TABLE OF CONTENTS

1. SCOPE

1.1 Identification.

This document applies to the Collision Detection Vehicle, version 1.1.8, release 1. It encompasses the Collision Detection Vehicle and its associated components. The operating system platform(s) to which this document applies is the Arduino IDE.

The Collision Detection Vehicle is programmed using the Arduino programming language and is equipped with the following components: Wire library for communication, LiquidCrystal_I2C library for interfacing with the LCD module.

The vehicle's hardware configuration includes the Motor driver setup using L298N module.
The vehicle is also equipped with ultrasonic sensors for collision detection.
Additional features of the Collision Detection Vehicle include a buzzer for audible alerts, and a variable to keep track of obstacle count.

The software functionality of the vehicle involves continuous monitoring of the sensors. If an obstacle is detected within a safe distance, the vehicle will stop its motors, activate the buzzer, display a message on the LCD screen, and increment the obstacle count.

To control the vehicle, registered operators can use the associated Android app. The app allows authorized operators to log in and operate the vehicle specifically within shopping aisles. However, if the vehicle detects shoppers or obstacles more than four times, indicating an unsafe environment, the operator must tap on the "Register Aisle" button in the app to register the problematic aisle in the database.

Overall, this document provides guidelines and instructions for operating the Collision Detection Vehicle, along with details of its hardware, software, and integration with the Android app for secure operation within shopping aisles.

1.2 System overview**.**

This system consists of a robotic vehicle equipped with obstacle detection capabilities and an Android app that allows registered operators to control the vehicle within a shopping aisle. The software components include Arduino code for the vehicle's behavior and an Android app for operator interaction. The system is designed to detect shoppers/obstacles and alert the operator when it detects them multiple times. The operator can register problematic aisles in a database via the Android app.

- General Nature: The system combines hardware (robotic vehicle) and software (Arduino code, Android app) components to enable safe navigation in shopping aisles.
- System Development, Operation, and Maintenance: The system was developed by Edwin Manda. It is currently operational and requires periodic maintenance to ensure proper functioning.

- Project Sponsor, Acquirer, User, Developer, and Support Agencies: Edwin Manda is the project sponsor, acquirer, user, and developer of the system. Support agencies are not explicitly mentioned.
- Operating Sites: The system can be deployed in shopping aisles where the robotic vehicle is intended to operate.
- Relevant Documents: This user manual serves as a guide for understanding and operating the system. Additional documents related to hardware specifications, Android app development, or database integration may exist but are not mentioned in this overview.

System Components:

- Robotic Vehicle: It incorporates a motor driver (L298N) and ultrasonic sensors (trig, echo, trig2, echo2) for obstacle detection. The vehicle's movements are controlled through the Arduino code.
- LiquidCrystal_I2C Library: Utilized for controlling a 16x2 LCD display (0x27 address) to provide visual feedback.
- Buzzer: An audible alert is triggered when obstacles or shoppers are detected.
- Arduino Board: The Arduino board runs the provided code and connects with the vehicle's hardware components.
- Android App: The app restricts access to registered operators and provides a user interface for controlling the vehicle and registering problematic aisles in the database.

System Operation:

- Setup: The setup() function initializes pins and components, including motor drivers, ultrasonic sensors, LCD display, buzzer, and serial communication.
- Command Execution: In the loop() function, the system checks for incoming commands from the Android app. Upon receiving a command, it executes the corresponding action using the executeCommand() function.
- Obstacle Detection: Ultrasonic sensors are used to measure distances in front of the vehicle. If an obstacle or shopper is detected within a safe range (10 cm), the vehicle stops, the buzzer sounds, and a warning message is displayed on the LCD. The obstacle count is incremented.
- Aisle Registration: When the obstacle count reaches four (indicating repeated detections in the same aisle), the vehicle stops, and a notification is displayed on the LCD. The operator must tap the "Register aisle" button on the Android app to record the problematic aisle in the database.
- Motor Control: The moveForward(), moveBackward(), moveLeft(), moveRight(), and stopMotors() functions control the vehicle's movement based on the commands received.
- LCD Display: The displayLCD() function provides visual feedback by displaying specific messages on the LCD screen during obstacle detection and aisle registration.

Android App Integration:

- The Android app ensures that only registered operators can log in and control the robotic vehicle.
- The app facilitates command transmission from the operator to the Arduino board via serial communication.

- When the vehicle detects obstacles multiple times, the app prompts the operator to tap the "Register aisle" button to register problematic aisles in the database.

1.3 Document overview**.**

Security and Privacy Considerations:

As mentioned, the Android app allows only registered operators to log in and operate the robotic vehicle. This security measure ensures that unauthorized individuals cannot control the vehicle. However, it is essential to implement appropriate security practices to protect the app, such as using secure authentication mechanisms and securely storing user data.

Regarding privacy, the manual does not provide explicit information on data collection or handling. It is recommended to adhere to applicable privacy regulations and ensure that any personal data collected through the Android app is handled securely and in compliance with privacy laws.

Operators should be aware of their responsibility to maintain the privacy and security of the system while operating it within a shopping aisle. They should avoid capturing or accessing any sensitive information and handle any recorded data appropriately.

2. REFERENCED DOCUMENTS

Collision Detection Robotic Vehicle
DESIGN DOCUMENTATION by Edwin Manda
PROPOSAL DOCUMENTATION by Edwin Manda
https://assets.ctfassets.net/xqu7pe39gxim/5KlJFsEPWLo57DlU9liDZ6/48e10d57b4da9122cd15cd85e0d50f21/920-0856-00.pdf
https://www.sqlite.org/docs.html
https://peppe8o.com/piezo-buzzer-with-arduino-uno-and-arduino-ide/
https://www.tutorialspoint.com/arduino/arduino_installation.htm
https://mhsc.org.za/sites/default/files/public/publications/CollisionAvoidance%20EDG%201Nov2018.pdf

3. SOFTWARE SUMMARY

3.1 Software application**.**

The software application in this system consists of two components: the Arduino code running on the robotic vehicle and the Android app for operator control. The intended uses of the software are to enable registered operators to control the robotic vehicle within a shopping aisle and facilitate obstacle detection and avoidance.

