CHAPTER 1

INTRODUCTION

From the past few years offline shopping is going down due to various reasons such as Covid-19 pandemic, offers available in online shopping and offline shopping has become very time consuming. To improve the shopping experience of customers in offline shopping, we are creating a smart cart system.

The proposed smart cart system is an innovative technological solution designed to enhance the traditional shopping experience. It consists of 300rpm motor which helps the cart system to move. We are adding convolutional neural network (CNN) program code into Raspberry Pi. 300rpm motor is connected to an actuator, and Arduino GSM module followed by a GPS module, RFID tags and RFID reader, buzzer, LCD are also part of the Arduino GSM module. The actuator will control the movement of the motor.

1.1. DESIGN

Designing a smart cart involves integrating various technologies to create a seamless and efficient shopping experience. Here's an outline of the design elements typically incorporated into a smart cart:

1. **Physical Structure:**
    - The physical structure of the smart cart should be sturdy, spacious, and ergonomic to accommodate various types and sizes of items while being easy to maneuver through aisles.
    - It should have a comfortable handle and smooth-rolling wheels for easy navigation within the store.
2. **Sensors:**
    - RFID (Radio-Frequency Identification) sensors or computer vision systems are integrated into the cart to identify items placed inside it.
    - Weight sensors may also be used to detect the addition or removal of items from the cart.
3. **Display and Interface:**
    - A user-friendly display screen is mounted on the cart, providing real-time information to the shopper.
    - The interface may include buttons or a touchscreen for shoppers to interact with the cart, view their shopping list, receive personalized recommendations, or check the total cost of items in the cart.
4. **Power Source:**
    - The cart is powered by a rechargeable battery to support its electronic components.
    - Charging ports may be included for shoppers to charge their devices while shopping.

5. **Security Features:**

    - Anti-theft alarms or sensors are integrated into the cart to detect and deter unauthorized removal of items.
    - RFID blocking technology may be incorporated to prevent RFID skimming or unauthorized access to shopper data.

6. **Personalization:**

    - The smart cart may use data analytics and machine learning algorithms to provide personalized recommendations to shoppers based on their purchase history and preferences.
    - Personalized promotions or discounts may be displayed on the cart's interface to enhance the shopping experience.

7. **Accessibility Features:**

    - Consideration should be given to designing the smart cart to be accessible to all shoppers, including those with disabilities or mobility impairments.
    - Features such as adjustable height handles or voice-activated controls may be included to accommodate different needs.

By incorporating these design elements, a smart cart can provide an enhanced shopping experience for customers while improving efficiency and convenience for both shoppers and retailers.

1.2. Working Principle

The working principle of a smart cart involves the integration of various technologies to enhance the shopping experience. Here's a breakdown of the key components and their functionalities:

1. **Sensors:** Smart carts are equipped with various sensors such as RFID (Radio-Frequency Identification), computer vision, or weight sensors. These sensors detect and identify items placed in the cart.

2. **RFID and The Raspberry Pi3 camera 5mps:**

    - RFID tags are attached to each item in the store. As items are placed in the cart, RFID readers embedded in the cart detect these tags and register the items in the cart's system.

    - Alternatively, smart cart use cameras and machine learning algorithms to identify people. Cameras installed in the cart capture images of the people, and image processing software recognizes and logs them in the cart's system.

3. **Item Tracking:** The smart cart's system keeps track of the items placed inside it in real-time. This information is then used for inventory management, personalized recommendations, and checkout purposes.

4. **Personalization:** Some smart carts utilize data collected from a shopper's past purchases to provide personalized recommendations or discounts in real-time. This enhances the shopping experience by offering tailored suggestions based on the shopper's preferences.

5. **Power Source:** Smart carts are powered by rechargeable batteries to support their electronic components and ensure uninterrupted functionality during shopping trips.

6. **Security:** To prevent theft or tampering, smart carts may incorporate security features such as anti-theft alarms or RFID blocking to detect and deter unauthorized removal of items.

Overall, the working principle of a smart cart involves the seamless integration of sensors, data processing algorithms, and wireless connectivity to enhance the shopping experience, improve inventory management, and streamline the checkout process.

1.2.1. Machine Learning used in Smart Cart

Machine learning (ML) plays a crucial role in enhancing the capabilities of smart carts, enabling them to provide personalized experiences, optimize inventory management, and improve overall efficiency. Here are several ways in which machine learning is utilized in smart carts:

1. Personalized Recommendations:
    - Machine learning algorithms analyze past purchase history and browsing behavior to generate personalized product recommendations for shoppers.
    - By understanding shoppers' preferences, ML models can suggest complementary or related items, leading to increased sales and customer satisfaction.
2. Predictive Analytics:
    - Machine learning algorithms can analyze historical sales data, seasonal trends, and other factors to predict future demand for products.
    - This enables retailers to optimize inventory levels, minimize stockouts, and reduce overstocking, leading to improved supply chain management and cost savings.
3. Checkout Optimization:
    - ML models can be used to optimize the checkout process by automatically detecting and scanning items placed in the smart cart.
    - Computer vision algorithms can identify products using image recognition techniques, while natural language processing (NLP) models can interpret voice commands or text input from shoppers.
    - This streamlines the checkout process, reducing wait times and enhancing the overall shopping experience.
4. Anomaly Detection:
    - Machine learning algorithms can detect anomalies or unusual patterns in shopping behavior, such as suspicious activities or potential theft.
    - By analyzing real-time data from sensors and cameras, ML models can flag irregularities and alert store staff to take appropriate action, improving security and loss prevention.

5. Dynamic Pricing:

    - Machine learning algorithms can analyze various factors, such as demand, competitor pricing, and customer demographics, to optimize pricing strategies in real-time.
    - Dynamic pricing models can adjust prices dynamically based on market conditions, maximizing revenue and profitability for retailers.

6. Sentiment Analysis:

    - Machine learning techniques such as sentiment analysis can analyze customer reviews, feedback, and social media data to understand shopper sentiment and preferences.
    - This information can be used to tailor marketing campaigns, product offerings, and customer service initiatives, enhancing customer satisfaction and loyalty.

Overall, machine learning enables smart carts to deliver personalized experiences, optimize operations, and drive business value for retailers by leveraging data-driven insights and predictive analytics.

1.3 Comparison between Traditional Cart and Smart Cart

Traditional shopping carts and smart carts represent two different approaches to the shopping experience, each with its own set of advantages and drawbacks. Here's a comparison between the two:

Traditional carts:

1. **Functionality:** Traditional shopping carts are simple in design and functionality. They provide a basic container for shoppers to place their items as they move through the store.
2. **Convenience:** Traditional carts require shoppers to manually scan and bag each item at the checkout counter, which can be time-consuming, especially for large purchases.
3. **Inventory Management:** Traditional carts do not provide any means for tracking inventory levels or product information.
4. **Personalization:** Traditional carts do not offer any personalized features for shoppers.
5. **Cost:** Traditional shopping carts are generally inexpensive to manufacture and maintain.
6. **Privacy and Security:** Traditional shopping carts do not collect any personal data about shoppers.
7. **User Experience:** The user experience with traditional carts is familiar and straightforward.

Smart carts:

1. **Functionality:**
    - Smart carts are equipped with advanced technology such as RFID (Radio-Frequency Identification) or computer vision systems.
    - They may include features like automatic item scanning, real-time inventory tracking, and personalized recommendations.

2. **Convenience:** Smart carts streamline the checkout process by automatically scanning items as they are placed in the cart. This can significantly reduce checkout times and improve overall convenience for shoppers.

