An-Najah National University

Faculty of Engineering & Information Technology

Department of Computer Engineering

Hardware Project

SHOP MATE

Supervisor:

Dr. Ashraf Armoush.

Students:

Eman Wasfy Ishtawi.

Toqa Omar Abd AlDayem.

January 23, 2026

Acknowledgment

We would like to express our sincere gratitude to everyone who contributed to the completion of the ShopMate Smart Shopping Cart hardware project.

First and foremost, we thank our families for their unconditional support, patience, and belief in us. Their sacrifices, encouragement, and constant presence formed the foundation that allowed us to stay focused, motivated, and resilient throughout this journey.

We are deeply grateful to our instructors and supervisors for their guidance, valuable feedback, and technical expertise. Their direction played a critical role in shaping the design, implementation, and overall quality of this project.

We also extend our appreciation to our friends and colleagues who supported us, shared ideas, and offered encouragement during challenging moments.

Dedication

This project is for Palestine, and for every dream interrupted by war.

For those who were prevented from growing up to fulfill their ambitions, and for the children who were never given enough time to imagine a future of their own.

It stands in memory of the martyrs whose lives ended too soon, leaving stories the world may never fully hear. It carries respect for the prisoners who continue to endure behind walls, holding onto hope under the hardest conditions, and for the wounded who live with pain as a permanent reminder of what was taken—and what still remains.

As Palestinian computer engineers, we dedicate ShopMate to all of them. Every effort invested in this project is a quiet statement that despite destruction, we continue to learn, to build, and to believe in a future shaped by knowledge, resilience, and purpose.

Abstract:

This project aims to enhance the shopping experience by helping customers easily locate items without prior knowledge of the supermarket layout. It saves time, reduces confusion, and supports individuals such as elderly people or first-time visitors. The system introduces automation and smart navigation into everyday retail environments, promoting the concept of intelligent shopping. The project focuses on several key aspects, including accurate indoor navigation to ensure autonomous and safe movement, collision avoidance using LiDAR, ultrasonic, accelerometer, gyroscope, and speed sensors, item scanning to allow users to view product prices and track their total cost, user-friendly interaction for entering shopping lists, and efficient path planning to minimize walking distance. The main objectives are to design and implement an autonomous cart that guides users to desired items, ensures smooth and safe movement through multi-sensor integration, and simplifies the shopping process by providing real-time product information. The methodology involves defining hardware and software components, using LiDAR for real-time mapping and localization, developing path-planning and obstacle-avoidance algorithms, integrating data from multiple sensors for stability, and building a simple user interface for inputting lists and scanning products. While some experimental smart carts exist globally, this project has not been developed before at our university, particularly not in this integrated form combining autonomous navigation with real-time LiDAR mapping and product price scanning.

Table of contents

List of Figures

Chapter 1 Introduction

1.1 General Background

The retail industry continues to face increasing pressure to improve efficiency, reduce operational costs, and enhance the overall customer shopping experience. Traditional shopping systems rely heavily on manual cart operation, cashier-based checkout processes, and human supervision, which often result in long queues, inefficient item handling, and increased labour requirements. These challenges are particularly evident in large supermarkets and busy retail environments.

Advances in robotics, embedded systems, and autonomous navigation have enabled the development of intelligent retail solutions that address these limitations. Smart shopping carts represent an emerging approach that integrates sensing, computation, and control technologies to assist customers throughout the shopping process. By incorporating autonomous navigation, real-time product verification, and digital interaction, smart carts aim to improve shopping convenience while supporting retailers in loss prevention and operational optimisation.

This project presents the design and implementation of a smart shopping cart capable of operating in both manual and autonomous modes. The system is built on ROS 2 as the core robotics middleware, enabling modular and efficient communication between localisation, navigation, and motion control components. A web-based user interface developed using Node.js allows users to interact with the cart by selecting items, receiving real-time system feedback, and completing the payment process. The system integrates LiDAR-based localisation and navigation, Arduino-based control and sensing, and a Raspberry Pi that serves as a central processing and communication bridge. The cart assists users throughout the shopping process, verifies items placed inside it, and incorporates multiple safety and security mechanisms to prevent misuse, unintended operation, or theft.

1.2 Objectives (Purpose and Aims) of the Work

The primary purpose of this project is to design and develop an intelligent smart shopping cart that enhances the shopping experience through autonomous assistance, secure item verification, integrated payment, and user-focused safety mechanisms. The system is intended to function as a reliable “shopping companion” that supports users throughout the shopping process while addressing common issues such as navigation difficulty, accidental or intentional item misplacement, and theft.

The main objectives of this work are to:

- Develop a smart shopping cart capable of operating in two distinct modes: manual mode and automatic mode.
- Implement a manual control system that allows users to directly command the cart to move forward, turn left or right, and stop.

- Design an automatic navigation system in which the cart autonomously navigates to items in the user's shopping list based on proximity, rather than the order in which items were added, thereby optimising travel distance and time.
- Perform accurate localisation and autonomous navigation in an indoor retail environment using LiDAR sensing, supported by rotational data from an MPU sensor to improve localisation accuracy.
- Use three ultrasonic sensors positioned to monitor the front, left, and right sides of the cart, providing proximity-based safety by automatically halting motion when obstacles are detected during forward movement or turning operations.
- Enable item scanning functionality that operates only when the cart is in a safe stop state, regardless of whether it is in manual or automatic mode.
- Implement an item verification mechanism that checks whether a scanned item exists in the user's cart list before allowing placement.
- Handle accidental scanning events by prompting the user to confirm whether they wish to add an unlisted item to their cart.
- Integrate a weight verification system that compares the measured weight of an item placed in the cart with its expected weight stored in a database.
- Prevent theft by activating an audible warning if a weight mismatch is detected and requiring RFID-based employee authorisation to resolve the alert.
- Detect and warn users when boycotted items are scanned to prevent unintentional purchases.
- Continuously monitor Bluetooth connectivity and immediately stop all cart functions if the connection to the user is lost, preventing unintended cart movement.
- Use a Raspberry Pi as a bridge between the Arduino-based control system and the LiDAR sensor, while also hosting a screen-based application for user interaction.
- Provide audio feedback through a speaker to support users who prefer or require voice-based interaction.
- Integrate a secure digital payment process using Stripe, allowing users to complete payment directly from the cart interface.
- Design the system to assist users who are unfamiliar with the store layout or local language, as well as elderly users or individuals with difficulty controlling a conventional shopping cart.

Through these objectives, the project aims to demonstrate a practical, secure, and user-centred smart shopping cart that integrates autonomous navigation, embedded control, intelligent verification, and payment within a single cohesive system.

1.3 Significance and Importance of the Work

The significance of this project lies in its relevance to current market demands for smart retail solutions that improve efficiency, security, and accessibility. Retailers face increasing losses due to theft, rising labour costs, and customer dissatisfaction caused by long checkout times. The proposed smart shopping cart addresses these challenges by integrating automated item verification, theft prevention mechanisms, and autonomous navigation.

From a customer perspective, the system offers a more convenient and guided shopping experience. The cart assists users in locating items efficiently, particularly benefiting customers who are unfamiliar with a store layout or shopping in a foreign country. Additionally, the system provides meaningful support for elderly users and individuals with mobility or control difficulties by reducing the need for manual cart handling.

From a technical and commercial standpoint, this project demonstrates how low-cost, widely available hardware components such as Arduino, Raspberry Pi, LiDAR sensors, and standard communication modules can be combined to create a sophisticated intelligent system. The inclusion of boycott item detection also introduces an ethical and user-aware feature that enhances informed purchasing decisions.

From an academic perspective, this work provides hands-on experience in robotics, embedded systems, and system integration within a real-world application. It aligns with current trends in autonomous systems and smart retail technologies.