Capabilities:

- Remote Control: The Android app allows registered operators to remotely control the movement of the robotic vehicle using predefined commands. Operators can move the vehicle forward, backward, left, and right or stop its motion.

- Obstacle Detection: The robotic vehicle is equipped with ultrasonic sensors that detect obstacles or shoppers in the vicinity. The software interprets the sensor readings and triggers appropriate actions to avoid collisions.
- LCD Display: The system includes an LCD display that provides visual feedback on the status of the vehicle and alerts operators when obstacles are detected.

Operating Improvements and Benefits:

- Enhanced Safety: By detecting obstacles and shoppers, the system enhances the safety of the shopping aisle environment by reducing the risk of collisions between the vehicle and individuals.
- Efficient Operations: The software enables operators to navigate the shopping aisle efficiently and maneuver the vehicle as needed, contributing to smoother operations and improved productivity.
- Operator Accountability: The Android app restricts access to registered operators, ensuring that only authorized personnel can control the robotic vehicle. This feature promotes accountability and prevents unauthorized usage.

By integrating the software application with the robotic vehicle, operators can remotely control its movement, while the obstacle detection capabilities improve safety and prevent collisions. The Android app's restricted access enhances security and ensures that only authorized operators can operate the vehicle within the shopping aisle. Ultimately, the software application aims to facilitate efficient and safe operations in the retail environment.

3.2 Software inventory**.**

The following software files are required for the operation of the system:

Arduino Code:

- File Name: RoboticVehicle.ino
- Description: This file contains the Arduino code that controls the robotic vehicle. It includes the setup and loop sections, as well as functions for motor control, obstacle detection, and LCD display.

Libraries:

- Wire.h: This library provides I2C communication functionality for connecting the LiquidCrystal_I2C display.
- LiquidCrystal_I2C.h: This library enables communication with the I2C-based LiquidCrystal display.

Android App:

- Description: The Android app allows registered operators to control the robotic vehicle in the shopping aisle. The app provides a user interface for sending commands to the vehicle and registering problematic aisles in the database.

Database:

- Name: TinyWebDB1

- Description: The database stores information about problematic aisles registered by the operators. It keeps track of the aisle details, such as aisle number or identifier, and the number of times obstacles have been detected in each aisle.

Security and Privacy Considerations:

- Arduino Code: The code running on the Arduino board does not involve any security or privacy-sensitive data. It primarily focuses on vehicle control and obstacle detection, without handling or storing personal or sensitive information.
- Android App: The Android app should implement appropriate security measures to ensure secure access for registered operators. This may include user authentication, password encryption, and secure communication protocols. It is essential to handle user data, such as login credentials, securely and adhere to privacy regulations.
- Database: The database should be protected with appropriate security measures, such as access control and encryption. It should ensure that only authorized personnel can access and modify the registered aisle information. Personal data of the operators or shoppers should be handled according to applicable privacy laws and regulations.

Emergency Operation:
In case of an emergency or system failure, the following software components are necessary to continue or resume operation:

- Arduino Code: The Arduino code running on the robotic vehicle is essential for its operation. Ensure that the Arduino board is properly functioning and the code is uploaded correctly to the board.
- Android App: The Android app is required for operators to control the vehicle. Ensure that the app is installed on a compatible device and can establish a connection with the vehicle.
- Database: The database is necessary to maintain the registered aisle information. Ensure that the database is accessible and functional to register problematic aisles if needed.
- Power: The robotic vehicle also runs on backup power in case the main power fails.

It is crucial to have backup copies of the Arduino code, Android app, and database to ensure continuity in case of emergencies, system failures, or data loss. Regular backups and proper version control are recommended to safeguard the software components and their associated data.

3.3 Software environment**.**

To install and run the software system, the following hardware, software, equipment, manual operations, and resources are required:

a. Computer Equipment:

- A computer capable of running the Arduino IDE and compiling/uploading code to Arduino boards.

- Memory: The memory requirements for the computer should meet the specifications of the Arduino IDE.
- Auxiliary Storage: Sufficient storage space to store the Arduino code file and other related files.

b. Communications Equipment:

- No specific communications equipment is required for the standalone operation of the robotic vehicle system. However, if the system is integrated with the Android app, a device capable of running the app and establishing a connection with the robotic vehicle is needed.

c. Other Software:

- Arduino IDE: The Arduino Integrated Development Environment (IDE) is required to compile and upload the Arduino code to the Arduino board.
- Operating System: The computer should be running a compatible operating system (e.g., Windows, macOS, Linux) that supports the Arduino IDE.
- Libraries: The following libraries should be installed in the Arduino IDE:
  - Wire.h: This library provides I2C communication functionality.
  - LiquidCrystal_I2C.h: This library enables communication with the I2C-based LiquidCrystal display.

d. Forms, Procedures, or Manual Operations:

- The Android app should provide a user interface that allows operators to log in, operate the robotic vehicle, and register problematic aisles in the database. The exact forms, procedures, or manual operations required would depend on the design and implementation of the Android app, which is not provided in the code snippet.

e. Other Facilities, Equipment, or Resources:

- Robotic Vehicle: The physical robotic vehicle, including the Arduino board, motor driver (L298N), ultrasonic sensors, buzzer, and LiquidCrystal display (I2C-based), should be properly assembled and connected according to the provided code.
- Android App: The Android app, which is not provided in the code snippet, should be installed on a compatible Android device.
- Database: A database system is required to store and manage the registered aisle information. The specific database software and associated resources needed depend on the implementation and requirements of the Android app.

3.4 Software organization and overview of operation.

a. Logical Components of the Software:

- Robotic Vehicle Control System: The provided code snippet represents the logic for controlling a robotic vehicle equipped with ultrasonic sensors, a buzzer, and an LCD display.
- Android App: The Android app, which is not included in the code snippet, allows registered operators to log in and operate the robotic vehicle in a

shopping aisle. It also provides a feature to register problematic aisles in a database.