3. **Inventory Management:** Smart carts can track inventory levels in real-time, providing valuable data to store managers for restocking and inventory management purposes.

4. **Personalization:** Smart carts can leverage data about a shopper's past purchases to provide personalized recommendations or discounts in real-time.

5. **Cost:** Smart carts are more expensive to develop and deploy due to the advanced technology they incorporate.

6. **Privacy and Security:** Smart carts may collect data about shoppers' purchasing habits and preferences, raising concerns about privacy and data security.

7. **User Experience:** Smart carts may require some adjustment for users accustomed to traditional shopping methods, but they offer a more technologically advanced and potentially seamless experience once users become familiar with them.

8. In summary, while traditional shopping carts are simple and familiar, smart carts offer advanced features such as automatic scanning and real-time inventory tracking, improving convenience and efficiency for both shoppers and retailers. However, they also raise concerns about privacy, data security, and cost.

CHAPTER 2

LITERATURE SURVEY

In [1], Alexander A S Gunawan et al (2019): Researchers have developed a smart cart system capable of tracking a user's location and moving autonomously. In the design of this cart, they have incorporated an IOIO microcontroller, an Android smartphone as sensors, and a two-wheeled mobile robot. The smartphone's compass is utilized for the robot's navigation. Furthermore, this system is equipped with an indoor positioning system that detects the user's position by using the Navisens framework, which relies on the smartphone's gyroscope and accelerometer. One significant limitation of this system is its lack of stability when the user is situated beyond a 3-meter radius, and it struggles to follow planned paths.

In [2], Apurva Sutar et al (2023): The proposed system comprises a camera that employs deep learning techniques to detect commodities and a load cell for measuring the weight of items added to the shopping cart. The YOLO model was utilized for image recognition. However, a major drawback of this system is its only focus on edible objects, such as fruits and vegetables.

In [3], Muhammad Atif Sarwar et al (2020): The self-checkout cart system, introduced by the researchers, is named "iCart". This innovative system is based on mobile cloud computing and deep learning cloud services. The Linux-based cloud server contained the YOLOv2 deep learning network. However, a notable drawback of this system is the necessity for the camera to record a video and transmit it to the cloud server for classification and segmentation, it is a time-consuming process.

In [4], Narayanan V Ramanathan et al (2020): The implemented smart cart system incorporates a computer vision unit capable of imaging a surveillance region to check the cart's load status, whether it is empty or not. This computer vision unit comprises a camera and an image processor programmed to execute the computer vision algorithm for load status determination. However, a significant limitation of this system lies in its exclusive focus on self-checkout functionality, while the primary purpose of a shopping cart, namely shopping assistance, remains unaddressed in this system.

In [5], Xuan Liu et al (2016): The system developed by the researchers can autonomously track its user by identifying the color and shape characteristics of the attire the user is wearing. They have created a mathematical model for dynamic target tracking. However, a significant limitation of this system is its predominant focus on the billing system. The accuracy of the image recognition employed in this system is insignificant.

In [6], Siddharth Sanghavi et al (2020): The proposed system is a smart shopping cart designed for human tracking. In the design of this cart, the researchers have incorporated a magnetometer, ultrasonic sensors, and a Microsoft Kinect sensor. Notably, they used magnetometers instead of IR sensors and cameras. However, a significant drawback of this system arises when the connection between the user and the cart is interrupted, as it can lead to the cart becoming immobile at its current location.

2.1 Problem Statement

Design and development of smart cart system which enhances the overall shopping experience of the customers by using machine learning algorithms, efficient payment system and smooth autonomous movement of the cart system. The customers' experience can be modelled using Bayesian Network.

2.2 Objectives

Shopping has become a very time consuming and boring activity, to address this problem we propose to develop a smart cart system that can perform following functions.

- Automated Navigation: Implementing machine learning algorithms for obstacles detection and avoidance to enable smart cart to navigate autonomously.
- User Interaction: Creating a website so that users can interact with smart cart system more conveniently.
- Easy payment system: RFID based payment systems use wireless technology to process transactions quickly and efficiently.
- Object Detection: convolution neural networks are used for object detection purposes; this enables cart system to manage inventory efficiently.
- Shopping Experience of Customers: Overall shopping experience of customers can be modelled using Bayesian network.

CHAPTER 3

METHODOLOGY

3.1. Smart cart system

Fig.1.1 depicts the methodology involved in the smart cart system we proposed. It involves various components that are used in the smart cart system. As you see in fig.1.1 camera capture the product and send it to the raspberry pi4 module as the input. Raspberry pi evaluates the data and send it to the Arduino GSM module. Arduino controls the motor system based on the input data from the Raspberry pi.

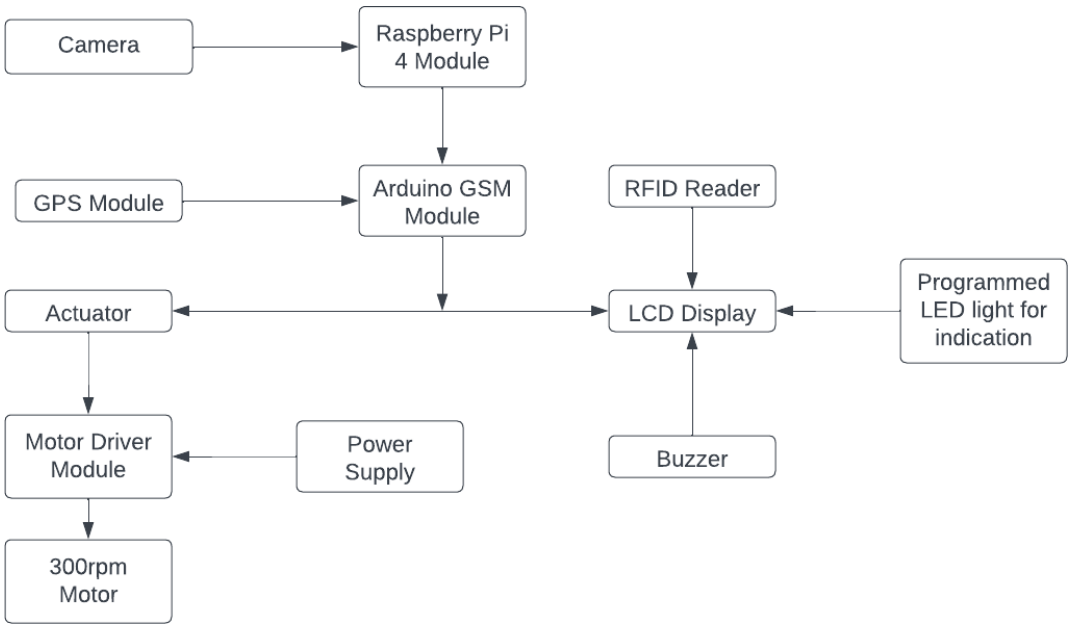

Figure 3.1: The Proposed System

1. Camera

The camera enables the cart system to visually perceive its surroundings, facilitates tasks such as object detection and navigation. The camera captures and processes real time visual data and enhances the cart systems functionality.

2. Raspberry Pi Module Integration

The Raspberry Pi 4 module serves as the central processing unit of the smart cart system, playing a very crucial role in various functions. Upon activation, the module receives signals from the camera module, providing the real time visual information about the cart systems surroundings.