1.4 Scope of the Project

This project focuses on the development of a functional prototype smart shopping cart designed for indoor retail environments. The scope includes hardware integration, software development for navigation and control, item verification mechanisms, obstacle collision prevention using ultrasonic sensors, and system safety features.

The system includes an implemented Stripe-based payment function operating in test mode, reflecting the prototype nature of the project while demonstrating a complete end-to-end shopping process.

While the system demonstrates advanced functionality, it is intended as a proof-of-concept rather than a fully commercialised product. Future enhancements may include large-scale deployment optimisation and further refinement of the user interface.

1.5 Organization of the Report

This report is organised into six chapters, each addressing a specific aspect of the smart shopping cart project. Chapter 1 introduces the project background, objectives, significance, and overall scope. Chapter 2 presents the theoretical background and reviews relevant previous work related to autonomous navigation, embedded systems, and smart retail technologies. Chapter 3 describes the methodology adopted in the design and implementation of the system, including hardware integration, software architecture, navigation strategy, safety mechanisms, and system constraints. Chapter 4 presents the experimental results and performance analysis of the developed system, focusing on navigation accuracy, item verification, user interaction, and safety behaviour. Chapter 5 discusses the results in relation to the project objectives, highlighting system capabilities, limitations, and practical implications. Finally, Chapter 6 concludes the report by summarising the key outcomes and proposing recommendations and future directions for further development.

Chapter 2 Theoretical Background and Previous Work

2.1 Previous Work Related to Smart Shopping Carts

Previous work on smart shopping carts and retail automation has focused on improving the shopping experience and reducing reliance on traditional checkout systems. Early developments introduced carts equipped with barcode scanners and screens that allowed customers to scan items while shopping. These systems reduced checkout time but depended largely on user honesty and offered limited protection against theft.

More advanced approaches incorporated camera-based item recognition and artificial intelligence to automatically detect products placed inside the cart. While effective, such systems often require high computational power, are sensitive to lighting conditions, and involve high implementation and maintenance costs. Other research efforts proposed semi-autonomous carts that follow users or navigate along predefined routes using ultrasonic sensors or basic vision-based tracking. These solutions typically lack accurate localisation and struggle in dynamic retail environments with narrow aisles and moving obstacles.

Overall, existing systems often focus on either automation or item tracking, but rarely integrate autonomous navigation, secure item verification, and user safety within a single cohesive platform. This project builds upon previous work by combining these elements into one integrated smart shopping cart system.

2.2 Theoretical Background of the Proposed Smart Shopping Cart System

The smart shopping cart developed in this project is based on several key theoretical concepts related to autonomous systems, embedded electronics, sensing, and human–machine interaction. These concepts form the technical foundation of the system design and implementation.

Autonomous robotics theory underpins the cart's ability to move independently within an indoor retail environment. An autonomous system must be capable of perceiving its surroundings, determining its position, planning movement toward a target, and controlling motion safely. In this project, the cart autonomously navigates to item locations when operating in automatic mode, while still allowing direct user control in manual mode.

LiDAR-based localisation and navigation provide the primary perception mechanism for the system. LiDAR sensors generate precise distance measurements that allow the cart to detect obstacles and understand its surrounding environment. This information is essential for indoor navigation, where GPS is unavailable. LiDAR is particularly suitable for retail environments because it performs reliably under varying lighting conditions and provides accurate spatial data.

To improve navigation accuracy, an MPU (Inertial Measurement Unit) is used to supply rotational and orientation data. The MPU helps compensate for rotational drift and supports smoother and more accurate movement, especially during turns. This sensor support enhances overall localisation reliability when combined with LiDAR data.

The system follows a distributed embedded architecture, a common approach in complex robotic systems. Low-level, time-critical tasks such as motor control, sensor reading, weight measurement, and safety handling are managed by an Arduino microcontroller. Higher-level processing, coordination, and communication are handled by a Raspberry Pi. This separation improves system stability and ensures that real-time control is not affected by computationally intensive processes.

Item verification is based on the theoretical principle of weight-based validation. Each product has a predefined weight stored in a database. When an item is scanned and placed inside the cart, the system compares the measured weight with the expected value. This approach provides a practical and reliable method for verifying that the scanned item matches the physical item placed in the cart.

Security and theft prevention are achieved through layered control mechanisms. If a weight mismatch is detected, the system activates an audible warning and restricts further operation. Resolution requires RFID-based employee authorisation, ensuring that only authorised staff can override security alerts. This mechanism addresses a major weakness of earlier smart cart systems that relied solely on scanning.

Human–machine interaction is implemented through a screen mounted on the cart and a speaker for audio feedback. The user interface is delivered through a Node.js application running locally on the Raspberry Pi, displayed directly on the cart screen. This allows users to interact with the system, view products, confirm actions, and receive feedback without requiring a mobile phone or external device.

User safety is further supported through continuous Bluetooth monitoring. The system immediately stops all cart functions if the Bluetooth connection between the user and the cart is lost, preventing unintended movement and ensuring the cart does not move away without user awareness.

Finally, accessibility principles are considered throughout the system design. The cart assists users who are unfamiliar with the store layout, elderly users, and individuals who may have difficulty controlling a conventional shopping cart. By providing autonomous navigation, guided assistance, and audio-visual feedback, the system functions as a “shopmate” that reduces both physical and cognitive effort during shopping.

Chapter 3 Methodology

This chapter describes the methodology used in the design and implementation of the smart shopping cart system. It details the materials, hardware, software tools, design standards, experimental procedures, and constraints considered throughout the project. The methodology is presented with sufficient technical detail to allow replication of the system by other engineering practitioners and students.

3.1 System Development Methodology

The smart shopping cart was developed using an iterative, system-integration-focused methodology. The project was divided into hardware development, software development, and system integration phases. Each subsystem was tested independently before full integration to ensure reliability, fault isolation, and safe operation.

The system architecture follows a distributed embedded design approach:

- An Arduino microcontroller handles low-level, real-time tasks such as motor control, sensor reading, weight measurement, RFID authentication, buzzer control, LED control, and safety interlocks.
- A Raspberry Pi performs higher-level processing, acts as a communication bridge between subsystems, interfaces with the LiDAR sensor, and runs the Robot Operating System (ROS) for autonomous navigation and localisation.
- A Node.js application runs locally on the Raspberry Pi and is displayed on the cart's screen, enabling user interaction without requiring a mobile device.
- Autonomous navigation and localisation are implemented using ROS-based modules, utilising LiDAR data supported by rotational feedback from an MPU sensor to improve localisation accuracy and movement stability.

The cart operates in two distinct modes:

- Manual mode, where the user directly controls cart movement.
- Automatic mode, where the cart navigates autonomously to items based on proximity optimisation.

The autonomous navigation subsystem follows a modular ROS 2–based architecture designed to integrate web-based user interaction with embedded hardware control. The system adopts a layered design in which user commands are generated through a cart-mounted web interface, processed by high-level navigation and planning modules, and executed through a dedicated motor control interface connected to the embedded Arduino platform. This separation of responsibilities improves system reliability, supports incremental development, and simplifies fault isolation during testing.

3.2 Apparatus and Materials

The smart shopping cart hardware platform was developed using commercially available components organised into functional subsystems. The system follows a distributed embedded architecture in which sensing, actuation, processing, safety, and user interaction are clearly separated. The following subsections describe each subsystem, its components, and its role within the overall system.

3.2.1 Processing and Control Subsystem

The processing architecture is divided into low-level real-time control and high-level computational processing.

Arduino Mega 2560

The Arduino Mega serves as the real-time embedded controller responsible for:

- Motor actuation
- Ultrasonic sensing
- Load cell data acquisition
- RFID authentication
- Bluetooth monitoring
- Safety interlocks

The Arduino was selected due to its large number of I/O pins and stable real-time timing behaviour.

