j) in chessboard_graph:
                    chessboard_graph[(i, j)].append((i - 1, j))
                if (i, j - 1) in chessboard_graph:
                    chessboard_graph[(i, j)].append((i, j - 1))

        chessboard_graph[obstacle] = []

    def heuristic(node, goal):
        return abs(node[0] - goal[0]) + abs(node[1] - goal[1])

    def astar(graph, start, goal):
        open_set = {start}
        came_from = {}

        g_score = {node: float("inf") for node in graph}
        g_score[start] = 0

        f_score = {node: float("inf") for node in graph}
        f_score[start] = heuristic(start, goal)

        while open_set:
            current = min(open_set, key=lambda node:
f_score[node])

            if current == goal:
                path = []
                while current in came_from:
                    path.append(current)
                    current = came_from[current]
                path.append(start)
                return path[::-1]

            open_set.remove(current)

            for neighbor in graph[current]:
                tentative_g_score = g_score[current] + 1

                if tentative_g_score < g_score[neighbor]:
                    came_from[neighbor] = current
                    g_score[neighbor] = tentative_g_score
                    f_score[neighbor] = tentative_g_score +

```

```
    heuristic(neighbor, goal)
        open_set.add(neighbor)

    return None

    return astar(chessboard_graph, start, finish)

def main():
    root = tk.Tk()
    app = WangQuanquanHandlingRobotControlSystem(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```