RAVI BOT:
AI Based Shopping Assistant

Group Members

Ahmad Naveed	19I-0837
Saad Mahboob	19I-0846
Muhammad Huzaifa	18I-0815

Project Supervisor

Dr. Muhammad Tariq Khan

Department of Electrical Engineering

National University of Computer and Emerging Sciences,
Islamabad 2023

Developer's Submission

"This report is being submitted to the Department of Electrical Engineering of the National University of Computer and Emerging Sciences in partial fulfillment of the requirements for the degree of BS in Electrical Engineering."

Developer's Declaration

We take full responsibility for the project work conducted during the Final Year Project (FYP) titled **"RAVI BOT AI based Shopping Assistant".** We solemnly declare that the project work presented in the FYP report is done solely by us with no significant help from any other person; however, small help taken is duly acknowledged. We have also written the complete FYP report by ourselves. Moreover, we have not presented this FYP (or substantially similar project work) or any part of the thesis previously to any other degree awarding institution within Pakistan or abroad.

We understand that the management of Department of Electrical Engineering of National University of Computer and Emerging Sciences has a zero-tolerance policy towards plagiarism. Therefore, we as the author of the above-mentioned FYP report solemnly declare that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced. Moreover, the report does not contain any literal citing of more than 70 words (total) even by giving a reference unless we have obtained the written permission of the publisher to do so. Furthermore, the work presented in the report is our own work and we have positively cited the related work of the other projects by clearly differentiating our work from their relevant work.

We further understand that if we are found guilty of any form of plagiarism in our FYP report even after our graduation, the University reserves the right to withdraw our BS degree. Moreover, the University will also have the right to publish our names on its website that keeps a record of the students who committed plagiarism in their FYP reports."

___________________	____________________	____________________
Muhammad Huzaifa	Ahmad Naveed	Saad Mahboob
BS(EE) 2018-0815	BS(EE) 2019-0837	BS(EE) 2019-0846

____

Certified by Supervisor

____

Verified by Plagiarism Cell Officer

Dated: __________

Abstract

The RAVI Bot AI based Shopping Assistant revolutionizes the retail industry by seamlessly blending artificial intelligence, robotics, and advanced algorithms. With its personalized recommendations, effortless product searches, and streamlined checkout process, it transforms the shopping experience into a delightful journey. Equipped with computer vision and sensor technologies, the assistant navigates through aisles, interacts naturally with customers, and ensures their satisfaction. It also enables targeted marketing strategies based on deep customer insights, allowing retailers to deliver personalized promotions and experiences that drive customer loyalty and increase revenue. Simultaneously, it empowers store owners with efficient inventory management, targeted marketing, and revenue optimization. As this cutting-edge solution continues to shape the future of retail, it creates a harmonious synergy between humans and intelligent robots, redefining convenience, efficiency, and engagement in the world of shopping.

Acknowledgements

First and foremost, we would like to express our sincere gratitude to our supervisors, Dr. Muhammad Tariq Khan for providing guidance, support, and valuable insights throughout the whole project. Their expertise and encouragement have been instrumental in shaping the project's success. We would specially thank Dr. Muhammad Tariq Khan for constantly motivating us to work hard and the importance of group work.

We would like to express our sincere gratitude to the institution that supported the project, providing resources, infrastructure, and funding. Their support has played a crucial role in enabling the project's execution and realization.

Table of Content

Contents

List of Figures

List of Tables

Chapter 1: Introduction

Emerging requirements for robots in the commercial sector. Much research is going on in our desired area, but real-time implementation has been challenging. “Humanoid Robot Market in the US” is expected to reach $17.3 billion by 2027, growing at a CAGR (Compound Annual Growth Rate) of 63.5% from 2022 [1]. The stats suggest that the robotic industry is growing exponentially in Europe and America. Whereas in Pakistan, this is a relatively new market with startups like LearnObots, Grit3Detc. Having been launched because of their FYP. We look forward to building a real-world system. During the past few years, startups have developed AI-based robots for various commercial setups.

We are looking to build a product that covers almost all the domains we have covered in our 4-year degree. From Engineering Drawing to Engineering Workshop to Control Courses to ECD and finally ML. A lot of research is going on in our desired area, but real-time implementation has been challenging. We look forward to building a real-world/time system.

As it is a new field here in Pakistan, the customers would be attracted to the brand. Saving Salary, and time, reducing staff, and freeing up your staff for the human-specific task. Walmart says, “We get robots to do the work our employees do not enjoy.” Customers can also provide verbal feedback to the robot during or after a store visit, and it has been proven that people are more honest in their feedback when speaking to a robot, as opposed to a human staff member [2].

Our bot would gather info about hot-selling products while roaming around all day and will upload it to the cloud at the end of the day. Based on which statistical analysis would be done.

1.1 Motivation

Firstly, customer expectations have undergone a significant shift in recent years. With the advent of e-commerce and online marketplaces, customers now demand personalized experiences, efficient transactions, and seamless interactions. Traditional brick-and-mortar stores are faced with the challenge of adapting to these changing expectations and finding ways to provide exceptional customer service in a highly competitive market.

Secondly, there is a growing need for retailers to optimize their operations and improve efficiency. Inventory management, stock replenishment, and personalized marketing strategies are vital components for success in the retail industry. The ability to leverage data, analytics, and automation to enhance these processes can significantly impact a retailer's bottom line and competitive advantage.

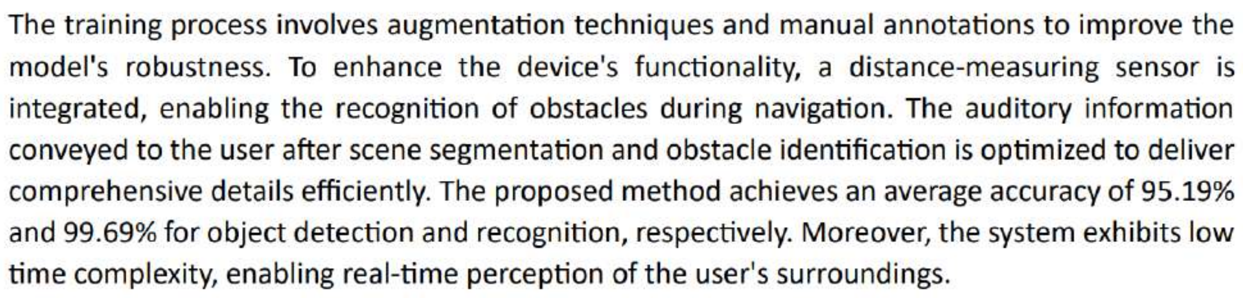
The integration of AI, robotics, and computer vision technologies presents an exciting opportunity to address these challenges and meet the evolving demands of customers and retailers alike. By developing the Bot AI based Shopping Assistant, we aim to leverage these cutting-edge technologies to revolutionize the retail landscape, create personalized shopping experiences, optimize inventory management, and drive revenue growth.

1.2 Problem Statement

Hospitals, shopping malls, and Stores must hire many workers for tedious and repetitive tasks. Hence, ending up sharing the income with fewer no employees. The problems faced by the owners are that humans tend to have more downtime due to numerous reasons, including sickness, family emergencies, personal events, holiday leaves, etc. They tend to impart biasedness, emotions, and judgments toward customers causing hindrances to corporate growth. Humans get to pass off mundane, repetitive tasks and adopt those that require critical thinking and problem-solving based on human intuition. Our solution is a Humanoid Robot. Machines do not fall ill; they do not need to isolate themselves to protect peers, and they do not need to take time off work. They can work around the clock and will not take bathroom breaks, and the owners will not have to raise the robot’s pay to meet the minimum wage criteria. Our solution will be a one-time investment with occasional maintenance and power charges.

1.3 Literature Review

Visually impaired people face numerous difficulties in their daily life, and technological interventions may assist them to meet these challenges. This research paper presents innovative assistive technology designed to address the challenges faced by visually impaired individuals. The proposed solution utilizes artificial intelligence to automatically recognize various objects in real-time and provides auditory feedback to the user, enhancing their understanding of the surrounding environment. A deep-learning model is trained using a diverse set of object images that are highly relevant to visually impaired individuals.



The training process involves augmentation techniques and manual annotations to improve the model's robustness. To enhance the device's functionality, a distance-measuring sensor is integrated, enabling the recognition of obstacles during navigation. The auditory information conveyed to the user after scene segmentation and obstacle identification is optimized to deliver comprehensive details efficiently. The proposed method achieves an average accuracy of 95.19% and 99.69% for object detection and recognition, respectively. Moreover, the system exhibits low time complexity, enabling real-time perception of the user's surroundings.

The near future has been envisioned as a collaboration of humans with mobile robots to help in the day-to-day tasks. The focus of the study is on developing a practical approach for real-time object detection and recognition using computer vision techniques. The proposed system aims to enable efficient indoor navigation for mobile robots, which are increasingly employed in home assistance, emergency services, and surveillance scenarios that demand quick response times. To achieve this, the paper introduces an algorithm based on a modified version of the You Look Only Once (YOLO) algorithm. This modified algorithm offers advantages such as reduced computational requirements and a smaller network structure, making it more suitable for real-time applications.

The performance of the proposed algorithm is compared with other conventional object detection and recognition algorithms in terms of metrics such as mean Average Precision score, mean inference time, weight size, and false positive percentage. Additionally, the framework utilizes the results of object detection and recognition to facilitate indoor navigation for mobile robots. The versatility of this framework suggests its potential for various applications involving indoor navigation robots in different service domains.

This research introduces a vision system based on the YOLO algorithm, designed to recognize stationary objects that may pose obstacles to a mobile robot. The system incorporates a Microsoft Kinect sensor to accurately identify objects and determine their distances. To enhance the image processing capabilities, a Nvidia Jetson TX2 GPU is employed. Experimental findings demonstrate the YOLO network's high reliability in detecting predefined obstacles it was trained on. Furthermore, the system effectively calculates distances using depth information provided by the Microsoft Kinect camera, with an error rate of less than 3.64%. These results highlight the system's proficiency in obstacle detection and distance estimation, contributing to safer navigation for mobile robots.

1.4 Scope of Project:

1.4.1 Project Objective:

To design and construct an AI-based shopping assistant robot named RAVI-BOT within 1 year.

1.4.2 Deliverables:

- To construct 4 feet tall shopping assistant robot.
- Robot should be driven through 4 high torque Pololu geared motors with 2-inch scooter tires.
- It has a 7-inch capacitive touch LCD for interfacing and display.
- Its battery should last up to 4 hours on full-load operation.

1.4.3 Milestones:

- Approve hardware design from supervisor -- 15 September.
- Submit hardware procurement form -- 18 September.
- Collect hardware -- 27 September.
- Create 3D model -- 15 October.

- 3D Print Parts -- 20 October
- Assemble hardware -- 28 October.
- Implement Yolov5 Algorithm -- 20 December

1.4.4 Technical Requirements:

- Robot needs to move at a max speed of 10 Km/s.
- It needs to localize itself through machine learning algorithms.
- Its base should not exceed 15x18 Inches.
- Its body should be 12x10 inches.

1.4.5 Limits and exclusions:

- University is responsible for providing all the hardware.
- Finished products will be the university's property.
- Supervisor can add or remove anything from the project.
- Project might be dismantled after completion

1.5 Report Outline

This report is organized into several chapters, providing a comprehensive overview of the project. Chapter 2 delves into the proposed solution, offering an in-depth analysis of the block diagram, process flowchart, and sensor integration with the processing module.

Moving on to Chapter 3, the focus shifts to the obtained results from extensive testing conducted on numerous subjects, aiming to enhance the device's accuracy. This chapter further explores the conclusions derived from the results and presents recommendations for future improvements and enhancements.

By structuring the report in this manner, readers will gain a comprehensive understanding of the proposed solution, the outcomes of testing, and potential avenues for further development.

Chapter 2: Solution Design and Implementation

This chapter discusses the complete design and implementation of the robot. It discusses the block diagram and module specifications. Further the details of the flow chart and hardware details are discussed. Lastly, guidance specifications along with calculations and physical design.

2.1 Block Diagram

Figure 2.1 shows the complete block diagram of the project. The details of each block with related technical specifications are discussed below.

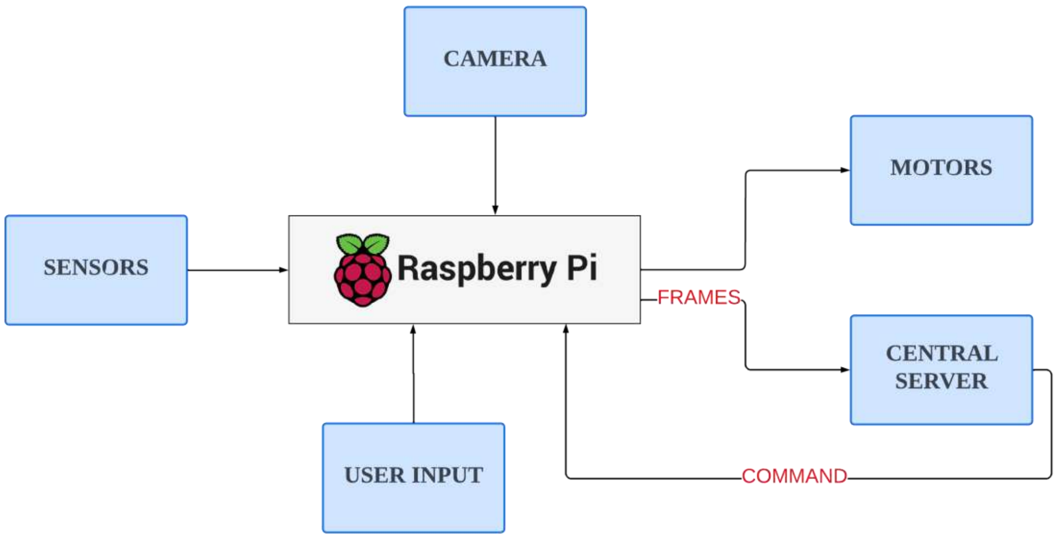


Figure 1 : System Block Diagram

2.2 Robot Specifications

The controller we are using is Raspberry Pi 4B along with 8MP Pi cam V2, speaker and microphone with amplifier, 7-inch touch screen LCD, Pololu geared motors, scooter wheels, Li-ion battery, L298N motor drivers, IR Sensors, proximity sensor and BMS.