Figure 3.1 : Arduino Mega

Raspberry Pi 4

The Raspberry Pi acts as the high-level processing unit. It performs:

- ROS 2 execution
- LiDAR data processing
- Localisation and path planning
- Node.js web interface hosting
- Communication bridge between ROS and Arduino

The separation between Arduino and Raspberry Pi improves system reliability by isolating time-critical tasks from computationally intensive processes.

Figure 3.2: Raspberry pi 4

3.2.2 Navigation and Localisation Subsystem

RPLiDAR A1

The RPLiDAR A1 is used for environment mapping and real-time obstacle detection. It generates 2D laser scans used by the ROS 2 SLAM and Nav2 stack to create occupancy grid maps and enable autonomous navigation.

Figure 3.3: RPLiDAR a1

MPU-6050 Inertial Measurement Unit (IMU)

The MPU-6050 provides rotational and orientation data (yaw measurement) to improve heading stability during motion. It compensates for rotational drift and supports smoother turning behaviour.

The LiDAR and IMU data together provide reliable indoor localisation without reliance on GPS.

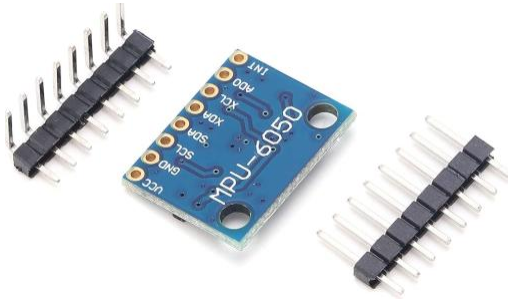

Figure 3.4: MPU-6050

3.2.3 Obstacle Detection Subsystem

Three ultrasonic sensors are mounted at the front, left, and right sides of the cart.

These sensors provide short-range proximity detection to:

- Prevent forward collision
- Restrict unsafe turning
- Enable hold–resume behaviour

Ultrasonic sensing was selected for short-range redundancy, as LiDAR is more effective for mid-range mapping but less reliable for very close obstacle detection.

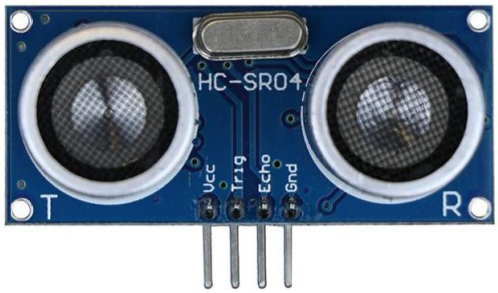

Figure 3.5: Ultrasonic

3.2.4 Weight Verification Subsystem

The weight verification system consists of four 50 kg strain-gauge load cells mounted at the four corners of the cart base. Together, they provide a total theoretical load capacity of 200 kg.

Using four load cells ensures stable and accurate measurement even when items are placed unevenly inside the cart. The sensors are connected in a Wheatstone bridge configuration and interfaced with the Arduino through an HX711 24-bit analog-to-digital amplifier, which enables precise digital weight measurement suitable for real-world shopping conditions.

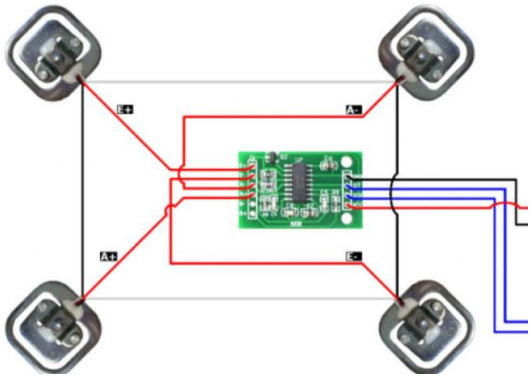

Figure 3.6: Load Cells with HX711 amplifier

3.2.5 Actuation Subsystem

The cart is driven by two hoverboard-style Brushless DC (BLDC) motors mounted on the rear axle.

Figure 3.7: Brushless DC motor

BLDC Motor Controller

A dedicated brushless motor controller regulates motor current and torque output.

Figure 3.8: Brushless Controller

MCP4725 Digital-to-Analog Converters (2 Units)

Two Adafruit MCP4725 DAC modules generate smooth analog throttle voltages.

PWM-based motor control was initially tested but resulted in abrupt speed changes and unstable motion. The DAC-based approach provided:

- Smooth acceleration and deceleration
- Reduced mechanical stress
- Improved navigation stability

Wheel velocities are computed using differential drive kinematics and converted to analog voltages. Reverse motion is disabled for safety.

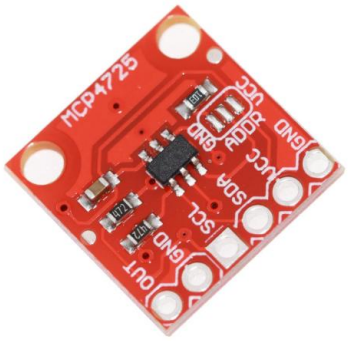

Figure 3.9: MCP4725

3.2.6 Identification and Security Subsystem

DFRobot GM60 Barcode Scanner

Used for item identification and price lookup.

Figure 3.10: GM60 scanner

RFID Reader and Tags

Used for employee authorisation during security alerts.

Figure 3.11: RFID

HC-05 Bluetooth Module

Continuously monitors user–cart connection and enforces immediate stop on disconnection.

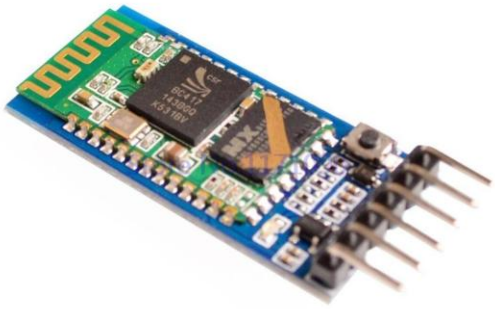

Figure 3.12: HC-05

3.2.7 Human–Machine Interface Subsystem

Touchscreen Display

Displays the Node.js web interface for user interaction.

Figure 3.13: Touchscreen

Speaker and Buzzer

Provide audio alerts and confirmation signals.

Figure 3.14: Buzzer and speaker

RGB LEDs (Controlled via IRF520 MOSFET Modules)

Provide visual state indication (waiting, verified, error, alert).

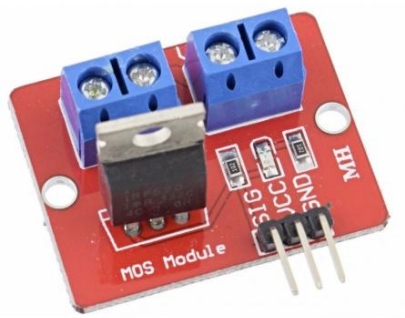

Figure 3.15: IRF520 Module

3.2.8 I²C Bus Architecture

To reduce wiring complexity and ensure synchronised communication, the MPU-6050 and both MCP4725 DAC modules share a common I²C bus on the Arduino. Each device is independently addressed.

This configuration minimises hardware overhead while maintaining stable timing performance.

3.2.9 Motor Control and Velocity Mapping

Motor control is implemented using two Adafruit MCP4725 digital-to-analog converters (DACs) to generate smooth analog throttle voltages for the BLDC motor controller. This approach was adopted to address sudden stops and jerky behaviour observed when pulse-width modulation (PWM) was previously used.

In automatic mode, commanded wheel velocities are converted to analog voltages using:

$$V_{out} = 1.24 + \left(\frac{v}{0.26}\right)(1.33 - 1.24)$$

where v is the wheel velocity in m/s. The DAC output is rate-limited to ensure gradual voltage changes and smooth acceleration and deceleration.

Differential drive kinematics are applied to compute individual wheel velocities:

$$v_L = v_x - \omega \frac{L}{2}, \quad v_R = v_x + \omega \frac{L}{2}$$

where vx is the forward velocity, ω is the angular velocity, and L=0.48 m is the wheel base. Wheel velocities are constrained to the range 0.22–0.26 m/s, with reverse motion disabled.

