IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 December 2023, accepted 8 January 2024, date of publication 17 January 2024, date of current version 13 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3355278

RESEARCH ARTICLE

Design and Development of FOODIEBOT Robot: From Simulation to Design

ATA JAHANGIR MOSHAYEDI[1], (Member, IEEE), ATANU SHUVAM ROY[2], LIEFA LIAO[1], AMIR SOHAIL KHAN[1], AMIN KOLAHDOOZ[3,4], AND ALI EFTEKHARI[5], (Member, IEEE)

[1]School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, Jiangxi 341000, China
[2]Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, Uttar Pradesh 208016, India
[3]Faculty of Technology, School of Engineering and Sustainable Development, De Montfort University, LE1 9BH Leicester, U.K.
[4]Faculty of Engineering, Universitas Negeri Padang, Padang, Sumatera Barat 25131, Indonesia
[5]Department of Mechanical Engineering, Khomeinishahr Branch, Islamic Azad University, Khomeinishahr 84181-48499, Iran

Corresponding author: Amin Kolahdooz (amin.kolahdooz@dmu.ac.uk)

This work was supported by the Institute of Engineering and DORA at De Montfort University, Leicester, U.K., through DMU "top-up" OA Fund.

ABSTRACT This investigation is centrally focused on the comprehensive evolution and enhancement of FOODIEBOT(shortened name for food delivery robot, an adaptive service automaton with a wide range of functionalities. Its capabilities encompass sophisticated image processing methods, seamlessly integrated via mobile applications (APP) and web interfaces, tailored specifically for intricate object manipulation in dining hall settings. During its developmental phase, the precise calibration of PID controller coefficients emerged as an essential requirement. The model underwent meticulous scrutiny through detailed simulations using MATLAB software. Following this phase, its operational efficiency navigating through circular, elliptical, spiral, and octagonal trajectories underwent rigorous examination, utilizing optimization methodologies like Beetle Antennae Search (BAS) Algorithm, Particle Swarm Optimization (PSO), Pelican Optimization Algorithm (POA), and Equilibrium Optimizer (EO). The exposition emphasizes the diverse dispersion of optimized coefficients within each algorithmic framework. The pinnacle of this effort involved a comprehensive evaluation of pathway performance, amalgamating insights from each optimization paradigm. The discussion extensively delineates both simulated and real-time performance metrics of the robot, validating the accuracy and reliability of simulation in deriving PID controller values. In the comprehensive evaluation of methodologies and the robotic system's effectiveness, the BAS technique excels in operational efficiency. This method consistently outperforms its counterparts in execution time, primarily due to its meticulous optimization of particle count. The comparative analysis across various trajectories reveals intriguing insights. The EO approach showcases outstanding accuracy in Path 1, while the POA method achieves optimal precision in Path 3. Impressively, the BAS technique demonstrates unparalleled accuracy in Path 4. Furthermore, in terms of solution optimization, the BAS method consistently displays the shortest execution times across all traversed pathways. When examining maximum velocity along these routes, the PSO method excels in Paths 1, 3, and 4, consistently achieving the highest speeds. Notably, Path 2 uniquely displays the peak velocity attained by the POA method. This article presents comprehensive insights into the constituent elements of the robotic system's design. The inquiry delves into the intricate nuances of optimization methodologies, elucidating their profound impact on the service automaton's performance across diverse orientations. The pragmatic implications underscore the critical role of temporal considerations in the judicious selection of these methodologies. The observed congruence between simulated and practical performance serves as a definitive validation, affirming the precision of simulation computations and the subsequent derivation of PID controller values.

INDEX TERMS FOODIEBOT, food delivery robot, service robot, particle swarm optimization (PSO), beetle antennae search algorithm (BAS), pelican optimization algorithm (POA), equilibrium optimizer (EO).

The associate editor coordinating the review of this manuscript and approving it for publication was Geng-Ming Jiang.

I. INTRODUCTION

In recent years, robots were predominantly confined to industry but have since diversified across sectors, from

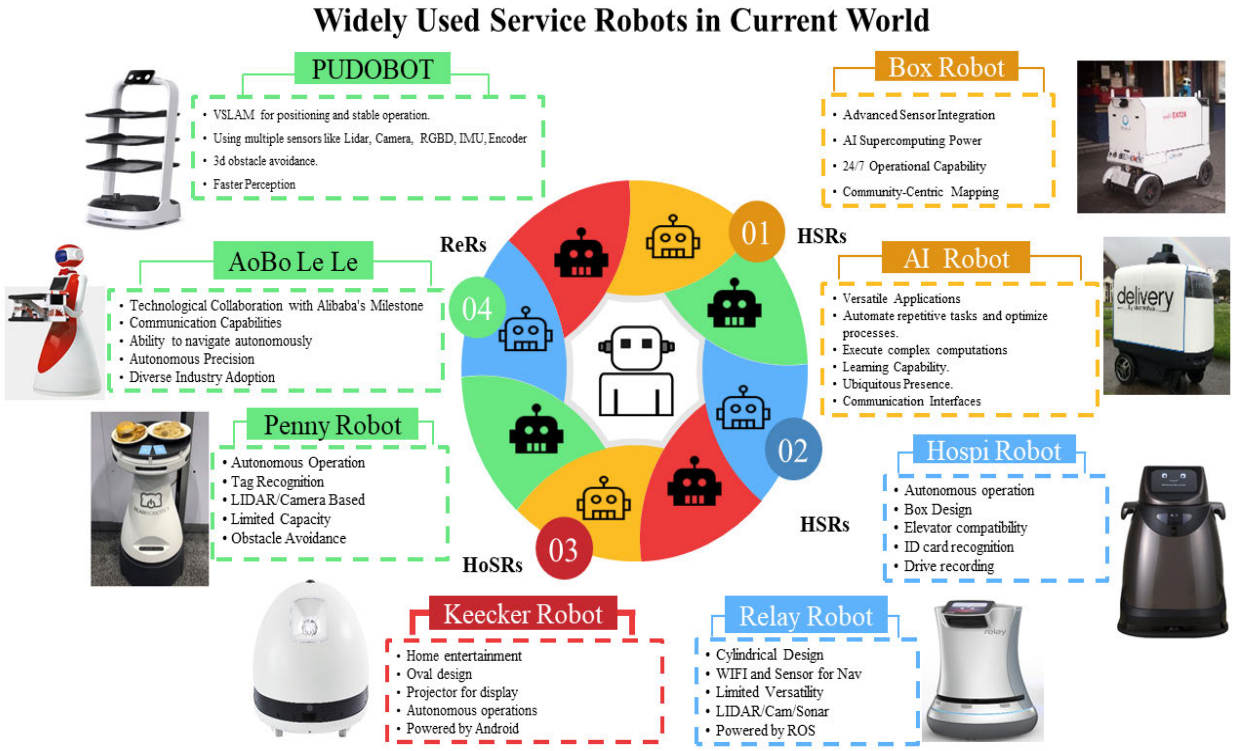


FIGURE 1. The current most utilized service robot spans across the primary categories of outdoor service robots: Outdoor Service Robots (OSRs),Hotel Service Robots (HSRs),Home Service Robots (HoSRs),Restaurant Robots (ReRs).

household chores to space exploration [1]. The term ''Service Robot'' has gained prominence in the world of robotics in recent times. A service robot is a robot designed to perform useful tasks for humans in various settings, from healthcare to logistics. These robots assist with tasks such as cleaning, transportation, and more, often improving efficiency and convenience in their respective industries [2]. Service robots are primarily deployed to alleviate human workloads, especially in industries facing challenges like adverse environmental conditions, labor shortages, and rising operational costs. Mohan et al. [3] focus on integrating robots into healthcare tasks like documentation and patient support, especially in scenarios with limited human presence. Using a patient-centric interface, the study assesses economic benefits by gathering insights from patients, caregivers, and medical professionals in major cities. It unveils a 9.1% rise in robotics adoption since 2022, supported by a robust model fit exceeding 80%, showcasing high acceptance levels. The overview showcases diverse robot classifications in hospitals, spanning from basic cleaning to advanced surgical systems, highlighting the widening applications in healthcare. However, the COVID-19 pandemic has ushered in fresh opportunities for service robots, particularly in areas once considered unthinkable. Notably, these robots have played a vital role in minimizing virus transmission by taking on tasks that necessitate direct human contact as hospital admissions and medicine delivery, significantly enhancing safety in subsequent interactions [4]. Adeleye [5] delved into service robotics research, emphasizing their role in daily tasks, highlighting limitations, and proposing solutions for seamless integration into home environments. The study showcased advancements in organizing groceries, rearranging household items, and exploring human-robot interactions, emphasizing the need for improved manipulation and adaptability with novel objects in real-world scenarios.

A. SERVICE ROBOT TYPES ASPECT AND ARENA

The investigation into service robot diversity can be categorized into two major groups: current products and ongoing research.

1) THE CURRENT PRODUCT

With the researcher's efforts, several products are currently being developed to create the service robot (Figure 1). Figure 1 illustrates the service robot divided primarily into four groups:

- **Outdoor Service Robots (OSRs):** OSRs are employed for efficient post and food delivery, navigating through urban landscapes to ensure timely and secure deliveries, and enhancing logistics in city environments.
- **Hotel Service Robots (HSRs):** HSR premises, service robots streamline operations by delivering assorted items, aiding guests with requests within the hotel premises, enhancing guest experience, and optimizing staff productivity.

- **Home Service Robots (HoSRs):** HoSRs are designed to assist with household tasks, offering support in chores, security monitoring, and personalized assistance to occupants, augmenting convenience and comfort within residential settings [6].
- **Restaurant Robots (ReRs):** ReRs are functioning as waiters, automate food service by efficiently navigating dining spaces, taking orders, and delivering meals, contributing to faster service and enhancing customer experiences in dining establishments.

These service robots' types within the extant framework and architectural configuration, a meticulous examination is conducted to scrutinize the diverse array of products elucidated in Table 1 and delineated graphically in Figure 1.

Figure 1 and Table 1 highlight the burgeoning popularity of service robots, notably the category of waiter service bots deployed across hotel, home, and restaurant settings. Tailored for specific environments, these robots have gained substantial traction in recent times. Table 1 presents a comparative analysis of service robot products, outlining their distinctions across four key reference criteria, particularly emphasizing their mission orientation as waiter substitutes. This diversity among these robots is pronounced. While certain models are tailored for specific industries like hospitality, others showcase adaptability across various domains, ranging from household to public spaces. Besides core design aspects such as shape, speed, and load capacity, the critical choice of position monitoring technologies stands as a pivotal factor. Robots integrate diverse monitoring methods, finely tuned to their respective environments and the precise tasks they are designed to fulfill. This diversity in robot design and functionality allows for more customized solutions to meet various service needs. Different designs showcase the diverse technologies and approaches used to create waiter robots with specific functionalities [7], from enhanced navigation and interactive capabilities to additional features like waste management and disinfection. The surge in demand for waiter service robots is driven by their contactless serving capabilities, efficiency enhancements, and the imperative to address labor shortages in the hospitality sector [8]. Waiter robots excel at tasks like delivering food and beverages to tables, taking orders, and navigating busy restaurant environments, offering contactless and efficient service [9]. These robots not only ensure a unique and memorable customer experience but also deliver cost-saving advantages. Their adaptability and customization options make them a sought-after choice in a range of service-oriented businesses [10].

2) ONGOING RESEARCH

On the other side, the review of previous research papers highlights diverse designs and technologies that enhance the robot's ability to fulfill its tasks and responsibilities effectively. Service robot design continually investigates human-robot interaction, adaptability, and safety measures. Research focuses on autonomy, materials, and long-term reliability for enhanced functionality. Ongoing advancements seek seamless user experiences and efficient robot performance in diverse settings. diverse settings.

1) **Human-Robot Interaction Challenges:** Creating intuitive interfaces and communication methods for effective interaction between humans and robots remains a significant hurdle.
2) **Autonomy and Navigation in Complex Environments:** Navigating diverse and dynamic environments without human intervention poses challenges for robots, especially in crowded or unstructured settings.
3) **Reliability and Maintenance:** Ensuring consistent and reliable performance over time, along with minimizing maintenance needs, is crucial, especially in commercial or industrial applications.
4) **Safety Concerns:** Developing robust safety protocols to prevent accidents and ensure the safety of both the robot and its surroundings is a prty.
5) **Adaptability to Varied Tasks:** Designing robots capable of adapting to a wide range of tasks and environments without compromising efficiency is an ongoing challenge.

In a study by Qing-Xiao et al. [11], Optical Character Recognition (OCR) technology and an Radio Frequency Identification (RFID) based positioning algorithm were employed to achieve real-time autonomous positioning of robots. Experimental simulations, mimicking a restaurant's layout, demonstrated impressive positioning accuracy, with the robot's positioning typically deviating by only about 3 cm from the target position. Lin et al. [12] focused on creating a humanoid robot capable of enhancing the dining experience by engaging customers in the finger-guessing game. Their design incorporated a high-speed camera to record the player's game movements, and a low Degree of Freedom (DOF) facial expression generation device for responsive interactions. Through gesture recognition experiments, the robot achieved a recognition rate of 70%. The study conducted by Cheong et al. [13], introduced a waiter robot featuring mechanical wheels, designed to function across various restaurants. This robot was equipped with the ROS (Robot Operating System) framework and employed a modular hardware design. Within the ROS framework, the restaurant environment map was generated through the SLAM (Simultaneous Localization and Mapping) algorithm and autonomous positioning and path planning were accomplished using Adaptive Monte-Carlo Localization (AMCL). In the research by Jahromi et al. [14], a waiter robot was designed with the ability to sense the number of customers using infrared sensors and navigate smoothly to their locations through ultrasonic sensors. Customers could place their food orders through the robot's LCD screen. Additionally, an automated table was devised to intelligently manage waste collection and disinfection to mitigate the spread of COVID-19. In another study Freeman et al. [15] investigates the potential of a collaborative robot system to perform essential healthcare tasks within a simulated intensive care

TABLE 1. The acclaimed service robot products excel in terms of feasibility and practicality.