- Databases and Data Files: The software interacts with a database to store information about registered operators and problematic aisles. The specific database management system (DBMS) and the data files used depend on the implementation of the Android app.

b. Performance Characteristics:

- Inputs: The inputs to the robotic vehicle control system are received through the serial communication interface. The system expects commands from the Android app via the serial connection.
- Outputs: The software can produce outputs in the form of motor control signals for the robotic vehicle, LCD display messages, and buzzer sounds.
- Response Time: The response time of the system depends on factors such as the speed of the microcontroller, communication latency, and the execution time of the code. It is typically fast for real-time control systems.
- Processing Time: The processing time for the provided code snippet is relatively low as it performs basic calculations and simple logic operations. The specific processing time depends on the hardware capabilities and the complexity of the complete software system.
- Limitations: The code snippet does not explicitly define limitations related to the number of events that can be tracked or any other specific limitations. However, the limitations can depend on the capabilities of the hardware components used, memory limitations, and the overall design and implementation of the complete software system.
- Error Rate: The error rate that can be expected depends on the reliability and accuracy of the hardware components used and the correctness of the code implementation.
- Reliability: The reliability of the software system depends on various factors, including the quality of the hardware components, the robustness of the code implementation, and the stability of the communication interfaces.

c. Relationship with Interfacing Systems/Organizations/Positions:

- The robotic vehicle control system interacts with the Android app, which serves as the interface for registered operators to control the vehicle and register problematic aisles in the database. The Android app communicates with the robotic vehicle control system through a serial connection to send commands and receive feedback.

d. Supervisory or Security Controls:

- The Android app can implement various security controls to manage access to the software and ensure that only registered operators can log in and operate the robotic vehicle. This can include user authentication mechanisms such as passwords or other authentication methods.
- Additionally, the Android app can implement supervisory controls to limit the actions or operations that operators can perform, ensuring they adhere to predefined rules and guidelines for operating the robotic vehicle safely.

3.5 Contingencies and alternate states and modes of operation**.**

Emergency Situations:

- In the event of detecting shoppers or obstacles for more than four times, the system enters an emergency state.
- The system will automatically stop the motors, activate the buzzer, and display appropriate messages on the LCD.
- During this emergency state, the operator will not be able to operate the robotic vehicle until the situation is resolved.
- The operator needs to tap on the "Register Aisle" button in the Android app to register the problematic aisle in the database.

Normal Operation:

- Under normal conditions, registered operators can log in to the Android app and operate the robotic vehicle in a shopping aisle.
- The operator can send commands to the robotic vehicle through the Android app, controlling its movement (forward, backward, left, right), and stop the motors.
- The vehicle's ultrasonic sensors detect obstacles (shoppers) and trigger appropriate actions such as stopping the motors, activating the buzzer, and displaying messages on the LCD.

Operator Authentication and Authorization:

- The Android app includes a user login system that allows only registered operators to access the vehicle control features.
- Before operating the robotic vehicle, operators must log in using their credentials (e.g., username and password).
- Unauthorized users who do not have valid credentials will not be able to log in and operate the vehicle.

Database Interaction:

- The Android app communicates with a database to store information about registered operators and problematic aisles.
- When an emergency situation occurs (obstacle count reaches the threshold), the operator needs to register the problematic aisle by tapping on the "Register Aisle" button.
- This information is then stored in the database for future reference or analysis.

3.6 Security and privacy**.**

The following points provide an overview of the security and privacy considerations associated with the software:

User Authentication and Access Control:

- The Android app includes a login system that allows only registered operators to access and operate the robotic vehicle.
- User authentication is implemented to ensure that only authorized personnel can log in and control the vehicle.
- It is essential to keep the login credentials confidential and not share them with unauthorized individuals.

Data Privacy:

- The system may collect and store data related to registered operators and problematic aisles.
- This data should be handled with care and stored securely to protect the privacy of individuals and comply with applicable data protection regulations.
- The database should have appropriate security measures in place, such as encryption and access controls, to safeguard the stored information.

Unauthorized Copying and Distribution:

- The user manual, software, and associated documents provided should not be copied or distributed without proper authorization.
- Unauthorized copying or distribution of the software or documentation may infringe on intellectual property rights and could lead to legal consequences.

Security Updates:

- It is important to regularly update the software system to incorporate security patches and enhancements.
- Stay informed about any updates or security advisories related to the software and ensure that the system is up to date with the latest security measures.

User Awareness:

- Users of the system should be educated about security best practices and the importance of maintaining the security and privacy of the system.
- Emphasize the significance of logging out from the Android app after use and not leaving the system unattended when logged in.

3.7 Assistance and problem reporting**.**

To obtain assistance and report any problems encountered while using the software, please follow the points of contact and procedures outlined below:

Technical Support:

- For technical assistance related to the robotic vehicle hardware or the software system, please contact our technical support team.
- You can reach our technical support team by sending an email to mandaeb93@outlook.com or by calling our support hotline at 071 299 3110.

Operator Registration and Login:

- If you are an operator using the Android app to control the robotic vehicle, make sure you have successfully registered as an authorized operator.
- In case you encounter any issues with the operator registration process or have difficulties logging into the app, refer to the user manual or contact the technical support team for guidance.

Reporting Problems:

- If you experience any problems or encounter errors while operating the robotic vehicle or using the Android app, it is important to report them promptly.
- Before reporting a problem, please ensure that you have followed the instructions provided in the user manual and have checked for any available troubleshooting steps.
- To report a problem, please send an email to support@example.com and include the following information:
  - Detailed description of the problem encountered
  - Steps to reproduce the problem, if applicable
  - Any error messages or relevant screenshots

Problematic Aisle Registration:

- If the robotic vehicle detects shoppers or obstacles for more than four times in a particular aisle, it is necessary to register the problematic aisle in the database.
- To register a problematic aisle, the operator should tap on the "Register Aisle" button in the Android app.
- Make sure to provide accurate information about the aisle, such as aisle number or location, to help identify and address the issue effectively.

Our technical support team will review and respond to your inquiries and problem reports in a timely manner. Please allow sufficient time for our team to investigate and provide assistance or solutions.