2.2.1 Raspberry PI 4B

The Raspberry Pi 4 is the main controller of the shopping assistant robot. It serves as the central processing unit, handling the execution of algorithms, data processing, and decision-making tasks. It sends camera frames to the external server in real time to process them and send commands back to controller to move the robot. Along with it runs the GUI displayed on LCD to take input from the user and greets the customer through audio voice.

- Data input: It takes output of sensor as an input.
- Data output: It transmits the output of sensor to the processing module.

2.2.2 Pi Cam V2

The camera captures images, allowing the shopping assistant robot to perceive its surroundings, detect objects, and analyze visual information.

- Data input: Camera Serial Interface (CSI) that allows high-speed data transfer between the camera module and the Raspberry Pi.
- Data output: Image and video data, such as JPEG for images and H.264 for videos.

2.2.3 Motor Drivers

L298N H-Bridge motor drivers are used to enable precise control over the speed and direction of the DC motors. By varying the voltage and polarity applied to the motor, the H-Bridge driver can determine the motor's rotational direction and speed, allowing the robot to move forward, backward, or turn with defined speed.

- Data input: 2-bit digital combinations for the control and PWM signal for speed control.
- Data output: Provide the required voltage and current to drive the motors.

2.2.4 Ultrasonic sensors

Ultrasonic sensors are used to detect the presence and distance of objects in the environment. They enable the robot to navigate through store aisles, avoid obstacles, and provide accurate positioning for efficient and safe movement.

- **Data input:** It take digital signal as input to send trigger pulse.
- **Data output:** Distance between sensor and object.

2.2.5 Touch LCD

The touch LCD acts as the primary input interface for customers. It allows them to interact with the shopping assistant robot by selecting products, inputting preferences, browsing through categories, and accessing additional information. The intuitive GUI on the touch LCD enhances the usability and engagement of customers during their shopping journey.

- **Data input:** It takes commands as input.
- **Data output:** Display the advertisements.

2.2.6 IR Sensor

The robot utilizes the output from the IR sensor to determine the position of the line. By analyzing the output signal, the robot can make decisions on how to adjust its movement to stay on the line. If the sensor detects the line is deviating to the left, the robot can make a correction by turning slightly to the right. By continuously reading the IR sensor output and making adjustments accordingly, the robot can follow the line accurately.

- **Data input:** It doesn’t require any input to operate.
- **Data output:** Provide a binary signal indicating the presence or absence of an object within the detection range.

2.3 Flow Chart

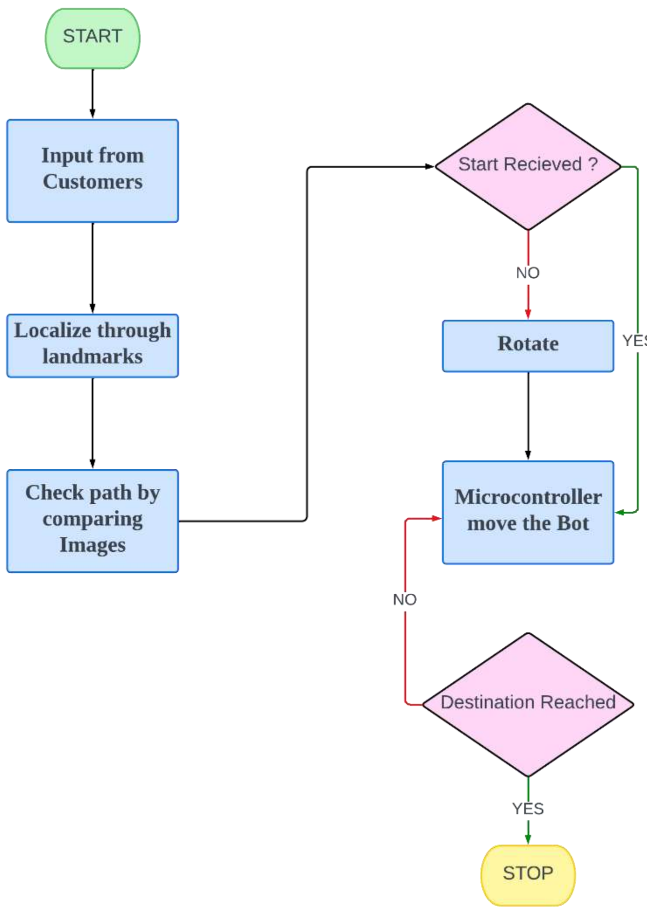

START
Input from Customers
Localize through landmarks
Check path by comparing Images
Start Recieved ?
NO
YES
Rotate
Microcontroller move the Bot
NO
Destination Reached
YES
STOP

Figure 2: Robot Movement Flow Chart

Description

The flowchart depicts the functioning of a shopping assistant robot, which begins by taking input from the customer. It then utilizes landmark localization techniques to determine its current position. By comparing images using correlation, the robot assesses the optimal path to reach the desired destination. Based on the customer's response, if the command is negative, the robot performs a rotation. The microcontroller subsequently directs the robot's movement. However, if the command is affirmative, the robot's moves in the same direction without any rotation. Once in motion, the robot periodically checks if the destination has been reached. If so, it ceases movement and informs the customer through voice.

2.4 Battery Management Systems

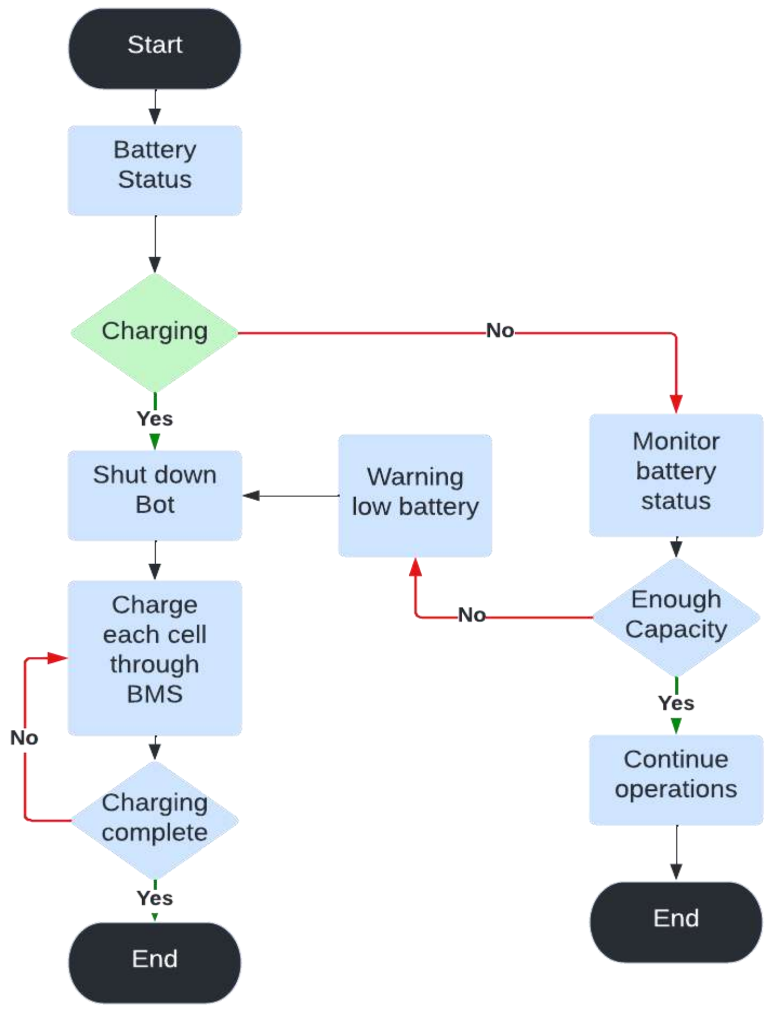


Figure 3: Battery Management System Flow Chart

Description

The shopping assistant robot is equipped with a battery system consisting of eighteen 18650 lithium-ion cells. During the robot's operation, it actively monitors the battery status to ensure sufficient charge capacity for movement. If the battery has enough charge to support the robot's tasks, it continues with its operational duties. However, if the battery becomes fully discharged, the robot gracefully shuts down and notifies the user to recharge the battery. The charging process is efficiently managed through a Battery Management System, which handles the charging cycle. Once the battery is fully charged, the robot is ready to resume its operations, ensuring uninterrupted functionality for its intended tasks.

2.5 Area Mapping

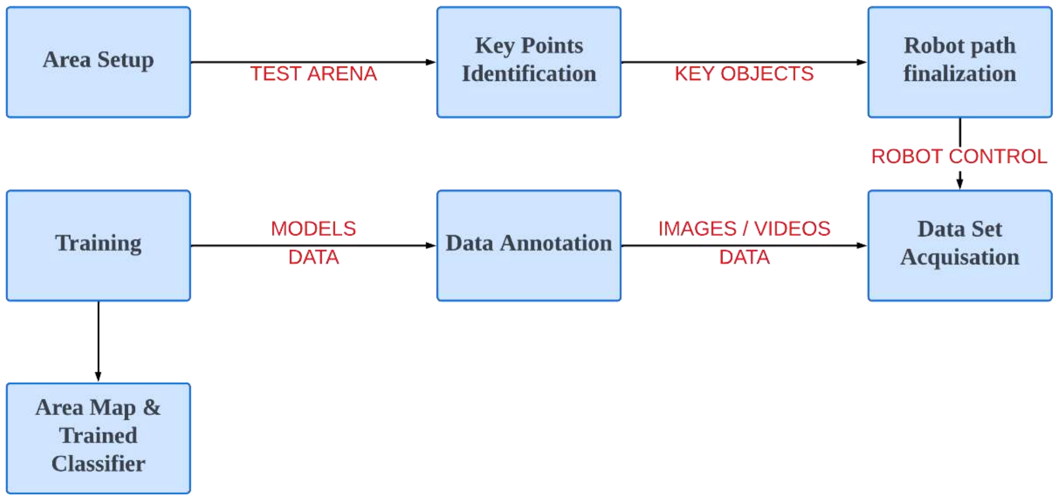


Figure 4: Area Mapping Block Diagram

2.5.1 Description

The camera serves a dual purpose in the robot system map making and localization. To begin, the designated area where the robot will operate needs to be set up. Within this area, various sections are strategically positioned as key points for identification. These sections act as reference points for the robot to determine its location accurately. Once the sections are established, the robot finalizes its path based on the identified key points. As the robot moves along its designated route, data acquisition takes place. This acquired data is manually annotated to provide meaningful information for subsequent analysis. The annotated data is then utilized to train our model on the map area and create a trained classifier. This trained classifier enables the robot to effectively recognize and navigate the different sections within the environment.

2.6 User Interaction:

- The user interacts with the robot by entering their desired location through LCD's user interface.
- The location information is sent from the robot to the central server for processing.

2.7 Image Processing and Analysis:

- The central server receives the location information and begins analyzing camera images received from the robot.

- The server utilizes computer vision techniques to determine the current position of the robot based on the camera images.
- To determine which path the robot is on, the server compares the features of the current image with two saved images of the object representing each possible path.
- The server uses image recognition algorithms to match the features of the current image with the saved images.
- The path with the minimum distance between the features of the current image and the saved image is selected as the current path of the robot.
- The server considers the current position of the robot along the selected path while generating movement instructions.

2.8 Instruction Generation:

- If the desired location is not yet reached, the central server generates movement instructions for the robot.
- Based on the analysis of camera images and the current path of the robot, the server determines whether the robot should rotate or proceed straight to reach the desired location.
- The movement instructions are sent from the central server to the robot.

2.9 Robot Movement:

- The robot receives the movement instructions from the central server.
- The robot executes the instructions, adjusting its movement accordingly.
- The robot continues to capture and transmit camera images to the central server for ongoing analysis.

2.10 Object Detection and Distance Verification:

- The central server analyzes the camera images received from the robot to detect the desired landmark based on the trained YOLO model.
- The server verifies if the detected object corresponds to the desired location landmark.
- The server calculates the distance to the object based on the number of pixels the object occupies in the frame.
- If the object is within a certain distance threshold, the server confirms that the desired location has been reached.
- By incorporating image recognition and path detection techniques into the system, the shopping assistant robot provides enhanced accuracy in determining the current position of the robot and enables it to select the appropriate path. This ensures efficient movement towards the desired location. Additionally, the system's object detection and distance verification capabilities further validate the robot's arrival at the specified landmark. Together, these features contribute to a reliable and effective shopping assistant robot.