A) Outdoor Robot	
Box Robot	**Task:** Autonomous delivery in urban environments. **Structure:** Body Chassis, Sensor Array, Nvidia Jetson TX1 Supercomputer, Navigation System, Cargo Compartment, Power System. **Position Monitoring:** Combination of sensors and mapping technologies **Advantages:** Box robots excel in navigating urban environments autonomously and AI processing to optimize delivery routes. **Limitations:** A potential limitation lies in the dependence on detailed three-dimensional maps for navigation.
AI Robot	**Task:** Automate and perform tasks with a level of intelligence and adaptability. **Structure:** Sensors, Processing Units, Actuators and Motors, Power Source, Memory Storage, Algorithms and software's. **Position Monitoring:** Continuous tracking and updating of the robot's location in its environment. **Advantages:** AI robots significantly enhance efficiency by automating repetitive tasks, and operating tirelessly 24/7. **Limitations:** One notable disadvantage is that AI robots take over routine and manual tasks, there is a risk of job loss for individuals in those roles.
B) Hotel Service Robot	
HOSPI	**Task:** Delivering amenities to hotel guests. **Structure:** Compact boxy design, storage capacity. **Position Monitoring:** Various sensors and localization technology. **Advantages:** Reliable service, frees up staff time. **Limitations:** Limited versatility, tailored to hotels.
Relay	**Task:** Delivering hotel room service. **Structure:** Compact cylindrical design, display screen. **Position Monitoring:** Wi-Fi and sensors for navigation. **Advantages:** Streamlines hotel services, reliable. **Limitations:** Limited versatility, designed for specific industry.
C) Home Service Robot	
KEECKER	**Task:** Home delivery of drinks, snacks, and entertainment **Structure:** Compact and stylish **Position Monitoring:** Computer vision, depth sensing, mapping **Advantages:** Multi-purpose, interactive, entertainment **Limitations:** May not be suitable for all home environments
D) Restaurant Robot	
Penny	**Task:** Food delivery in restaurants **Structure:** Compact and mobile design **Position Monitoring:** LiDAR and camera-based navigation **Advantages:** Contactless service, efficient, obstacle avoidance **Limitations:** Limited carrying capacity, may not handle complex tasks
AoBo LeLe Food delivery robot	**Task:** Mainly for delivery purposes in restaurants, hospitals, business buildings etc. **Structure:** Plastic, glass fiber, metal steel plate. 24V15AH polymer lithium battery. **Position Monitoring:** WiFi, 4G, Bluetooth Module, **Advantages:** Long lasting battery and laser navigation obstacle avoidance **Limitations:** Costly
PUDUBOT	**Task:** An intelligent Delivery robot **Structure:** 2D radar, 3D Stereo sensors, Lidar, Camera and multiple sensors **Position Monitoring:** Camera, smart sensors (Automatically detects obstacles) **Advantages:** Safety and obstacle avoidance, Great precision, Visual Positioning **Limitations:** Expensive
Smart Delivery Robot	**Task:** Autonomous delivery robots transport food parcels **Structure:** Chassis, Sensors(Lidar, Camera, ultrasonic sensors), processing unit, battery, communication system, navigation system **Position Monitoring:** GPS, Lidar, Encoders, Inertial Measurement Units, Visual Odometery for navigation and updating of location. **Advantages:** Increased efficiency in the delivery process **Limitations:** Current constraint on payload capacity.

unit (ICU) for COVID-19 patients. The researchers tested five discrete medical tasks, including interacting with intravenous pole machinery, adjusting a ventilator knob, and responding to false alerts on an ICU monitor. The results indicate that these tasks were successfully completed robotically after a training period of 45 minutes to 1 hour for each task. The study suggests that utilizing collaborative robots in healthcare settings, particularly for routine tasks, may reduce the need for personal protective equipment (PPE) and minimize healthcare workers' exposure to the SARS-CoV-2 virus. However, the findings call for further validation and healthcare worker training. Among the reported cases, the use of LiDAR and cameras for positioning is common, but it can be expensive, especially in stable internal environments. Magnetic tapes are a cost-effective alternative, particularly in smaller spaces [16]. Additionally, there's limited mention of a comprehensive evaluation system for precise customer diagnosis and efficient delivery of multiple orders along a single path, which is an area for potential improvement in waiter robot design.

Deploying a robot waiter poses multifaceted challenges, notably in areas like human interaction, order accuracy,

customization, and maintenance. Additionally, considerations such as the final design cost, safety protocols, monitoring capabilities, and environmental adaptability are pivotal factors that demand attention. Addressing these challenges necessitates ongoing research, innovative solutions, and the seamless integration of advanced technologies into restaurant operations. This article endeavors to explore these complexities by delving into the complete lifecycle of FOODIEBOT's design (shortened name for food delivery robot), construction stages, and modeling. This versatile robot holds potential across diverse applications, including catering services, hotel management, healthcare facilities, elderly care, and various service sectors [17].

The FOODIEBOT consists of a data resource sharing platform, distributed task ideas, and highly integrated modular expansion functions. In summary, the primary contributions and focal points of this work encompass:

1) **Regarding design:**
   - Development of an interactive robot waiter facilitating food orders via a mobile app, recognizing customers, and executing precise deliveries through coded instructions. This includes automated order handling via a robotic arm or manual customer control, enhancing operational efficiency and offering varied options.
   - Equipping the robot for seamless round-the-clock online service, ensuring swift and continuous assistance.
   - Creation of a localized website dedicated to assessing customer satisfaction levels based on food quality, yielding valuable statistical insights for ongoing improvements.
2) **In design and simulation:**
   - Proposal of an extended Automated Guide Vehicle (AGV) model, lauded for its effective and efficient performance, capable of accurately emulating real-world movement scenarios. Introduction of a simulation model mirroring the robot's design, practically implemented and rigorously compared with real-world robot behavior. This assessment involves evaluating the performance of four nature-inspired controllers, referred to as BAS, PSO, POA, and EO, to optimize path-following behaviors.
3) **Regarding Navigation Algorithms:**
   - Precisely calibrate controller coefficients to enhance the robot's operational precision, employing meticulous simulations using MATLAB software.
   - Scrutinize the robot's performance across diverse pathways - circle, ellipse, spiral, and octagon - leveraging optimization methodologies such as BAS, PSO, POA, and EO.
   - Highlight the dispersion of optimized coefficients within each algorithmic framework.
   - Integrate insights from diverse optimization paradigms to rigorously evaluate the robot's performance in combined pathways, both in simulation and real-world scenarios.
   - Validate the accuracy and efficacy of simulation models in deriving PID controller values.
   - Examine and articulate the profound impact of optimization methodologies on the multifaceted performance orientations of the service robot.

The article is organized into distinct sections: Part 1 scrutinizes the physical framework, peripheral elements encompassing the mobile application and the web-based analysis system for robot data, along with the mathematical model and simulations performed using MATLAB software. Part 3 elaborates on the optimization algorithms implemented within the robot's path controller. In Part 4, the outcomes and assessment of the robot's performance across various pathways are presented, culminating in an evaluation of combined path scenarios. Finally, the article concludes by comparing methodologies, scrutinizing the model, and outlining potential avenues for future research.

II. FOODIEBOT ROBOT DESIGN

The FOODIEBOT robot is a comprehensive system comprising both hardware (combination of various sensors) and software (Mobile APP, and the Web Application) components working in harmony to execute a diverse range of tasks. Among these tasks, a pivotal one involves navigating and following predefined paths, placing food orders, identifying individuals at the ordering counter, delivering food or packages to recognized recipients, etc., which are described as follows;

A. ROBOT HARDWARE SYSTEM

The robot's physical structure comprises four wheels (4WD), strategically positioned on both sides, with individual instructions from the processors to control the movement of the robot. At the top of the robot, its design features a robotic arm manipulator equipped with 4 DOF [18], along with Vision Camera1 (Vision 1) for user visual input, a Food Dispenser for handling food items, an Obstacle Avoidance System, Pixi Camera 2 (Vision 2) Ensuring user verification and accurate order delivery to the intended recipient as the additional imaging capabilities, a speaker for audio output, and an LCD for displaying information. The tasks for each part are described in Table 2 and Figure 2. As depicted in Table 2, the comprehensive system comprises six primary components: Body movement motor, User interaction, Processor, Sensors, Food Dispenser, and Power Key. The integration of these hardware components empowers the robot to execute delivery tasks. Given the higher sensor count and the cost implications in design, the orchestration among the three CPUs, managed by the main processor with a multitasking principle, efficiently coordinates the system (Figure 2).

TABLE 2. The FOODIEBOT robot section and hardware list.

NO	Section	Main Part	Description	Ref
1	Body Movement	Four Number, Motor Wheel (Zhongling New 4-inch micro robot hub 24v DC motor 1024 line encoder AGV wheel type)	Primary Robot Motor Motion System	[19]
		Driver(Zhong Ling servo driver ZLAC706-CAN robot hub motor driver Brand new)		
2	User interaction	Lcd (HDMI -7 inch)	Display the video captured by Vision 1 and also show the recorded Clip	[20]
		MP3 player module	Playback of the recorded audio	[21]
		Speaker (USB 2, 3W ,5v Stereo Speakers	Play the recorded audio and interface it with the MP3 module	
3	Processor	Main Processor : Raspberry Pi 3 B+ (1.4 ARM GHz CPU with 1GB RAM)	Connected to the primary LCD, capable of managing data received from all Processor 1 to 4. Additionally, it is linked to Vision 1 camera for facial detection and user interaction and Enable remote control	[22]
		Processor 1 : Arduino Mega 2560	Overseeing the operation of the robotic arm's driver and motor driver	[23]
		Processor 2 : Arduino Mega 2560	Interfacing with the Pixy camera and ultrasonic sensor	
		Processor 3 :Wmos	App connection	
		Processor 4 : Raspberry Pi 3 B+	Used for web page host	
4	Sensors	Magnetic Sensor(N-pole magnetic strip AGS-16N)	Detecting the path line and following the magnetic tape path	[24]
		Ultrasonic: HYSRF05	Avoiding obstacles.	
		Vision 1: USB intelligent Endoscope Camera with LED) model:8.0 MM, view angle 70, HD 640*480	Utilized for consumer face detection and transmission of vision data to the main CPU	
		Vision 2: Pixy cam Pixy 2.1 Smart Vision Sensor(NXP LPC4330, 204 MHz; Sensor: Omnivision OV9715)	Utilized for reading the color code on the customer table	[25]
5	Food Dispenser	Main Dispenser plastic designed Dispenser for box/plates	Used for handling four objects with a maximum size of 9 x 13 cm and a height of 10 cm each."	[26]
		Armand Gripper (Arm 3DOF Mechanical Robot Arm with Gripper for 3D Printing, with 17 Nema Servo Motor 3 Axis Full Metal Frame Robotic Manipulator with Stepping Motors and along driver A4988)	Used for picking up and transporting objects from the dispenser to the customer.	[27]
6	Power KEY	Six on/ off switch	To control the power status of various components, including the main power, sound, arm, etc	[28]
		Battery: Tattu Plus(6 Cells , 22.2 v, 22000)	System Power Supply | FOODIEBOT Runtime: 1 Hour - Continuous | 3 Hours - Standby	[29]

As depicted in Table 2, the comprehensive system comprises six primary components: Body movement motor, User interaction, Processor, Sensors, Food Dispenser, and Power Key. The integration of these hardware components empowers the robot to execute delivery tasks. Given the higher sensor count and the cost implications in design, the orchestration among the three CPUs, managed by the main processor with a multitasking principle, efficiently coordinates the system (Figure 2).

As Figure 2-B shows in this design three CPUs are used to ensure the quickest response time and enhance coordination among all robot components, the Multitasking Principle, [30], is employed. This approach enables the robot to boost processing speed by employing smaller processors, resulting in faster task execution, quicker maintenance, and troubleshooting procedures. and improved overall coordination. The choice of using specific processors, such as the Raspberry Pi, Arduino Mega, and WeMo's, is based on their compatibility with the sensors and actuators used in the system. These microcontrollers offer a rich set of input/output pins, allowing for seamless integration with various components. Additionally, their extensive community support and a vast library of pre-built code make them ideal choices for rapid prototyping and development. Utilizing two Arduino Mega 2560 boards as Processor 1 and 2, a Raspberry Pi 3B+ serves as the Main Processor. The WeMo's, functioning as the third CPU, collaborates with the Raspberry Pi 3B+ for web-related functions within the system These components are responsible for executing precise movements and controlling the physical actions of the robot. The main processor, which is the Raspberry Pi, and Camera1 (USB camera) vision system governs the audio/Speech section, and GUI, then the primary responsibilities of the FOODIEBOT are delegated to the other controller, while other controllers work collaboratively under the main controller's guidance. Processor 1 orchestrates movement commands to the right and left motors via PID control and magnetic sensors, ensuring obstacle avoidance using ultrasound sensors. It also manages the buzzer and warning functionalities within the system. Sound controlling and robotic arm controlling. Processor 2 operates the Pixy Cam (Vision 2) as a secondary vision system with the color recognition principle, facilitating customer order recognition from a mobile application (APP) and controlling the robotic arm. The Pixy camera stands as a remarkably user-friendly and adaptable vision sensor, empowering seamless and effective object detection and

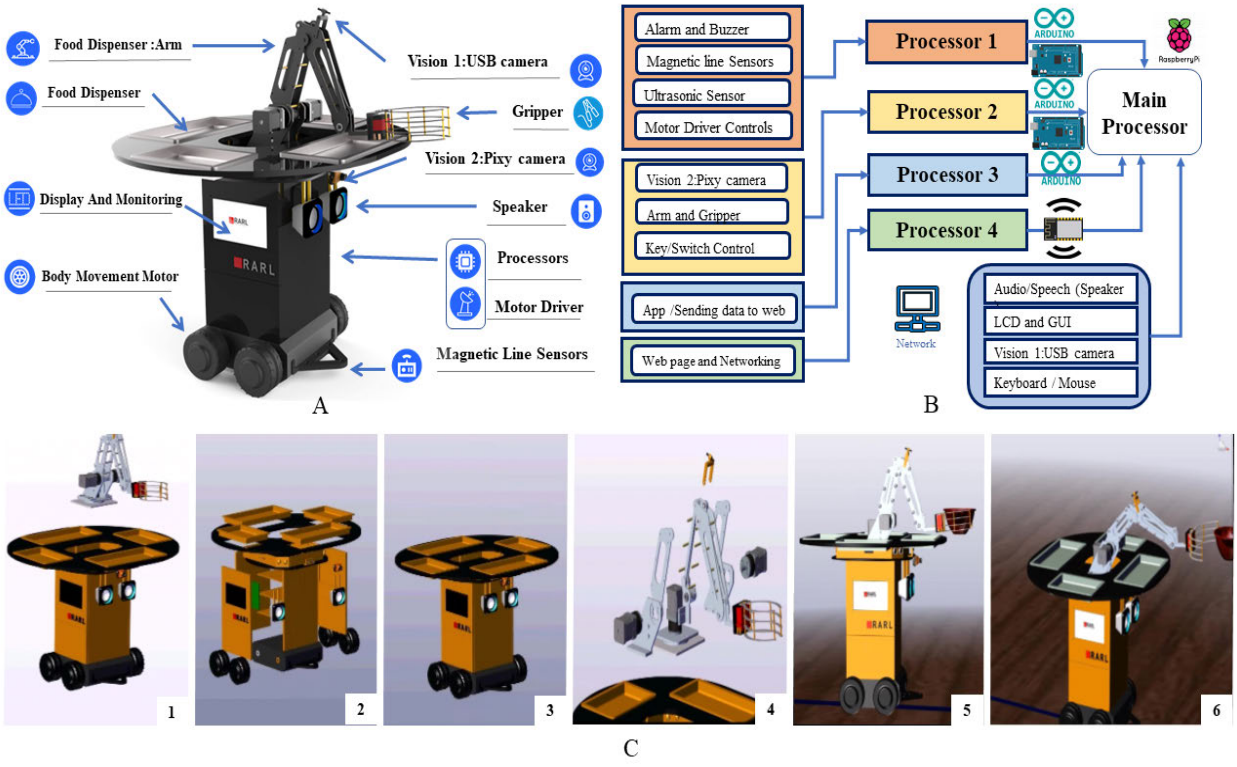


FIGURE 2. FOODIEBOT encompasses:A. A four-wheel drive skid-steered (4WD) robot featuring a manipulator platform designed for grasping and handling various objects.C. The robot's CPU with multitasking processors, each dedicated to specific parts and dutiesB. Different perspectives of FOODIEBOT: 1. Arm dislocated, 2. Body disassembled, 3. Body assembled, 4. Arm disassembled, 5. Robot holding food, 6. Robot arm moving forwards.

tracking. Processor 3 WeMo's [31], are used to connect the mobile apps order via Wi-Fi and send the data to the webpage part. The Processor 4 (Raspberry Pi B+) hosts the webpage and networking principle for food ordering management. The Raspberry Pi serves as the communication hub within the internal network, connecting the Android app to the robot, and facilitating connectivity with the database through the Raspberry Pi's Wi-Fi network.