3. GPS and GSM Module Integration

To ensure accurate and precise location tracking, the GPS (Global Positioning System) and GSM (Global System for Mobile Communication) models are integrated with the Arduino UNO microcontroller.

4. Motor Driver Module [L298N]

The motor driver module, specifically L298N, acts as a crucial component for controlling the cart systems movement.

5. Arduino Microcontroller

The Arduino acts as the brain for the locomotion of the cart. The cost-effective integration of Arduino allows the cart system to navigate automatically and ultimately improving the overall efficiency of the smart cart system.

3.2. Overall Shopping Experience

The overall shopping experience of the customers is modelled using Bayesian network as shown in fig.1.2. The shopping experience of the customers depends on conditional probability of th both software and hardware components.

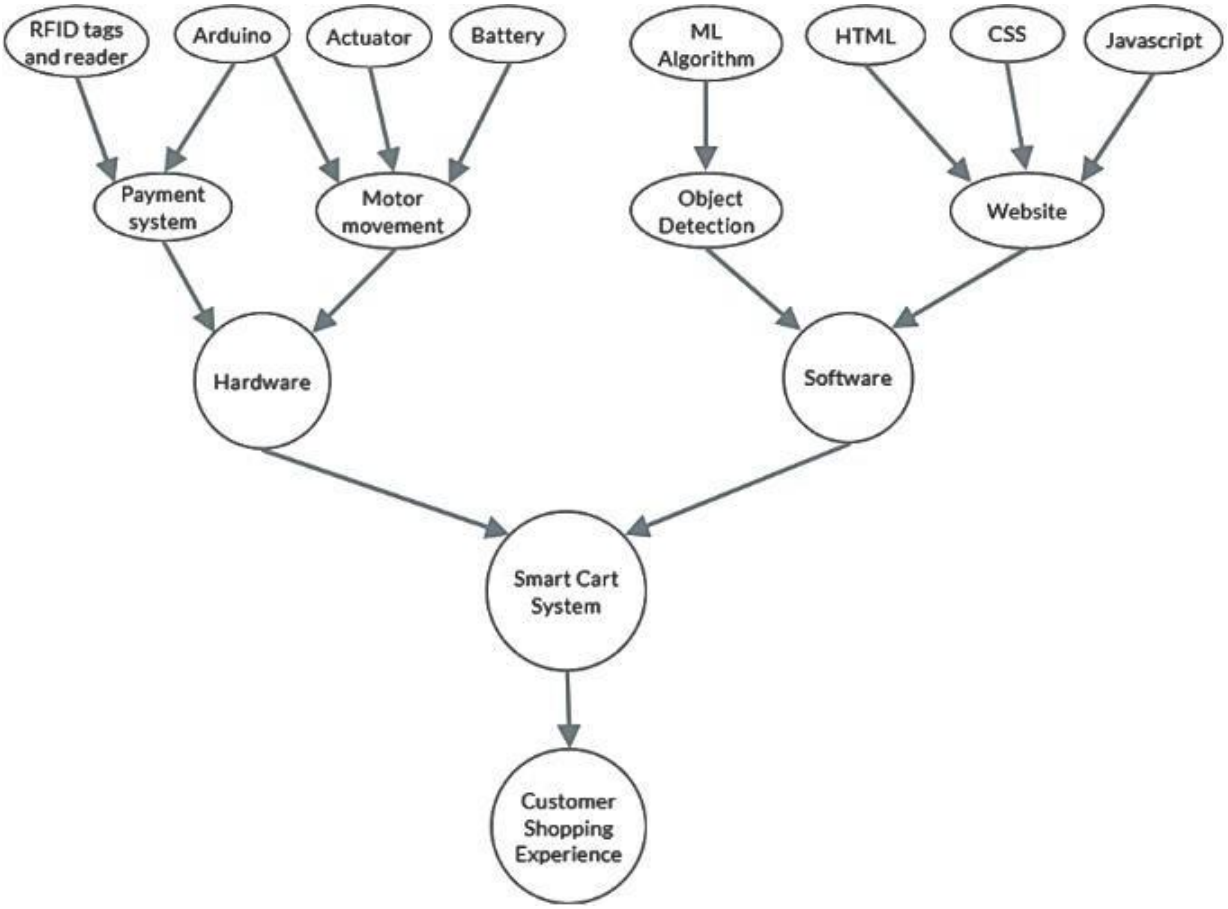

Figure 3.2: **Bayesian Network**

A Bayesian network is a probabilistic graphical model for representing uncertain domain where each node corresponds to a random variable and each edge represents the conditional probability for the corresponding random variable.

Bayesian network can be represented as

P(B/A) = P (A and B) / P(A)

P(A) = Occurred event

P(B) = Uncertain event

3.3 Project Requirements

3.3.1 Raspberry Pi

Raspberry Pi is a series of small, affordable, single-board computers developed by the Raspberry Pi Foundation, a UK-based charity organization. These credit card-sized computers are designed to promote the teaching of basic computer science and programming skills in schools and developing countries, as well as to serve as versatile platforms for many projects and prototyping.

Various aspects of Raspberry Pi:

1. **Hardware:**

   - Raspberry Pi boards typically consist of a single circuit board with a CPU, RAM, GPU, USB ports, HDMI output, Ethernet port (in some models), GPIO (General Purpose Input/Output) pins, and a power connector.
   - The components may vary slightly between different models, with variations in CPU, RAM, and connectivity options.
   - Raspberry Pi boards are powered by ARM-based processors, which offer a good balance between performance and power efficiency.
   - The GPIO pins allow users to connect external sensors, actuators, and other electronic components for hardware interfacing and DIY projects.

2. **Operating System and Software:**

   - **Raspberry Pi supports various operating systems, including:**
     - Raspberry Pi OS (formerly Raspbian): A Debian-based Linux distribution optimized for Raspberry Pi.
     - Ubuntu, Fedora, and other Linux distributions can also be installed on Raspberry Pi.
     - Windows 10 IoT Core: A lightweight version of Windows 10 designed for IoT (Internet of Things) applications.
     - A wide range of software packages and development tools are available for Raspberry Pi, including programming languages (Python, C/C++, Java), development environments (IDLE, Thonny, Visual Studio Code), and

libraries/frameworks for various purposes (TensorFlow for machine learning, OpenCV for computer vision, etc.).

3. Applications and Use Cases:

- **Education:** Raspberry Pi is widely used in educational settings to teach programming, electronics, and computer science concepts. It provides an affordable and accessible platform for hands-on learning and experimentation.
- **DIY Projects:** Raspberry Pi is popular among hobbyists and makers for building various DIY projects, such as home automation systems, media centers, retro gaming consoles, weather stations, and IoT devices.
- **Prototyping:** Raspberry Pi is used for rapid prototyping of electronic devices and IoT solutions. Its GPIO pins allow for easy integration with sensors, actuators, and other hardware components.
- **Server and Networking:** Raspberry Pi can be used as a low-cost server for hosting websites, file servers, media servers, and other network services. It can also serve as a router, VPN server, or wireless access point.
- **Robotics:** Raspberry Pi is commonly used as the brain of robotic projects, providing computation power and connectivity for controlling motors, sensors, and other components.

4. Community and Support:

- Raspberry Pi has a large and active community of users, developers, and enthusiasts who share knowledge, tutorials, and project ideas through online forums, social media, and community events.
- The Raspberry Pi Foundation provides official documentation, tutorials, and resources to support users in getting started with Raspberry Pi and exploring its capabilities.