2.11 Calculations

Table 1: Power Calculations

Sr no.	Equipment Name	Max Current	Power Consumption
1	Raspberry Pi 4B	1280 mA	6.4 W
2	Pi cam	280mA	1.4 W
3	7-inch TFT Touch Screen LCD	800mA	4 W
4	Geared motor and wheel x 4	350mA	16.8 W
5	L298N Motor Driver x 2	36 mA	0.18 W
6	BMS	40mA	0.2 W
7	Voltage regulator x 2	10mA	0.1 W
Total		3.323 A	29.08 W

To Run System for continuous 4.5 hours we need

$$29.08W \times 4.5h = 130.86Wh$$

Converting into Amp Hour

$$\frac{130.86\,Wh}{3.7V} = 35.36\,Ah$$

As our battery capacity is 6000mAh so we need

$$\frac{35.36\,Ah}{6000mAh} = 5.89\,units$$

So, we are using a bank of 6 cells with 3 banks in series to the power system for approx. 4.5 hours.

Our battery capacity is

$$6000mAh \times 3.7V = 22.2\,Wh$$

For 18 units

$$22.2\,Wh \times 18 = 399.6Wh$$

Now In 1 energy unit we can charge our system for

$$\frac{1000\,Wh}{399.6\,Wh} = 2.5\ times$$

As 1kWh costs around average 32 Rs. So cost per charge is

$$\frac{32\,Rs}{2.5} = 12.8\,Rs$$

2.12 Software Implementation

For creating the PCB (Printed Circuit Board) of our robot, we employ KICAD, a powerful software tool. Through KICAD, we meticulously design the schematic and PCB layout, integrating the entire circuitry of our robot. This enables us to optimize the arrangement and connectivity of electronic components, ensuring a well-structured and efficient embedded system at the core of our robot.

Here are listed the schematic and layout of printed circuit boards.

2.13 PCB Schematic

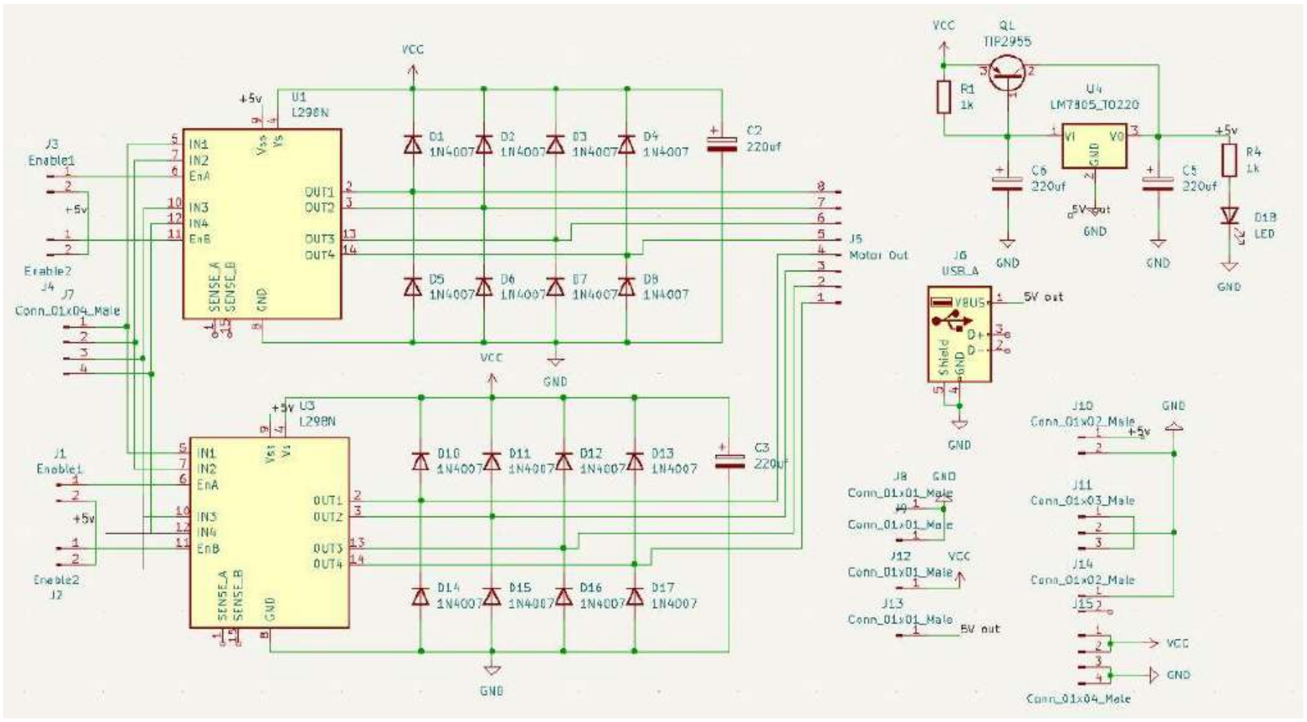


Figure 5: PCB Schematic

The system comprises two L298N motor driver modules connected in a parallel configuration to drive the motors. Diodes are incorporated to safeguard the circuitry from the back electromotive force (EMF) generated by the motors. A 5V voltage regulator, specifically the LM7805, is employed in conjunction with a current boost circuit to ensure a stable 5V output. Furthermore, JST connectors are utilized for the input, output, and power connections, providing a reliable and standardized interface.

2.14 PCB Layout

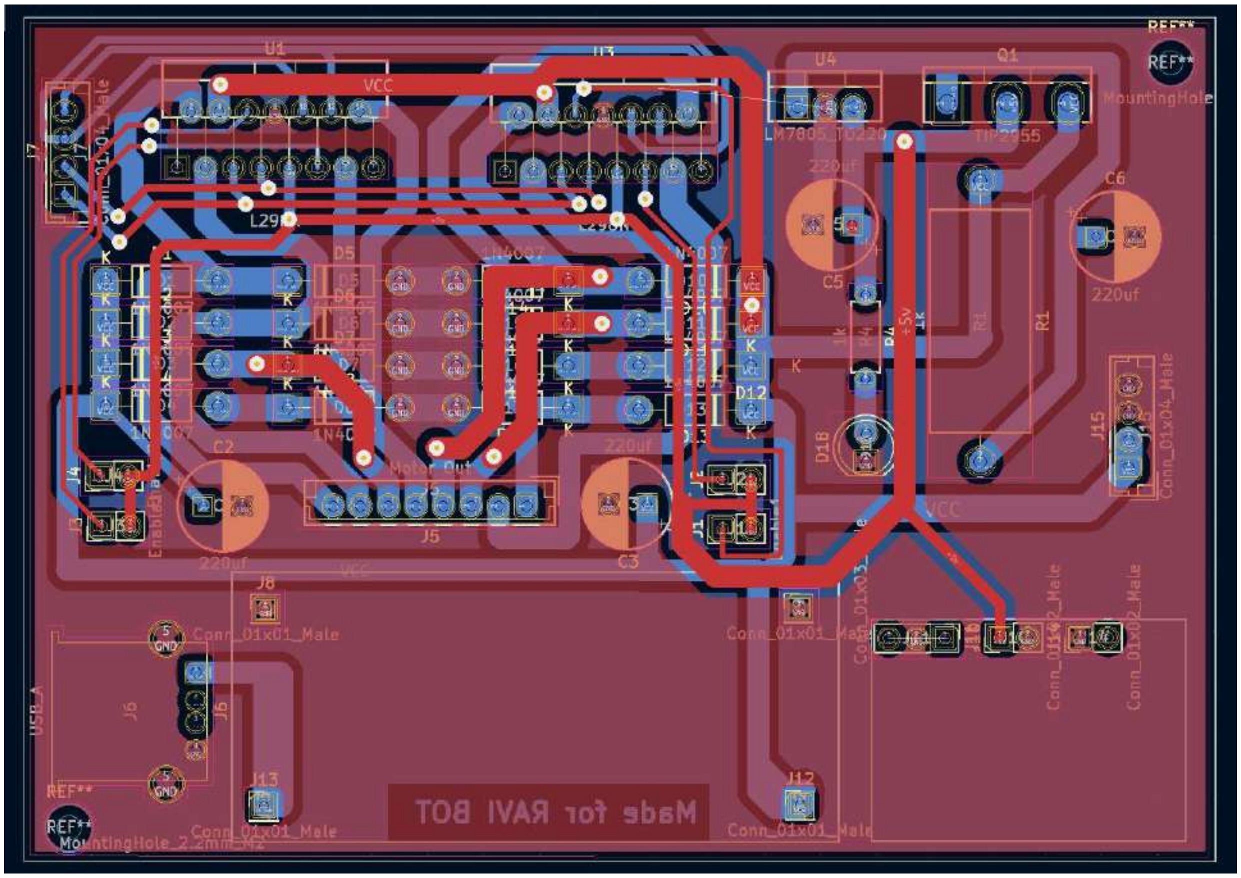


Figure 6: PCB Layout

The PCB design consists of two layers with distinct routing specifications. The power lines and motor input routes are allocated a thickness of 4mm, ensuring efficient current flow. On the other hand, the signal wires are designed with a thickness of 1.5mm, optimizing their transmission capabilities.

Considering the circuit is intended for fabrication at home, the via diameter is set at 1.5mm. This dimension allows for manual drilling, ensuring ease of fabrication while maintaining the necessary electrical connectivity between layers.

2.15 Graphical User Interface

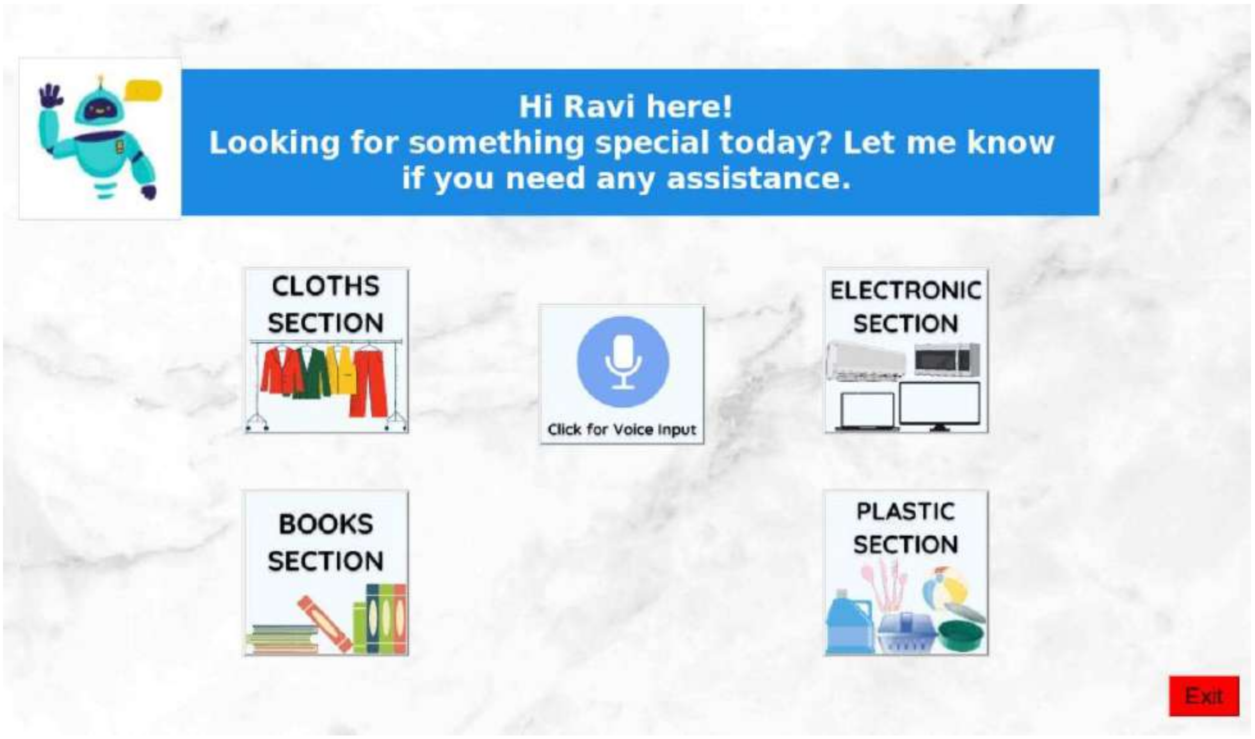


Figure 7: Graphical User Interface

We have created a Graphical User Interface (GUI) for the robot that offers a visually pleasing and intuitive experience to users. The GUI is designed using Python and utilizes the tkinter library, known for its robust features. By leveraging the power of tkinter, we have developed an interactive interface that seamlessly integrates with the robot's capabilities. The GUI provides customers with a clear representation of various sections on an LCD screen, enabling easy navigation and enhancing the overall user experience. With its user-friendly design, our GUI serves as a visual guide, ensuring customers can effortlessly explore different sections and interact with the robot's functionalities.

2.16 Hardware Implementation

Hardware implementation was carried out in the following two stages.