B. ROBOT SOFTWARE SYSTEM

In the software component, the designed structure encompasses both a website and a mobile application.

1) THE MOBILE APPLICATION

The 'FOODIEBOT Robot' is a mobile application designed for ordering foods and controlling a robot arm in a manual setting. The app par Developed using MIT app-inventor, the app offers various functionalities and interfaces to enhance the user experience with the workflow shown in figure 3. The app page as it is shown in Figure 3 contains the six-part:

- **Page 1 (Welcome Page or Login Procedure):** Welcomes users to the app and directs them to enter the main app after selecting an entry section. Displays a login page prompting users to input their mobile number and table number. If the entry is correct, it displays a "Successful!!!" message; otherwise, it prompts users to try again (Figure 3).
- **Pages 2 to 5 (ORDERPAGE Procedure):** Display food information (numbered 1 to 6) with images, checkboxes for food selection, food names, and a user feedback section allowing evaluations (e.g., like/dislike) for each food item. Each page includes navigation options for Next Page and FrontPage. Upon selecting the options for food ordering, various categories are shown on the screen such as breakfast, dinner, lunch, drink, etc (figure 3).
- **User Selection Data Page:** Displays the user's selections across pages, possibly summarizing chosen food items. It offers a submission feature where, upon pressing "submit," the data is sent to the web page. An icon or indicator confirms data submission. After submitting the list of chosen foods, a color (blue, green, red, or gray) corresponding to four food locations is assigned to the user. This color, determined through Pixy camera recognition, will be used when the worker transports the food. The color code is randomly assigned to one of the four users for each robot (figure 3).
- **SETINPAGE Procedure:** Establishes a connection with a WEMOS board, creating a web server using the WEMOS and ESP8266 [32]. A dedicated control section allows users to manage food placement on their table and transmit data to a local webpage. This section also offers

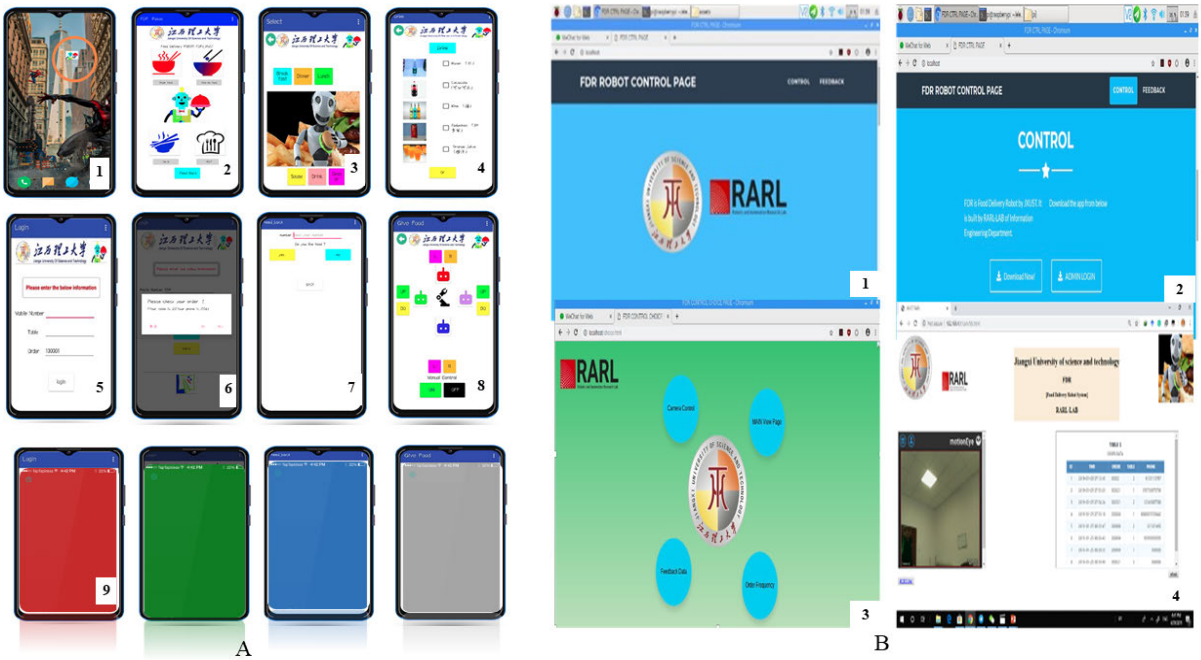


FIGURE 3. A: The mobile application, the web application.(1:app selection in user mobile phone, 2&3: welcome page or login procedure task selection),4: ORDERPAGE procedure,5: user selection data page, 6: DATAUPDATE Procedure,7: WEMOS and ESP8266,8: ROBOT CONTROL Procedure 9: the color code generated by app, B: web app landing page, 1: app download option, 2: Control Panel, 3: order management page 4: robot camera record order.

configuration settings that facilitate the establishment of a connection with the WEMOS board, the creation of a web server utilizing WEMOS and esp8266, and access to the server's IP address for various web pages (figure 3).

- **ROBOT CONTROL Procedure:** Establishes a connection to the WEMOS board and offers controls for robot operations. It interacts with a web interface to send commands and control the robot's movements (figure 3).
- DATAUPDATE Procedure: Uploads data to a server using the POST method, utilizing JSON format for data transmission. It structures the data into lists across various pages for management and updates (figure 3). These procedures represent the functionalities related to a App interface for managing user login, food orders, robotics control, and data management.

2) THE WEB APPLICATION

The design is equipped with a web application that greatly enhances system management and monitoring capabilities. The designed webpage offers a range of functionalities and pages to serve various purposes. It all begins with the index page, providing download links for the app and control page (Figure 3-B). The control page is the gateway for administrators upon successful login. Within the choice page (Figure 3-C), users can explore different options, including the camera control (for camera settings control), main view page, feedback data, and order frequently (for user feedback management), (Figure 3-C). the order management and the robot's camera through a service called Motion eye as the final page of the webpage part shown (Figure 3-D) along with the data export to CSV files option for analysis and storage. The combined functionalities within the system ensure effective management. They incorporate statistical analyses of food demand for future planning and facilitate real-time monitoring of FOODIEBOT. This setup offers administrators valuable insights and control over the system. To link the Mobile App with the web page, mobile devices interact with the database by receiving orders and providing feedback. The MySQL database is divided into two distinct sections: 'Data 1' for managing user orders and 'Data 2' for collecting feedback from customers regarding their food experience. Furthermore, an Arduino-controlled USB camera with a rotating motor is connected to the Raspberry Pi, enabling the capture and uploading of images from the network interface to the FOODIEBOT server. This module connects to the database and sends the food order to the backend for it to be prepared so that the robot can deliver it to the proper table.

III. FOODIT ROBOT 4WD MODEL AND SIMULATION

To facilitate Robot simulation, three primary sections are identified for the simulation task: the physical system, navigation sensor, and control logic (Figure 4).

A. PHYSICAL SYSTEM

The physical system encompasses the DC motor model, incorporating both inverse kinematics and a dynamic model. These elements are elaborated as follows:

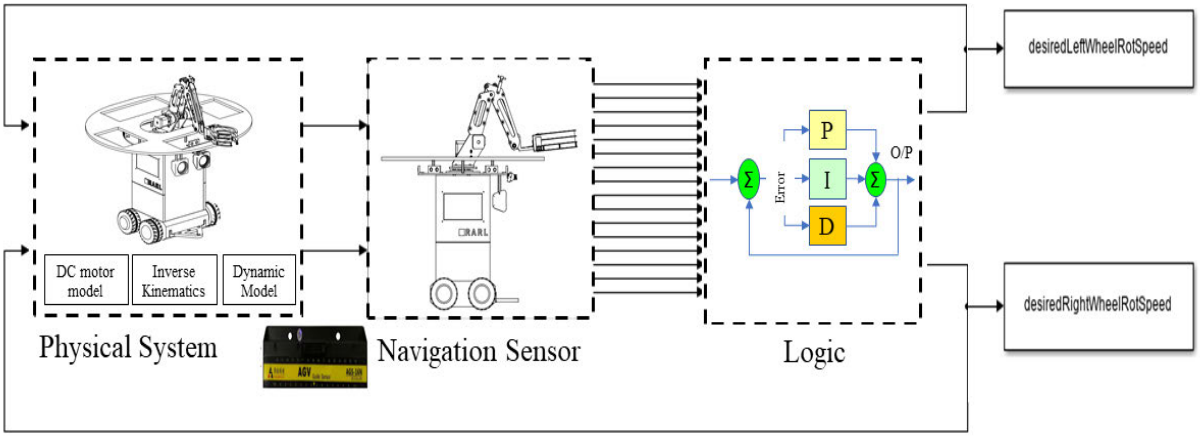


FIGURE 4. The FOODIEBOT robot simulation block parts.

1) DC MOTOR MODELING

The physical system incorporates four DC motor model [1000] four-inch brushless electric wheel hub motors (24 V DC, 280 rpm), with specifications sourced from the manufacturer's laboratory. The mathematical model for a simple DC motor can be expressed using electrical and mechanical equations. Let's consider a basic DC motor with a separately excited DC motor configuration summarized in Table 3.

The equation displayed in Table 3 stands as the mathematical representation of the DC motor model. This comprehensive model harmonizes the intricate interactions of electrical and mechanical components within a DC motor, unveiling how variations in input voltage can impact the resultant angular velocity, and illustrating the system's dynamic behavior. The electrical equation (Eq. 3) in a DC motor encapsulates the impact of resistance (R), inductance (L), and the back electromotive force (E) on the applied voltage (V), which influences the current (I) passing through the system. This equation Eq. (3) reflects the interplay of electrical elements in influencing the current in the motor, considering the resistance to the flow of current, the inductive behavior due to coils and windings, and the generated back EMF resulting from the motor's rotation. The mechanical equation (Eq. 3) in the context of a DC motor describes the balance between mechanical forces and parameters governing the motor's motion and torque generation. It encapsulates the relationship between the torque (T) produced by the motor and the factors influencing it, notably the current (I) and the angular velocity (ω) of the motor's rotation. This equation delineates how the generated torque is influenced by the current-driven torque (determined by the torque constant K_t) and the opposing force due to friction (F) acting against the motor's motion, as related to its angular velocity (ω). The Equation Eq. 3 highlights the direct correlation between the back EMF (E) and the angular velocity (ω) of the motor. As the motor spins faster, the back EMF increases proportionally according to the back EMF constant. The Back EMF Equation is crucial in understanding how the motor generates a counter-voltage that opposes the applied voltage, affecting current flow and motor performance. The Torque Equation (Eq. 3) depicts the relationship between the generated torque (T) and the current (I) passing through the motor. This equation serves as a fundamental representation of the motor's ability to produce torque based on the electrical current applied to it. The equation 3 illustrates that the torque produced by the motor is directly proportional to the current passing through it, as determined by the torque constant K_t. It signifies that as the current increases, the generated torque also increases linearly according to the torque constant. Understanding this equation is crucial in predicting and controlling the torque output of a DC motor based on the applied electrical current. Equation (Eq. 3) as Transfer Function represents the relationship between the input voltage (V_{in}) applied to the motor and the resulting angular velocity (Ω) of the motor's rotation. It encapsulates the system's dynamic behavior, linking the input voltage and the motor's response in terms of angular velocity through a mathematical function. It describes how changes in the input voltage influence the motor's speed and how the motor's electrical and mechanical properties interact to determine the motor's response in terms of angular velocity.

2) KINEMATICS AND DYNAMIC MODEL OF THE 4WD AND SIMULATION

The robot is configured with four wheels, where two wheels on the right-side function as a single unit, and the two wheels on the left side act as another unit. This setup defines the robot as a "differential drive robot with four wheels," effectively partitioning it into two distinct differential drive systems: one for the right side and one for the left side. In this design, the fundamental principles of differential drive apply independently to each side, much like they do in a conventional two-wheel differential drive robot. This means that each side can autonomously regulate the speeds and directions of its wheels, enabling differential motion. To simulate and control this robot, an extended simulation model based on the previously introduced two-wheel differential drive model is

TABLE 3. Symbols and variables for motor equations. V is the applied voltage, I_a is the armature current, R_a is the armature resistance, E is the back electromotive force (EMF), K_b is the motor constant, K_t is the torque constant, T is the torque developed by the motor, J is the moment of inertia of the rotor, ω is the angular velocity of the motor shaft, B is the damping coefficient, $\Omega(s)$ is the Laplace transform of angular velocity, Vin is the Laplace transform of input voltage, K is the motor constant relating applied voltage to angular velocity, J is the moment of inertia representing resistance to changes in angular velocity, B is the damping coefficient accounting for viscous friction, and s is the laplace variable.