This control strategy ensures smooth motor operation, eliminates abrupt speed changes, and reduces mechanical stress on the drivetrain.

3.3 Experimental and Operational Procedures

3.3.1 Manual Mode Operation

In manual mode, the user directly controls cart motion through the cart-mounted Node.js touchscreen interface. Directional commands, including forward, left, right, and stop, are transmitted to the embedded Arduino controller for execution. This mode is intended for users who prefer direct control of the cart's movement.

Manual operation incorporates multiple safety interlocks to ensure predictable behaviour and reduce collision risk in indoor retail environments. Three ultrasonic sensors are positioned at the front, left, and right sides of the cart to provide real-time proximity monitoring. During forward motion, the front ultrasonic sensor continuously checks for obstacles within a predefined safety threshold and immediately halts movement if an obstacle is detected. Similarly, the left and right ultrasonic sensors prevent the initiation or continuation of turning manoeuvres when insufficient side clearance is available.

Turning in manual mode is executed as fixed 90-degree rotations. This turning strategy is well suited to supermarket layouts, where aisles and shelf intersections are typically arranged at right angles. By limiting turns to discrete 90-degree movements, the cart maintains predictable motion and alignment with aisle geometry.

3.3.2 Automatic Mode Navigation

In automatic mode, the cart navigates autonomously to shopping item locations defined within the store map. Users select items through the touchscreen interface, after which navigation goals are generated and passed to the ROS-based navigation framework. The system continuously estimates the cart's position and plans collision-aware paths to each target location.

Before autonomous navigation begins, the user may define or reset the cart's starting position through the interface. This allows the localisation system to initialise correctly regardless of where the cart is placed within the store.

When multiple items are present in the shopping list, the system determines the next navigation target based on proximity rather than the order of item selection. This distance-based prioritisation minimises unnecessary travel and improves overall shopping efficiency.

Upon reaching a navigation goal, the cart enters a waiting state and requires the user to scan the corresponding item. The system does not allow navigation to the next goal until the item associated with the current location is either successfully scanned and verified or explicitly removed from the shopping list. This ensures a strict correspondence between navigation goals and item acquisition.

3.3.3 Environment Mapping and Map Generation

Prior to autonomous navigation, an environment map of the indoor retail space was generated to support localisation and path planning. Mapping was performed using the LiDAR sensor within a ROS-based SLAM framework. The resulting two-dimensional occupancy grid represents static environmental features such as walls, shelves, and aisle boundaries and serves as the global reference for all navigation tasks.

During mapping, the cart was manually driven through the environment to ensure sufficient spatial coverage and loop closure. The final map was saved and reused during autonomous operation, allowing the localisation system to estimate the cart's position relative to a fixed reference frame.

In addition to the static map, the navigation system generates planned paths between the cart's current position and target item locations. These paths are computed by the ROS navigation stack and represent collision-aware trajectories that respect environmental boundaries. The planned path is continuously updated during motion execution to account for localisation updates and nearby obstacles.

Figure X illustrates the generated occupancy grid map used for localisation, while Figure Y shows an example of a planned navigation path overlaid on the map during automatic operation. The visualised path demonstrates the cart's ability to follow smooth trajectories within confined indoor spaces while maintaining appropriate clearance from surrounding structures.

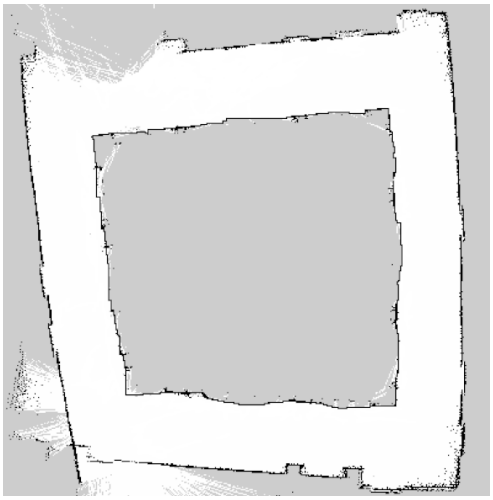

Figure 3.16: Generated indoor environment occupancy grid map used for LiDAR-based localisation.

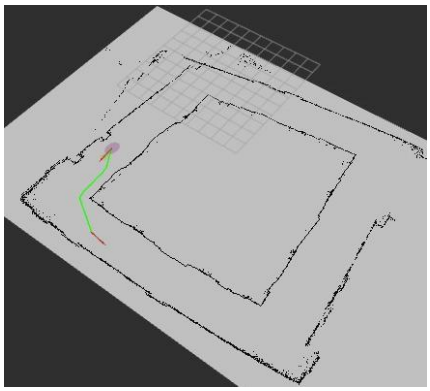

Figure 3.17: Example autonomous navigation path overlaid on the environment map during automatic operation.

3.3.4 Navigation Control Strategy

To ensure safe and stable navigation in narrow retail environments, a custom navigation control strategy was implemented. Default navigation controllers were observed to prioritise direct orientation toward the target location, which can lead to unsafe proximity to shelves and aisle boundaries.

The implemented strategy continuously monitors the curvature of the planned navigation path. When a significant change in heading is detected, indicating an approaching corner, the controller reduces forward velocity and increases rotational control to execute a smooth turn. During straight path segments, steady forward motion is maintained while small heading corrections are applied to remain aligned with the planned path.

This approach improves motion stability, preserves safe clearance from nearby obstacles, and is better suited to supermarket environments than standard navigation controllers.

3.3.5 System States and Mode Logic

The embedded Arduino controller implements a hierarchical state machine to manage cart behaviour. Five operational states are defined:

- **ST_STOP:** Cart stationary; sensor monitoring and item scanning enabled
- **ST_MOVE:** Cart in forward motion; obstacle monitoring active
- **ST_TURN:** Cart executing a 90-degree turn (manual mode only); yaw monitoring active
- **ST_WAIT_WEIGHT:** Item scanned; awaiting placement and weight measurement
- **ST_WAIT_RESULT:** Weight measurement complete; awaiting user confirmation

Mode-specific logic governs state transitions. In manual mode, turning commands initiate the ST_TURN state, allowing controlled 90-degree pivots using inertial feedback. In automatic mode, the turning state is bypassed entirely, and continuous velocity commands received from the navigation system are applied directly. In both modes, item scanning is only permitted when the cart is fully stopped and not executing a turn.

3.3.6 Obstacle Detection and Hold–Resume Behaviour

Ultrasonic sensors provide short-range safety monitoring during both manual and automatic operation. Rather than performing active obstacle avoidance through dynamic path replanning, the system implements an obstacle hold–resume mechanism.

When an obstacle is detected during forward motion, the motors are stopped immediately and the intended command is temporarily paused. Once the obstacle is cleared, motion resumes automatically without requiring additional user input. During turning manoeuvres, obstacle detection pauses motor output while preserving the target orientation and turn progress. When clearance is restored, the cart resumes the same turn direction and completes the manoeuvre.

This approach was selected instead of dynamic obstacle avoidance using LiDAR-based replanning because the operating environment consists of narrow supermarket aisles with limited alternative paths. In such constrained spaces, attempting to generate a new path around temporary obstacles-such as nearby customers or carts-can lead to unstable navigation behaviour, oscillations, or deadlock conditions. Temporarily stopping and resuming motion after the obstacle clears provides a safer, more reliable, and context-appropriate solution for indoor retail environments.

3.3.7 Item Scanning and Verification

Item scanning is restricted to safe operational conditions. Scanning is only enabled when the cart is stationary and not turning. Once a barcode is scanned, the system verifies that the item exists in the user's shopping list and prompts the user to place the item inside the cart.