1. Electronics of the Robot
2. Body of the Robot

2.16.1 Electronics of the Robot

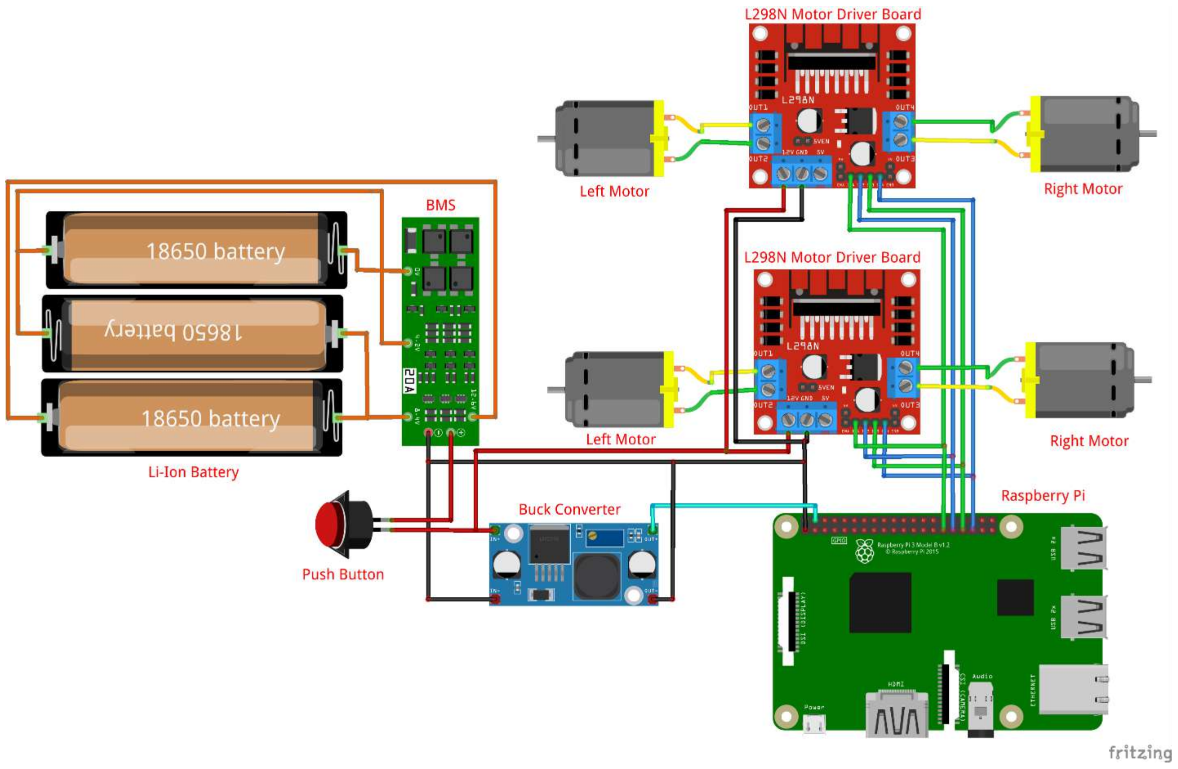


Figure 8: Electronic Block Diagram

The shopping assistant robot system comprises several key components for its operation. At the core of the system, the Raspberry Pi 4B serves as the central controller. For precise and controlled movement, a motor driver L298N is integrated, providing accurate motion control capabilities. Power management is handled by a buck converter, which efficiently regulates the voltage. The robot's locomotion is powered by 12V geared motors, providing the necessary force for seamless navigation. To supply the required power, a dedicated battery pack is utilized, while a power switching mechanism ensures smooth transitions between power sources. Lastly, a charging jack enables convenient recharging of the robot's battery pack, ensuring uninterrupted functionality.

2.16.1.1 Raspberry Pi 4B:

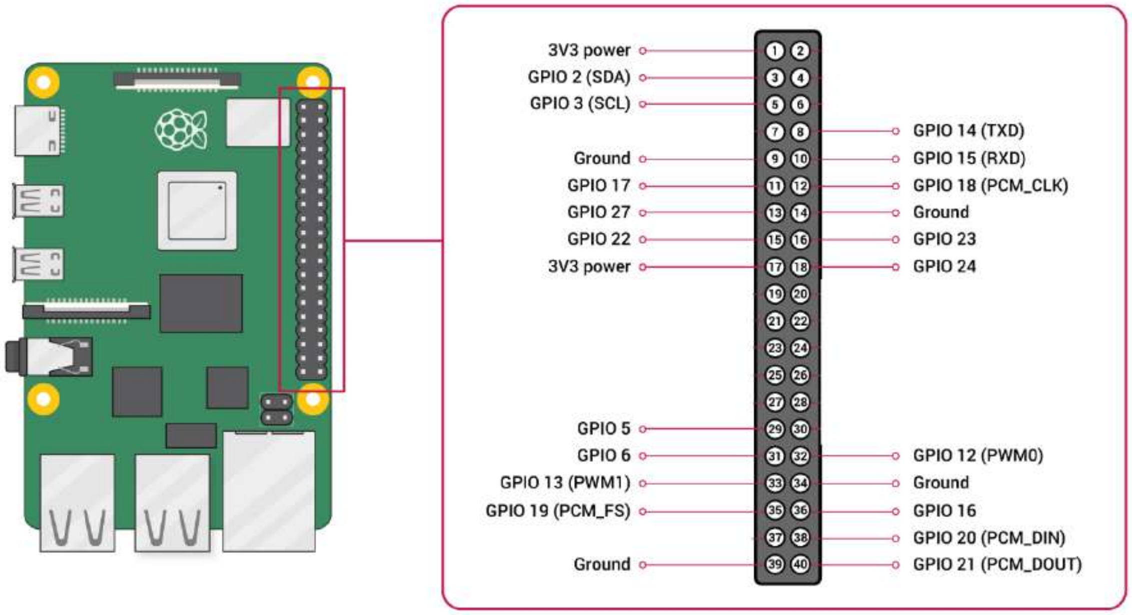
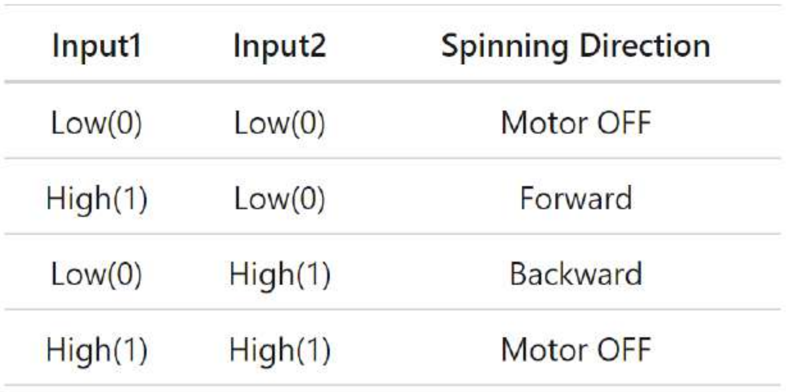


Figure 9: Raspberry Pi 4B Pinout

The board incorporates a high-performance quad-core ARM Cortex-A72 processor, delivering significant computational prowess. It boasts GPIO (General Purpose Input/Output) pins, enabling seamless integration with external electronic components and devices. In our system, we leverage these GPIO pins for various sensors, allocating 6 pins for IR sensors, 2 pins for an ultrasonic sensor, 5 pins for motor drivers, and additional pins for supporting complementary operations.

2.16.1.2 Motor Driver L298N



Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

Figure 11: Motor Driver Control Sequence

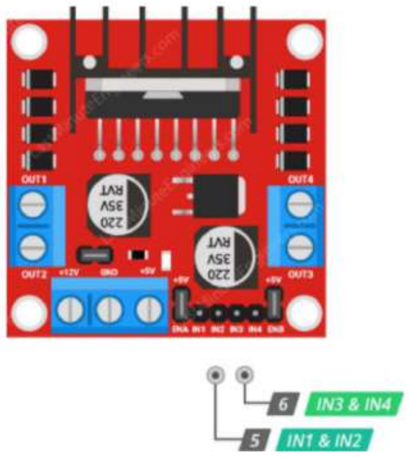


Figure 10: L298N Motor Driver

The L298N motor driver is a dual H-bridge module that can control the direction and speed of two DC motors independently. It works by utilizing H-bridge circuitry, which consists of four switches (transistors or MOSFETs) that can be controlled to manipulate the motor's behavior.

2.16.1.3 LCD Display

Figure 12: LCD Display

The 7-inch touch screen display used to take input form user and for showing Ads to the customers.

2.16.1.4 Motor and wheels

Figure 13: Pololu geared Motor

Figure 14: Scooter Wheel

For high torque and low speed, we utilize the Pololu geared motor with rotary encoder for precise measurement of movement. Along with Scooter wheels are used with coupler to couple them with motors shaft.

2.16.2 Body of the Robot

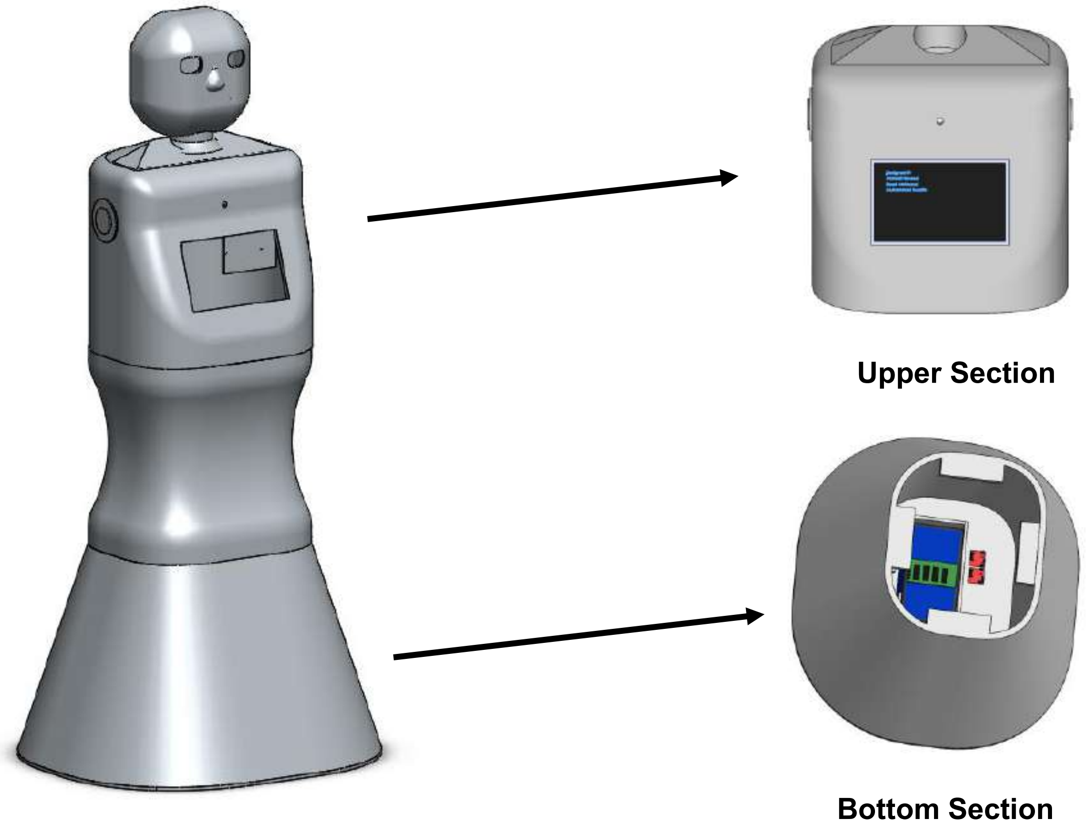


Figure 15: Robot SOLIDWORKS Model

The body of the robot is designed using SOLIDWORKS, a robust computer-aided design (CAD) software commonly utilized for modeling and designing mechanical components and systems. The body is divided into three sections: the bottom, upper, and face.

The face and upper sections are manufactured using a 3D printer. Due to limitations in the printing area, these sections are divided into multiple parts for printing. Afterward, the printed parts are joined together using adhesive to create a cohesive unit. The upper section accommodates the display and controller components. On the other hand, the bottom section is constructed using 5mm and 3mm Acrylic sheets. This section houses the battery, motors, and associated control circuitry.