Equation Name	Equation	Description
Electrical Equation	$V = I_a \cdot R_a + E$ (1)	Relates the applied voltage to the armature current and back electromotive force (EMF).
Mechanical Equation	$T = K_t \cdot I_a$ (2)	Relates the torque developed by the motor to the armature current using the torque constant.
Back EMF Equation	$E = K_b \cdot \omega$ (3)	Describes the back electromotive force (EMF) generated in the armature in terms of angular velocity.
Torque Equation	$T(s) = J \cdot s \cdot \Omega(s) + B \cdot \Omega(s)$ (4)	Relates the torque to the moment of inertia, angular acceleration, and damping in the system.
Transfer function	$\frac{V_m(s)}{\Omega(s)} = \frac{K_c}{\delta} \cdot \frac{1}{(Ls+R)(Js+b)+K_t \cdot K_c}$ (5)	Relates the Laplace transform of angular velocity to the Laplace transform of input voltage, where K is the motor constant, J is the moment of inertia, B is the damping coefficient, and s is the Laplace variable.

employed. This extended model is instrumental in replicating the robot's behavior and dynamics, facilitating the adjustment and fine-tuning of control parameters, such as those utilized in PID (Proportional-Integral-Derivative) controllers. These controllers help manage and optimize the robot's movements, making it a versatile and controllable platform for various applications. the kinematic and Dynamic models can be breifly shown in Table 4.

The kinematic model provides a simplified representation of the robot's motion in terms of its linear and angular velocities. It's useful for path planning and control but doesn't consider the robot's dynamics. The dynamic model is a more complex representation that takes into account the robot's physical properties, such as mass, inertia, and wheel friction. It relates wheel torques to linear and angular accelerations, making it suitable for more accurate simulations and control of the robot. These models are fundamental for understanding and controlling the motion of differential drive robots. The choice between the kinematic and dynamic models depends on the level of accuracy required for a given application. The dynamic model is typically used for tasks that involve precise control and consideration of physical constraints, while the kinematic model is often used for simpler tasks like path planning and navigation.

3) NAVIGATION AND LOGIC

In the upcoming segment of the Foodit modeling, the sensor component and logic section are tasked with executing the reference line detection function. In this robot design, an array of magnetic sensors is strategically positioned at the front of the vehicle, functioning in a manner akin to infrared sensors, as previously detailed in our publication [34]. These sensors are oriented towards magnetic tape, which serves as the demarcation line. When a sensor approaches a magnetic strip, it registers peak values, aiding in the detection process. Conversely, as the sensor moves away from the magnetic strip, minimal reflection occurs, rendering the sensor inactive. The simulation program emulates the positions of 16 sensors through computational determination, placing each magnetic sensor and comparing their locations with those of the reference track. The current robot position ($x_{\text{pos}}, y_{\text{pos}}$) and the robot position from the previous simulation frame ($x_{\text{last\_pos}}, y_{\text{last\_pos}}$) serve as inputs to the Matlab function block. The outputs of the Matlab function block represent the states of each magnetic sensor (Figure 6).

As the Logic segment encompasses the computation of errors and the implementation of the PID controller (Figure 6), the motor's direction hinges on sensor readings and PID parameters within this module. The error computation segment undertakes the task of gathering readings from 16 sensors. Subsequently, based on the respective error sensor values, these are extracted and fed into the PID segment. Operating as a classical controller, the PID controller relies on error feedback and three parameters of incremental significance—K_p (proportional), K_i (integral), and K_d (derivative).

The proportional, integral, and derivative components are leveraged to generate corresponding PWM signals, which, in turn, dictate the robot's speed. As delineated in the primary controller section, the PID controller is adept at harmonizing the speed differential between both sides of the robot, enabling it to traverse the designated path. The input value (Figure 6) to this PID controller is the error representing the deviation between the robot's heading and the target direction of the path. This error is computed based on the readings from the magnetic sensors.

TABLE 4. The two-wheel robot differentiation model involves several parameters and equations. Let V represent the linear velocity of the robot, ω denote the angular velocity of the robot, ω_l represent the angular velocity of the left wheel, and ω_r represent the angular velocity of the right wheel. The parameter r stands for the radius of the robot's wheels, and w represents the distance between the wheels (wheelbase). Torques on the left and right wheels are denoted by T_l and T_r respectively. The moment of inertia of the wheels is denoted by I_w, while α_l and α_r represent the angular accelerations of the left and right wheels respectively [33].

Model Name	Paramter	Equation	Description
Kinematic Model:	Linear velocity (V)	$V = \frac{r}{2}(\omega_l + \omega_r)$ (6)	Explains the robot's movement in terms of its linear and angular wheel velocities.
	Angular velocity (ω)	$\omega = \frac{r}{L}(\omega_r - \omega_l)$ (7)	Linear velocity is computed as the average of left and right wheel velocities, and angular velocity is determined as the difference divided by the wheelbase (L).
Dynamic Model	Torque on the left wheel (Tl)	$T_l = \frac{I_w}{2} \cdot \alpha_l$ (8)	Explains the wheel torques and their impact on linear acceleration.
	Torque on the right wheel (Tr)	$T_r = \frac{I_w}{2} \cdot \alpha_r$ (9)	Torques applied to the right wheel result in both linear and angular acceleration.
	Linear acceleration (a)	$a = \frac{r}{2}(\alpha_l + \alpha_r)$ (10)	Calculates the robot's linear acceleration by considering wheel torques
	Angular acceleration α	$\alpha = \frac{r}{w}(\alpha_r - \alpha_l)$ (11)	Computes the robot's angular acceleration by analyzing wheel torques.

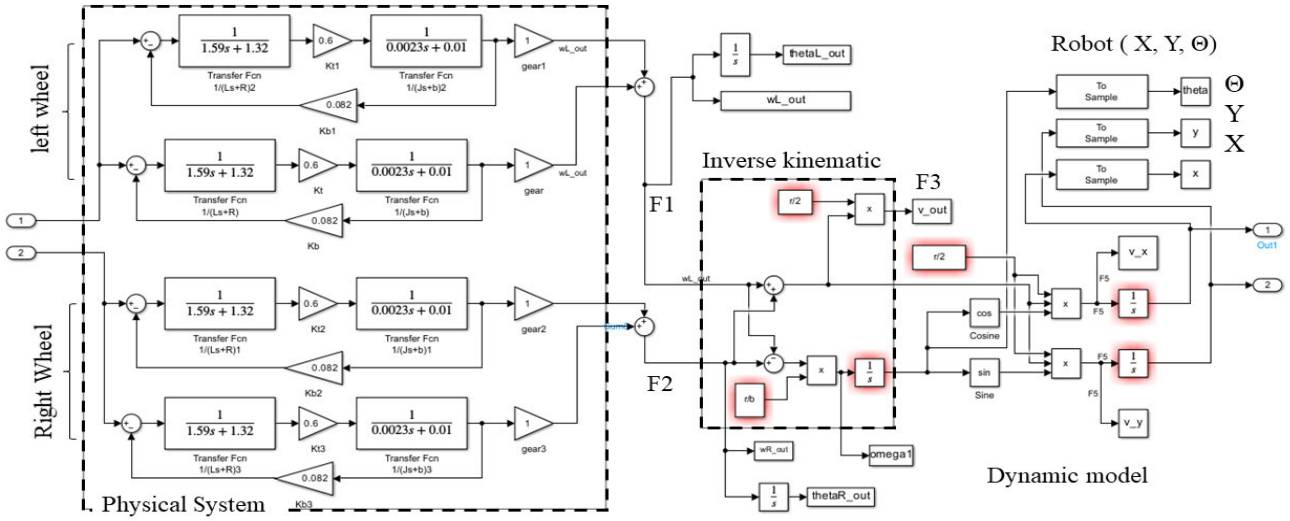


FIGURE 5. Physical system with dc motor parameters and inverse kinematic & dynamic model. The DC motor used in the system has the following values: armature resistance (R_a: 1.32 Ohm), Armature Inductance (L_a: 1.59 mH), Rotational Inertia (J_m: 0.0023 Kg.m^2), Back-EMF Constant (K_e: 0.082 V sec/rad), Motor Torque Constant (K_t: 0.6 N m/A), and Viscous Friction (B_m: 0.01 N/rad/sec) [36]. In the inverse kinematic and dynamic model, the angular velocity of the left wheel is denoted as $F_1 = \omega_L$, while the angular velocity of the right wheel is marked as $F_2 = \omega_R$. The robot's velocity (F_3) is calculated as $(F_1 + F_2) \cdot \frac{r}{2}$, and the angular velocity of the robot (F_4) is obtained through $(F_2 - F_1) \cdot \frac{r}{b}$. For the robot's motion analysis, the x-axis velocity (F_5) is computed as $F_3 \cdot \cos\theta$, the y-axis velocity (F_6) as $F_3 \cdot \sin\theta$, and the robot's x-axis position (x) is determined by $\int F_5$. Similarly, the robot's y-axis position (y) is derived from $\int F_6$, and the robot's angle (θ) is obtained through $\int F_4$.

The robot adeptly follows the path after extracting error values from sensor readings. The assigned error values range from -14 to -1, 0, and $+1$ to $+14$ for the left, middle, and right sensors, respectively. The PID value is then calculated to determine the robot's speed. In instances where none of the sensors are activated, indicating a deviation from the reference path, the PID controller is bypassed. In this scenario, the output values on both sides of the wheels are set to constants, ensuring the robot continues moving forward. Each sensor is assigned a coefficient number contingent

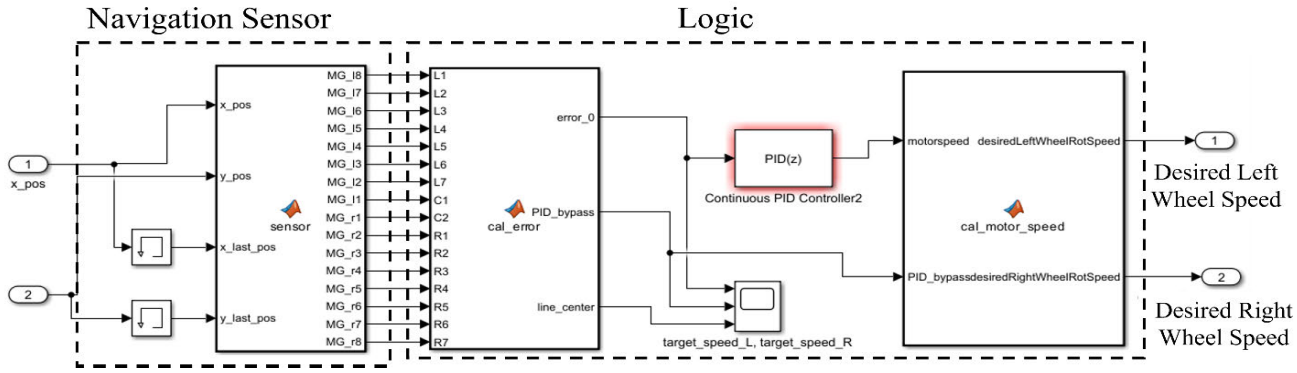


FIGURE 6. The navigation sensor and logic section of foodit simulation.

on its position. Sensors on the left side possess negative coefficients, while those on the right side have positive coefficients. The error value is the sum of the coefficients of all activated sensors. Ultimately, the voltage for the left wheel motor is set as a constant augmented by the PID output value, while the voltage for the right wheel is set as a constant reduced by the PID output value. In this research, the PID values ($K_p = 3.263175$, $K_i = 43.677826$, $K_d = 0.060948$) undergo an initial tuning using the empirical method [35], followed by optimization through the application of four distinct optimization algorithms.

IV. PID CONTROLLER OPTIMIZATION ALGORITHMS

To refine the empirical PID values, four bio-inspired algorithms—BAS, Particle Swarm Optimization (PSO), POA, and EO—are employed to achieve optimized PID controller values.

A. THE BEETLE ANTENNAE SEARCH ALGORITHM (BAS)

The Beetle Antennae Search Algorithm (BAS) is a nature-inspired optimization method that uses a population of beetles with random positions to find optimal solutions. It operates iteratively, with beetles moving towards improved solutions found by their neighbors and conducting local searches. The algorithm gradually reduces the search radius over time and continues until it converges. BAS mimics the foraging behavior of beetles and doesn't rely on explicit mathematical formulations, adapting its approach to each specific problem. It starts by initializing a group of individualized solution vectors and uses a custom fitness function to evaluate their quality. The algorithm also employs a dynamic search radius (R) that changes during its evolution, starting with an initial value (R-initial) and progressively decreasing as the algorithm runs.

The Beetle Antennae Search (BAS) algorithm as shown in Figure 7 initiates a population of N beetles, each with a random position, and utilizes parameters like maximum iterations, trials, initial and minimum search radius, and step size. The algorithm identifies the best solution in the population and returns this optimal outcome, leveraging the collective

Algorithm 1 Beetle Antennae Search (BAS)

Data: Initialize a population of N beetles with random positions
Data: Initialize parameters: *max_iterations*, *max_trials*, *initial_radius*, *minimum_radius*, *step_size*
for *iteration* = 1 to *max_iterations* **do**
  for each beetle in the population **do**
    Randomly select a neighboring beetle x_j within the current search radius Calculate fitness values F_i for the current beetle and F_j for the neighboring beetle **if** $F_j > F_i$ **then**
      Move the current beetle x_i towards x_j: $x_i = x_i + step\_size \cdot (x_j - x_i)$(12)
    else
      Randomly move the current beetle within its search radius: $x_i = x_i + random\_step$(13)
    Optionally, apply local search operators to improve x_i: $x_i = LocalSearch(x_i)$(14)
  Reduce the search radius R, possibly following a schedule or decay factor: $R = ReduceRadius(R)$(15) **if** R falls below *minimum_radius* **then**
    Reset it to *initial_radius*
Find the best solution in the population: $x_{best} = \text{argmax}(F_i)$ for all beetles in the population (16) Return the best solution x_{best} found by the beetles

exploration and exploitation behaviors inspired by beetle foraging. In the subsequent movement phase, occurring consistently in each iteration, the algorithm grants individual beetles (indexed as 'i') the autonomy to make deliberate movements toward their neighboring counterparts (indexed as 'j'), contingent on the superior quality of 'j's solution. This orchestrated mobility adheres to a defined algorithmic expression (Equation 12). When 'j' provides an inferior solution, the current beetle employs a tactic of random exploration within its local area. This haphazard exploration is guided

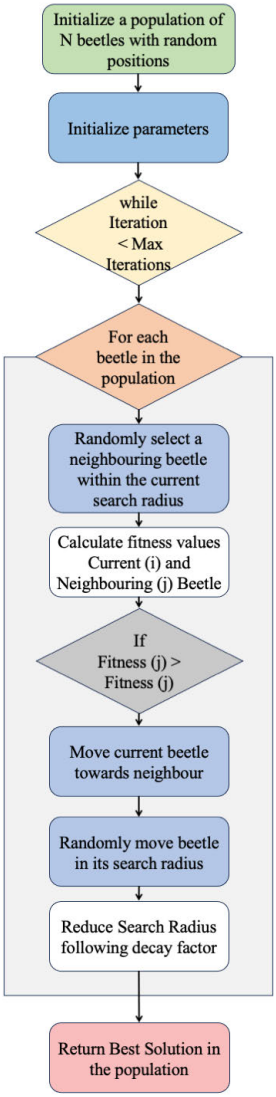


FIGURE 7. The beetle antennae search algorithm (BAS) algorithm flowchart.

by another algorithmic directive (Equation 13).Subsequently, the algorithm introduces the option for Local Search integration, enhancing the adaptability of the heuristic. This allows the incorporation of problem-specific optimization strategies to fine-tune individual beetle positions, as elucidated in Equation 14. This orchestration progresses with the potential contraction of the Search Radius. The controlled reduction of 'R' is facilitated through a prospective mechanism, which may follow a predetermined schedule or be guided by a decay factor, exemplified as Equation 15.The algorithm concludes with the anticipation of Termination, strictly adhering to predefined stopping criteria. It ceases its operations when it reaches either a specified number of iterations or attains a predefined convergence criterion. Ultimately, the optimization process yields the Optimal Solution, represented by the final disposition of the most efficient beetle within the population, succinctly denoted as Equation 16. It is essential to emphasize that the practical application of BAS invariably demands careful parameterization tailored to the specific characteristics of the problem domain. As a result, the heuristic's effectiveness lies in its adaptive response to the unique nuances of various problem domains, making it a versatile tool among heuristic optimization methodologies. Equations 12 through 16 describe various aspects of the algorithm's operation.