A weight verification workflow is then initiated. A baseline weight is recorded, followed by detection of item placement. After a short settling delay, the item weight is averaged and compared to the expected value stored in the database. If the measured weight matches the expected range, the item is approved and the cart returns to a stopped state. If a mismatch is detected, further operation is blocked and a security alert is triggered.

3.3.8 Theft Prevention and Authorisation

In the event of a weight verification failure, the cart enters a locked security state. All motion and user commands are disabled, and a visual alert remains active. Resolution of this state requires employee authorisation using RFID authentication.

A dedicated RFID reader continuously monitors for authorised cards. Only a designated employee card can reset the system and restore normal operation. This mechanism ensures that security events cannot be bypassed by users and provides a robust theft prevention layer.

3.3.9 Boycott Item Detection

All scanned items are checked against a boycott list stored in the system database. If a scanned item is identified as restricted, the user is notified through the interface, allowing informed purchasing decisions and preventing unintentional selection of prohibited products.

3.3.10 Payment Procedure

When the user is ready to complete the purchase, the system verifies the total weight of items placed in the cart against the expected total weight of all scanned items stored in the shopping list. Any detected mismatch must be resolved before payment can proceed.

Once verification is successful, payment is initiated using Stripe, which is fully integrated into the Node.js application running locally on the Raspberry Pi and displayed on the cart's touchscreen. This enables secure, real-time payment directly through the cart interface without the need for external devices. Upon successful payment, the system confirms the transaction and provides visual and audio feedback to the user.

3.3.11 Bluetooth Safety Monitoring

Bluetooth connectivity between the user interface and the cart is continuously monitored. If the connection is lost, all cart functions immediately stop to prevent unintended movement or loss of user control. Normal operation resumes only after the connection is re-established.

3.4 Standards and Specifications (Codes)

The design of the smart shopping cart follows established engineering standards, software conventions, and safety principles to ensure reliable, secure, and user-friendly operation. These standards guided both hardware and software design decisions throughout the project.

3.4.1 Robotics Software Standards and Conventions

The system is built using the Robot Operating System 2 (ROS 2) framework, which provides a standardised architecture for modular robotic systems. ROS 2 was used for localisation, navigation, and communication between software components. The system follows standard ROS conventions for message passing, node separation, and lifecycle management.

The cart adheres to the ROS REP-105 coordinate frame convention, using a hierarchical frame structure:

map → odom → base_link → sensor frames (LiDAR, MPU)

This convention ensures consistent localisation, navigation, and sensor data fusion in indoor environments. LiDAR-based localisation is used as the primary positioning method, with MPU rotational data supporting orientation accuracy.

A ROS–Node.js communication bridge is used to enable real-time interaction between the ROS navigation system and the cart-mounted user interface.

3.4.2 Embedded Systems and Control Standards

The system follows standard embedded systems design principles, particularly the separation of real-time control and high-level processing:

- The Arduino handles time-critical tasks such as motor control, sensor acquisition, and safety interlocks.
- The Raspberry Pi manages high-level processing, ROS execution, navigation coordination, and the user interface.

This separation improves reliability and prevents timing issues caused by computational load.

3.4.3 Wireless Communication Standards

Two wireless communication methods are utilised in the system with clearly separated roles.

- Wi-Fi is used solely to support cloud connectivity and backend services, enabling the Node.js application and Firebase database to operate correctly for user interface updates, item management, and payment-related communication. It is not involved in real-time control, localisation, or navigation, which are performed entirely on-board.
- Bluetooth is employed for short-range safety monitoring, enabling immediate cart stoppage in the event of user disconnection.

This separation ensures reliable system operation while maintaining safety and minimising dependency on network availability

3.4.4 Electrical and Power Standards

The cart uses a low-voltage DC power distribution system suitable for indoor robotic platforms. Electrical safety considerations include:

- Common grounding across all electronic components
- Proper current handling for motors and actuators
- Colour-coded wiring to improve maintainability and fault tracing

All components operate within their rated electrical limits.

3.4.5 Operational Safety Standards

Operational safety is enforced through software and control constraints, including:

- Speed limiting in confined or crowded areas
- Obstacle avoidance using LiDAR and ultrasonic sensors
- Automatic motor shutdown under unsafe conditions (e.g. Bluetooth disconnection or verification failure)

These measures align with general robotic safety principles for operation in public indoor environments.

3.4.6 Human–Machine Interaction Standards

Human–machine interaction follows usability and accessibility principles. Visual feedback is provided through the touchscreen and RGB LEDs, while audio feedback reinforces system states and warnings. The interface is designed to be intuitive and minimise user error during operation.

3.4.7 Payment and Data Security Standards

Secure digital payment is implemented using Stripe in test mode. Stripe ensures that sensitive payment data is handled according to industry-standard security and data protection practices. No payment information is stored locally on the cart, reducing security risk.

These standards and specifications ensure that the smart shopping cart operates safely, reliably, and securely. The system integrates ROS 2 software conventions, embedded systems best practices, wireless communication standards, electrical safety principles, and secure payment handling within a unified and practical design.

3.5 Design Constraints

Several technical constraints were encountered during the development and testing of the smart shopping cart, which influenced system performance and design decisions.

A major constraint was the overall weight of the cart, which significantly impacted the motion subsystem. Initial prototypes employed drill motors; however, these motors were unable to sustain continuous motion under load and stalled after short operating periods. Automotive wiper motors were subsequently evaluated as an alternative due to their higher torque characteristics. Despite this improvement, the motors still experienced excessive current draw when driving the fully loaded cart, leading to overheating and, in some instances, damage to the Arduino controller. These outcomes highlighted the mechanical limitations imposed by the cart's weight and the need for more robust motor and power solutions.

Another constraint was associated with LiDAR-based perception. The LiDAR sensor was unable to reliably detect transparent and reflective surfaces such as glass walls. This limitation resulted in incomplete and noisy environment maps during the SLAM process, negatively affecting localisation accuracy. As a temporary mitigation, cardboard was placed over glass surfaces to enhance laser reflectivity and improve map quality. While this approach proved effective during experimentation, it is not suitable for practical deployment in real retail environments.

Localization accuracy was further constrained by the absence of wheel encoders. The hoverboard-style motors used in the system do not readily support encoder installation, preventing direct measurement of wheel rotation. Consequently, odometry estimation relies on commanded velocity inputs rather than actual wheel feedback. For example, when a velocity of 0.22 m/s is issued, the system assumes ideal motion at that speed regardless of real-world factors. Variations caused by battery voltage fluctuations, wheel slippage, or changes in payload are therefore not captured, leading to accumulated positional error over time and reduced navigation precision.

These constraints collectively influenced system performance and highlight areas for improvement in future iterations, particularly in mechanical design, sensing robustness, and motion feedback integration.

3.6 Safety and Risk Considerations

Safety was a primary design consideration due to the cart's operation in public indoor retail environments. Potential hazards include unintended movement, electrical faults, and user misuse. These risks are addressed through a combination of software controls, hardware safeguards, and user feedback mechanisms.

The system enforces immediate stopping under unsafe conditions, including loss of Bluetooth connectivity between the user and the cart. During item scanning and verification, the cart is required to remain stationary, preventing movement while users interact with the cart or place items inside it. In addition, motion is restricted during security alerts, ensuring that abnormal or unauthorized states cannot result in cart movement.

Obstacle monitoring during motion provides an additional safety layer by stopping the cart when objects are detected at close range. Clear visual and audio feedback is provided to inform users of system states, warnings, and required actions. Electrical safety is addressed through secure wiring practices and proper insulation of all electronic components.

Together, these measures reduce the likelihood of unsafe behaviour and support reliable operation in crowded indoor environments.

Chapter 4 Results and Analysis

This chapter presents the results obtained from testing the smart shopping cart system and analyses its performance with respect to navigation accuracy, item verification reliability, system safety, and user interaction. Only data relevant to evaluating system functionality and objectives is included. All tests were conducted in an indoor environment representative of a retail setting.