4. ACCESS TO THE SOFTWARE

4.1

4.1.1 Access control.

The software system includes access and security features to ensure proper control and privacy. The following items provide an overview of the access and security features visible to the user:

a. Obtaining a Password:

- The password for accessing the Android app and operating the robotic vehicle will be provided by the system administrator or an authorized personnel.
- To obtain a password, contact the system administrator or the designated person responsible for managing user access.

b. Managing Passwords:

- Users have the ability to add, delete, or change passwords under their control to enhance security.
- To add a new password or change an existing password, follow the instructions provided in the Android app's settings or user profile section.
- To delete a password, navigate to the appropriate section in the app and follow the instructions for removing a password.

c. Security and Privacy Considerations:

- When it comes to the storage and marking of output reports and other generated media, security and privacy considerations should be taken into account.
- Output reports and media generated by the software should be stored in secure locations to prevent unauthorized access or tampering.
- It is recommended to follow your organization's security protocols and guidelines regarding the storage, handling, and disposal of sensitive information.
- If any security or privacy concerns arise related to the storage or marking of output reports and media, consult your organization's data protection policies or contact the system administrator for guidance.

Note: It is important to maintain the confidentiality of passwords and ensure they are not shared with unauthorized individuals. Regularly updating passwords and adhering to best security practices will help protect the system and maintain data privacy.

Please remember that the access control and security measures mentioned above are general guidelines. For specific instructions or further details, refer to the user manual provided with the software system or consult with the system administrator.

4.1.2 Installation and setup.

To install and set up the system comprising the software and the Android app, follow the procedures outlined below:

Identification and Authorization:

- To access or install the software on the equipment, ensure that you have the necessary permissions and authorization from the system administrator or authorized personnel.
- Obtain the required login credentials or registration details from the system administrator to gain access to the Android app.

Installation:

- Begin by installing the Arduino development environment on your computer if it is not already installed. Visit the Arduino website (https://www.arduino.cc/) and follow the instructions specific to your operating system.
- Connect the Arduino board to your computer using a USB cable.
- Open the Arduino IDE and create a new sketch.
- Copy and paste the provided system code into the Arduino sketch.
- Verify and upload the sketch to the Arduino board by clicking on the appropriate buttons in the Arduino IDE.
- Ensure that the necessary libraries, such as Wire.h and LiquidCrystal_I2C.h, are installed in the Arduino IDE. If any libraries are missing, they can be installed via the Library Manager in the Arduino IDE.

Configuration:

- Connect the motor driver (L298N) and other required components according to the wiring instructions provided in the system code or any additional documentation.
- Adjust the pin assignments in the system code if necessary to match your hardware configuration.
- Configure any other parameters within the system code as per your specific requirements, such as adjusting motor speed or sensor thresholds.
- Save the modified system code.

Android App Installation:

- On your Android device, navigate to the Google Play Store.
- Search for the system's Android app by its name or provided app title.
- Install the app on your device by following the on-screen instructions.
- Once the installation is complete, locate the app icon on your device's home screen or app drawer.

User Registration and Login:

- Launch the installed Android app by tapping on its icon.
- If you are a new user, tap on the registration option and follow the provided instructions to register as an operator.
- Provide the necessary information, such as username, password, and any other required details, to complete the registration process.
- If you are a registered user, enter your login credentials (username and password) to access the operator features.

Operating the Robotic Vehicle:

- After logging in as a registered operator, ensure that the Arduino board is properly connected to the Android device via a compatible communication interface (e.g., USB, Bluetooth).
- Follow the on-screen instructions or refer to the app's user interface to control the robotic vehicle's movement in the shopping aisle.
- Use the provided buttons or gestures within the app to navigate the vehicle in the desired direction (forward, backward, left, right) or to stop its motion.
- Be cautious while operating the vehicle to avoid collisions with shoppers or obstacles.

Registering a Problematic Aisle:

- If the robotic vehicle detects shoppers or obstacles for more than four times and becomes unable to operate safely, tap on the "Register Aisle" button within the Android app.
- Follow the instructions provided on the screen to register the problematic aisle in the system's database.
- Enter the required information, such as aisle number or location details, and submit the registration to store the information in the database for further analysis or action.

4.2 Initiating procedure.

Follow the step-by-step procedures below to initiate the system and begin work:

Hardware Setup:

- Ensure that the robotic vehicle is properly connected to the Arduino board.
- Connect the necessary components, such as the motor driver (L298N), ultrasonic sensors (trig and echo pins), and the buzzer, according to the provided wiring instructions or any additional documentation.

Software Setup:

- Install the Arduino development environment on your computer if it is not already installed. Visit the Arduino website (https://www.arduino.cc/) and follow the instructions specific to your operating system.
- Open the Arduino IDE and create a new sketch.
- Copy and paste the provided system code into the Arduino sketch.
- Verify and upload the sketch to the Arduino board by clicking on the appropriate buttons in the Arduino IDE.
- Ensure that the required libraries, such as Wire.h and LiquidCrystal_I2C.h, are installed in the Arduino IDE. If any libraries are missing, they can be installed via the Library Manager in the Arduino IDE.

Power On:

- Connect the power supply to the Arduino board or ensure that it is powered via USB if connected to a computer.
- Power on the robotic vehicle by supplying power to the motor driver and other necessary components.

Android App:

- Ensure that the Android device running the operator app is powered on and in a functional state.
- Locate the app icon on the home screen or app drawer of the Android device and tap on it to launch the app.
- If prompted, enter your registered operator credentials (username and password) to log in. If you are a new user, follow the registration process outlined in the app to create a new operator account.

Establish Connection:

- Ensure that the Arduino board and the Android device are connected via a compatible communication interface (e.g., USB, Bluetooth). Refer to the app's connection settings or instructions for more details on establishing the connection.

Begin Work:

- After successfully logging into the operator app and establishing the connection with the Arduino board, you are ready to begin work.
- Use the provided buttons or gestures within the app's user interface to control the robotic vehicle's movement in the shopping aisle. Options may include buttons for moving forward, backward, left, right, and stopping the vehicle.
- Observe the LCD display on the robotic vehicle for any relevant information or notifications.

Problem Determination Checklist:

- If you encounter any difficulties while operating the system, refer to the following checklist for problem determination:
  - Ensure that the Arduino board is powered on and properly connected to the necessary components.
  - Verify that the Android device is connected to the Arduino board and the operator app.
  - Check the wiring connections of the motor driver, ultrasonic sensors, buzzer, and other components for any loose connections or errors.
  - Confirm that the necessary libraries, such as Wire.h and LiquidCrystal_I2C.h, are correctly installed in the Arduino IDE.
  - Review the system code for any errors or modifications specific to your hardware setup.
  - Consult the troubleshooting section in the user manual or seek assistance from the system administrator or technical support if the issue persists.