B. PARTICLE SWARM OPTIMIZATION (PSO)

Particle Swarm Optimization (PSO)is a nature inspired optimization algorithm that simulates the social behavior of birds or fish in search of optimal solutions. In PSO, a population of particles explores a solution space, adjusting their positions and velocities iteratively to improve their fitness based on a defined objective function. The Particle Swarm Optimization (PSO) algorithm embarks on its quest for optimization excellence with the crucial Initialization step, where a swarm of particles is artfully placed within the problem's solution space. Each particle possesses distinct attributes, delineating their positions and velocities, setting the stage for an intricate journey. In the subsequent Fitness Evaluation phase, during each iterative cycle, every particle meticulously assesses its fitness by invoking the objective function germane to the optimization problem. This discerning function, infused with mathematical rigor, acts as the arbiter of solution quality, scrutinizing the spatial configuration represented by each particle. Personal Best Update follows, where particles vigilantly safeguard their most illustrious past achievements. These particles maintain a repository of personal best solutions, venerating prior triumphs. Yet, when the current position heralds a more commendable fitness, a steadfast transition transpires as particles promptly update their personal best to reflect this newfound zenith. Global Best Identification stands as an operation of profound significance, where the algorithm astutely identifies the paramount solution within the collective swarm. Velocity and Position Update ensues as a magnum opus within the PSO orchestration, an intricate choreography governing the fluidity of particle movement. Influenced by three key factors, this ballet commences with the Inertia Weight, akin to an anchor, preserving particles existing velocities. Subsequently, the Cognitive Coefficient entices particles towards their personal best solutions, while the Social Coefficient orchestrates a collective pursuit of the global zenith.

These velocity updates harmoniously balance exploration and exploitation while imposing constraints to curtail undue excursions. Termination Criteria, a vigilant sentinel, oversees the algorithm's progression, scrutinizing.

Conditions that signal its culmination. These conditions encompass a designated maximum iteration threshold or criteria rooted in fitness enhancement. Upon meeting these discerning criteria, the algorithm gracefully concludes; otherwise, it resumes its iterative pursuit. The PSO odyssey culminates with the unveiling of the Optimal Solution,

Algorithm 2 Particle Swarm Optimization (PSO)

Data: Initialize a swarm of particles with random positions and velocities
Data: Initialize parameters:
Data: *max_iterations*
Data: *cognitive_coefficient*
Data: *social_coefficient*
Data: *inertia_weight*
Data: *maximum_velocity*
Data: *termination_criteria_threshold*
for *iteration* = 1 to *max_iterations* **do**
  for each particle in the swarm **do**
    Calculate the fitness of the current particle's position: $fitness_i = EvaluateFitness(position_i)$ Update the particle's personal best position if the fitness is better: **if** $fitness_i > personal\_best\_fitness[i]$ **then**
      $personal\_best\_position[i] = position_i$
      $personal\_best\_fitness[i] = fitness_i$
  Identify the global best position among all particles in the swarm: $global\_best\_position = GetGlobalBestPosition()$
  for each particle in the swarm **do**
    Update the particle's velocity: $velocity_i = inertia\_weight \cdot velocity_i + cognitive\_coefficient \cdot rand() \cdot (personal\_best\_position[i] - position_i) + social\_coefficient \cdot rand() \cdot (global\_best\_position - position_i)$ Limit the particle's velocity to a maximum allowable value if necessary: **if** $||velocity_i|| > maximum\_velocity$ **then**
      $velocity_i = maximum\_velocity \cdot \frac{velocity_i}{||velocity_i||}$
    Update the particle's position: $position_i = position_i + velocity_i$
  Check the termination criteria: $||global\_best\_fitness - previous\_global\_best\_fitness|| < termination\_criteria\_threshold$ **if** $||global\_best\_fitness - previous\_global\_best\_fitness|| < termination\_criteria\_threshold$ **then**
    Exit the loop
Identify the best solution found by the swarm: $best\_solution = global\_best\_position$ Return the best solution: *best_solution*

a singular gem fashioned from the collective endeavors of the swarm. This zenith, a representative of the most superlative solution uncovered by any particle within the ensemble, provides a resounding answer to the optimization riddle. The valediction, realized in the Output phase, signifies the bequeathal of the optimal solution, accompanied by its attendant fitness value. This latter metric, a yardstick of unparalleled precision, serves as a testament to the solution's quality. The Particle Swarm Optimization (PSO) algorithm aims to find an optimal solution by simulating the behavior of a swarm of particles in a search spaceas shown in brief

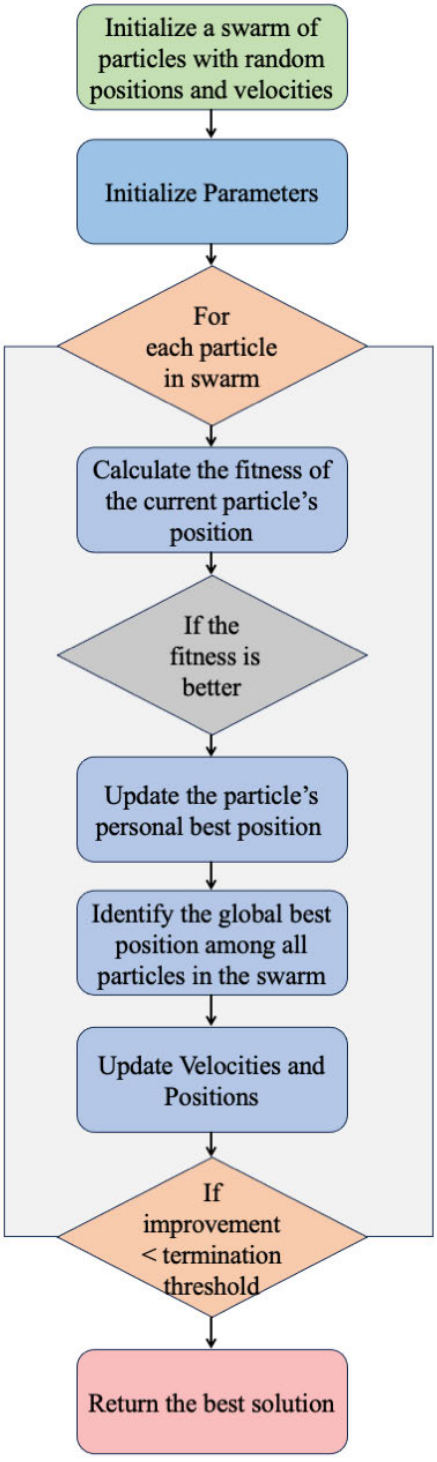


FIGURE 8. Particle Swarm Optimization (PSO) algorithm flowchart.

in Figure 8. Each particle represents a potential solution with a position and velocity. In each iteration, the fitness of each particle is evaluated, and if a particle's current position yields a better fitness than its personal best, the personal best is updated. The particles adjust their velocities based on inertia, cognitive influence (towards personal best), and social influence (towards global best). The algorithm continues iterating until a termination criteria and a best solution is returned In the whole, these meticulously choreographed stages, perpetuated within an unceasing iterative continuum, epitomize the bedrock of the PSO algorithm. With unwavering fidelity, they underpin an intricate waltz of particle positions and velocities, harmoniously balancing exploration and exploitation. In this orchestrated symphony, PSO adeptly navigates the intricate mazes of complex solution spaces, diligently seeking optimal or near-optimal solutions across a vast landscape of pragmatic optimization challenges.

C. PELICAN OPTIMIZATION ALGORITHM (POA)

The Pelican Optimization Algorithm (POA) is a nature-inspired optimization algorithm that draws inspiration from the behavior of pelicans, specifically their foraging and hunting strategies.

The variable 'w' denotes the inertia weight, gradually decreasing to strike a balance between exploration and exploitation. 'c1' and 'c2' stand for cognitive and social coefficients, steering the pelicans towards their personal best and global best positions. 'rand()' generates random values within the range of 0 to 1. 'current Position', 'personal Best Position', and 'global Best Position' are vectors representing the positions of the pelicans in the algorithm. The algorithm starts by creating a population of pelicans, each with a random initial position in the search space. These positions represent potential solutions to the optimization problem.

In the Fitness Evaluation step, In each iteration, every pelican evaluates its current position's fitness using a predefined objective function. The fitness function quantifies how good a solution is in the context of the problem being optimized. If a pelican discovers a better position (higher fitness) than its personal best, it updates its personal best accordingly. Global Best Identification step, The algorithm identifies the pelican with the best fitness as the global best. This pelican's position represents the best solution found across the entire population. Velocity and Position Update step, Pelicans update their positions in the search space based on their current velocities.

The Pelican Optimization Algorithm (POA) starts by initializing a population of pelicans with random positions as shown in Figure 9. Each pelican's fitness is evaluated using an objective function, and their personal best position and fitness are updated if a better fitness is achieved. The global best pelican is identified based on the best fitness. The algorithm iterates until termination criteria are met. In each iteration, pelicans' velocities are calculated using a formula involving inertia weight (w), cognitive coefficient (c1), and social coefficient (c2). Pelicans' positions are updated accordingly, and fitness is evaluated for the new positions. If the fitness surpasses personal or global bests, corresponding updates are made. The inertia weight decreases over time to reduce exploration. The algorithm concludes by returning the global best position and fitness as the optimal solution.

Algorithm 3 Pelican Optimization Algorithm (POA)

Data: Initialize population of pelicans with random positions
for each pelican in population **do**
  Evaluate the fitness of the pelican's position using the objective function **if** fitness is better than the pelican's personal best fitness **then**
    Update pelican's personal best position and fitness
Find the pelican with the best fitness as the global best
while termination criteria not met **do**
  for each pelican in population **do**
    Calculate velocity using formula: $velocity = w \cdot velocity + c1 \cdot rand() \cdot (personal\_best\_position - current\_position) + c2 \cdot rand() \cdot (global\_best\_position - current\_position)$
    Update pelican's position using formula: $position = current\_position + velocity$
    Evaluate the fitness of the new position using the objective function **if** fitness is better than the pelican's personal best fitness **then**
      Update pelican's personal best position and fitness
    if fitness is better than the global best fitness **then**
      Update global best position and fitness
  Decrease w (inertia weight) over time to reduce exploration
Return the global best position and fitness as the optimal solution

D. EQUILIBRIUM OPTIMIZER (EO)

The Equilibrium Optimizer (EO) algorithm is a nature-inspired optimization approach that mimics equilibrium-seeking behavior in natural systems to find optimal solutions for optimization problems. The algorithm utilizes random values, position vectors, cognitive and social coefficients, and velocity updates to guide solutions within certain bounds. The process involves:

- **Initialization:** Creating a population of potential solutions with random positions in the solution space.
- **Fitness Evaluation:** Assessing each individual's fitness by applying the objective function.
- **Global Best Identification:** Identifying the best-performing individual, known as the global best, which

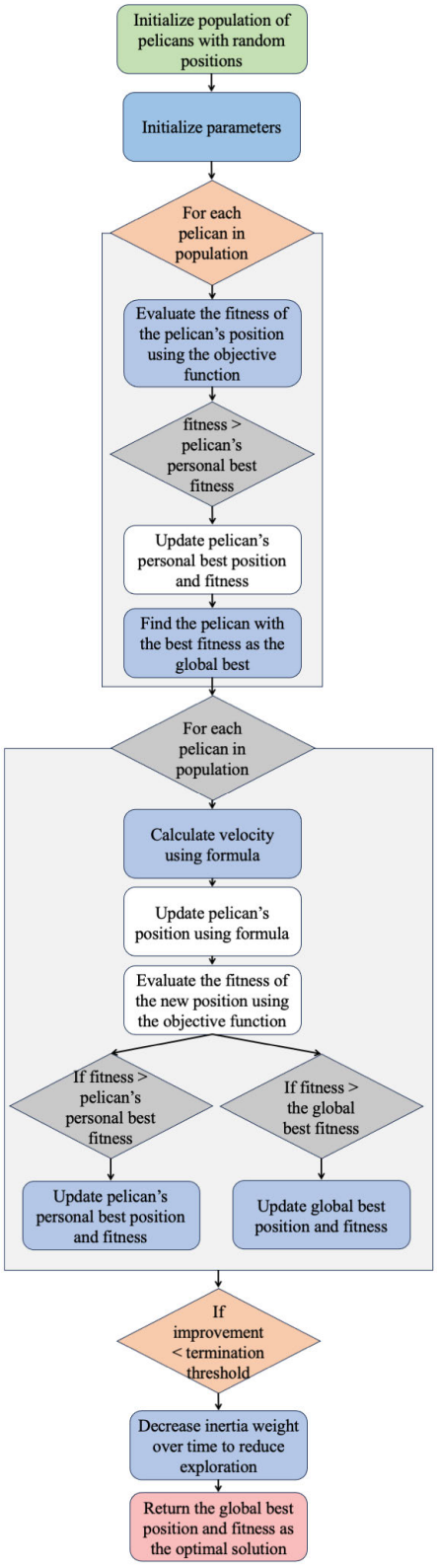


FIGURE 9. Pelican optimization algorithm (POA algorithm flowchart.

represents the best solution found in the entire population in terms of fitness.

During each iteration, individuals in the algorithm calculate their velocities, which are influenced by a randomization factor for exploration, cognitive coefficients for personal learning, and social coefficients for collective learning. Using these velocities, they update their positions within the solution space to effectively explore and exploit the search space while staying within predefined boundaries.

The algorithm continues iterating until specific termination criteria are met, which could be a maximum number of iterations or convergence based on fitness improvement. As the algorithm progresses, it gradually reduces the randomization factor. EO's strength lies in its ability to balance exploration and exploitation effectively, inspired by the equilibrium-seeking behavior found in various natural systems.