2.17 Physical Model

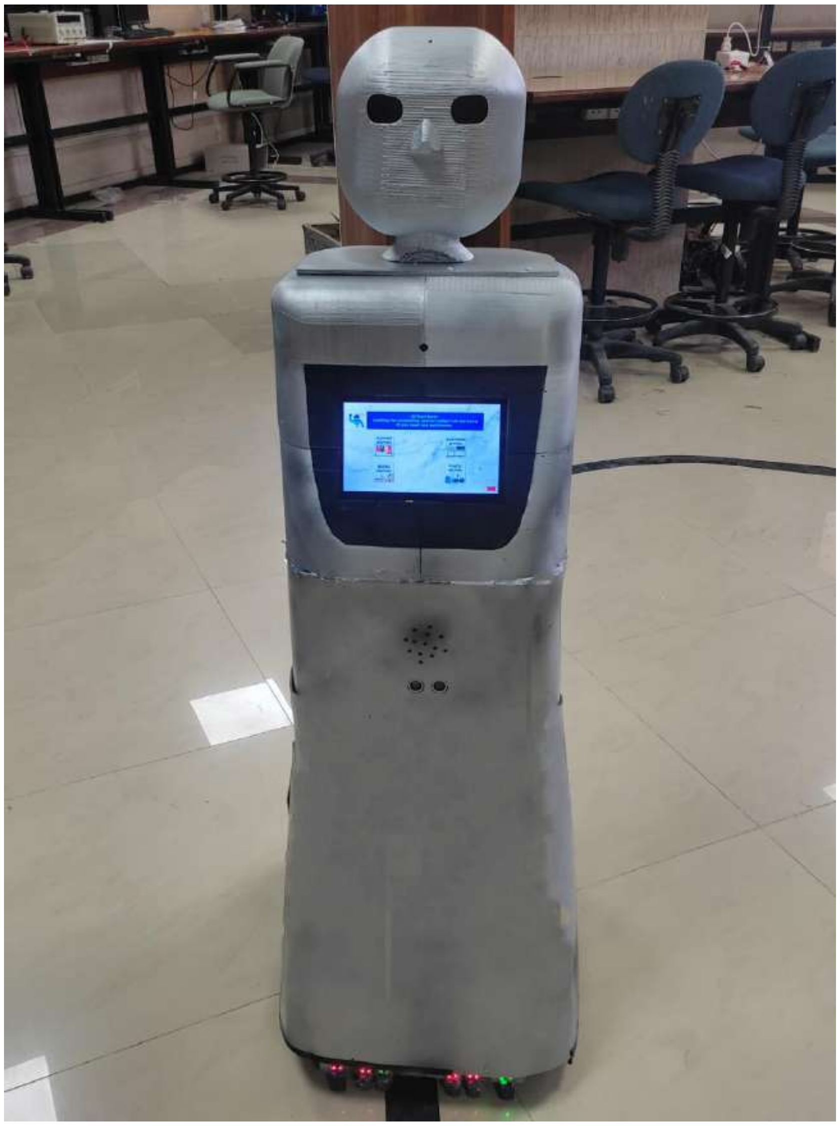


Figure 16: Physical Picture of Robot

Chapter 3: Results and Recommendations

The shopping assistant robot can revolutionize the retail industry in Pakistan by providing a unique and interactive shopping experience. It can assist customers in finding products, provide information, and offer personalized recommendations. With the robot's ability to navigate and locate items within the store, it can help optimize the shopping process, saving time for both customers and staff. This increased efficiency can lead to higher customer satisfaction and increased sales.

The integration of assistive technologies, such as object recognition and auditory feedback, can greatly benefit visually impaired individuals, improving their independence and accessibility while shopping. The introduction of a shopping assistant robot project in Pakistan showcases the country's commitment to innovation and technological advancements. It can attract attention and interest from both consumers and investors, fostering a culture of technological growth.

While automation can streamline certain tasks, it can also create new job opportunities related to robot maintenance, software development, and customer service roles, thereby contributing to economic growth and employment.

3.1 Project Progress:

Following are the deliverables of the project

3.3.1 FYP 1 Deliverables:

1. Integrated Vehicle
2. Designing of Hardware

Specifications:

- Body Design on SOLIDWORKS
- Calibration of Camera and Raspberry Pi
- Develop Battery Management System

3.1.2 FYP 2 Deliverables:

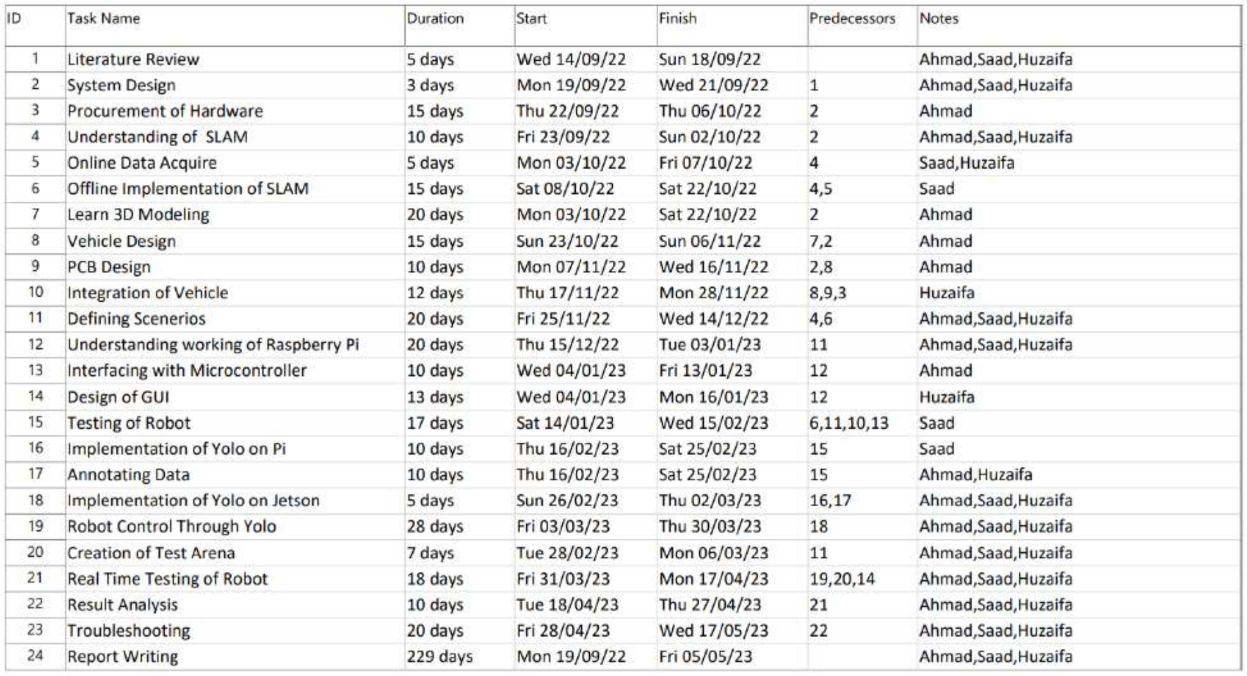
1. Final Product Prototype
2. Displaying real-time YOLOv5

Specifications:

- Deployment of Prototype
- Connecting Raspberry Pi to train model
- Showcasing test-arena

A project Schedule was prepared with the help of planning and management software (Microsoft Project). The above milestones were broken down into sub activities for easier monitoring and control. A screenshot of the schedule and the achieved progress by the end of final presentation of the project is shown below

Table 2: Project Schedule



ID	Task Name	Duration	Start	Finish	Predecessors	Notes
1	Literature Review	5 days	Wed 14/09/22	Sun 18/09/22		Ahmad,Saad,Huzaifa
2	System Design	3 days	Mon 19/09/22	Wed 21/09/22	1	Ahmad,Saad,Huzaifa
3	Procurement of Hardware	15 days	Thu 22/09/22	Thu 06/10/22	2	Ahmad
4	Understanding of SLAM	10 days	Fri 23/09/22	Sun 02/10/22	2	Ahmad,Saad,Huzaifa
5	Online Data Acquire	5 days	Mon 03/10/22	Fri 07/10/22	4	Saad,Huzaifa
6	Offline Implementation of SLAM	15 days	Sat 08/10/22	Sat 22/10/22	4,5	Saad
7	Learn 3D Modeling	20 days	Mon 03/10/22	Sat 22/10/22	2	Ahmad
8	Vehicle Design	15 days	Sun 23/10/22	Sun 06/11/22	7,2	Ahmad
9	PCB Design	10 days	Mon 07/11/22	Wed 16/11/22	2,8	Ahmad
10	Integration of Vehicle	12 days	Thu 17/11/22	Mon 28/11/22	8,9,3	Huzaifa
11	Defining Scenerios	20 days	Fri 25/11/22	Wed 14/12/22	4,6	Ahmad,Saad,Huzaifa
12	Understanding working of Raspberry Pi	20 days	Thu 15/12/22	Tue 03/01/23	11	Ahmad,Saad,Huzaifa
13	Interfacing with Microcontroller	10 days	Wed 04/01/23	Fri 13/01/23	12	Ahmad
14	Design of GUI	13 days	Wed 04/01/23	Mon 16/01/23	12	Huzaifa
15	Testing of Robot	17 days	Sat 14/01/23	Wed 15/02/23	6,11,10,13	Saad
16	Implementation of Yolo on Pi	10 days	Thu 16/02/23	Sat 25/02/23	15	Saad
17	Annotating Data	10 days	Thu 16/02/23	Sat 25/02/23	15	Ahmad,Huzaifa
18	Implementation of Yolo on Jetson	5 days	Sun 26/02/23	Thu 02/03/23	16,17	Ahmad,Saad,Huzaifa
19	Robot Control Through Yolo	28 days	Fri 03/03/23	Thu 30/03/23	18	Ahmad,Saad,Huzaifa
20	Creation of Test Arena	7 days	Tue 28/02/23	Mon 06/03/23	11	Ahmad,Saad,Huzaifa
21	Real Time Testing of Robot	18 days	Fri 31/03/23	Mon 17/04/23	19,20,14	Ahmad,Saad,Huzaifa
22	Result Analysis	10 days	Tue 18/04/23	Thu 27/04/23	21	Ahmad,Saad,Huzaifa
23	Troubleshooting	20 days	Fri 28/04/23	Wed 17/05/23	22	Ahmad,Saad,Huzaifa
24	Report Writing	229 days	Mon 19/09/22	Fri 05/05/23		Ahmad,Saad,Huzaifa

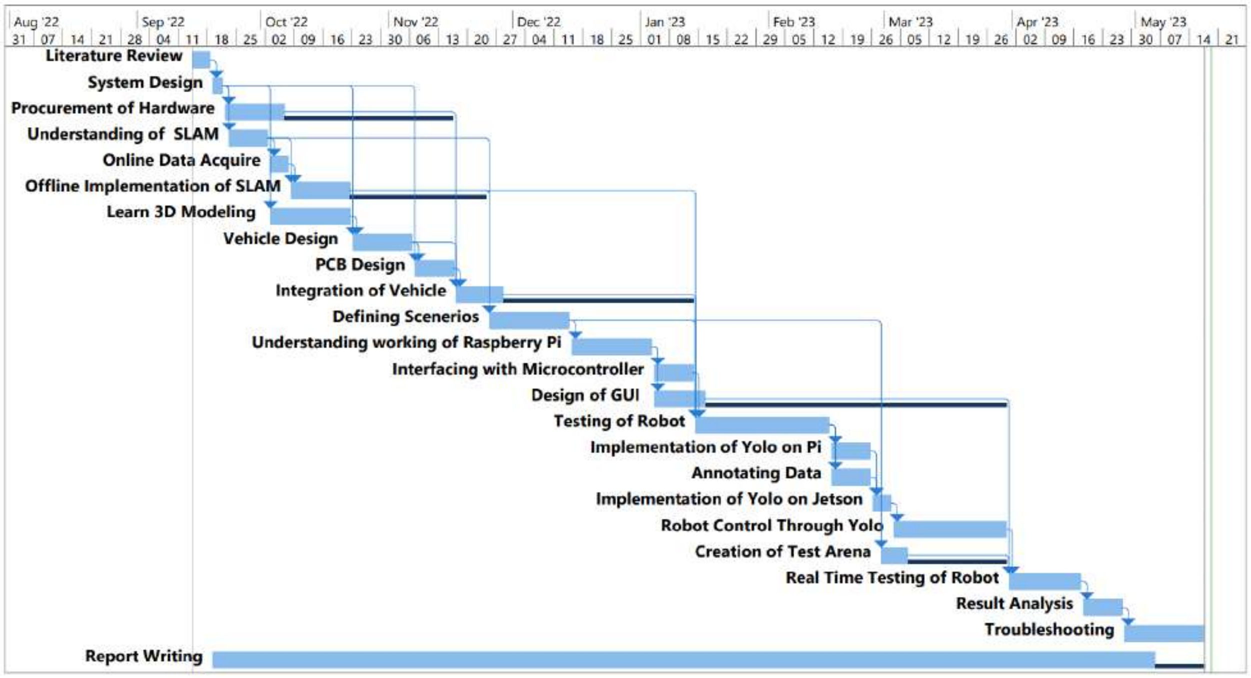


Figure 17: Gantt Chart

The work was distributed amongst the three group members for effective working on project. The group worked on the project on weekly basis, with dedicated targets for each week. Overall the project is completed within the planned period.

3.2 Results of Trained Model

Yolov5, a state-of-the-art object detection model, has been meticulously trained in four distinct sections - Book, Plastic, Clothing, and Electronics. This advanced training equips our robot with the ability to accurately identify and communicate with the specific section it has reached, ensuring a seamless shopping experience for our customers. Harnessing the power of Yolov5, our robot effortlessly navigates through these sections, providing precise location awareness and enhancing the overall convenience and efficiency of the shopping journey. Below figures shows the accurate real time detection of different sections in shopping stores.