4.1 Autonomous Navigation Performance

Autonomous navigation is implemented using LiDAR-based localisation techniques suitable for indoor retail environments. The cart continuously estimates its position relative to its surroundings and plans paths to predefined item locations while actively avoiding obstacles. Item locations are defined within the system and used as navigation goals during automatic operation.

The navigation and control framework is implemented using the Robot Operating System (ROS). Several ROS nodes are employed to support localisation, path planning, and motion control. The rplidar_ros node is used to process LiDAR data and generate obstacle information for navigation. Rotational data from the MPU sensor is acquired through the mpu_serial node, which supports orientation accuracy and reduces drift during motion. The Nav2 stack is used for global path planning, obstacle avoidance, and goal-based navigation. In addition, a custom motion controller, corner_controller, is integrated to enhance navigation precision.

Communication between the ROS-based navigation system and the Node.js user interface is handled through rosbridge_suite, enabling real-time command exchange, system state monitoring, and seamless interaction between the cart interface and the navigation stack.

To improve navigation behaviour in narrow indoor environments such as supermarket aisles, a custom controller, corner_controller, was developed. This controller continuously monitors the curvature of the planned path. When a significant change in heading is detected, indicating an approaching corner, the controller adjusts wheel velocities to ensure smooth and safe turning. This approach maintains appropriate clearance from nearby obstacles while preserving stable motion.

The default navigation controller provided by the Nav2 stack was found to be unsuitable for this application, as it prioritises direct orientation toward the target location without sufficient consideration of nearby obstacles. In confined retail environments, this behaviour may lead to unsafe proximity to shelves or aisle boundaries. The custom corner_controller addresses this limitation by incorporating path-awareness and safety constraints, making it more appropriate for real-world shopping environments.

In automatic mode, the cart determines the optimal navigation sequence by selecting the nearest required item rather than following the order in which items were added to the shopping list. When the cart reaches a navigation goal, the system enters a waiting state and requires the user to scan the corresponding item before proceeding to the next goal. Movement to the next target is disabled until the item is scanned and verified, ensuring that navigation and item collection remain synchronised. If the user chooses not to collect the item, it can be deleted from the shopping list

through the interface, after which the user may select the next goal and allow the cart to continue navigation.

MPU rotational data is used to support orientation accuracy throughout navigation, further reducing drift and improving movement stability.

4.2 Manual Mode Control Accuracy and Obstacle Awareness

Manual mode tests were conducted to evaluate the responsiveness, controllability, and safety of the cart when operated directly by the user. Directional commands, including forward, left, right, and stop, were issued through the Node.js interface running on the cart-mounted screen.

The cart responded immediately to user inputs, with no noticeable delay between command issuance and motor actuation. Directional movements were smooth and predictable, and the stop command consistently halted the cart within a short distance. This confirmed that the low-level motor control implemented on the Arduino provided stable and responsive behaviour.

During early testing, direct PWM-based control from the Arduino to the brushless motor controller resulted in abrupt speed changes and occasional sudden stops. This behaviour was attributed to the sensitivity of the brushless DC motor controller to discrete PWM throttle signals. To resolve this issue, a Digital-to-Analog Converter (DAC) was introduced to generate smooth analogue throttle voltages from digital control commands. The use of the DAC significantly improved speed stability, eliminated sudden stops, and resulted in smoother acceleration and deceleration during manual operation.

Obstacle awareness during manual mode was enhanced using three ultrasonic sensors, positioned to monitor the front, left, and right sides of the cart. These sensors continuously measured short-range distances and prevented motion when an obstacle was detected within a predefined safety threshold. Forward motion was inhibited when an obstacle was detected ahead, while turning commands were restricted if obstacles were present on the corresponding side. This ensured that the cart could not collide with shelves, objects, or people due to user input.

No unintended movement or unsafe behaviour was observed during manual mode testing. The combination of smooth DAC-based motor control, multi-directional ultrasonic obstacle detection, and enforced stop logic resulted in reliable, controlled, and safe manual operation in an indoor environment.

4.3 Item Verification and Weight Measurement Results

Item verification tests were performed using products with predefined reference weights stored in the system database. After scanning an item, the user placed it into the cart, and the load cells measured the resulting weight change.

During the verification process, RGB LED indicators were used to provide immediate visual feedback to the user. The LED turned blue while the system was waiting for the user to place the scanned item into the cart. Once the weight measurement was completed, the LED turned green if the measured weight matched the expected value within the allowable tolerance, and red if a mismatch was detected.

The measured weight values closely matched the expected reference values for correctly placed items. Small variations were observed between measured and expected weights due to factors such as load cell resolution, mechanical vibration during placement, and item positioning within the cart. To account for these real-world effects, a tolerance range was applied during verification.

This tolerance reflects practical real-life conditions rather than ideal laboratory conditions. In real shopping environments, slight variations in measured weight are unavoidable and should not result in false rejection of valid items. The applied tolerance ensured reliable operation without compromising security.

It is important to note that in high-end commercial systems, industrial-grade weighing mechanisms with milligram-level sensitivity can reduce measurement error to near zero. In such systems, changes at the milligram scale would be negligible, effectively eliminating the need for larger tolerance ranges. However, for a prototype system using commercially available components, the observed tolerance was appropriate and realistic.

Items placed within the acceptable tolerance range were consistently approved, while items outside the range triggered a verification failure. No false approvals were observed during testing, confirming that the weight-based verification mechanism effectively balanced reliability and practicality.

4.4 Theft Prevention and RFID Authorisation Results

To evaluate the theft prevention mechanism, deliberate weight mismatches were introduced by placing incorrect items or additional objects into the cart after scanning.

In all tested cases, the system correctly detected the mismatch, activated the audible buzzer, and locked cart movement. The cart remained in the locked state until an authorised RFID card was presented.

RFID authentication was consistently recognised within a short detection time, after which the system reset to normal operation. No false authorisations were observed during testing, indicating reliable access control.

4.5 Bluetooth Safety Monitoring Results

Bluetooth connectivity tests were conducted by intentionally disconnecting the user device during cart operation. Upon loss of connection, the system immediately halted all cart movement and disabled further actions.

The response time between Bluetooth disconnection and cart stoppage was observed to be sufficiently short to prevent unintended cart movement. Reconnection restored normal operation without requiring system restart.

These results confirmed that Bluetooth monitoring effectively enhanced user safety and system reliability.

4.6 User Interaction and System Feedback

User interaction was evaluated through the cart-mounted touchscreen and audio feedback system, both driven by the Node.js application running locally on the Raspberry Pi. The application was designed to manage user sessions, shopping logic, and safety-related interactions while maintaining system robustness.

The system supported user sign-up and login functionality, allowing each shopping session to be associated with a specific user. This ensured that shopping lists, item quantities, verification states, and payment actions were correctly linked to the active user session. Authentication operated reliably during testing, with no observed session conflicts or unintended data carryover between users.

Users were able to switch seamlessly between manual and automatic modes through the interface. In automatic mode, the system allowed the user to define the starting position of the cart. This feature enabled localisation to be initialised or re-initialised at any time, ensuring that autonomous navigation could begin accurately regardless of where the cart was positioned within the store. This proved particularly useful when restarting navigation after manual operation or interruptions.

The application enabled comprehensive shopping list management. A delete function allowed users to remove items from the cart. If an item had not yet been physically scanned and verified, it could be deleted after confirmation without affecting system state. If an item had already been verified, deleting it triggered a recalculation of the expected total cart weight by subtracting the stored reference weight of the removed item. Any subsequent scan or payment attempt compared the measured total weight against this updated value, and discrepancies resulted in a security alert.