It has been applied to a wide range of optimization challenges across different domains, including engineering, finance, and machine learning. As with any optimization algorithm, parameter tuning and adaptation to specific problem domains are crucial for achieving optimal results.

In the Figure 10, the optimization algorithm is shown in brief. The Equilibrium Candidate-based Particle Swarm Optimization (ECPSO) algorithm aims to optimize a given fitness function. It initializes particle populations, assigns equilibrium candidates' fitness a large number, and sets free parameters. The algorithm iterates until the maximum iteration count is reached. In each iteration, it calculates fitness values for particles and equilibrium candidates. Based on fitness comparisons, particles in the equilibrium pool are updated, and the pool is recalculated. After a certain iteration threshold, a secondary loop generates random vectors and constructs global best position of the equilibrium candidate particle (GCP) based on mathematical equations. Then concentrations calculated are updated using these parameters, and the process continues until the maximum iteration count is reached. The algorithm's objective is to find the optimal solution by dynamically adjusting equilibrium candidates and concentrations.

V. RESULTS AND DISCUSSION

A. RESULT OF OPTIMIZER PERFORMANCE

At the outset, the PID values were initially tuned using the empirical method outlined in the [36]. However, in pursuit of optimized performance, four optimization algorithms (BAS, PSO, POA, EO) were employed. These algorithms aimed to refine the PID controller's track and value, leveraging the settings specified in Table 5.

Following the specified settings and expected PID values for each optimizer algorithm, the robot's movement was tested across four distinct paths: Circle Shape, Ellipse Shape, Spiral Path, and Eight Shape Path. The figure 11 illustrates the robot's movement, showcasing path trajectory, tracking error, tangential velocity, and angular velocity across the four distinct paths. In consideration of the optimization algorithms' trajectories and the pertinent parameters, illustrated in the accompanying figure, it is evident that all paths,

Algorithm 4 Equilibrium Candidate-based Particle Swarm Optimization

Data: Start and Initialization
Data: Initialize the particle's populations $i = 1, \ldots, n$
Data: Assign equilibrium candidates' fitness a large number
Data: Assign free parameters: $a = 2, a = 1, GP = 0.5$
Data: Iter = 0, Max_iter = 1000, I = 500
while Iter < Max_iter **do**
  for $i = 1$ to n **do**
    Calculate fitness of i-th particle fit_G = calculate_fitness(G) fit_Ceq1 = calculate_fitness(Ceq1) fit_Ceq2 = calculate_fitness(Ceq2) fit_Ceq3 = calculate_fitness(Ceq3) fit_Ceq4 = calculate_fitness(Ceq4) fit_C = calculate_fitness(C) fit_Cegi = calculate_fitness(Cegi)
    if fit_G < fit_Ceq1 **then**
      replace(Ceq1, Gi, fit_G, C)
    else if fit_G > fit_Ceq1 and fit_G < fit_Ceq2 **then**
      replace(Ceq2, Gi, fit_G, C)
    else if fit_G > fit_Ceq1 and fit_G > fit_Ceq2 and fit_G < fit_Ceq3 **then**
      replace(Ceq3, Gi, fit_G, C)
    else if fit_C > fit_Cegi and fit_G > fit_Ceq2 and fit_G > fit_Ceq3 and fit_C < fit_Ceq4 **then**
      replace(Ceq4, Gi, fit_G, C)
    Update equilibrium pool Ceq_ave = (Ceq1 + Ceq2 + Ceq3 + Ceq4) / 4 Ceq_pool = [Ceq1, Ceq2, Ceq3, Ceq4, Ceq_ave]
    if Iter > I **then**
      Iter = Iter + 1 **for** $i = 1$ to n **do**
        Randomly choose one candidate from the equilibrium pool (vector) candidate = random.choice(Ceq_pool) Generate random vectors of A from Eq (16) Construct $F = a \cdot \text{sign}(f - 0.5) \cdot e^{-t-1}$ (Eq. 16) Construct $GCP = 0.5 \cdot \eta G$ (Eq. 17) Construct $Go = GCP(Ceg - Ceq)$ (Eq. 18) Construct $G = Go \cdot F$ (Eq. 19) Update concentrations $Ceg = Ceg + (G - Ceg) \cdot (1 - e^{-t})$ (Eq. 20) Iter = Iter + 1
Result: End of while loop

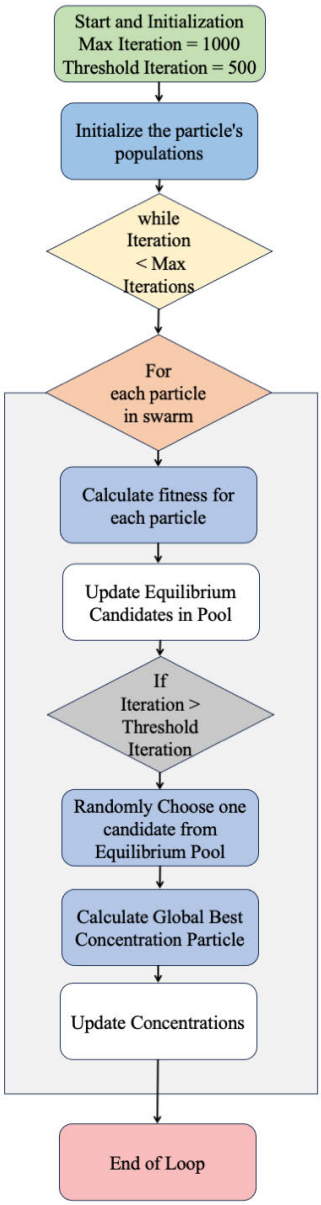


FIGURE 10. Equilibrium candidate-based Particle Swarm Optimization (PSO) algorithm flowchart.

defined by coefficients from four distinct methods, have been successfully traversed by the robot. Notably, the path error remains minimal for all trajectories, except for the spiral path, where it escalates with an increase in radius. for all this process each algorithm repeated 10 times.

Table 5 shows different optimization methods(BAS, PSO, POA, EO) applied to various scenarios characterized by the number of particles(indicates the number of particles used in the optimization process), maximum iterations(maximum number of iterations for the optimization process), and paths(Path1 to path4), Elapsed time(time taken for the optimization process to complete in second), Avg Corr (Average Correlation) average correlation coefficient obtained as a result of the optimization, Speed (m/s)(the speed of the robot achieved using the PID coefficients optimized through the respective method). The outcomes of the robot's movement utilizing

TABLE 5. The optimizer algorithm settings and PID extracted values.Maximum Iteration(MAX -It),random between 1 to zero(rand [0-1]).

BAS Setting		PSO Setting		POA Setting		EO Setting	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
MAX -It	10	MAX -It	10	MAX -It	10	MAX -It	10
d0	0.001	population	10	pelicans number	10	a1	2
d1	3	Inertia weight	1			a2	1
Eta	0.95	c1, c2	2.05			GP	0.5
Step length	0.8					r1, r2	rand[0-1]
Eta step	0.95	r1, r2	rand[0-1]			particles	2
P	0.03	P	0.025498209	P	0.019967	P	0.026285697
I	0.0001	I	0.001652876	I	0.000747	I	0.001865601
D	0.001	D	0.017505199	D	0.005252	D	0.00473531

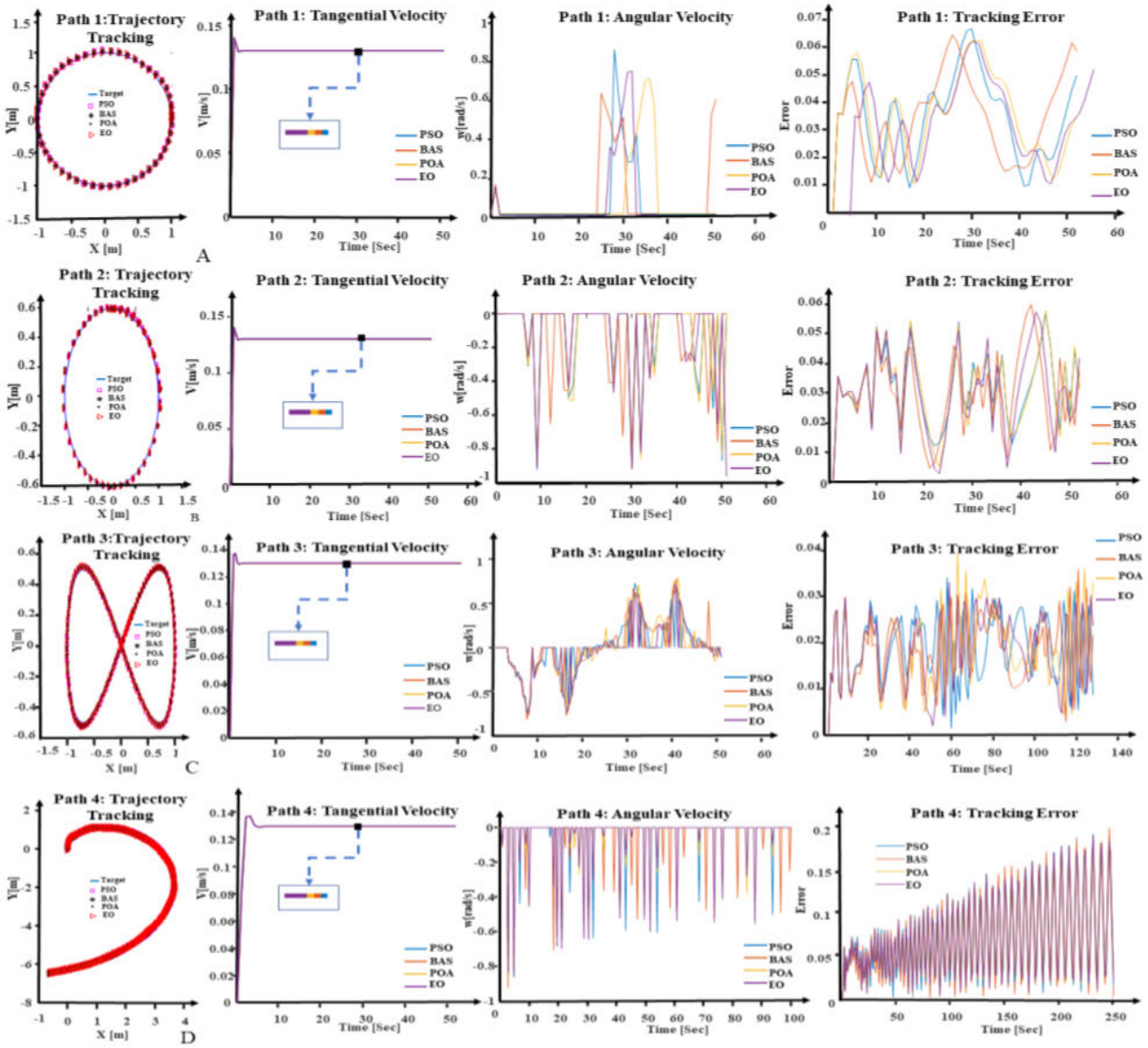


FIGURE 11. Circle shape ($x = 0.3 \times \cos(0.3 \times t)$, $y = 0.3 \times \sin(0.3 \times t)$, Length: 7.9293 meters), Ellipse Shape ($x = 0.5 \times \sin(0.3 \times t)$, $y = 0.3 \times \cos(0.3 \times t)$, Length: 7.9301 meters), Spiral Path ($x = 0.1 \times t \times \cos(0.3 \times t)$, $y = 0.1 \times t \times \sin(0.3 \times t)$, Length: 7.9309 meters), Eight Shape Path ($x = 0.5 \times \sin(0.3 \times t)$, $y = 0.5 \times \sin(0.3 \times t) \times \cos(0.3 \times t)$, Length: 7.9282 meters).

the optimized PID values for each path are detailed in Table 6.

Table 6 shows the optimal conditions for achieving either the highest speed along the route or the greatest accuracy

TABLE 6. The optimization algorithm (BAS, PSO, POA, EO) results for circle shape(shown as 1), ellipse shape(shown as 2), spiral path(shown as 3), and Eight Shape Path(shown as 4) and elapsed time in the second, average correlation and speed meter/ second.

Method	Particles No	Max Iteration	Path	Elapsed Time (Second)	Avg Corr	Speed (m/s)
BAS	1	1	1	42.4175754	**0.99998794**	1.396619302
	1	1	2	36.4147978	0.999972937	1.396606709
	1	1	3	35.3343722	0.999949912	**1.39662711**
	1	1	4	35.6712477	0.999862529	1.396564588
PSO	7	2	1	144.6500631	0.999986944	1.443536275
	10	1	2	214.6614239	0.999977003	1.473419644
	7	8	3	425.8249164	**0.999949883**	1.488781939
	9	8	4	529.2995626	0.999918135	**1.505637686**
POA	6	3	1	286.2698212	0.9999881	1.397981994
	10	3	2	558.6823481	0.999978142	**1.484725055**
	4	3	3	189.3700008	**0.999950453**	1.410752058
	9	7	4	1105.031704	0.999917872	1.469588752
EO	4	2	1	75.7053047	0.999988145	1.366124428
	9	8	2	610.1978842	0.999978799	1.45704462
	5	10	3	460.3919135	**0.999950193**	**1.473996489**
	7	5	4	312.7322637	0.999922315	1.459489351

for each route. As shown, the elapsed time for each scenario varies significantly, with some scenarios taking longer to optimize than others. For example, the PSO method with 9 particles, 8 maximum iterations, and Path 4 took the longest time at 1105.03 seconds. It appears that the speed tends to be higher for some scenarios with certain parameter settings. Different methods have varying performances across different scenarios. For example, EO with 4 particles, 2 maximum iterations, and Path 1(Circle) achieved the lowest speed at 1.366 m/s, while PSO with 7 particles, 2 maximum iterations, and Path 1 achieved the highest speed at 1.443 m/s. The optimizer performance over the speeds (Table 7) shows that over the four paths, the BAS method demonstrates consistent performance, achieving similar speeds across all paths (approximately 1.396 to 1.397). In contrast, the PSO method attains slightly higher speeds but exhibits greater variability, with speeds ranging from approximately 1.443 to 1.506. The POA method also displays variable speeds across paths, ranging from approximately 1.410 to 1.485, notably outperforming BAS on Path 3. Meanwhile, the EO method consistently maintains speeds ranging from approximately 1.366 to 1.474, showcasing stable performance across all paths.