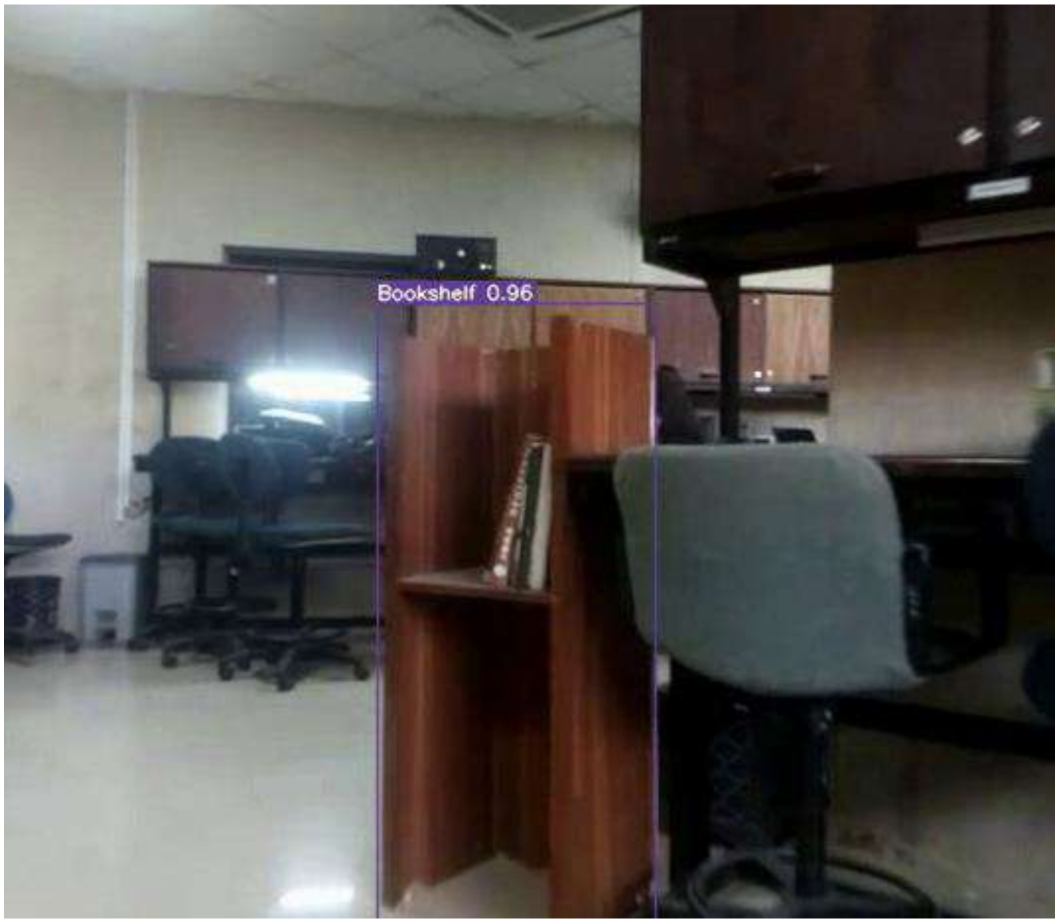


Figure 18:Detection of Books Section

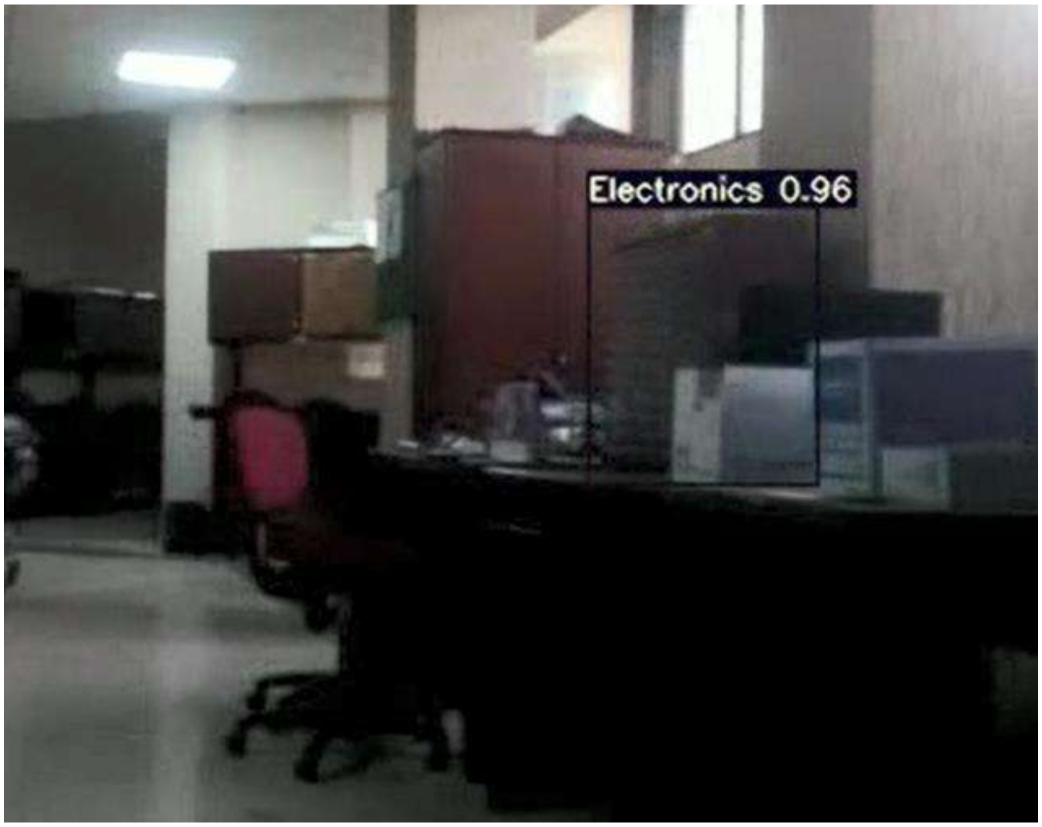


Figure 19: Detection of Electronics Section

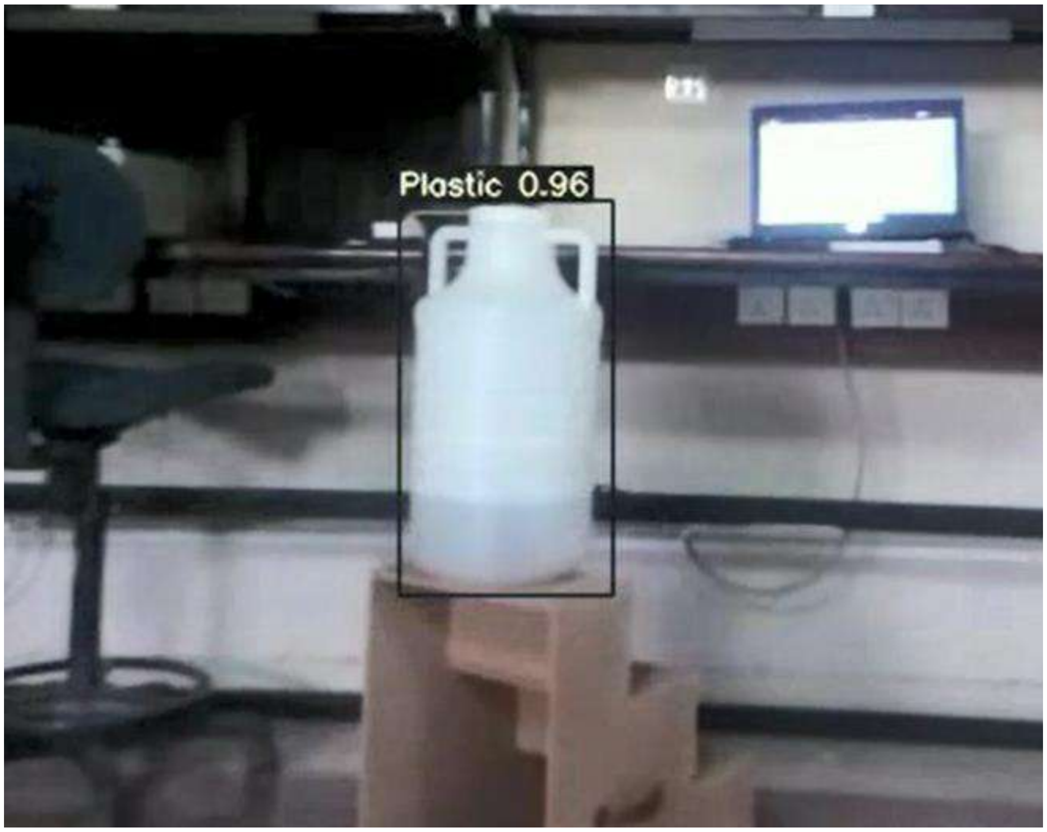


Figure 20:Detection of Plastic Section

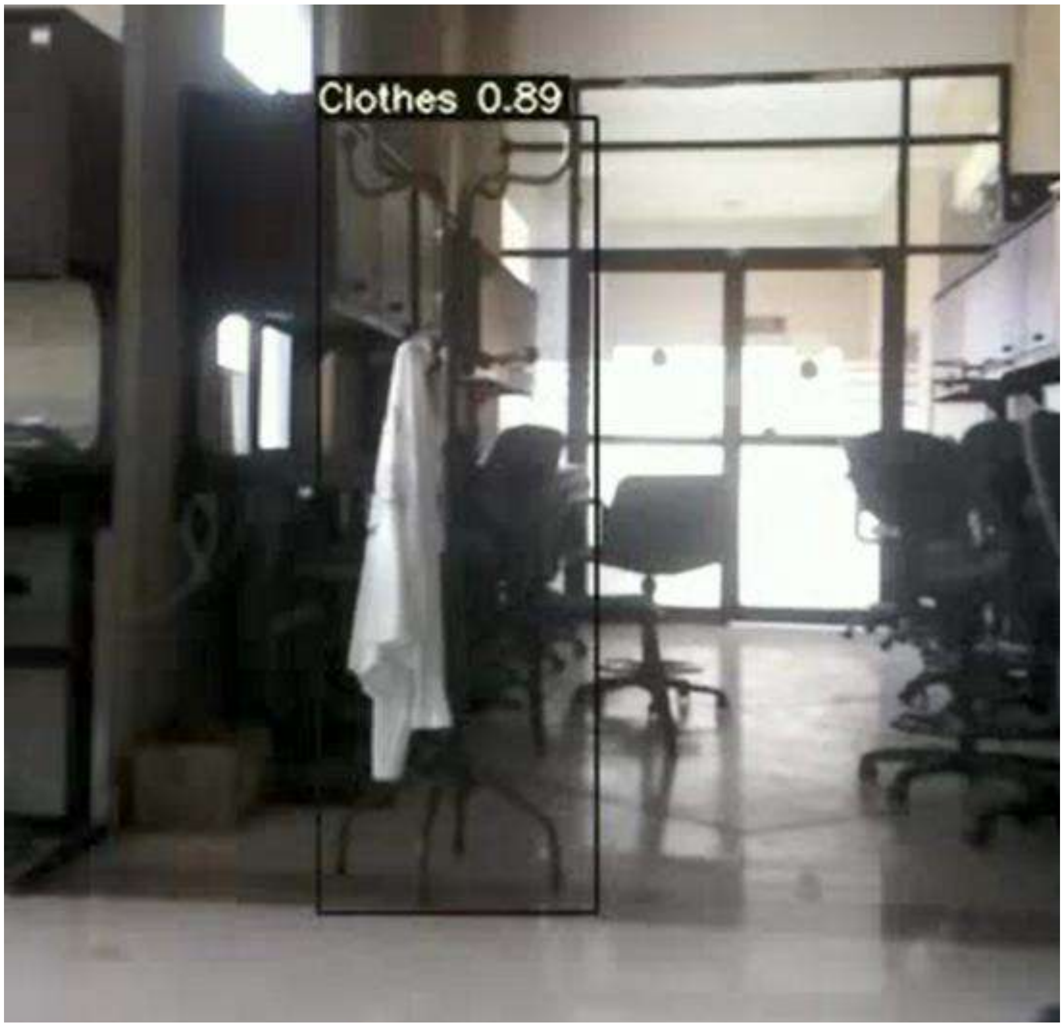


Figure 21: Detection of Clothes Section

3.3 Conclusion

In conclusion, the development of an AI-based shopping assistant robot marks a significant milestone in revolutionizing the retail industry. By harnessing the power of artificial intelligence, computer vision, and advanced robotics, this project has successfully created a cutting-edge solution to enhance the shopping experience for customers.

Through object recognition, localization, and intelligent navigation, the shopping assistant robot streamlines the process of locating products, providing personalized recommendations, and assisting visually impaired individuals. Its integration of a comprehensive graphical user interface, along with efficient circuitry and motor control, ensures seamless interaction and smooth operation.

Moreover, the use of state-of-the-art technologies such as Yolov5 for object detection, SolidWorks for precise modeling, KICAD for PCB design, and Raspberry Pi for control and computation, showcases the project's commitment to innovation and technological advancement.

With its potential to increase efficiency, improve accessibility, and elevate customer satisfaction, the AI-based shopping assistant robot represents a significant stride forward in merging human and robotic collaboration in the retail sector. It paves the way for a future where intelligent robots work alongside humans, transforming the way we shop and setting new standards for convenience and assistance.

As this project takes its place in the market, it holds the promise of reshaping the retail landscape, creating opportunities for increased automation, job creation, and technological growth. With ongoing advancements and continuous refinement, the AI-based shopping assistant robot has the potential to redefine the retail experience and set new benchmarks for innovation and customer service.

A key aspect of an engineering project is its role and impact on society and the environment. The primary goal of this project was to offer equal opportunities to all and provide ease in daily life activities. The project maps to one sustainable goal defined in the Sustainable Development Goals (SDGs) set by the United Nations General Assembly:

- SDG 09: Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

The retail industry is one of the oldest functioning sectors of market. Our product provides an innovative feature for physical retailers that will result in growth of industry and have long lasting impact on infrastructure.

In summary, this remarkable project showcases the immense potential of AI and robotics in reimagining the shopping journey, providing a glimpse into a future where intelligent machines work hand in hand with humans to create a more efficient, personalized, and accessible retail environment.