The system also supported quantity selection for items. When a quantity greater than one was selected, the system multiplied the reference item weight by the chosen quantity and verified the measured weight accordingly. This allowed a single scan to represent multiple units while preserving verification accuracy.

During item scanning, users were able to cancel the addition of an item if it was not already part of the shopping list. When a boycotted item was scanned, the system immediately issued visual and audio warnings to prevent unintentional purchase.

Throughout testing, prompts, confirmations, warnings, and alarms were displayed clearly on the screen and reinforced through audio feedback. The Node.js application remained stable, with no observed crashes, freezes, or incorrect state transitions, demonstrating reliable handling of user interaction, session management, and shopping logic.

4.7 Payment Process Validation

The payment workflow was evaluated by completing full shopping sequences and initiating checkout directly through the cart-mounted interface. Before payment was allowed, the system verified that the measured total weight of items in the cart matched the expected total derived from the scanned items and their quantities. Once verification was successful, transactions were processed using Stripe in test mode, fully integrated into the Node.js application running on the Raspberry Pi. Payment confirmation was returned to the system and presented to the user through visual and audio feedback. No payment data was stored locally on the cart, ensuring secure transaction handling. The payment process completed without errors during testing, validating its reliable integration into the overall system.

Chapter 5 Discussion

This chapter interprets the results obtained from system testing and evaluates the extent to which the project objectives were achieved. The discussion compares the observed system behaviour with the original design intent and current approaches in smart retail and autonomous systems, while also identifying limitations and opportunities for further development.

5.1 Evaluation of Problem Resolution

The primary problem addressed in this project was the lack of an integrated shopping assistance system that combines autonomous navigation, secure item verification, user safety, and accessibility within a single smart shopping cart platform. Based on the experimental results, this problem was largely resolved.

The cart successfully operated in both manual and automatic modes, allowing users to either directly control movement or rely on autonomous navigation. In automatic mode, the cart optimised item collection by navigating to the nearest required item rather than following the order of addition, thereby improving efficiency. This behaviour directly addressed the goal of reducing unnecessary movement within a store environment.

The integration of item scanning with weight-based verification and RFID-authorised intervention effectively addressed theft prevention, a common limitation of earlier smart cart systems. Bluetooth monitoring and enforced stop logic further resolved safety concerns related to unintended cart movement.

5.2 Contribution of the Work

This project contributed a fully integrated prototype that combines several features typically implemented separately in prior systems. The main contributions include:

- A dual-mode smart shopping cart supporting both manual control and autonomous navigation.
- Proximity-based item collection logic that improves navigation efficiency in automatic mode.
- A practical and reliable weight-based item verification system integrated with RFID-authorised resolution.
- A layered safety framework incorporating ultrasonic obstacle detection, Bluetooth disconnection handling, and enforced stop states.
- A cart-mounted user interface implemented as a locally running Node.js application, removing dependence on mobile devices.
- The inclusion of boycott item detection as an ethical and user-aware feature.

These contributions demonstrate how low-cost, commercially available components can be combined to deliver a robust and user-centred smart retail solution.

5.3 Interpretation and Implications of Results

The navigation results indicated that LiDAR-based localisation, supported by MPU data, was sufficient for indoor retail environments. While small positional errors were observed, these did not prevent the cart from reaching target locations or performing its intended function. This suggests that the chosen sensing and navigation approach is appropriate for prototype and near-commercial applications.

The weight verification results demonstrated that real-world tolerance is necessary when using consumer-grade load cells. The applied tolerance balanced reliability and practicality without compromising security. Importantly, the discussion highlights that higher-grade industrial weighing systems could further reduce measurement uncertainty, supporting scalability to commercial deployment.

Safety-related results, particularly Bluetooth monitoring and ultrasonic obstacle detection, had important implications for public deployment. The immediate halt on connection loss and obstacle proximity significantly reduced risk, reinforcing the importance of layered safety mechanisms in autonomous retail systems.

5.4 Limitations of the Work

Despite its successful operation, the system has several limitations:

- Navigation accuracy was affected by wheel slip, surface conditions, and sensor noise.
- Weight measurement accuracy was limited by the resolution of the load cell used.
- Item locations were predefined and did not adapt dynamically to store layout changes.
- Testing was conducted in a controlled environment rather than a live commercial store.
- The system relied on a single-cart setup and was not evaluated for multi-cart coordination.

These limitations are typical of prototype-stage systems and do not undermine the overall validity of the results.

5.5 Comparison with Existing Approaches

Compared to camera-based smart cart solutions, the proposed system avoided high computational cost and sensitivity to lighting conditions by relying on LiDAR and weight verification. While vision-based systems may offer higher automation, they introduce complexity and cost that may not be justified for all retail contexts.

In contrast to basic self-scanning carts, this system provided stronger theft prevention, autonomous navigation, and enhanced safety. The combination of features positions the proposed cart as a balanced solution between simplicity, cost, and functionality.

Chapter 6 Conclusions and Recommendation

6.1 Conclusion

This project successfully designed, implemented, and evaluated a functional prototype of a smart shopping cart that integrates autonomous navigation, secure item verification, user interaction, and digital payment into a single cohesive system. The primary objective of enhancing the shopping experience while improving safety, efficiency, and theft prevention was achieved.

The results demonstrated that the cart could reliably operate in both manual and automatic modes. In automatic mode, LiDAR-based localisation supported by MPU rotational data enabled accurate indoor navigation, while proximity-based item selection reduced unnecessary movement. Manual mode operation was stable and responsive, aided by smooth DAC-based motor control and obstacle detection using three ultrasonic sensors.

Item verification through weight measurement proved effective in distinguishing correct and incorrect item placement under realistic conditions. The application of tolerance values appropriately reflected real-world variability while maintaining security. The integration of RFID authorisation, RGB LED feedback, audible alerts, and total-weight validation before payment provided a layered and robust theft-prevention mechanism.

The Node.js application running locally on the Raspberry Pi enabled reliable user interaction, including sign-up and login, shopping list management, quantity handling, item deletion with weight recalculation, and mode switching. Secure digital payment using Stripe in test mode completed the end-to-end shopping process directly from the cart, eliminating the need for traditional checkout queues.

Overall, the project demonstrated that a low-cost, modular hardware and software architecture can deliver a practical and intelligent retail solution suitable for indoor environments.

6.2 Key Learnings

Through this project, several important conclusions were drawn:

- Distributed embedded architectures improve system stability by separating real-time control from high-level processing.
- LiDAR-based navigation, when combined with inertial sensing, is suitable for indoor retail environments.
- Consumer-grade load cells can be effectively used for item verification when realistic tolerances are applied.
- Layered safety mechanisms are essential for autonomous systems operating near the public.
- Integrating payment directly into the cart interface significantly enhances user experience and system completeness.

These findings reinforce the importance of system integration, safety-first design, and realistic engineering trade-offs in autonomous retail applications.

6.3 Recommendations for Performance Improvement

Several cost-effective and feasible improvements are recommended to enhance system performance:

- Replacing the load cell with a higher-precision industrial weighing mechanism to reduce tolerance requirements.
- Improving wheel encoders to further enhance navigation accuracy.
- Optimising ROS navigation parameters for dynamic obstacle handling in crowded environments.
- Refining the user interface to further simplify interaction for first-time or elderly users.

These improvements can be implemented without fundamentally changing the system architecture.

6.4 Future Work and Open Problems

Future work may extend the system beyond its current prototype capabilities. In the current implementation, item locations are predefined within the system. This can be improved by integrating dynamic mapping and smart shelf identification, allowing product locations to be updated automatically and reducing reliance on manual configuration.

Another potential extension is the integration of a robotic arm mounted on the cart. This would enable the cart not only to navigate autonomously but also to physically pick and place items into the cart, further reducing user effort and moving toward higher levels of automation.