To compare the correlation over the paths as shown in Table 8 BAS generally exhibits a high correlation with all paths, indicating that it performs well across different routes. PSO also demonstrates a strong correlation with all paths, suggesting it is effective in optimizing for these objectives. POA appears to have a high correlation as well, indicating its effectiveness in optimizing the specified objectives along the paths. EO shows high correlations, suggesting its effectiveness in achieving the desired objectives on the given routes. On the whole, all four methods (BAS, PSO, POA, EO) seem to perform well in optimizing for the specified objectives along the different paths, as indicated by their high correlation values. Utilizing the aforementioned outcomes, we can now compute the PID coefficients for the new route employing the four optimization methods. In this context, the particle count is held constant at a predefined value, as determined from the preceding results, and each algorithm is executed ten times. This trend persists in a newly explored route, showcasing increased errors in paths with larger radii. This observation advocates for the recommendation to design robot trajectories with either straight or gently curved paths, aiming to minimize radii. Regarding velocity analysis, the graphs indicate that the robot achieves a maximum linear speed of 0.13 m/s. This speed, limited by the capabilities of the engine and gearbox, aligns with the intended application of the robot for food transportation.

Additionally, the angular speed is capped at a maximum of 1 rad/s, an appropriate parameter for a food-serving robot to prevent spillage, as higher speeds could lead to unintended consequences. Analyzing the angular speed across different paths, it's notable that elliptical and spiral paths exhibit negative angular speeds, while the circular path displays a positive angular speed. This discrepancy arises from the clockwise traversal of oval and spiral paths, where the left wheel's speed surpasses that of the right. As a result, the robot's angular speed, defined by the difference between the right and left wheel speeds, becomes negative. Conversely, in the counterclockwise circular path, the angular speed remains positive. Path eight introduces a unique dynamic by transitioning from clockwise to counterclockwise traversal, resulting in a change in the sign of angular velocity at the midpoint of the path.

TABLE 7. Best speed for optimization algorithm (BAS, PSO, POA, EO) over paths for circle shape(shown as 1), ellipse shape(shown as 2), spiral path(shown as 3), and eight shape path (shown as 4).

Method	Path 1	Path 2	Path 3	Path 4
BAS	1.396619302	1.396606709	1.39662711	1.396564588
PSO	**1.443536275**	1.473419644	**1.488781939**	**1.505637686**
POA	1.397981994	**1.484725055**	1.410752058	1.469588752
EO	1.366124428	1.45704462	1.473996489	1.459489351

B. OPTIMIZED PID VALUE EVALUATION

To assess the gathered data, the comparison involves two main stages:

TABLE 8. Best correlation for each method and path.

Method	Path 1	Path 2	Path 3	Path 4
BAS	0.99998794	0.999972937	0.999949912	**0.999862529**
PSO	0.999986944	0.999977003	0.999949883	0.999918135
POA	0.9999881	0.999978142	**0.999950453**	0.999917872
EO	**0.999988145**	**0.999978799**	0.999950193	0.999922315

1) Assessing the performance on the compound path and determining the success rate of traversing this route.
2) Investigating the relationship between variations in optimized controller speeds and correlating these changes with alterations in speed and error rates.

1) COMPOUND PATH EVALUATION

To thoroughly validate the optimized PID values, a 470-meter compound path comprising both lines and arcs was meticulously tested. This path includes all crucial sections from previous iterations, ensuring a comprehensive assessment. This evaluation aimed to affirm the robot's performance across various parameters, including path trajectory, tracking error, tangential velocity, and angular velocity figure 12.
The compound path figure 12, was tested over all optimized algorithms and the robot performance was tested. As the sample Figure 12, it observes the values of tangential speed, angular speed, and the speeds of the left and right wheels of the robot, all of which have been determined using the PID coefficients obtained through the PSO method. As illustrated, the robot has achieved a linear speed exceeding 0.1 m/s, an ideal performance level for its intended application. Moreover, the simulation's desired speed closely matches the actual speed of the robot. secondly, the angular speed of the left and right wheels, averaging around 2.4 radians/s and peaking at 3.2 radians/s, comfortably below the maximum capacity of the robot's propulsion system.

The maximum angular velocity along the prescribed path is graphically presented. This depiction illustrates that the robot's maximum angular speed in practical operation reaches 0.2 radians/second, a fitting rate for its specific functionality. The path diagram unmistakably reveals the robot's adeptness in faithfully following the designated route, comprising diverse segments. Additionally, an analysis of the error diagram indicates that errors are more pronounced in path segments with larger radii, whereas they are negligible in other portions, notably in straight paths and minor curves.

2) OPTIMIZED PID AND SPEED VARIATIONS

Each optimized algorithm, configured with the specified settings outlined in table 9, underwent 10 iterations for rigorous evaluation. The outcome of each algorithm iteration was recorded and analyzed to discern performance consistency and ascertain the algorithm's effectiveness across BAS: For the BAS, the duration of the optimization process fluctuates based on various parameter setups. The optimization duration ranges approximately between 103 seconds to 156 seconds for these setups, which encompass diverse combinations of PID coefficients. The values of kp, Ki, and Kd exhibit variations across rows, sometimes displaying alterations within the same column (such as differing Ki and Kd values while maintaining a fixed kp). Across all cases, the average correlation coefficients remain notably high, signifying a robust correlation between the PID coefficients and the desired outcomes. These coefficients consistently approach a value of 1, indicating a positive association. Moreover, the robot's achieved speed demonstrates slight fluctuations among different parameter configurations, maintaining relative consistency at around 0.1297 m/s.

The average correlation coefficients are relatively high in all cases, indicating a strong correlation between the PID coefficients and the desired outcomes. The values are consistently close to 1, which is a positive sign. The speed achieved by the robot also varies slightly across different parameter configurations. The values are relatively consistent, hovering around 0.1297 m/s. Overall, the BAS optimization method performs consistently well in optimizing PID coefficients for the given task, with slight variations in the speed achieved. The strong correlation between the coefficients and the desired outcomes suggests that the optimization process is effective. For the PSO The elapsed time for the optimization process varies across different parameter configurations.

TABLE 9. Best correlation for each method and path.

Elapsed Time	Kp	Ki	Kd	Avg Corr	Speed (m/s)
136.3848507	0.03	0.0001	0.001	0.999911420	0.129708219
118.1761428	0.03	0.0001	0.001	0.999911420	0.129708219
110.8884990	0.03	0.0020	0.020	0.999910927	0.129699790
109.3432232	0.03	0.0001	0.020	0.999911420	0.129708219
109.7107103	0.03	0.0020	0.020	0.999910927	0.129699790
103.8125510	0.02	0.0020	0.020	0.999911780	0.129723278
123.1355683	0.03	0.0001	0.001	0.999911420	0.129708219
135.7216938	0.03	0.0001	0.020	0.999910642	0.129723278
108.5949755	0.03	0.0020	0.001	0.999910927	0.129699790
156.9771500	0.03	0.0021	0.020	0.999911320	0.129723278

It ranges from approximately 104 seconds to 158 seconds Different combinations of PID coefficients were tested. The values of kp, Ki, and Kd vary across rows, with some variations in the same column (e.g., different values of Ki and Kd with a fixed kp). The values are close to 0.9999 or 1, which is a positive sign. The speed achieved by the robot is relatively consistent across different parameter configurations, with values around 0.1297 m/s.the PSO optimization method performs consistently well in optimizing PID coefficients for the given task. The strong correlation between the coefficients and the desired outcomes indicates the effectiveness of the optimization process. The PSO method parameters are their results are shown in table 10.

For the POA The elapsed time for the optimization process varies across different parameter configurations, ranging from approximately 178 seconds to 201 seconds. Different combinations of PID coefficients were tested. The values of kp, Ki, and Kd vary across rows, with some variations in the same column (e.g., different values of Ki and Kd with a fixed kp). The values are close to 0.9999 or 1, which is a positive sign The speed achieved by the robot is relatively

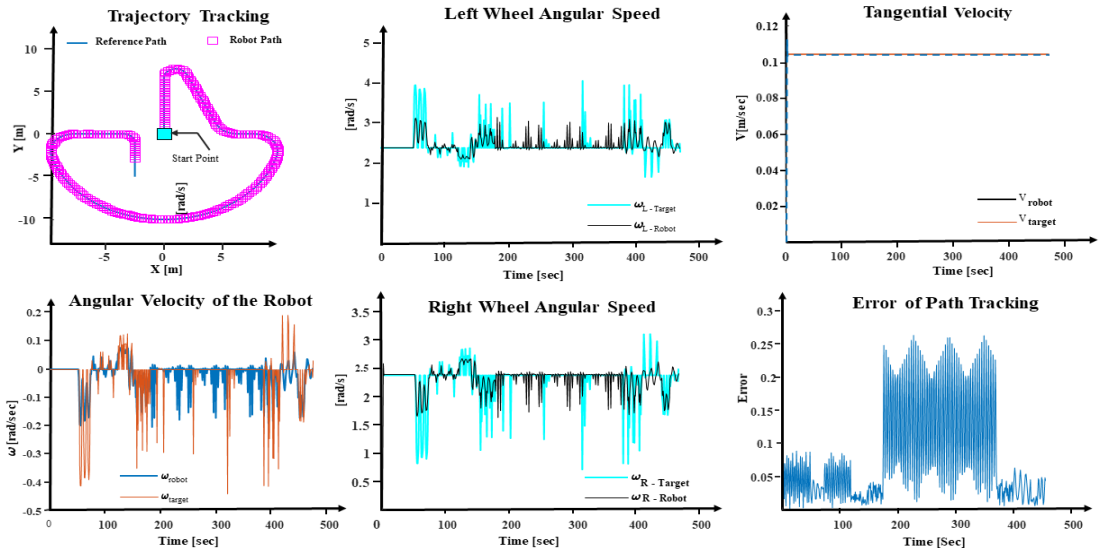


FIGURE 12. The PSO algorithm PID values(P = 0.025498209,I = 0.001652876,D = 0.017505199) and robot performance.

TABLE 10. PSO method parameters and results.

Elapsed Time	Kp	Ki	Kd	Avg Corr	Speed (m/s)
158.7505866	0.025844	0.001923	0.013459	0.99991092	0.12970545
103.4054296	0.023897	0.000702	0.019054	0.99991125	0.129706788
127.4605722	0.016808	0.001212	0.005252	0.99990959	0.129697924
114.9302595	0.026286	0.000563	0.018656	0.99991127	0.129708032
116.9969842	0.025583	0.001875	0.003468	0.99991094	0.129704075
144.4785902	0.023784	0.001521	0.00956	0.99991112	0.129697902
108.9736054	0.026001	0.00092	0.018302	0.99991119	0.129708785
144.7121539	0.018036	0.000244	0.005558	0.99991111	0.129706236
140.5809013	0.014834	0.000867	0.002833	0.99994245	0.12971449
104.5355302	0.02094	0.000663	0.015149	0.99991117	0.129705222

consistent across different parameter configurations, with values around 0.1297 m/s.the POA optimization method performs consistently well in optimizing PID coefficients for the given task. The strong correlation between the coefficients and the desired outcomes indicates the effectiveness of the optimization process. The PSO method parameters and their results are shown in table 11.

TABLE 11. PSO method parameters and results.

Elapsed Time	Kp	Ki	Kd	Avg Corr	Speed (m/s)
191.1464814	0.017285	0.000245	0.004148	0.9999102	0.129704961
201.5609415	0.0159	0.000702	0.01681	0.99991124	0.129703674
186.7680825	0.026913	0.001404	0.004885	0.99991115	0.129708301
195.7247694	0.028724	0.001959	0.001	0.99991107	0.129705117
190.3095571	0.018252	0.001237	0.004406	0.99991101	0.129703502
178.1313745	0.017771	0.000617	0.004386	0.99991094	0.129704835
193.030461	0.03	0.000109	0.018739	0.99991141	0.129707998
194.1156603	0.025435	0.001743	0.018867	0.99991106	0.129705881
183.0816421	0.03	0.000979	0.005362	0.99991123	0.129708771
198.3042482	0.022392	0.001804	0.015009	0.99991072	0.129716859

for the EO method The elapsed time for the optimization process varies across different parameter configurations, ranging from approximately 53 seconds to 65 seconds. Different combinations of PID coefficients were tested. The values of kp, Ki, and Kd vary across rows, with some variations in the same column (e.g., different values of Ki and Kd with a fixed kp). The values are close to 0.9999 or 1, which is a positive sign. The speed achieved by the robot is relatively consistent across different parameter configurations, with values around 0.1297 m/s. the EO optimization method performs consistently well in optimizing PID coefficients for the given task. The strong correlation between the coefficients and the desired outcome indicates the effectiveness of the optimization process. All four optimization methods (BAS, PSO, POA, EO) exhibit consistent performance in optimizing PID coefficients. They achieve high average correlation coefficients, indicating a strong correlation between the optimized coefficients and the desired outcomes.

TABLE 12. EO method parameters and results.

Elapsed Time	Kp	Ki	Kd	Avg Corr	Speed (m/s)
58.4965176	0.027957	0.001057	0.016569	0.9999112	0.12970831
60.3219016	0.025511	0.000918	0.018959	0.9999112	0.12970883
59.6193852	0.026974	0.001975	0.006157	0.9999111	0.12970819
62.0429704	0.028788	0.001017	0.010952	0.9999112	0.12970777
63.2222409	0.03	0.001286	0.001	0.9999112	0.12970867
59.1801343	0.019731	0.001728	0.001	0.999911	0.12970363
53.5160243	0.027544	0.000983	0.01807	0.9999112	0.12970832
62.3023747	0.026837	0.0001	0.006689	0.9999114	0.12970872
57.6801106	0.029469	0.002	0.010173	0.9999111	0.12970659
65.8226863	0.013555	0.0001	0.003711	0.99991	0.12970726

The EO method parameters are their results are shown in table 12. The speed achieved by the robot is also relatively consistent across different parameter configurations for each method. With the obtained PID coefficients, violin plots can be created for each method. These visualizations not only allow us to assess the range of resulting parameters but also provides a clear view of data dispersion and distribution. In the violin plots presented, each algorithm is executed ten times, and the PID parameters are calculated. Based

on the generated graphs, it is evident that all algorithms exhibit a broad range for KP and KD values, while the diversity in Ki values is considerably lower. This observation suggests that Ki parameters have a significant impact on the robot's accuracy and should not be altered extensively. In contrast, the influence of KP and KD parameters on controller accuracy is less pronounced, allowing for more noticeable variations in their values. Another noteworthy observation when examining these charts is the distribution of data points within the obtained range. These figures reveal a distinct pattern: among the four optimization methods, only the BAS method has produced parameter values at the extreme ends of the range. In contrast, the other three methods exhibit a more scattered distribution of data points across the entire parameter space. This phenomenon appears to be attributed to the specific characteristics of the BAS optimization algorithm. Additionally, it's worth noting that each algorithm has identified a unique optimal region within the PID coefficient space. Remarkably, all these algorithms have effectively enabled the controller to track the desired path with minimal error. In other words, achieving the desired objective, which is precise path tracking with minimal error, is feasible with different sets of PID parameters. Notably, these variations in controller coefficients have resulted in nuanced adjustments in the robot's accuracy and speed in path following, albeit relatively minor and generally inconsequential. It is essential to refer to the error diagram, speed table, and correlation coefficient figures for a comprehensive understanding of these findings. In the path error diagram, one can observe that the curved path with a large radius tends to exhibit a larger error compared to other segments of the path (such as straight sections or smaller curves). Importantly, despite these discrepancies, the robot remains within the prescribed path boundaries. The optimization method and the PID deviation saptial is shown in figure 13.