4.3 Description of inputs.

4.3.1 Input conditions.

The following conditions should be observed when preparing input for the system:

a. Reason for Input: The inputs serve to control the movement of the robotic vehicle in the shopping aisle and provide information regarding obstacles or shoppers detected. Inputs can include normal status reports and the need to update data related to the vehicle's operation.

b. Frequency of Input: Inputs may vary in frequency depending on the specific requirements. For example, the operator can provide inputs on demand, whenever they want to initiate a particular movement or receive information about the vehicle's status. The frequency can be determined by the operator's interaction with the Android app controlling the vehicle.

c. Origin of Input: Inputs originate from registered operators using the Android app. Only authorized operators are allowed to log in and operate the robotic vehicle in the shopping aisle. The input originates from the organization or station that has the authority to generate such inputs.

d. Medium of Input: The input is provided through the Android app's user interface, utilizing the available controls such as buttons or gestures. The Android device serves as the medium through which the operator communicates the desired commands to the robotic vehicle.

e. Related Inputs: There may be related inputs that need to be entered simultaneously or in a specific sequence. For example, the operator may need to input the desired direction (forward, backward, left, right) along with the speed of movement or any additional commands required for the specific operation.

f. Other Applicable Information: Additional information relevant to the inputs can include:

- Priority: The urgency or importance of the input can be indicated, specifying whether it requires immediate attention or can be queued based on the system's capabilities.
- Security and Privacy Considerations: Input may be subject to security measures to ensure that only authorized operators can access and control

the vehicle. Privacy considerations may also apply to protect the personal information of registered operators or sensitive data related to the organization or shoppers.

4.3.2 Input formats.

The following layout formats should be used when preparing inputs for the system. These formats explain the information that can be entered in the various sections and lines of each format. The inputs are primarily entered through the Android app's user interface.

Login Credentials Format:

Section 1: Operator Credentials
Line 1: Username (registered operator's username)
Line 2: Password (registered operator's password)

Vehicle Control Format:

Section 1: Control Commands
Line 1: Command (numeric value representing the desired action)
0: Move forward
1: Move backward
2: Turn left
3: Turn right
4: Stop motors

Aisle Registration Format:

Section 1: Aisle Information
Line 1: Aisle ID or Name (identifier for the problematic aisle)
Line 2: Description (optional, additional information about the aisle issue)

4.3.3 Composition rules.

To prepare inputs for the system and interact with the Android app effectively, the following rules and conventions must be observed:

a. Input Transaction Length:

There is no specific maximum length mentioned for input transactions. However, it is recommended to keep the input within a reasonable length to ensure clarity and avoid potential errors.

b. Format Conventions:

All input items must be entered in a left-justified format, aligning the information to the left side of the input fields or lines.
Ensure proper spacing and indentation to maintain readability and clarity of the inputs.

c. Labeling:

The usage of identifiers to denote major data sets is not explicitly defined in the provided system code. However, in the Android app, you may encounter labeling conventions such as:
Operator Credentials: Username and Password
Aisle Information: Aisle ID or Name, Description

d. Sequencing:

The order and placement of items in the input are determined by the specific requirements of the Android app and the system's functionality.
Follow the instructions provided within the app to enter inputs in the correct sequence and at the appropriate locations.

e. Punctuation:

The system code provided does not require specific punctuation or symbols for input.
Ensure proper spacing between items and use appropriate punctuation marks (e.g., commas, periods) where necessary to maintain clarity and readability.

f. Restrictions:

The provided system code does not explicitly mention any restrictions on the usage of particular characters or parameter sets.
However, when entering inputs through the Android app, follow any restrictions or guidelines provided by the app itself to ensure the inputs are accepted correctly.
Avoid using special characters or symbols that may conflict with the app's input processing or database operations.

4.3.4 Input vocabulary.
To prepare inputs for the system and interact with the Android app, the following legal character combinations or codes can be used:

Operator Login:

- Username: Alphanumeric characters (a-z, A-Z, 0-9), underscore (_), and hyphen (-).
- Password: Alphanumeric characters (a-z, A-Z, 0-9), underscore (_), and hyphen (-).

Robotic Vehicle Operations:

- Move Forward: Enter the code '0' to move the robotic vehicle forward.
- Move Backward: Enter the code '1' to move the robotic vehicle backward.
- Move Left: Enter the code '2' to move the robotic vehicle to the left.
- Move Right: Enter the code '3' to move the robotic vehicle to the right.
- Stop Motors: Enter the code '4' to stop the motors of the robotic vehicle.

Register Aisle:

- Tap on the "Register Aisle" button in the Android app to register a problematic aisle in the database.

4.3.5 Sample inputs.
a. Headers denoting the start of input:

- The system does not require specific headers to denote the start of input.

b. Text or body of the input:

- Operator Login:
  - Example 1:
    - Username: "john_doe"
    - Password: "password123"
  - Example 2:
    - Username: "jane_smith"
    - Password: "securepass"
    -
- Robotic Vehicle Operations:
  - Example 1: Move the robotic vehicle forward
    - Input: 0
  - Example 2: Stop the robotic vehicle
    - Input: 4
- Register Aisle:
  - Tap on the "Register Aisle" button in the Android app to initiate the registration process.

c. Trailers denoting the end of input:

- The system does not require specific trailers to denote the end of input.

d. Portions of the input that may be omitted:

- In the case of Operator Login, omitting the username or password will result in a login failure.

e. Portions of the input that may be repeated:

- Robotic Vehicle Operations:
  - The commands to move the robotic vehicle (forward, backward, left, right) can be repeated by sending the corresponding input code multiple times.
  - For example, to move forward twice, send "0" twice consecutively: 0 0

4.4 Description of outputs.

4.4.1 General description.

a. Reasons why the output is generated:

- The output is generated to provide information, status updates, and notifications related to the operation of the robotic vehicle in a shopping aisle. It notifies the operator about the presence of shoppers or obstacles and prompts appropriate actions.

b. Frequency of the output:

- The output is generated whenever certain conditions are met, such as detecting shoppers or obstacles in the shopping aisle. It occurs dynamically based on the real-time sensing of the environment.

c. Modifications or variations of the basic output:

- The basic output, "SHOPPER DETECTED!" and "Kindly wait...", can be modified or customized to suit specific requirements or preferences. The text or messages displayed on the LCD screen can be changed to convey different instructions or alerts.

d. Media:

- The output is displayed on a LiquidCrystal_I2C screen attached to the robotic vehicle. It utilizes the screen to present visual information to the operator.

e. Location where the output will appear:

- The output appears on the LCD screen attached to the robotic vehicle. It is typically located within the vicinity of the vehicle, allowing the operator to easily view the information.

f. Additional characteristics:

- Priority: The output related to shoppers or obstacles takes priority as it indicates potential safety concerns. Prompt attention and appropriate action are required when such outputs are displayed.
- Security and Privacy Considerations: The output displayed on the LCD screen should be kept confidential and not visible to unauthorized individuals to maintain security and privacy. Access to the robotic vehicle's operation and related output should be restricted to registered operators only.
- Associated Outputs: In addition to the displayed output, the system may generate audible alerts, such as a buzzer sound, to attract the operator's attention when shoppers or obstacles are detected. These additional outputs complement the visual information provided on the LCD screen, enhancing the operator's awareness of the situation.

4.4.2 Output formats.

a. Security and privacy markings:

- The output format generated by the system does not explicitly include security and privacy markings. However, it is important to ensure that the output data, including any sensitive information, is handled securely and protected from unauthorized access. The system should adhere to relevant security and privacy regulations and guidelines.

b. Data that may appear in headers:

- The output format does not include explicit headers. However, if the system generates any headers or additional information, it should include relevant details such as system identification, version number, and timestamp to provide context for the output.

c. Information that may appear in the body or text of the output:

- The primary information displayed in the body or text of the output is related to the detection of shoppers or obstacles. The LCD screen presents the following information:
  - "SHOPPER DETECTED!" - Indicates the presence of a shopper in the shopping aisle.
  - "Kindly wait..." - Provides an instruction to the operator, indicating that they should wait until the situation is resolved or the obstacle is cleared.

d. Data that may appear in trailers:

- The output format does not include explicit trailers. However, if any additional information or status messages are displayed after the primary output, it should be relevant to the current operation or context of the system.

e. Additional characteristics, such as the meaning of special symbols:

- The output format in this system does not involve the use of special symbols. However, it is important to define and explain the meaning of any special symbols used in the output format if they are introduced in the specific implementation or customization of the system.

4.4.3 Sample outputs

LCD Output:

- Description: The LCD display provides information about the detection of shoppers or obstacles in the shopping aisle.

- Meaning and Use:

  - Line 1: "SHOPPER DETECTED!" - Indicates that a shopper has been detected in the aisle.
  - Line 2: "Kindly wait..." - Instructs the operator to wait until the situation is resolved or the obstacle is cleared.
- Source: Generated by the robotic vehicle system.

- Characteristics:
  - When Omitted: If the LCD output is omitted or not displayed, the operator will not receive real-time information about shoppers or obstacles.
  - Range of Values: N/A
  - Unit of Measure: N/A

Android App Output:

- Description: The Android app allows registered operators to log in and operate the robotic vehicle in a shopping aisle. It includes a feature to register problematic aisles in the database.
- Meaning and Use:
  - Login Screen: The login screen prompts the operator to enter their credentials (e.g., username and password) to access the app.
  - Operator Interface: The operator interface provides controls to operate the vehicle, such as buttons for forward, backward, left, right, and stop movements.
  - Register Aisle Button: When the operator is no longer able to operate the vehicle due to repeated shopper/obstacle detections, they can tap on the "Register Aisle" button to register the problematic aisle in the database.
- Source: Generated by the Android app.
- Characteristics:
  - When Omitted: If the app output is omitted or not functioning, the operator will not be able to log in, operate the vehicle, or register problematic aisles.
  - Range of Values: N/A
  - Unit of Measure: N/A

4.5 Recovery and error correction procedures.

Error Codes:

Error Code: E001

- Meaning: No Response from Sensors
- Corrective Action:
  - Check the sensor connections and ensure they are properly connected.
  - Verify that the sensors are powered on and functioning correctly.
  - Restart the system and try again.

Error Code: E002

- Meaning: Motor Control Failure
- Corrective Action:
  - Check the motor connections and ensure they are properly connected.
  - Verify that the motor driver is powered on and functioning correctly.
  - Restart the system and try again.

Error Code: E003

- Meaning: Communication Error
- Corrective Action:
  - Check the serial communication connections between the Arduino and the Android app.
  - Ensure that the baud rate and communication settings are configured correctly.
  - Restart both the Arduino and the Android app and try again.

Recovery and Continuity of Operations:

In the event of emergencies or system failures, follow these procedures to recover and ensure the continuity of operations:

Restart Procedure:

- Turn off the power supply to the system.
- Wait for a few seconds and then turn on the power supply.
- Allow the system to initialize and boot up completely.
- Verify that all components, including the Arduino, sensors, motor driver, LCD, and Android app, are functioning correctly.

Recovery Procedure:

- If an error occurs during operation, refer to the corresponding error code and take the appropriate corrective action as described above.
- Perform necessary checks and troubleshooting steps to identify and resolve the issue.
- If required, consult the system administrator or technical support for further assistance.

Emergency Situations:

- In case of critical emergencies or when the system detects shoppers/obstacles for more than four times, the operator should immediately stop the vehicle and ensure the safety of shoppers and surroundings.
- Tap on the "Register Aisle" button in the Android app to register the problematic aisle in the database for further analysis and action.
- Follow the established emergency protocols and notify the relevant authorities or personnel as per the organization's guidelines.

4.6 Stopping and suspending work.

To cease or interrupt the use of the software and stop the operation of the robotic vehicle, follow these steps:

Operator Control:

- In the Android app, tap on the "Stop" or "Cease Operation" button to send the stop command to the robotic vehicle.
- The vehicle will immediately stop its movement and remain stationary.