3.4 Recommendations / Future Work

Scalability and Adaptability: As the project expands, it is recommended to design the shopping assistant robot to be easily scalable and adaptable to different store layouts and environments. This could involve implementing advanced mapping and localization techniques that can quickly adapt to new store configurations without the need for extensive retraining.

Integration of Natural Language Processing: Incorporating natural language processing (NLP) capabilities into the shopping assistant robot can enable it to understand and respond to customer queries, providing personalized recommendations and assistance. This would enhance the interactive experience and provide more comprehensive support to customers.

Integration with Mobile Applications: Developing a companion mobile application that complements the shopping assistant can further enhance the customer experience. The application can provide additional features such as personalized shopping lists, real-time inventory updates, and seamless integration with online shopping platforms.

Integration with Online Shopping Platforms: Expanding the capabilities of the shopping assistant robot to seamlessly integrate with online shopping platforms can provide customers with a holistic shopping experience. This could involve incorporating features such as order placement, online payment integration, and tracking of online purchases.

Appendix-A: Project Codes

A.1 Main.py (Server)

The below code is to be run on the server side; this is where all the processing is being done. The code establishes a socket connection to receive video frames and location from the robot. The virtual map is first initialized of the shopping mall with the distance thresholds. The script enters a loop to continuously receive and process the video frames. It begins by reading the size of the frame and determining if it's a string or frame data is being sent. If it is the string, it unpacks the size data and proceeds to receive the location information. If the size of the frame is below a threshold, it sets a flag to indicate localization is required. For the next frames, it reads the size of the frame and proceeds to decode and process the frame. It runs a model on the frame to identify objects and their bounding box coordinates. If localization is flagged, it crops the frame based on the bounding box coordinates and compares it with reference images to determine the location based on the visible object to check the path on which it's standing and whether it has to rotate 180 degrees to get there through the shortest path or just start moving forward. Based on the location and the object class, it determines a route and direction. If the object class matches the location, it checks if the object is within certain dimensions to initiate a stop signal. The processed frames are displayed, and control signals are sent to the remote device based on the determined actions. The script continues this process until interrupted, and it cleans up resources upon completion.

```
import socket
import struct
import cv2
import numpy as np
import time
from myutils import Model_Run,compare_images

# Configure the socket connection
rasb_host = "192.168.79.102"
s_port = 9000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((rasb_host, s_port))
print("Connected to host")

# Mapping of location names to their corresponding numeric codes
names = {"CBookshelf": 0, "CElectronics": 1, "CClothes": 2, "CPlastic": 3, "ABookshelf": 4,
"AElectronics": 5, "AClothes": 6, "APlastic": 7}

# Path for storing comparison images
img_initial_path = "compare/"

# Create a server socket to listen for connections
```

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 8000))
t1 = time.time()
model_run = Model_Run()
server_socket.listen(1)
t2 = time.time()
print(t2 - t1)

# Accept a connection from a client
connection = server_socket.accept()[0]
print("Connected")
connection = connection.makefile('rb')
i = 0
thresh = 50000  # Threshold for size of the frame
localize_flag = False

# Mapping of regions, categories, and paths for comparison
maps = {
    "R1": {
        "Plastic": ["Bookshelf,Plastic", "CC"],
        "Clothes": ["Electronics,Clothes", "AC"],
        "Bookshelf": ["Bookshelf", "S"],
        "Electronics": ["Electronics", "S"]
    },
    "R2": {
        "Plastic": ["Plastic", "S"],
        "Clothes": ["Plastic,Clothes", "CC"],
        "Bookshelf": ["Bookshelf", "S"],
        "Electronics": ["Bookshelf,Electronics", "AC"]
    },
    "R3": {
        "Plastic": ["Plastic", "S"],
        "Clothes": ["Clothes", "S"],
        "Bookshelf": ["Plastic,Bookshelf", "AC"],
        "Electronics": ["Clothes,Electronics", "CC"]
    },
    "R4": {
        "Plastic": ["Clothes,Plastic", "AC"],
        "Clothes": ["Clothes", "S"],
        "Bookshelf": ["Electronics,Bookshelf", "CC"],
        "Electronics": ["Electronics", "S"]
    }
}

# Paths for comparison images
img_comparison_paths = {
    "Bookshelf": ["BS_A.png", "BS_C.png"],
    "Plastic": ["Plastic_A.png", "Plastic_C.png"],
    "Clothes": ["Clothes_A.png", "Clothes_C.png"],
    "Electronics": ["Electronics_A.png", "Electronics_C.png"]
}

# Stop conditions for each category
stop_close = {
    "Bookshelf": [250, 108],
    "Plastic": [140, 68],
    "Clothes": [400, 130],
    "Electronics": [208, 170]
}

first = 1

try:
    while True:
        if first:
            # Unpack the size of the frame as a 4-byte integer
            size_data = connection.read(4)

        if size < thresh:
            first = 0
            s_time = time.time()
            print("Receiving Location")
            location = connection.read(size)
            location = location.decode('utf-8') # Location recieved
            print("Received Location:", location)
            localize_flag = True

        if not first:
            for i in range(0, 5):
                size_data = connection.read(4)
                if not size_data:
                    break
                size = struct.unpack('>L', size_data)[0]  # Unpack the size of the frame as a 4-byte integer
                frame_data = connection.read(size)  # Read the frame data

            # Convert the frame data to a NumPy array
            frame = np.frombuffer(frame_data, dtype=np.uint8)

            # Decode the frame
            frame = cv2.imdecode(frame, cv2.IMREAD_COLOR)

            classes, bbox_data = model_run.run(frame, i)  # Run the model on the frame and get class
labels and bbox data
            a = []

            if bbox_data:
```

```
        classes = classes[0]
        classes = classes.split()[0]
        w = bbox_data[0]
        for j in w:
          a.append(float(j.cpu())) # converting the bbox from tensor to array
        print("Height:", a[3] - a[1], "Width:", a[2] - a[0])
        if localize_flag:
          print(classes)
          if a:
            cropped_frame = frame[int(a[1]):int(a[3]), int(a[0]):int(a[2])] # Cropping only the object
and comparing it to find the path
            cv2.imwrite("cropped_frame.jpg", cropped_frame)
            localize_flag = False
            paths = img_comparison_paths[classes] # Compare with the saved images
            dist = 10000
            img_name = []
            for p in paths:
              img = cv2.imread(img_initial_path + p)
              ## Finding the image with the least distance/most corrleation
              corelation_score = compare_images(img, cropped_frame)
              if dist > corelation_score:
                dist = corelation_score
                img_name = p.split('.')[0]
            print(img_name)
            ## Determine route using a virtual Map
            if img_name == "BS_A" or img_name == "Electronics_C":
              determined_route = "R1"
            elif img_name == "BS_C" or img_name == "Plastic_A":
              determined_route = "R2"
            elif img_name == "Clothes_A" or img_name == "Plastic_C":
              determined_route = "R3"
            elif img_name == "Clothes_C" or img_name == "Electronics_A":
              determined_route = "R4"
            print(maps[determined_route][location])
            path = maps[determined_route][location][0]
            direction = maps[determined_route][location][1]
            # Rotate 180 degrees if not facing the object on the path to go to
            if path.split(',')[0] != img_name:
              print("Rotate 180 degrees")
              string_data = "R"
              s.sendall(string_data.encode())# send data to rpi
            # Send start so that robot starts moving
            else:
              print("Start")
              string_data = "F"
              s.sendall(string_data.encode())# send data to rpi
        else:
          classes, bbox_data = model_run.run(frame, i)  # Run the model on the frame again
```

```
                print(classes)
                name = []
                conf = []
                maxconf_name = ''
                # Find the class with the max confidence score to deal with false detections
                for c in classes:
                    category, confidence = c.split()
                    name.append(category)
                    conf.append(confidence)
                if classes:
                    maxconf_index = conf.index(max(conf))
                    maxconf_name = name[int(maxconf_index)]
                else:
                    maxconf_name = ""
                print(location)
                # Check if the desired location reached
                if maxconf_name == location:
                    a = []
                    print("Classes same checking to stop")
                    w = bbox_data[maxconf_index]
                    for j in w:
                        a.append(float(j.cpu())) # converting the bbox from tensor to array
                    width = a[2] - a[0]
                    height = a[3] - a[1]
                    print("Height:", height, "Width:", width)
                    print("Width:", width, "Height:", height)
                    if stop_close[location][0] < height and stop_close[location][1] < width:
                        print("Stopping")
                        string_data = "s"
                        s.sendall(string_data.encode()) # send data to rpi
                        print(string_data)

        i += 1
        print(classes)
        cv2.imshow('Video Stream', frame) # Shpw the image on the server
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        e_time = time.time()
        if e_time - s_time >= 1:
            s_time = e_time
            s.sendall(string_data.encode()) # send data to rpi
            print(string_data)

finally:
    # Clean up the resources
    cv2.destroyAllWindows()
    connection.close()
    server_socket.close()
    s.close()
```

A.2 Helping Functions:

Compare images uses OpenCV’s built-in function to check the similarities between 2 different images and sends a value based on that. The smaller the value the closer the images are.

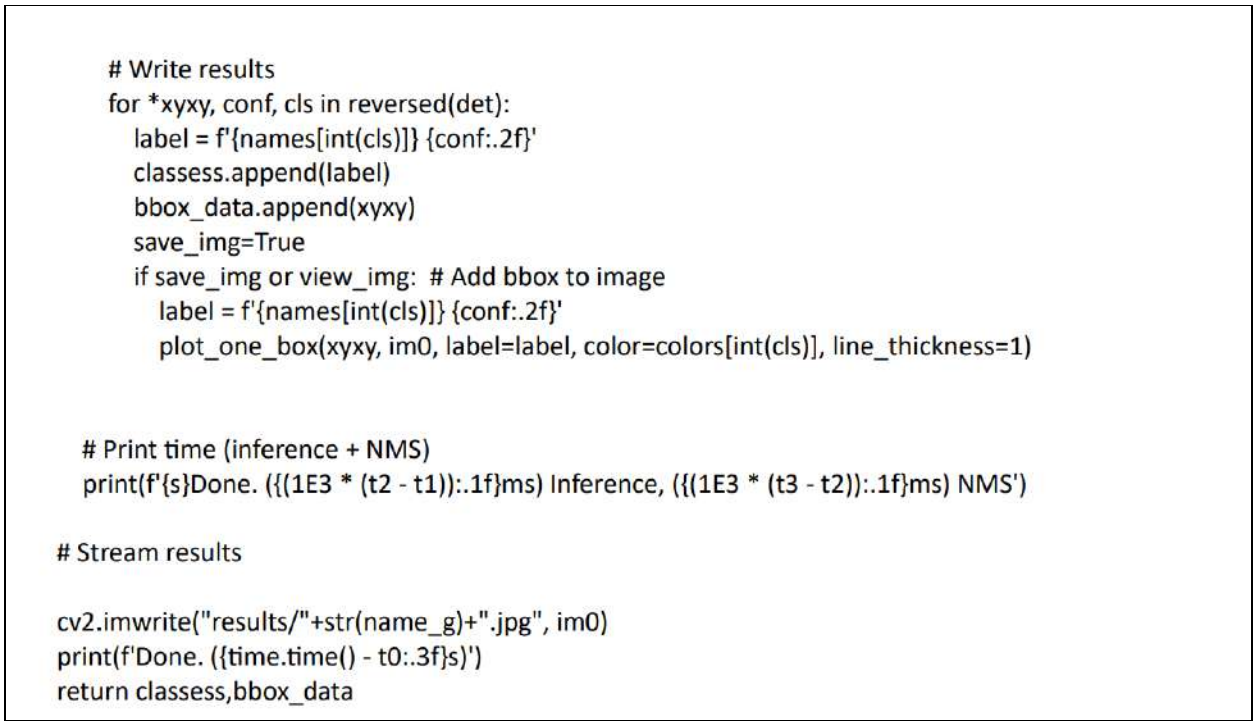
```
import cv2
import numpy as np

def compare_images(img1,img2):
# Convert images to grayscale
    gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

    # Create a SIFT object and detect keypoints and descriptors in the images
    sift = cv2.SIFT_create()
    kp1, desc1 = sift.detectAndCompute(gray1, None)
    kp2, desc2 = sift.detectAndCompute(gray2, None)

    # Create a brute force matcher and match the descriptors
    bf = cv2.BFMatcher()
    matches = bf.match(desc1, desc2)

    # Compute the average distance between matches
    distances = [m.distance for m in matches]
    avg_dist = np.mean(distances)

    return avg_dist
```

A.3 Inference using Yolo

The below codes are used to make inference using Yolo

```
import argparse
import time
from pathlib import Path
import numpy as np
import time
import argparse
import time

import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random
import numpy as np
from models.experimental import attempt_load
from utils.datasets import LoadStreams
from utils.general import check_img_size, check_requirements, check_imshow,
non_max_suppression, apply_classifier, \
   scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device, load_classifier, time_synchronized, TracedModel
import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random


def LoadImages(image, img_size=640, stride=32):  # for inference
        img0 = image
        assert img0 is not None, 'Image Not Found '
        #print(f'image {self.count}/{self.nf} {path}: ', end='')

        # Padded resize
        img = letterbox(img0, img_size, stride=stride)[0]

        # Convert
        img = img[:, :, ::-1].transpose(2, 0, 1)  # BGR to RGB, to 3x416x416
        img = np.ascontiguousarray(img)

        return  img
def letterbox(img, new_shape=(640, 640), color=(114, 114, 114), auto=True, scaleFill=False,
scaleup=True, stride=32):
   # Resize and pad image while meeting stride-multiple constraints
   shape = img.shape[:2]  # current shape [height, width]
   if isinstance(new_shape, int):
      new_shape = (new_shape, new_shape)