Additional future enhancements include:

- Supporting remote shopping, where users can prepare shopping lists from home and allow the cart to perform item collection autonomously.
- Improving scalability through multi-cart coordination and collision avoidance.
- Integrating the system with store inventory databases for real-time product availability.
- Conducting extended testing in live retail environments and addressing regulatory and safety certification requirements.

Addressing these areas would improve system robustness and move the smart shopping cart closer to practical commercial deployment.

6.5 Final Remarks

This project demonstrates the feasibility of intelligent, autonomous shopping systems using commercially available hardware and open-source software tools. The smart shopping cart developed in this work provides a strong foundation for future research and development in automated retail, with clear pathways toward full autonomy and enhanced user convenience.

Appendices

Appendix A: Technical Deep Dive - ROS2-Based Autonomous Navigation System

A.1 System Architecture and Design Approach

The autonomous navigation system developed for the ShopMate smart shopping cart is based on a modular ROS2 (Robot Operating System 2) architecture. The design follows a clear separation of concerns, enabling seamless integration between user-facing software, high-level navigation logic, and low-level hardware control.

The system is structured as a four-tier computational stack:

- **Web Client Layer** – User interaction via an HTML/JavaScript interface
- **Nav2 Navigation Framework** – High-level path planning and goal execution
- **Custom Corner-Detection Controller** – Path-following logic optimised for retail environments
- **Serial Communication Bridge** – Interface between ROS2 and the Arduino-based motor controller

This layered architecture ensures that navigation decision-making remains independent of motor actuation and sensor hardware, thereby improving maintainability, scalability, and fault isolation across the system.

A.2 Navigation Flow and Operational Methodology

Autonomous navigation is executed through coordinated communication across ROS2 topics, services, and action servers, enabling reliable goal-driven motion.

User Interaction and Goal Publication

Users interact with a web-based interface that communicates with the ROS2 ecosystem via WebSocket bridging. When a shopping item is selected, the interface publishes a goal pose to the /web_goal topic using the geometry_msgs/PoseStamped message type.

Goal Forwarding and Localization

The nav_bridge node subscribes to /web_goal and forwards validated goals to the Nav2 action server /navigate_to_pose. Concurrently, the Adaptive Monte Carlo Localization (AMCL) node continuously estimates the robot's pose by publishing updates on /amcl_pose, maintaining real-time localization relative to the pre-built store map.

Coordinated Execution

This parallel operation ensures that path planning, execution, and localization remain synchronized, allowing the robot to continuously evaluate its current position relative to the target goal throughout navigation.

A.3 Path Planning and Control Strategy

Once a goal is accepted, the Nav2 stack computes a collision-free path between the robot's current pose and the target location. Path execution is handled by a custom C++ controller plugin, CornerController, designed specifically for structured retail environments.

Corner Detection and Dynamic Speed Control

The controller analyses sequential waypoints along the planned path to detect sharp directional changes exceeding 0.5 radians. When approaching a corner, linear velocity is reduced and angular velocity is increased to ensure precise turning. On straight path segments, the robot travels at a maximum linear speed of 0.25 m/s while applying proportional yaw correction (Kp = 1.5) to maintain alignment with the path.

Retail-Specific Optimisation

This dual-mode control strategy balances efficiency and precision, allowing rapid traversal of straight aisles while ensuring safe and accurate manoeuvring near shelves, corners, and constrained spaces.

A.4 Hardware Integration and Odometry Feedback

The CartSerialBridge node forms the interface between the ROS2 navigation stack and the Arduino-based motor control system.

Velocity Command Translation

The node subscribes to /cmd_vel (geometry_msgs/Twist) messages and translates linear and angular velocity commands into serial messages transmitted to the Arduino at 115,200 baud.

Telemetry Processing and Odometry Estimation

The Arduino provides telemetry feedback over the same serial link, including IMU-derived yaw measurements and a motor movement flag. The CartSerialBridge node performs dead-reckoning by integrating yaw data directly from the IMU while estimating linear displacement based on whether the robot is moving at a predefined maximum velocity (max_vx = 0.21 m/s) or stationary.

Transform Framework Integration

The resulting odometry is published to the /odom topic using nav_msgs/Odometry messages and broadcast as a TF2 transform from odom to base_link. This ensures that all ROS2 components share a consistent spatial reference.

A.5 Safety and Synchronization Mechanisms

To ensure reliable stopping behaviour and prevent unsafe state transitions, the system implements a topic-based synchronisation mechanism using /nav_done.

Dual-Redundancy Stop Enforcement

When a navigation action reaches a terminal state (success, cancellation, or failure), the nav_bridge node publishes nav_done = True. Both the CornerController and CartSerialBridge subscribe to this signal and immediately enter a latched stop state, forcing zero velocity output.

Stop Burst Reinforcement

To ensure that stop commands propagate reliably through the serial pipeline, the system transmits a burst of ten consecutive zero-velocity commands at 30 ms intervals, mitigating the effects of buffering or transient communication delays.

Cycle Reset and Goal Re-enablement

When a new goal is received, nav_done is reset to False, releasing both controllers from the stop latch and enabling the next navigation cycle. This dual-layer enforcement provides robustness against communication failures and state inconsistencies.

A.6 Initialization and Localization Setup

Before autonomous navigation can commence, the robot's initial pose must be defined.

Manual Pose Initialization

The web interface allows an operator to set the robot's starting pose by publishing a geometry_msgs/PoseWithCovarianceStamped message to the /initialpose topic.

AMCL Reset and Anchoring

AMCL uses this message to reset its particle filter and anchor the localization estimate to the specified location. This step is essential to prevent cumulative drift in systems relying on dead-reckoning.

System-Wide Consistency

Explicit initialization ensures that planners, controllers, and validators operate from a consistent spatial reference, enabling reliable reinitialization at any point during operation.

A.7 Distance-Based Goal Prioritization Algorithm

The web client implements an automated goal-selection mechanism to optimise shopping efficiency.

Greedy Nearest-Neighbour Strategy

When multiple shopping items are selected, the system retrieves all unchecked items from the Firebase database. The sendNextGoal() function computes the Euclidean distance between the robot's current position (from /amcl_pose) and each item's stored map coordinates, selecting the closest unchecked item as the next goal.

Travel Distance Optimisation

This greedy nearest-neighbour approach minimises total travel distance and reduces overall shopping time, improving system usability and user experience.

Alignment with System Objectives

This algorithm directly supports the project objective outlined in Section 1.2 of the main report, prioritising navigation efficiency over the order in which items are added to the shopping list.

A.8 System Validation and Robustness

Multiple defensive programming strategies are incorporated to enhance reliability and fault tolerance:

- **Serial Communication Safety** – Mutex-based locking prevents concurrent read/write conflicts
- **Race Condition Prevention** – Explicit state checks distinguish between navigation completion and new goal initiation
- **Communication Efficiency** – Velocity command caching avoids redundant serial transmissions
- **Graceful Degradation** – Corner detection safely defaults to forward motion when path data is insufficient
- **Defensive Bounds Checking** – Prevents index and buffer overrun errors

These measures ensure stable operation under communication latency, sensor noise, and unexpected planning behaviour, reinforcing the system's safety-first design philosophy.

References

1. Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic Robotics. MIT Press.
2. Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. IEEE Intelligent Transportation Systems Magazine, 2(4), 31–43.
3. Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo localization for mobile robots. In Proceedings of IEEE ICRA.
4. Open Robotics. (2025). ROS 2 Documentation (Humble/Jazzy). https://docs.ros.org/
5. Slamtec. (2018). RPLIDAR A1—Datasheet. https://www.slamtec.com/
6. Luxonis. (2024). DepthAI / OAK-D Documentation. https://docs.luxonis.com/
7. Avia Semiconductor. (2016). HX711 Load Cell Amplifier—Datasheet.
8. RobotShop. (2023). Load Cells—Technical Overview.
9. GY Robotics. (2020). GRF-60 Laser Scanner—Datasheet.