Emergency Situations:

- In case of emergencies or critical situations, it is important to prioritize safety.
- Immediately stop the vehicle by tapping the emergency stop button in the Android app or using any emergency stop mechanism available in the hardware.
- Follow the established emergency protocols and guidelines to ensure the safety of shoppers and surroundings.

Determining Normal Termination or Cessation:

To determine whether the software has terminated normally or the cessation of the software has occurred, follow these indications:

Android App:

- The Android app may display a confirmation message indicating that the software has terminated or ceased operation.
- The user interface of the app may change to reflect the inactive or stopped state.

Robotic Vehicle:

- The LCD display connected to the Arduino may show a specific message indicating the termination or cessation of the software.
- The robotic vehicle's motors and actuators will stop operating and remain in a stationary state.

Note: It is important to always ensure that the vehicle is safely stopped and there is no immediate danger before determining normal termination or cessation. Follow any specific guidelines or instructions provided by the system administrator or the manufacturer of the software and hardware components.

Once the operator is no longer able to operate the vehicle due to the detection of shoppers/obstacles for more than four times, the operator needs to take the following action:

Registering the Problematic Aisle:

- Tap on the "Register Aisle" button in the Android app.
- This action will register the problematic aisle in the database for further analysis and necessary actions to address the issue.

It is recommended to consult the system administrator or technical support for any further guidance or assistance related to stopping or suspending the work of the robotic vehicle and handling problematic aisles.

5. PROCESSING REFERENCE GUIDE

5.1 Capabilities:
The software, consisting of an Arduino-based system and an Android app, provides the following capabilities to facilitate the operation of a robotic vehicle in a shopping aisle:

Robotic Vehicle Control:

- The software enables registered operators to control the movement of the robotic vehicle within the shopping aisle.
- Operators can send commands to the vehicle using the Android app, which communicates with the Arduino-based system.

Motor Control:

- The software controls the motor drivers (L298N) connected to the robotic vehicle.
- It provides functions to move the vehicle forward, backward, left, and right, allowing precise navigation within the aisle.

Obstacle Detection:

- The software utilizes ultrasonic sensors to detect shoppers or obstacles present in the vehicle's vicinity.
- It continuously monitors the sensor readings and triggers appropriate actions when an obstacle is detected.

Operator Authentication:

- The Android app includes a login system that restricts access to registered operators only.
- Each operator needs to authenticate themselves before gaining control over the robotic vehicle.

Problematic Aisle Registration:

- When the vehicle detects shoppers or obstacles for more than four times, the software prompts the operator to take action.
- The operator can tap on the "Register Aisle" button in the Android app to register the problematic aisle in the database.

The interrelationships between these capabilities provide a comprehensive solution for operating a robotic vehicle in a shopping aisle while ensuring shopper safety and efficient navigation. The software combines motor control, obstacle detection, operator

authentication, and aisle registration functionalities to create a seamless and controlled experience for the operator.

5.2 Conventions:

- Colors in displays: The software uses a LiquidCrystal display with I2C communication. The backlight color is adjustable.
- Audible alarms: The software utilizes a buzzer for audible notifications, such as when obstacles are detected.
- Abbreviated vocabulary: The software uses abbreviations for pin names and motor commands, as specified in the code.
- Rules for assigning names or codes: The software follows specific rules for assigning names and codes to pins, sensors, and commands. These rules are defined in the code

5.3 Processing procedures:

- The subsequent paragraphs are organized by functionality and menu options.
- Procedures should be followed in the given order to ensure proper usage and functionality.

5.4 Aspect of software use:

- Function: Robotic Vehicle Operation
- Menus: The Android app provides a login screen for registered operators, and once logged in, it allows the operator to control the robotic vehicle using directional commands.
- Graphical icons: The Android app may include graphical icons for navigation and control.
- Data entry forms: The Android app may include forms for user input, such as login credentials or registration of problematic aisles.
- User inputs: The operator can input commands through the Android app, which are then transmitted to the robotic vehicle.
- Inputs from other software or hardware: The robotic vehicle receives obstacle detection inputs from ultrasonic sensors.
- Outputs: The outputs include motor commands to control the movement of the robotic vehicle and visual notifications on the LCD display.
- Diagnostic or error messages or alarms: The software displays error messages and alarms when obstacles are detected, signaling the need for the operator to take appropriate action.
- Help facilities: The Android app may provide on-screen descriptive or tutorial information to assist the operator in using the software.

5.5 Related processing:

- There may be background processing performed by the software, such as obstacle detection and continuous monitoring.

5.6 Data backup:

- The user manual should describe procedures for creating and retaining backup data, but since the provided code does not involve data storage or manipulation, data backup may not be applicable in this context.

5.7 Recovery from errors, malfunctions, and emergencies:

- The user manual should provide detailed procedures for restarting or recovering from errors or malfunctions during processing.
- It should also describe measures to ensure continuity of operations in the event of emergencies.

5.8 Messages:

- The user manual should list or refer to an appendix that lists all error messages, diagnostic messages, and information messages that can occur while using the software.
- Each message should be explained, and the recommended actions to be taken after each message should be described.

5.9 Quick-reference guide:

- If applicable, the user manual should provide or reference a quick-reference card or page summarizing frequently-used function keys, control sequences, formats, commands, or other aspects of software use.

6. QUERY PROCEDURES

This database stores relevant information regarding collision detection events. The following details are provided for each data element:

a. Data element name: Aisle Name
b. Synonymous names: None
c. Definition: The name or description of the registered aisle.
d. Format: Text.
e. Range and enumeration of values: No specific range or enumeration defined.
f. Unit of measurement: None
g. Data item names, abbreviations, and codes: None

6.2 Query Capabilities:

The software provides both preprogrammed and ad hoc query capabilities. Preprogrammed queries are predefined queries that can be executed with specific parameters or criteria. These queries are designed to retrieve commonly requested information or perform common operations. Ad hoc queries, on the other hand, allow users to create custom queries based on their specific requirements. Users can define their own criteria and parameters to retrieve specific data from the database.

6.3 Query Preparation:

- Determine the purpose of your query and the specific information you want to retrieve or actions you want to perform.
- Identify the appropriate query type, whether preprogrammed or ad hoc.
- If using a preprogrammed query, refer to the available predefined queries provided by the software. Each query will have a specific format or structure.
- If using an ad hoc query, use the query interface provided by the software to construct your query. Follow the syntax and rules specified by the software for building valid queries.
- Specify the criteria or parameters for your query. This may include specifying data filters, sorting options, or other relevant conditions.