Overall, Raspberry Pi offers a versatile and affordable platform for learning, experimentation, and innovation in the fields of computer science, electronics, and DIY projects. Its simplicity, accessibility, and community support make it an ideal choice for beginners and experienced users alike.

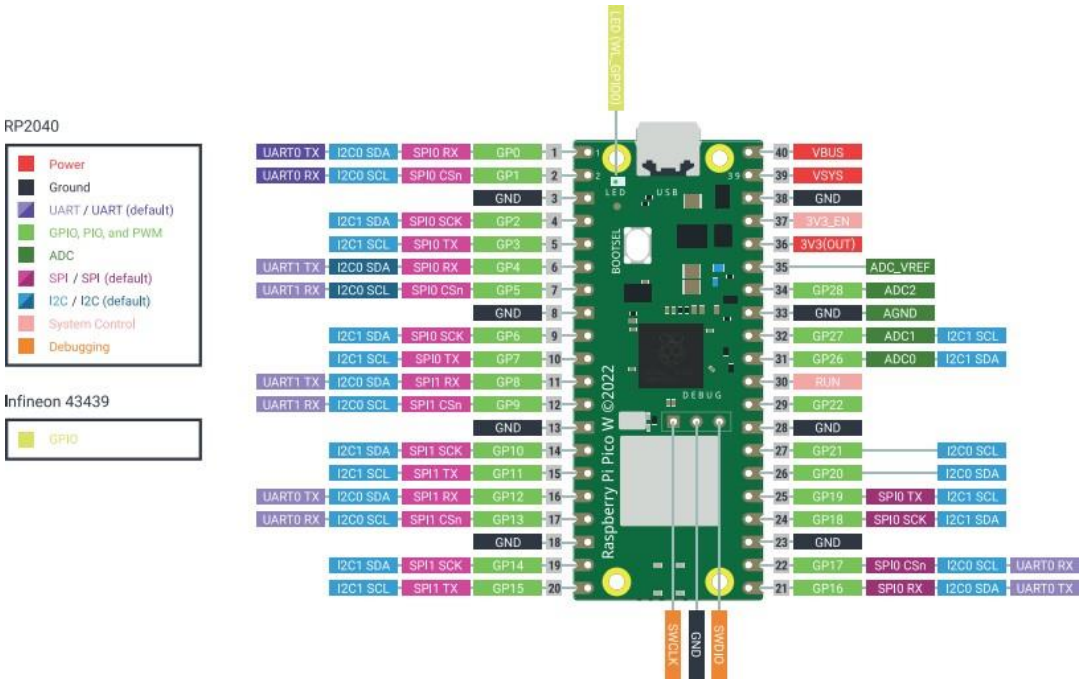

Fig.3.3: Raspberry pi

3.3.2 Arduino IDE

Arduino represents an open-source platform catering to a wide spectrum of electronic projects, encompassing both hardware and software IDEs. Its user-friendly interface and simplicity make it an ideal choice for beginners, requiring nothing more than a USB cable to upload code. Utilizing a straightforward C++ syntax, Arduino streamlines microcontroller operation, simplifying complex tasks for users of varying skill levels. Within the Arduino community, members actively contribute to the development of microcontroller-based modules, fostering collaboration and innovation.

In contrast to the assembly language utilized in AVR Studio, the Arduino IDE is written in C, offering access to an extensive library. This vast library provides users with a plethora of resources, enriching the programming experience and expanding the possibilities for project realization.

1. **Extensive community support:**

   - The Arduino platform enjoys the support of a robust and engaged community comprising developers, enthusiasts, and industry professionals.
   - Within this dynamic ecosystem, members actively contribute by sharing projects, tutorials, and invaluable problem-solving insights.
   - This collaborative environment not only promotes knowledge dissemination but also fosters innovation in various domains, including energy development and the implementation of solutions like the smart cart.
   - Through collective efforts and shared expertise, the Arduino community continues to drive advancements and inspire creativity in electronic projects and beyond.

2. **Cross-platform compatibility:**

   - The Arduino IDE boasts compatibility with a diverse range of operating systems, including Windows, macOS, and Linux, ensuring widespread accessibility to users across various platforms.
   - This seamless cross-platform integration offers simplicity and user-friendliness, empowering developers to leverage their preferred workflows with ease.
   - By providing a unified development environment regardless of the operating system, Arduino facilitates efficient and streamlined programming experiences for its diverse user base.

3. **Integrated Development Environment (IDE):**

   - The Arduino IDE offers an intuitive and user-friendly platform for composing, compiling, and exporting code tailored for Arduino-compatible microcontrollers.
   - Its streamlined interface and integrated code editor contribute to a straightforward development process, particularly beneficial for novices and individuals with limited experience in programming.
   - This cohesive environment fosters efficient code creation and enhances accessibility, empowering users of all proficiency levels to embark on their programming endeavors with confidence.

4. **Simple programming language:**

- The Arduino IDE adopts a straightforward C++ programming language model, tailored to accommodate beginners and individuals with minimal programming proficiency.
- Renowned for its simplicity and accessibility, this syntax facilitates rapid comprehension and learning, thereby expediting the development process.
- Its user-friendly nature empowers users with limited programming experience to navigate the software efficiently, promoting faster and more efficient coding practices.

5. **Plug and play functionality:**

- The Arduino board seamlessly interfaces with a computer through a USB cable, facilitating effortless loading and debugging of program code.
- This plug-and-play functionality streamlines the installation process, alleviating the necessity for intricate hardware configurations.
- Such simplicity renders the Arduino IDE particularly well-suited for expedited prototyping endeavors within airship projects.

6. **Rich library support:**

- The Arduino IDE offers a comprehensive array of pre-written codes and libraries, encompassing a multitude of functions and device functionalities.
- These libraries serve to streamline the development process and minimize application efforts by simplifying tasks such as sensor interaction, actuator control, and external device communication.
- By leveraging these resources, developers can significantly reduce development time while ensuring efficient implementation of various functionalities within their projects.

7. **Open Source Platform:**

- The Arduino IDE stands as an open-source platform, fostering an environment where users can tailor and enhance its capabilities to suit their individual requirements.
- This openness cultivates a culture of innovation and collaboration within the Arduino community, promoting the continual development and refinement of new features.
- Such a collaborative ethos serves to drive progress and adaptation, particularly in the context of advancing functionalities relevant to airship applications.

8. **Integration with sensors and modules:**

- The Arduino IDE facilitates seamless integration with a diverse range of sensors, modules, and peripherals frequently employed in aerospace applications.
- This capability enables developers to effortlessly incorporate temperature sensors, humidity sensors, GPS modules, and wireless communication modules into their projects.
- Such integration empowers the acquisition of vital weather data from airships, facilitating its transmission to ground stations or weather servers for analysis and dissemination.

9. **Hardware Compatibility:**

- The Arduino IDE boasts compatibility with a wide array of Arduino-compatible microcontroller boards, encompassing popular models such as the Arduino Uno, Arduino Nano, and Arduino Mega.
- Renowned for their affordability and versatility, these boards offer ample functionality for a myriad of applications, including integration within airships.
- Their robust features and cost-effectiveness render them well-suited for deployment in such contexts, aligning with the demands of airborne projects.