   # Scale ratio (new / old)
   r = min(new_shape[0] / shape[0], new_shape[1] / shape[1])
   if not scaleup:  # only scale down, do not scale up (for better test mAP)
      r = min(r, 1.0)

   # Compute padding
```

```
    ratio = r, r  # width, height ratios
    new_unpad = int(round(shape[1] * r)), int(round(shape[0] * r))
    dw, dh = new_shape[1] - new_unpad[0], new_shape[0] - new_unpad[1]  # wh padding
    if auto:  # minimum rectangle
      dw, dh = np.mod(dw, stride), np.mod(dh, stride)  # wh padding
    elif scaleFill:  # stretch
      dw, dh = 0.0, 0.0
      new_unpad = (new_shape[1], new_shape[0])
      ratio = new_shape[1] / shape[1], new_shape[0] / shape[0]  # width, height ratios

    dw /= 2  # divide padding into 2 sides
    dh /= 2

    if shape[::-1] != new_unpad:  # resize
      img = cv2.resize(img, new_unpad, interpolation=cv2.INTER_LINEAR)
    top, bottom = int(round(dh - 0.1)), int(round(dh + 0.1))
    left, right = int(round(dw - 0.1)), int(round(dw + 0.1))
    img = cv2.copyMakeBorder(img, top, bottom, left, right, cv2.BORDER_CONSTANT, value=color)  #
  add border
    return img, ratio, (dw, dh)

  class Model_Run:  # for inference
    def __init__(self):
      self.weights="best.pt"#best,yolov7
      self.imgsz=640
      self.viewimg=False
      self.conf_thres=0.45
    # Directories
    # Initialize
      set_logging()
      self.device = select_device('')
      self.half = self.device.type != 'cpu'  # half precision only supported on CUDA


    # Load model
      self.model = attempt_load(self.weights, map_location=self.device)  # load FP32 model
      self.stride = int(self.model.stride.max())  # model stride
      self.imgsz = check_img_size(self.imgsz, s=self.stride)  # check img_size
      print("In init")



    def run(self,frame,name_g):
      classess=[]
      bbox_data=[]
      names = self.model.module.names if hasattr(self.model, 'module') else self.model.names
 colors = [[random.randint(0,25) for _ in range(3)] for _ in names]
      if self.half:
         self.model.half()  # to FP16
      if self.device.type != 'cpu':
         self.model(torch.zeros(1, 3, self.imgsz,
 self.imgsz).to(self.device).type_as(next(self.model.parameters())))  # run once
      old_img_w = old_img_h = self.imgsz
      old_img_b = 1
      t0 = time.time()
      image=LoadImages(frame, img_size=self.imgsz, stride=self.stride)
      img = torch.from_numpy(image).to(self.device)
      img = img.half() if self.half else img.float()  # uint8 to fp16/32
      img /= 255.0  # 0 - 255 to 0.0 - 1.0
      print("Img shape is ",img.shape)

      if img.ndimension() == 3:
          img = img.unsqueeze(0)

      # Warmup
      if self.device.type != 'cpu' and (old_img_b != img.shape[0] or old_img_h != img.shape[2] or
 old_img_w != img.shape[3]):
          old_img_b = img.shape[0]
          old_img_h = img.shape[2]
          old_img_w = img.shape[3]
          for i in range(3):
              print("Imf shp",img.shape)
              self.model(img, augment=False)[0]

      # Inference
      t1 = time_synchronized()
      with torch.no_grad():   # Calculating gradients would cause a GPU memory leak
          pred = self.model(img, augment=False)[0]
      t2 = time_synchronized()

      # Apply NMS
      pred = non_max_suppression(pred, self.conf_thres, 0.45, classes=None, agnostic=False)
      t3 = time_synchronized()


      s=''
      # Process detections
      for i, det in enumerate(pred):  # detections per image
          # p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame', 0)
          im0=frame
          gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
          if len(det):
              # Rescale boxes from img_size to im0 size
              det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()


              # Print results
              for c in det[:, -1].unique():
                  n = (det[:, -1] == c).sum()  # detections per class
                  s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string
```



```
        # Write results
        for *xyxy, conf, cls in reversed(det):
            label = f'{names[int(cls)]} {conf:.2f}'
            classess.append(label)
            bbox_data.append(xyxy)
            save_img=True
            if save_img or view_img:  # Add bbox to image
                label = f'{names[int(cls)]} {conf:.2f}'
                plot_one_box(xyxy, im0, label=label, color=colors[int(cls)], line_thickness=1)


    # Print time (inference + NMS)
    print(f'{s}Done. ({(1E3 * (t2 - t1)):.1f}ms) Inference, ({(1E3 * (t3 - t2)):.1f}ms) NMS')

# Stream results

cv2.imwrite("results/"+str(name_g)+".jpg", im0)
print(f'Done. ({time.time() - t0:.3f}s)')
return classess,bbox_data
```

A.4 GUI Program with audio output

This code displays the GUI on LCD Screen in Full screen format. Further, it outputs the audio when user clicks on the button on the screen

```
import tkinter as tk
from tkinter import *
import tkinter.font as TkFont
from PIL import ImageTk, Image
import pygame

# Initialize Pygame mixer
pygame.mixer.init()

# Create the main window
def exit_fullscreen(event=None):
    root.attributes("-fullscreen", False)
  # Clean up Pygame mixer
    pygame.mixer.quit()
    root.destroy()

root = tk.Tk()
root.geometry("1024x600+0+0")
root.attributes("-fullscreen", True)
root.title("Shopping Assistant Robot")
root.bind("<Escape>", exit_fullscreen)
# Get the size of the screen
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

background_image = PhotoImage(file="background.png")
background_image = ImageTk.PhotoImage(Image.open("background.png"))
# Create a label to display the background image

background_label = tk.Label(root, image=background_image)
background_label.place(x=0, y=0, relwidth=1, relheight=1)


custom_font = TkFont.Font(family="Roboto", size=18, weight="bold")
# Create a label to display a welcome message
welcome_label = tk.Label(root, text="Hi Ravi here!\n Looking for something special today? Let me
know if you need any assistance.", font=custom_font,
                    bg="#1E88E2",
                    fg="white",
                    padx=35,
                    pady=15,
                    wraplength=700,
                    justify=tk.CENTER)
welcome_label.pack(pady=20)
welcome_label.place(x=150, y=55)


def audio():
  # Load the audio file
  audio_file = "welcome.wav"
  pygame.mixer.music.load(audio_file)

  # Play the audio file
  pygame.mixer.music.play()

  # Wait for the audio to finish playing
  while pygame.mixer.music.get_busy():
   continue


# Book Section
image1 = tk.PhotoImage(file="1.png")
book_button = tk.Button(root,image=image1,command=audio)
book_button.pack()
book_button.place(x=200, y=400)

# Cloth Section
image2 = tk.PhotoImage(file="2.png")
```

```
cloth_button = tk.Button(root,image=image2,command=audio)
cloth_button.pack()
cloth_button.place(x=200, y=220)

# Electronic Section
image3 = tk.PhotoImage(file="3.png")
electronic_button = tk.Button(root,image=image3,command=audio)
electronic_button.pack()
electronic_button.place(x=670, y=220)

# Plastic Section
image4 = tk.PhotoImage(file="4.png")
Plastic_button = tk.Button(root,image=image4,command=audio)
Plastic_button.pack()
Plastic_button.place(x=670, y=400)

# Exit
exit_button = tk.Button(root, text="Exit", font=("Helvetica", 14),bg="red", command=exit_fullscreen)
exit_button.pack()
exit_button.place(x=950, y=550)

# Welcome robot
image6 = tk.PhotoImage(file="Welcome.png")
label = tk.Label(root, image=image6)
label.pack()
label.place(x=19, y=50)


# Run the GUI
root.mainloop()
```

A.5 Sending and receiving frames and command for Motor control

This code is used to send and receive data from server to control motor movement and speed

```
import socket
import RPi.GPIO as GPIO             # ------> Import GPIO module so that we can controll pins
import time



GPIO.setwarnings(False)             #-------> To disable warnings
GPIO.setmode(GPIO.BOARD)                #-------> GPIO pins scheme (here its set to physical board)

GPIO.setup(12, GPIO.OUT)
GPIO.setup(29, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(31, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(33, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(35, GPIO.OUT, initial=GPIO.LOW)

# Set GPIO pins as input for sensor inputs
GPIO.setup(36, GPIO.IN)
GPIO.setup(38, GPIO.IN)
#GPIO.setup(40, GPIO.IN)
GPIO.setup(11, GPIO.IN)
GPIO.setup(13, GPIO.IN)
#GPIO.setup(15, GPIO.IN)

# Set GPIO for ultrasonic
GPIO_TRIGGER = 16
GPIO_ECHO = 18
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = '' # Server IP address
port = 12345 # Server port

server_socket.bind((host, port))
server_socket.listen(1)

client_socket, address = server_socket.accept()
print("Accepted connection from", address)

pwm = GPIO.PWM(12, 100)

forward_speed=40
turn_speed=50

def left():
    print (" LEFT")
    pwm.start(turn_speed)                    # setting speed
    GPIO.output(29, GPIO.HIGH)             #-----> turning left
    GPIO.output(31, GPIO.LOW)
    GPIO.output(33, GPIO.LOW)
    GPIO.output(35, GPIO.HIGH)

def right():
    print ("right")
    pwm.start(turn_speed)                    # setting speed
    GPIO.output(29, GPIO.LOW)
    GPIO.output(31, GPIO.HIGH)             #-----> turning right
    GPIO.output(33, GPIO.HIGH)
    GPIO.output(35, GPIO.LOW)

 def forward():
    print ("FORWARD")
    pwm.start(forward_speed)                    # setting speed
    GPIO.output(29, GPIO.HIGH)
    GPIO.output(31, GPIO.LOW)             #-----> move forward
    GPIO.output(33, GPIO.HIGH)
    GPIO.output(35, GPIO.LOW)

 def rotate():
    print ("ROTATE")
    pwm.start(turn_speed)                     # setting speed
    GPIO.output(29, GPIO.LOW)
    GPIO.output(31, GPIO.HIGH)            #-----> rotating 180
    GPIO.output(33, GPIO.HIGH)
    GPIO.output(35, GPIO.LOW)

 def stop():
    print ("STOP")
    GPIO.output(29, GPIO.LOW)
    GPIO.output(31, GPIO.LOW)             #-----> stop
    GPIO.output(33, GPIO.LOW)
    GPIO.output(35, GPIO.LOW)

 def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)
    # set Trigger after 0.01ms to LOW
    time.sleep(0.00009)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()
    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
       StartTime = time.time()
    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
StopTime = time.time()
    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
    return distance

# constant loop
```

```
while True:                           #------> run the below functions in loop
   data= server_socket.recv(1024)          #-----> declaring variable "data" as the data received from the
client
   data = data.decode('UTF-8')            #-----> the data recieved will be in the form of byes
                                                #        so we will convert it into strings
   print ("Received: ", data)
   dist = distance()
   print ("Measured Distance = %.1f cm" % dist)

   s1 = GPIO.input(36)
   s2 = GPIO.input(38)
   s3 = GPIO.input(40)
   s4 = GPIO.input(11)
   s5 = GPIO.input(13)
   s6 = GPIO.input(15)

   if (data == "F" and s1 == 0 and s2 == 0 and s4 == 0 and s5 == 0 and dist>=30
      forward()                         # function forward

   elif (s1 == 1 or s2 == 1 and dist>=30):
      left()                            # function left

   elif (s4 == 1 or s5 == 1 and dist>=30):
      right()                           # function right

   elif (data == "R" and dist>=30) :
      rotate()                          #  function rotate

   elif (data == "s" or dist<30):
      stop()                            #  function stop
```

Appendix-B: Project Codes

B.1 List of Hardware Used:

Table 3: Hardware used

Sr.	**Product**
1	Raspberry Pi 4B
2	8MP Pi cam V2
3	Speaker and mic with an amplifier
4	SD Card
5	7-inch TFT Touch Screen LCD
6	Geared motor and wheel
7	3.7V 6000mAh Li-ion Cells
8	Acrylic sheet A4 (5mm, 3mm)
9	L298N Motor driver
10	3S BMS
11	IR Sensor
12	Ultrasonic Sensor
13	3D Printing

Bibliography

[1] "MARKETSANDMARKETS," April 2022. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/humanoid-robot-market-99567653.html.

[2] “Study: People are “more honest” when chatting to a robot” July 2014

[Online]. Available https://siliconangle.com/2014/07/24/study-people-are-more-honest-when-chatting-to-a-robot/

[3] S. Y. M. K. D. a. C. M. T.-G. Rakesh Chandra Joshi, "Efficient Multi-Object Detection and Smart Navigation Using Artificial Intelligence for Visually Impaired People," *Entropy,* vol. 22, no. 9, 27 August 2020.

[4] D. S. K. K. T. A. S. a. X.-Z. G. Kiran Jot Singh, "Computer-Vision Based Object Detection and Recognition for Service Robot in Indoor Environment," 12 October 2021.

[5] D. W. M. A. D. S. L. C. a. D. F. T. G. Douglas Henke Dos Reis, "Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm," November 2019.