- Validate your query to ensure it is properly formed and will produce the desired results.
- Execute the query and review the results.

Note: The specific instructions and steps for query preparation may vary depending on the software implementation and interface used.

6.4 Control Instructions:

Control instructions are necessary for the sequencing of runs and other actions required to extract responses to query requests. These instructions include control statements that may be required by the computer system or software. To ensure proper control and execution of queries, follow the instructions below:

- Determine the appropriate sequence of actions required to initiate a query.
- Identify any specific control statements or commands provided by the software to execute queries.
- Follow the software's guidelines for initiating queries, including any required input or parameters.
- If there are additional control instructions related to query execution, such as error handling or result processing, adhere to the guidelines provided by the software.
- Review the software documentation or user manual for any specific considerations or best practices related to query control instructions..

7. **NOTES**

1. General Information

This user manual provides instructions and guidelines for operating the robotic vehicle in a shopping aisle using the accompanying Android app and the integrated software system. The system is designed to assist registered operators in safely navigating the vehicle while detecting shoppers and obstacles in the vicinity. The manual covers various aspects of the system, including setup, operation, query capabilities, and control instructions.

2. Acronyms and Abbreviations

The following acronyms and abbreviations are used throughout this document:

- ENA: Enable Motor A
- ENB: Enable Motor B
- IN1: Input 1
- IN2: Input 2
- IN3: Input 3
- IN4: Input 4
- LCD: Liquid Crystal Display
- POSITIVE: Positive polarity
- trig: Trigger pin
- echo: Echo pin
- buzzer: Buzzer pin

3. Terms and Definitions

The following terms and definitions are relevant to understanding this document:

- Android App: Refers to the mobile application installed on an Android device that allows registered operators to log in and control the robotic vehicle.

- Collision Detection: The capability of the system to detect shoppers and obstacles in the shopping aisle to prevent collisions.
- Database: A structured collection of data that stores relevant information about collision detection events and other system data.
- Preprogrammed Query: A predefined query with specific parameters or criteria that can be executed to retrieve commonly requested information or perform common operations.
- Ad hoc Query: A custom query created by users based on their specific requirements, allowing them to define their own criteria and parameters to retrieve specific data from the database.
- Query: A request for information or an action performed on the database to retrieve specific data or perform operations.
- Query Capabilities: The preprogrammed and ad hoc query functionalities provided by the software to retrieve information or perform actions on the database.
- Query Preparation: The process of preparing queries, including defining the purpose, selecting the appropriate query type, specifying criteria, and validating the query before execution.
- Control Instructions: Instructions and commands necessary for sequencing runs and other actions required to extract responses to query requests, including any control statements or commands required by the computer system or software.

8. REFERENCE

Arduino website: https://www.arduino.cc/
Arduino is a popular platform for building electronics projects, including collision detection systems. Their website offers tutorials, project examples, and a community forum where you can find relevant information.

Adafruit Learning System: https://learn.adafruit.com/
Adafruit provides a learning platform with a wide range of tutorials and guides on electronics and DIY projects. You can search for topics related to collision detection, Bluetooth modules, ultrasonic sensors, LCD displays, and more.

Instructables: https://www.instructables.com/
Instructables is a website where users share step-by-step guides for various projects. You can search for collision detection systems or specific components to find detailed instructions and examples.

GitHub: https://github.com/
GitHub is a code hosting and collaboration platform. You can search for open-source projects related to collision detection systems or specific components. Many projects include documentation and code that can help you understand the implementation.

Additionally, you can search on technical forums such as Stack Overflow (https://stackoverflow.com/) and electronics-specific forums like Electronics Stack Exchange (https://electronics.stackexchange.com/) for discussions, questions, and answers related to collision detection systems and their components.

A. APPENDIXES

Appendix A: System Components and Code

This appendix provides detailed information about the system components and the associated Arduino code used in the implementation of the robotic vehicle. It includes the pin configurations, motor driver setup, sensor connections, and the code for controlling the vehicle's movements, obstacle detection, and display functionality.

A.1. System Components
The following components are used in the system:

- Motor Driver: L298N
- Sensors: Ultrasonic distance sensors (trig and echo)
- LCD Display: LiquidCrystal_I2C (0x27)
- Buzzer: Connected to a digital pin

A.2. Pin Configurations

The pin configurations for various components are as follows:

Motor Driver (L298N):

  - IN1: Pin 7
  - IN2: Pin 6
  - IN3: Pin 5
  - IN4: Pin 4
  - ENA: Pin 9
  - ENB: Pin 3

Ultrasonic Distance Sensor (1):

  - Trigger Pin: 11
  - Echo Pin: 12

Ultrasonic Distance Sensor (2):

  - Trigger Pin: 8
  - Echo Pin: 13

Buzzer: Pin 10

LCD Display (LiquidCrystal_I2C):

Address: 0x27
Pins: 2, 1, 0, 4, 5, 6, 7, 3
A.3. Arduino Code
Below is the Arduino code used in the system. It includes the setup and loop sections, as well as functions for executing commands, controlling motor movements, obstacle detection, and displaying information on the LCD.

```
// By Edwin Manda

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

/* Define section */

// Motor driver setup L298N
const int IN1 = 7;
const int IN2 = 6;
const int IN3 = 5;
const int IN4 = 4;

const int ENA = 9;
const int ENB = 3;

const int trig = 11;
const int echo = 12;

const int trig2 = 8;
const int echo2 = 13;

int buzzer = 10;
int command = 0;

long duration;
int dis;
int safe;

// New feature: obstacle count
int obstacleCount = 0;

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

/* Setup section */

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);

  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(trig2, OUTPUT);
  pinMode(echo2, INPUT);

  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.backlight();
  lcd.backlight();
}
```

```
/* Loop section */

void loop() {
  if (Serial.available() > 0) {
    command = Serial.read();
    executeCommand(command);
  }

  // First sensor
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  duration = pulseIn(echo, HIGH);
  dis = (duration * 0.034) / 2
```