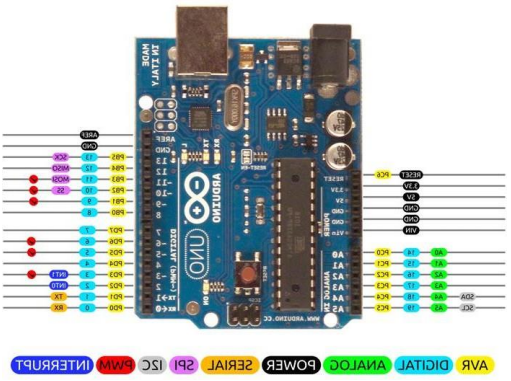

Fig.3.4: Arduino

Fig.3.5: Raspberry pi

3.3.3 Details of the components

The hardware and software tools that power smart carts are:

1. Raspberry Pi3 camera 5mps:

- **Resolution and Megapixels:**
  - The Raspberry Pi Camera Module V2 features a 5-megapixel Omnivision OV5647 sensor.
  - It captures images with a maximum resolution of 2592 × 1944 pixels, providing sharp and detailed photos.
- **Video Capabilities:**
  - In addition to capturing still images, the camera module supports video recording.
  - It can record video in various resolutions and frame rates, including 1080p (Full HD) at 30 frames per second (fps) and 720p (HD) at 60 fps.
- **Lens and Field of View:**
  - The camera module comes with a fixed-focus lens, providing a standard field of view suitable for a wide range of applications.
  - It has a horizontal field of view (FOV) of approximately 54 degrees.

- **Connectivity:**
    - The camera module connects directly to the Raspberry Pi board via a dedicated ribbon cable.
    - It utilizes the MIPI CSI-2 (Mobile Industry Processor Interface Camera Serial Interface 2) interface for high-speed data transfer and communication between the camera and the Raspberry Pi.
- **Image Quality and Features:**
    - The camera module delivers high-quality images with good color reproduction and clarity.
    - It supports various image adjustments and features, including exposure control, white balance adjustment, and image effects (e.g., grayscale, sepia).
    - Users can adjust camera settings programmatically using the Raspberry Pi Camera Module's official Python library (picamera) or through command-line tools.
- **Low-Light Performance:**
    - While not specifically designed for low-light conditions, the Raspberry Pi Camera Module V2 performs reasonably well in moderate lighting environments.
    - For improved low-light performance, users can augment the camera module with external lighting sources or utilize software-based enhancements.
- **Applications:**
    - The Raspberry Pi Camera Module V2 finds applications in various fields, including:
    - Photography and videography projects, including time-lapse photography and surveillance systems.
    - Robotics and computer vision applications, such as object detection, facial recognition, and gesture recognition.
    - Scientific experiments and research projects requiring image or video capture.

- IoT (Internet of Things) projects involving remote monitoring, environmental sensing, and image-based analytics.

Fig. 3.6: Raspberry Pi3 camera 5mps

2. RFID Tags/Reader:

RFID Tags:

- RFID tags are small electronic devices containing a microchip and an antenna.
- They come in various form factors, including cards, labels, stickers, and implants.
- RFID tags can be either passive, active, or semi-passive.
- Passive RFID tags do not have an internal power source and rely on energy harvested from the RFID reader's electromagnetic field to transmit data.
- Active RFID tags have their own power source (e.g., battery) and can actively transmit data to RFID readers.
- Semi-passive RFID tags have a battery for powering certain functions (e.g., onboard sensors) but rely on the RFID reader's energy for data transmission.
- RFID tags can store unique identifiers (e.g., serial numbers) and additional data such as product information, expiration dates, or maintenance records.
- They operate at various frequencies, including low frequency (LF), high frequency

(HF), and ultra-high frequency (UHF), each offering different read ranges and data transfer rates.

- RFID tags are commonly used in applications such as inventory management, asset tracking, access control, payment systems, and supply chain management.

RFID Readers:

- RFID readers, also known as interrogators or scanners, are devices used to communicate with RFID tags.
- They emit radio waves via an antenna to energize nearby RFID tags and read the data stored on them.
- RFID readers come in different types and form factors, including handheld readers, fixed readers, and integrated readers (e.g., built into access control systems or point-of-sale terminals).
- Readers can be designed to operate at specific frequencies to match the frequency of the RFID tags they are intended to read.
- RFID readers can be further categorized based on their functionality and connectivity.
- Basic readers: Simply read data from RFID tags and transmit it to a computer or backend system for processing.
- Smart readers: Have additional processing capabilities and can perform tasks such as filtering data, performing data encryption, or triggering events based on tag reads.
- Network-connected readers: Connect to local networks or the internet, enabling real-time data transfer and remote management.
- RFID readers can be integrated with other systems and technologies, such as barcode scanners, GPS, sensors, and databases, to enhance functionality and enable more advanced applications.

Operation:

- When an RFID tag comes within range of an RFID reader, it receives electromagnetic energy from the reader's signal.
- The tag uses this energy to power its microchip and antenna, allowing it to transmit

data back to the reader.

- The reader captures the data transmitted by the tag and forwards it to a computer or backend system for further processing and interpretation.
- Depending on the application, RFID systems can be configured for various modes of operation, including inventory tracking, access control, authentication, and data logging.

RFID technology provides a powerful means of automatically identifying and tracking objects in various applications. RFID tags and readers work together seamlessly to enable efficient data capture, enabling businesses and organizations to streamline operations, improve visibility, and enhance security.

Fig. 3.7: RFID Reader/Tags

CHAPTER 4

Result and Discussion:

The Raspberry Pi3 camera 5mps successfully captured the real-time objects and people. Based on the data sent by the Raspberry Pi3 camera 5mps, motors will automatically navigate either forward or backward based on the data. The motors are connected to the Arduino and Raspberry Pi and Motor driver. RFID tags and reader are connected to the Arduino. . RFID tags and reader used for payment process are fully functional, fulfilling the project's objectives.

4.1. Users feedback after using smart cart

Table 1 : "Feedback about Cart Movement"

Performance	Percentage
Poor	16.6
Good	50
Excellent	33.3

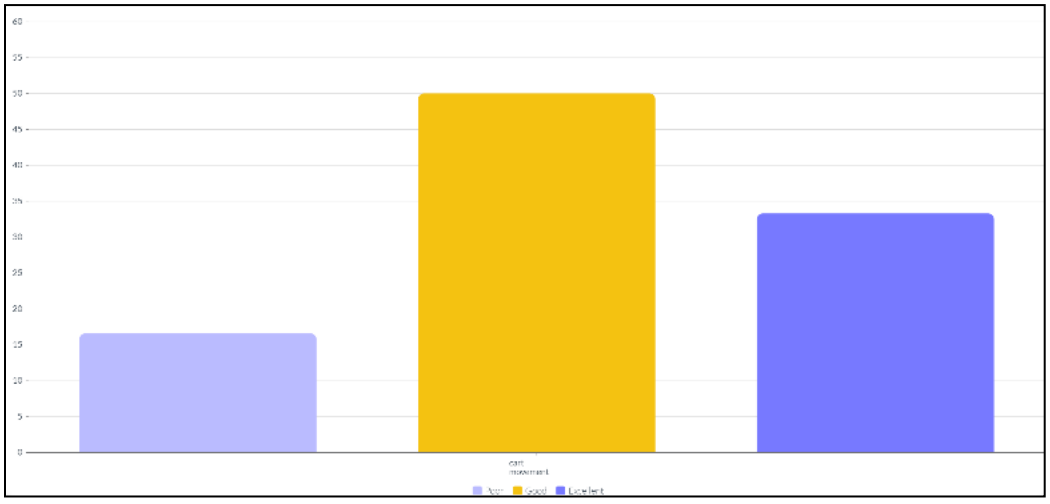

Fig. 4.1: Graph for feedback about cart movement

Table 2: "Feedback about Payment System"

Performance	Percentage
Poor	20
Good	60
Excellent	20

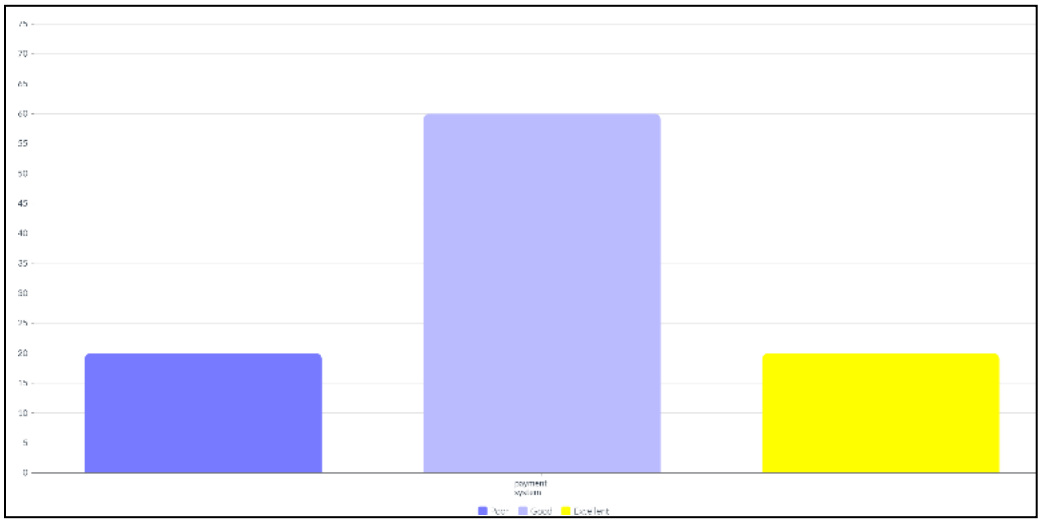

Fig. 4.2: Graph for feedback about payment system

Table 3 : "Time consumed by traditional cart vs smart cart"

Avg. time consumed by traditional cart	10	15	20	25
Avg. time consumed by smart cart	6	9	12	15

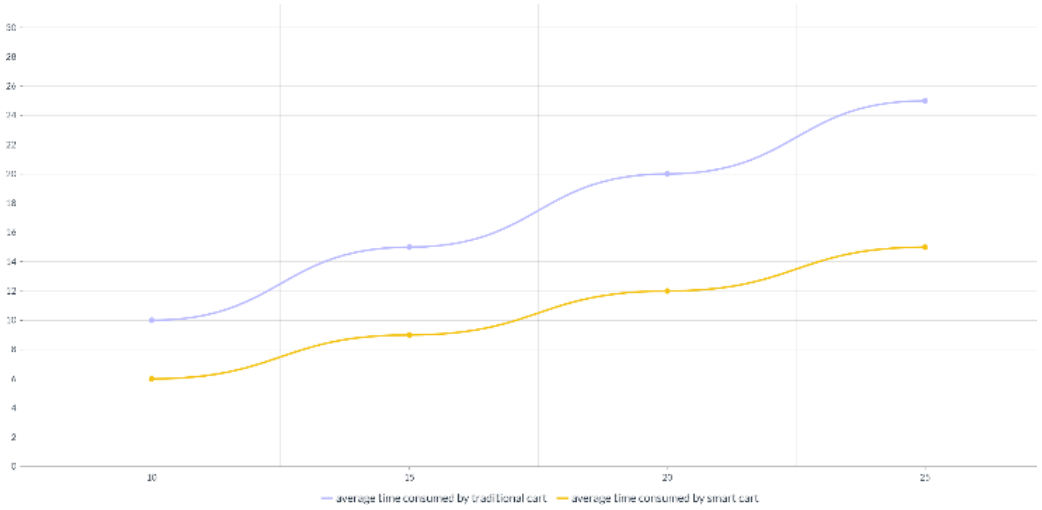

Fig. 4.3: Graph for avg. time consumed by traditional cart vs smart cart

CHAPTER 5

Conclusion

The design and development of smart cart based on machine learning present a promising and innovative solution for enhancing the shopping experience. The project not only showcases the potential of machine learning in everyday applications but also sets the stage for future advancements in retail industries.

We are implementing convolution neural networks for object detection. Our smart cart system features an adaptive navigation system, it guides the users through the store efficiently, suggesting optimal routes based on their shopping list and the store layout. This not only saves time but also ensures a smooth and convenient shopping experience.

From retailers' perspective, the integration of machine learning aids in inventory management. The project represents a practical and impactful application of machine learning in retail sector, offering benefits to both users and shop owners.

We are also modelled the overall shopping experience of the customers and it can be analysed using the Bayesian network.

CHAPTER 6

Future Scope

1. Enhanced Personalization: Further refinement of machine learning algorithms to provide even more accurate and personalized product recommendations based on individual preferences and shopping history.

2. IoT Integration: Incorporation of Internet of Things (IoT) devices to enable seamless communication between the smart cart and other connected devices within the retail environment.

3. Mobile App Integration: Development of a companion mobile app that syncs with the smart cart, allowing users to create shopping lists, receive real-time promotions, and track their purchases even when not physically present at the store.

4. Checkout Automation: Integration with mobile payment systems and digital wallets for a completely streamlined and contactless payment experience.

5. Data Analytics and Insights: Utilization of data for predictive analysis to anticipate and meet customer demands effectively.

6. Accessibility Features: Inclusion of features to make the smart cart more accessible to individuals with disabilities, such as voice-activated controls, text-to-speech capabilities, and tactile feedback.

References

1. Alexander A S Gunawan, Valdi Stevanus, Albertus Farley, Heri Ngarianto, Widodo Budiharto, Herman Tolle, Muhammad Attamimi, “Development of smart trolley system based on android smartphone sensors”, 4th International Conference on Computer Science and Computational Intelligence 2019.

2. Apurva Sutar, K. J. Karande, A. D. Harale, “Deep Learning based Automated Billing Cart” in International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) (2023).

3. Muhammad Atif Sarwar, Yousef-Awwad Daraghmi, Kuan-Wen Liu, Hong-Chuan Chi, Ts`ı-U´ı ˙Ik, Yih-Lang Li “Smart Shopping Carts Based on Mobile Computing and Deep Learning Cloud Services” Department of Computer Science, College of Computer Science, National Chiao Tung University 1001 University Road, Hsinchu City 30010, Taiwan (2020).

4. Narayanan V. Ramanathan, Scott J. Carter, Stephen E. Hannah, “shopping basket monitoring using computer vision and machine learning” United States Patent Application Publication (2020).

5. Xuan Liu, Haitao Zhang, Jingxian Fang, Guan Guan, Yundi Huang, “intelligent shopping cart with quick payment based on dynamic target tracking” Proceedings of CCIS2016 (2016).

6. Siddharth Sanghavi, Palash Rathod, Preet Shah, Dr. Narendra Shekokar, “Design and implementation of a human following smart cart”, Proceedings of the Third International Conference on Smart Systems and Inventive Technology, 2020.

7. Bruce G. Marcot, Trent D. Penman, “Advances in Bayesian network modelling: Integration of modelling technologies”, Environmental Modelling and Software ELSEVIER, 2019

8. Sudipta Ranjan Subudhi, Ponnalagu R. N, "An Intelligent Shopping Cart with Automatic Product Detection and Secure Payment System" Conference Paper, December 2019.

9. Sakorn Mekruksavanich, "The Smart Shopping Basket Based on IoT Applications", IEEE 2019.

10. Tapan Kumar Das, "A Smart Trolley for Smart Shopping", IEEE ICSCAN 2020

11. Kowshika S, Madhu mitha S.S, Madhu Varshini G, Megha V, Lakshmi K, "IoT based Smart Shopping Trolley with Mobile Cart Application", 2021, 7th International Conference on Advanced Computing & Communication Systems (ICACCS)

12. Apurva Sutar, K.J.Karande, A.D.Harale, "Deep Learning based Automated Billing Cart", International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), April 2023.

13. Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, Wolfram Burgard, "Multimodal Deep Learning for Robust RGB-D Object Recognition", 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

14. Muhammad Atif Sarwar, Yousef-Awwad Daraghmi, Kuan-Wen Liu, Hong-Chuan Chi, Tsi-Ui Ik, Yih-Lang Li, "Smart Shopping Carts Based on Mobile Computing and Deep Learning Cloud Services", IEEE, 2020.

15. Bipin Kumar Yadav, Akash Burman, Abhoy Mahato, Manish Choudhary, Anirban Kundu, "Smart Cart: A Distributed Framework", 2020 IEEE International Conference for Convergence in Engineering

Annexures

/*Machine Learning code for human detection*/

```
import numpy as np

import sys

import time

import serial

import cv2


arduino = serial.Serial('COM6', 9600)

time.sleep(2) # Wait for initialization

print("Initialized")


directions = {

    1: "Left Back",

    2: "Backward",

    3: "Backward Right",

    4: "Left",

    5: "Stay Still",

    6: "Right",

    7: "Forward Left",

    8: "Forward",

    9: "Forward Right"

}
```

```
def send_direction(direction):
    print("Direction:", directions[direction])
    arduino.write(bytes([direction]))


def compute_direction(bound, init_area=40000):
    center = (320, 240)
    curr = (bound[0] + bound[2] / 2, bound[1] + bound[3] / 2)
    out = 5  # Stay still by default

    if bound[2] * bound[3] > init_area + 5000 or bound[1] < 50:
        out = 2  # Move backward if object is approaching or too close
    elif bound[2] * bound[3] < init_area - 5000 or (bound[1] + bound[3]) > 430:
        out = 8  # Move forward if object is moving away or too far
    elif curr[0] > center[0] + 100:
        out = 6 # Move right if object is to the right of the center
    elif curr[0] < center[0] - 100:
        out = 4 # Move left if object is to the left of the center
    elif curr[1] < center[1] - 50:
        out = 7 # Move forward-left if object is above the center
    elif curr[1] > center[1] + 50:
        out = 9 # Move forward-right if object is below the center
    return out


def detect_and_display(frame):
```

```
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=3, minSize=(30, 30), maxSize=(500, 500))


    if len(faces) > 0:

        max_area = -1

        max_area_idx = 0

        for i, (x, y, w, h) in enumerate(faces):

            if w * h > max_area:

                max_area = w * h

                max_area_idx = i


        rect = faces[max_area_idx]

        cv2.rectangle(frame, (rect[0], rect[1]), (rect[0] + rect[2], rect[1] + rect[3]), (0, 255, 0), 2)

        cv2.putText(frame, f'x: {rect[0] + rect[2] / 2} y: {rect[1] + rect[3] / 2} size: {rect[2] * rect[3]}',

                    (rect[0], rect[1] + rect[3]), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255))


        direction = compute_direction(rect)

        send_direction(direction)

    else:

        print('Search...')

        send_direction(5)  # No face detected, stay still


    cv2.imshow('frame', frame)
```

```
cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

try:
    cap = cv2.VideoCapture(1)
    if not cap.isOpened():
        raise IOError("Failed to open camera")

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Failed to retrieve frame")
            break

        detect_and_display(frame)

        if cv2.waitKey(1) == 27: # ESC key
            break

except Exception as e:
    print("Error:", e)

finally:
   cap.release()
```

```
    cv2.destroyAllWindows()
    arduino.close()
```

/* code for smart cart system movement.*/

```
const int leftMotorPin1 = 2;
const int leftMotorPin2 = 3;
const int rightMotorPin1 = 4;
const int rightMotorPin2 = 5;
const int ledPin = 13; // LED pin
int data[1];

long duration;
int distance;

const int trigPin = 10;
const int echoPin = 11;

void setup() {
  Serial.begin(9600);

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);

  pinMode(leftMotorPin1, OUTPUT);
```

```
  pinMode(leftMotorPin2, OUTPUT);
  pinMode(rightMotorPin1, OUTPUT);
  pinMode(rightMotorPin2, OUTPUT);
  pinMode(ledPin, OUTPUT); // Set LED pin as output
  pinMode(8, OUTPUT);
    digitalWrite(8 , HIGH);
    pinMode(7,  OUTPUT);
    digitalWrite(7 , HIGH);
}
void loop() {
  if (Serial.available() > 0) {
data[0] = Serial.read();
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
distance = duration * 0.034 / 2;
if (distance >10){
   int direction = data[0];
```

```
    processDirection(direction);


    // Turn on the LED while the robot is moving
    if (direction != '5') { // Assuming '5' corresponds to staying still
      digitalWrite(ledPin, HIGH); // Turn on LED
    } else {
      digitalWrite(ledPin, LOW); // Turn off LED when staying still
    }
  }
   }
  }


  void processDirection(int direction) {
   switch (direction) {
    case '1': // Back Left
      moveBackLeft();


      break;
    case '2': // Back
      moveBack();


      break;
    case '3': // Back right
      moveBackRight();
```

```
  break;
case '4': // Left
  moveLeft();

  break;
case '5': // Stay still
  stopMoving();

  break;
case '6': // Right
  moveRight();

  break;
case '7': // Front Left
  moveForwardLeft();

  break;
case '8': // Forward
  moveForward();

  break;
case '9': // Forward right
  moveForwardRight();
```

```
      break;
    default: // Unknown direction or stay still
      stopMoving();

      break;
  }
}

void moveForward() {
  digitalWrite(leftMotorPin1, HIGH);
  digitalWrite(leftMotorPin2, LOW);
  digitalWrite(rightMotorPin1, HIGH);
  digitalWrite(rightMotorPin2, LOW);
  delay(80);
}

void moveBack() {
  digitalWrite(leftMotorPin1, LOW);
  digitalWrite(leftMotorPin2, HIGH);
  digitalWrite(rightMotorPin1, LOW);
  digitalWrite(rightMotorPin2, HIGH);
  delay(80);
}
```

```
void moveLeft() {
  digitalWrite(leftMotorPin1, LOW);
  digitalWrite(leftMotorPin2, HIGH);
  digitalWrite(rightMotorPin1, HIGH);
  digitalWrite(rightMotorPin2, LOW);
  delay(80);
}

void moveRight() {
  digitalWrite(leftMotorPin1, HIGH);
  digitalWrite(leftMotorPin2, LOW);
  digitalWrite(rightMotorPin1, LOW);
  digitalWrite(rightMotorPin2, HIGH);
  delay(80);
}

void moveForwardLeft() {
  digitalWrite(leftMotorPin1, LOW);
  digitalWrite(leftMotorPin2, HIGH);
  digitalWrite(rightMotorPin1, HIGH);
  digitalWrite(rightMotorPin2, LOW);
  delay(80);
}
```

```
void moveForwardRight() {
  digitalWrite(leftMotorPin1, HIGH);
  digitalWrite(leftMotorPin2, LOW);
  digitalWrite(rightMotorPin1, LOW);
  digitalWrite(rightMotorPin2, HIGH);
  delay(80);
}
void moveBackLeft() {
  digitalWrite(leftMotorPin1, LOW);
  digitalWrite(leftMotorPin2, HIGH);
  digitalWrite(rightMotorPin1, LOW);
  digitalWrite(rightMotorPin2, HIGH);
  delay(80);
}
void moveBackRight() {
  digitalWrite(leftMotorPin1, HIGH);
  digitalWrite(leftMotorPin2, LOW);
  digitalWrite(rightMotorPin1, HIGH);
  digitalWrite(rightMotorPin2, LOW);
  delay(80);
}
void stopMoving() {
  digitalWrite(leftMotorPin1, LOW);
```

```
  digitalWrite(leftMotorPin2, LOW);
  digitalWrite(rightMotorPin1, LOW);
  digitalWrite(rightMotorPin2, LOW);
  delay(80);
}
```

/* Code for payment system */

```
Ns #include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal.h>

const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2 ;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
const int remove_button = A0;
const int add_button = A1;
const int reset_button = A2;
const int buzzer_Pin = A3;
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance.

struct item
{
  String item_name;
  String item_number;
```

```
  int item_price;
};

const int number_of_item = 4;
const item item_list[number_of_item]=
  {
  //Item Name       Item RFID Number  Item Price
  {"Tata salt",     "2E 07 1E 85",        100},
  {"Milk Powder",   "B8 D6 4C 33",        50},
  {"Bournvita",     "71 B6 1D 85",        20},
  {"Ball_Pen",      "F7 AF 51 1B",        10},
  };

int bill_amount = 0;
int remove_buttonState = 0;
int add_buttonState = 0;
int reset_buttonState = 0;
int add_item_flag = 1;
int remove_item_flag = 0;
void setup()
{
   lcd.begin(16, 2);
  pinMode(remove_button, INPUT);
  pinMode(reset_button, INPUT);
```

```
  pinMode(add_button, INPUT);
  pinMode(buzzer_Pin, OUTPUT);
  Serial.begin(9600); // Initiate a serial communication
  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();  // Initiate MFRC522
  Serial.println("Approximate your card to the reader...");
  Serial.println();
  digitalWrite(buzzer_Pin, LOW);


  lcd.clear();
  // Set cursor (Column, Row)
  lcd.setCursor(0, 0);
  lcd.print("Smart Trolley");
  lcd.setCursor(0,1);
  lcd.print("Billing System");
  delay(2000);
  lcd.clear();
  // Set cursor (Column, Row)
  lcd.setCursor(0, 0);
  lcd.print("Start purchasing");
  lcd.setCursor(0,1);
  lcd.print("your item");
  delay(100);
 }
```

```
void loop()
{
  remove_buttonState = digitalRead(remove_button);
  add_buttonState = digitalRead(add_button);
  reset_buttonState = digitalRead(reset_button);
  if(remove_buttonState)
  {
    add_item_flag = 0;
    remove_item_flag = 1;
    lcd.clear();
    // Set cursor (Column, Row)
    lcd.setCursor(0, 0);
    lcd.print("You Can now");
    lcd.setCursor(0,1);
    lcd.print("Remove your item");
    delay(2000);
  }
  else if(add_buttonState)
  {
    add_item_flag = 1;
    remove_item_flag = 0;
    lcd.clear();
    // Set cursor (Column, Row)
    lcd.setCursor(0, 0);
```

```
  lcd.print("You Can now");
  lcd.setCursor(0,1);
  lcd.print("add your item");
  delay(2000);
 }
 else if(reset_buttonState)
 {
  lcd.clear();
  // Set cursor (Column, Row)

   lcd.setCursor(0, 0);
  lcd.print("please pay");
  lcd.setCursor(0,1);
  lcd.print(bill_amount);
  delay(3000);
lcd.setCursor(0, 0);
  lcd.print("Thankyou for");
  lcd.setCursor(0,1);
  lcd.print("Shopping");
  delay(2000);

  lcd.setCursor(0, 0);
  lcd.print("Resetting");
  lcd.setCursor(0,1);
```

```
  lcd.print("Trolley data");
  delay(2000);
  lcd.clear();
  // Set cursor (Column, Row)
  lcd.setCursor(0, 0);
  lcd.print("Start Purchasing");
  lcd.setCursor(0,1);
  lcd.print("Your item");
  delay(2000);
  bill_amount = 0;
 }

 // Look for new cards
 if ( ! mfrc522.PICC_IsNewCardPresent())
 {
  return;
 }
 // Select one of the cards
 if ( ! mfrc522.PICC_ReadCardSerial())
 {
  return;
 }
 //Show UID on serial monitor
 Serial.print("UID tag :");
```

```
String content= "";
byte letter;
for (byte i = 0; i < mfrc522.uid.size; i++)
{
  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
  Serial.print(mfrc522.uid.uidByte[i], HEX);
  content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
  content.concat(String(mfrc522.uid.uidByte[i], HEX));
}
Serial.println();
Serial.print("Message : ");
content.toUpperCase();
for(int i = 0 ;i < sizeof(item_list) ; i++)
{
 if (content.substring(1) == item_list[i].item_number) //change here the UID of the card/cards that you want to give access
  {
   if(add_item_flag == 1)
    {
       bill_amount = bill_amount + item_list[i].item_price;
//     Serial.println("you have purchased = " + item_list[i].item_name);
//     Serial.print("Purchased item price");
//     Serial.println(item_list[i].item_price);
//     Serial.print("Total billing amount");
```

```
//      Serial.println(bill_amount);
//      delay(1000);
        digitalWrite(buzzer_Pin, HIGH);
        delay(500);
        digitalWrite(buzzer_Pin, LOW);
        lcd.clear();
        // Set cursor (Column, Row)
        lcd.setCursor(0, 0);
        lcd.print(item_list[i].item_name );
        lcd.setCursor(0,1);
        lcd.print(String(item_list[i].item_price)+" Rs");
        delay(2000);
    }
    else if(remove_item_flag == 1)
    {
      if(bill_amount > 0)
       {
        bill_amount = bill_amount - item_list[i].item_price;
//      Serial.println("you have purchased = " + item_list[i].item_name);
//      Serial.print("Purchased item price");
//      Serial.println(item_list[i].item_price);
//      Serial.print("Total billing amount");
//      Serial.println(bill_amount);
//      delay(1000);
```

```
      digitalWrite(buzzer_Pin, HIGH);
      delay(500);
      digitalWrite(buzzer_Pin, LOW);
      lcd.clear();
      // Set cursor (Column, Row)
      lcd.setCursor(0, 0);
      lcd.print(item_list[i].item_name );
      lcd.setCursor(0,1);
      lcd.print("removed");
      delay(2000);

    }
    else
    {
      lcd.clear();
      // Set cursor (Column, Row)
      lcd.setCursor(0, 0);
      lcd.print("Your trolley is" );
      lcd.setCursor(0,1);
      lcd.print("empty now");
      delay(2000);
    }
  }
}
```

```
  }
  lcd.clear();
  // Set cursor (Column, Row)
  lcd.setCursor(0, 0);
  lcd.print("Total Billing");
  lcd.setCursor(0,1);
  lcd.print(String(bill_amount)+" Rs");
  delay(2000);
}
```