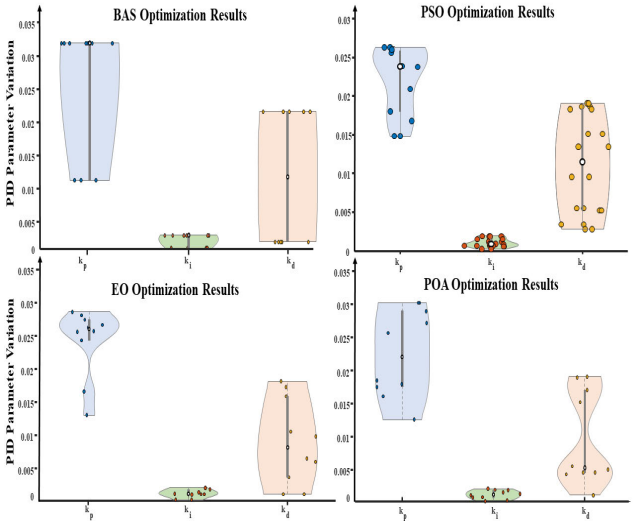


FIGURE 13. Optimzation method vs PID deviation saptial.

C. FINAL DEMONSTRATION TEST AND IMPLEMENTATION

In the final phase, the acquired PID values were implemented within the robot's logic section. As depicted in Figure 14, the robot adeptly tracked the predefined path for its delivery mission. In the controlled testbed scenario, the designated area encompasses the testing laboratory and the adjoining department corridor within a specific floor. The predetermined path for experimentation is established within this spatial context. The experimental setup involves the utilization of tables situated within the controlled laboratory environment. This careful selection ensures a controlled and replicable testing environment, facilitating systematic observations and assessments. As for the robot operations, firstly, the robot adheres to a predefined path using magnetic strips until it arrives at the specified location. Upon reaching the door, the ultrasonic sensor detects obstacles, triggering an alarm. Subsequently, the user stands before the robot, displaying the pre-entered color code from the app.as it mentioned in the Foodit Delivery Process the color code generated by the app is used to identify the user and Pixy camera. These unique color signatures encapsulate the distinct properties of color, including hue, saturation, and brightness and trained by the Pixy camera as the color signatures. This dispenser is equipped with four distinct color-coded places: Blue, Red, Green, and Yellow. By detecting these colors from a user's mobile device when ordering food, the robot accurately ascertains the user's location, facilitating the precise retrieval of the ordered food. The Pixy camera's role here is pivotal. It initiates the process, reads the number of detected color blocks, and communicates the specifics of the first detected block, including its color. Since each place in the Food Dispenser head corresponds to a specific color, the robot's arm can selectively retrieve the food from the designated location and present it to the user. The procedure begins with a check of the arm's position. If the current position does not align with the desired state (State0), the system promptly adjusts the position. Subsequently, the arm executes the pre-programmed movements corresponding to the food's location on the dispenser, deftly retrieving the item and delivering it to the user. For enhanced service quality, face detection algorithms come into play, aided by a USB camera. This innovation allows the robot to identify the user's face for more personalized service and customer satisfaction. The color signature assumes a pivotal role as it serves as the unique identifier for the objects the Pixy camera recognizes. The code continuously loops, diligently monitoring for fresh detection's and relaying this information to the serial monitor. Taking a broader perspective, the robot leverages the OpenCV API to engage in real-time face detection, utilizing the Raspberry Pi and a USB camera. This approach ushers in a versatile environment, enabling the robot to interact with individuals.

Upon recognizing signs of ordering, the robot re-positions itself and waits, providing an opportunity for users to place their orders. Subsequent to this waiting phase, the

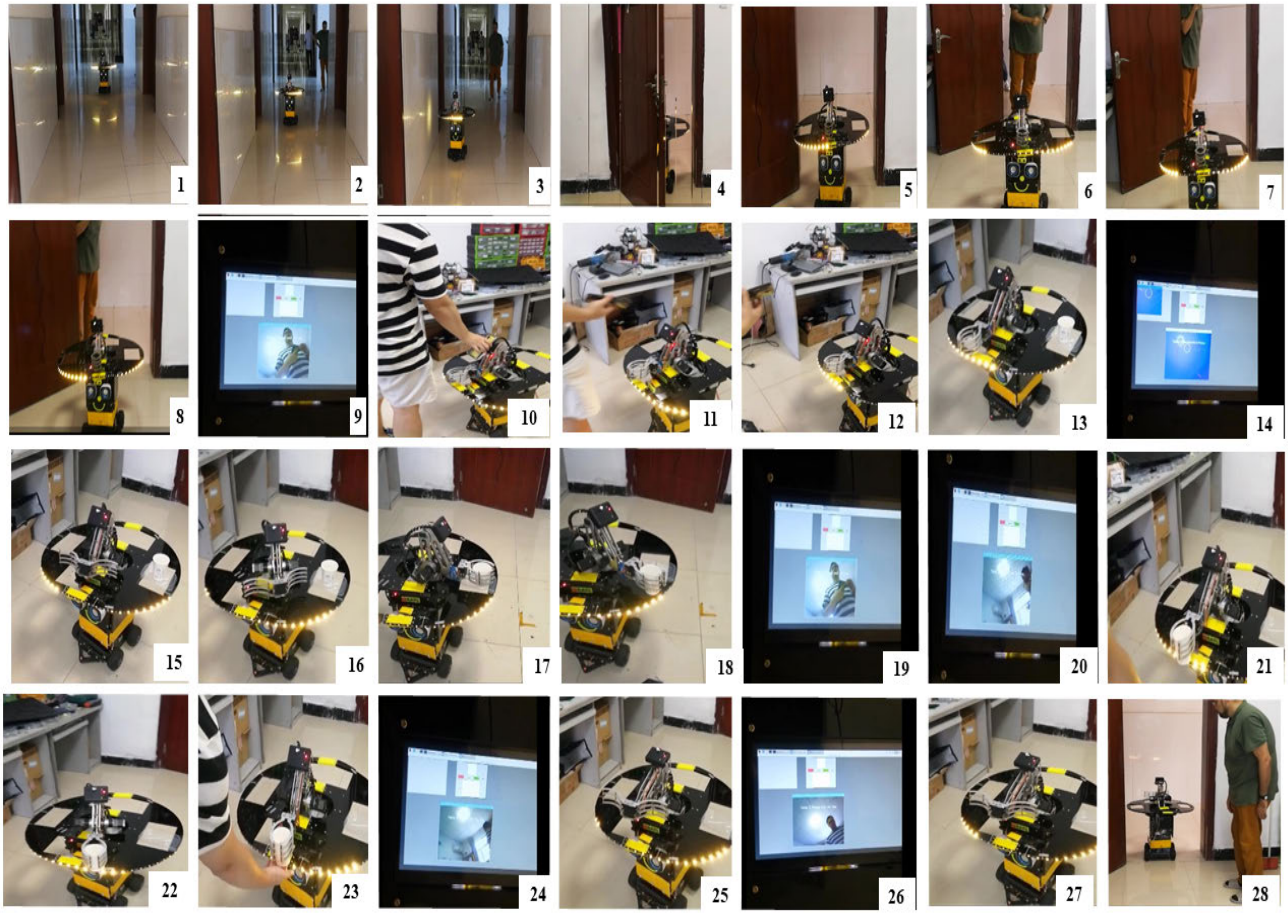


FIGURE 14. The Foodit robot Step 1: Navigation and Prep(Figure 1 to 8), Step 2: Plating and Delivery(Figure 9 to 25),Step 3: Customer Interaction(Figure 9 to 14), Step4: Returning and Reset(Figure 27 to 28).

system verifies if all orders have been collected, then proceeds to process them accordingly. To optimize resource utilization, the algorithm incorporates delays during face detection, reducing computational overhead and ensuring the efficient use of the Raspberry Pi's hardware resources. In summary, this algorithm underscores the integration of Raspberry Pi 3B+ and OpenCV, facilitating intelligent robot behavior in face detection and interaction scenarios. Notably, FOODIEBOT is enriched with a voice performance system, allowing it to communicate verbally and engage in partial human-computer interaction. Additionally, this system can provide entertainment by playing music. Moreover, the integration of obstacle sensors, such as ultrasonic sensors, serves as a crucial safety feature. These sensors are employed to detect and avoid collisions with users. When the robot's proximity to an object or user falls below a critical threshold, typically 20 cm, the robot halts its motion and emits an alarm sound, ensuring the safety of both the robot and the people it interacts with. Using this color code, the robot guides its arm to the corresponding colored area and, recognizing the user's face, delivers the specified package. Once the task is completed, it departs the area, tracing back its route along the magnetic lines.

VI. CONCLUSION

In this article, the process of designing and simulating a service robot named FOODIT has been discussed.In contrast to commercialized models, the designed structure boasts distinctive features, particularly its economical pricing. Notably, it accommodates the transport of four dishes, facilitating delivery to multiple customers an advancement compared to existing structures limited to serving a single table. Additionally, this model allows seamless integration with multiple platforms. Its straightforward routing system simplifies operations. Furthermore, it enables customer interaction through a mobile application, allowing patrons to assess restaurant service and food quality, while also facilitating real-time online environmental monitoring. In the simulation, the designed model of MATLAB software has been checked along with the initial setting of the PID controller parameters. After that, to check the performance of the robot, the cost values were evaluated by four methods BAS, PSO, POA, and EO optimally, and the performance during the four paths Circle Shape, Ellipse Shape, Spiral Path, and Eight Shape PathThe obtained results show that Among the three optimization methods (EO, PSO, OA), the highest accuracy and correlation coefficient, with $R = 0.99995$, were achieved

for the third path. In contrast, the BAS method demonstrated its best accuracy with R = 0.999988 for the first path. It's noteworthy that the choice of the optimization method played a significant role in accuracy outcomes, but there wasn't a clear pattern relating the method to maximum speed across the different paths studied. Each method attained its peak speed on a distinct path. An interesting finding is that superior results were not necessarily tied to employing the largest number of particles and repetitions. This observation carries implications for the practical implementation of the method, considering the time-intensive nature of the optimization process. Additionally, it's worth noting that various methods yielded different speeds for the various routes. For instance, for the first route, the PSO method achieved the highest speed at 1.443536 meters per second, while the EO method recorded the lowest speed at 1.366124 meters per second for the same route. A similar trend was observed for the second and fourth routes, where the PSO method consistently delivered the highest speed, with the third route seeing POA as the top performer. In terms of execution time, the BAS method generally outperformed other methods, likely due to the number of particles it utilized. This information underscores the importance of considering execution time in the choice of optimization method. In the context of the four distinct paths, the choice of the most accurate optimization method, as determined by the correlation coefficient, varies. For the first and second paths, the EO method stands out, while the third path demonstrates superior accuracy when employing the POA method. Remarkably, the BAS method yields the most favorable results for route number 4. When considering the optimization solution time for all four paths, it becomes evident that the BAS method exhibits the shortest execution time, highlighting its efficiency. Examining the maximum speed achieved along each route, the pattern reveals the following trends: The PSO method attains the highest speed in paths 1, 3, and 4, whereas route 2 showcases the highest speed when optimized with the POA method. the last obtained coefficients were finally implemented on the robot processor and the robot was used. The performance of the robot based on the coefficients obtained in reality and the mission assigned to the robot indicates the correctness of the calculations in the simulation and obtaining the optimal values for the PID controller. The current robot phase faces a major limitation in the delivery speed to the main tray, requiring two minutes due to arm operation. An improved design is essential to optimize the food delivery process. the forthcoming phase of development involves evaluating the integration of this structured design into a swarm format. This progression marks a pivotal stride toward system refinement, fostering comprehensive development and culminating in the creation of innovative functionalities, thereby advancing towards final completion.

REFERENCES

[1] L. Acosta, E. J. González, J. N. Rodríguez, A. F. Hamilton, J. A. Méndez, S. Herneández, M. Sigut, and G. N. Marichal, "Design and implementation of a service robot for a restaurant," *Int. J. Robot. Autom.*, vol. 21, no. 4, p. 273, 2006.

[2] N. Kulaç and M. Engin, "Developing a machine learning algorithm for service robots in industrial applications," *Machines*, vol. 11, no. 4, p. 421, Mar. 2023.

[3] R. Mohan, A. A. Prakash, N. U. Devi, S. A. Sharma, N. A. Babu, and P. Thennarasi, "Smart patient engagement through robotics," in *Proc. Hum.-Mach. Interface, Making Healthcare Digital*, 2023, pp. 115–159.

[4] P. A. Hancock, D. R. Billings, K. E. Schaefer, J. Y. C. Chen, E. J. de Visser, and R. Parasuraman, "A meta-analysis of factors affecting trust in human–robot interaction," *Human Factors*, vol. 61, no. 2, pp. 201–262, 2019.

[5] A. O. Adeleye, "Enabling assistive service robots to contextually organize household objects," Ph.D. dissertation, Dept. Comput. Sci., Univ. California San Diego, La Jolla, CA, USA, 2023.

[6] A. J. Moshayedi, N. M. I. Uddin, X. Zhang, and M. Emadi Andani, "Exploring the role of robotics in Alzheimer's disease care: Innovative methods and applications," *Robotic Intell. Autom.*, vol. 43, no. 6, pp. 669–690, Nov. 2023.

[7] A. L. Thomaz and G. Hoffman, "Computational human–robot interaction," *J. Hum.-Robot Interact.*, vol. 8, no. 1, pp. 1–3, 2019.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[9] S. Thrun, "Robotic mapping: A survey," *Exploring Artif. Intell.*, vol. 2, pp. 237–267, Feb. 2002. [Online]. Available: http://robots.stanford.edu/papers/thrun.mapping-tr.pdf

[10] W.-K. Kao and Y.-S. Huang, "Service robots in full- and limited-service restaurants: Extending technology acceptance model," *J. Hospitality Tourism Manage.*, vol. 54, pp. 10–21, Mar. 2023.

[11] Y. Qing-xiao, Y. Can, F. Zhuang, and Z. Yan-zheng, "Research of the localization of restaurant service robot," *Int. J. Adv. Robotic Syst.*, vol. 7, no. 3, p. 18, Sep. 2010.

[12] C.-Y. Lin, C.-C. Huang, L.-W. Chuang, B.-S. Lin, K.-Z. Lin, and C.-S. Fahn, "Towards finger gaming humanoid robot: Mechanism and perception development," *J. Chin. Inst. Engineers*, vol. 38, no. 5, pp. 621–635, Jul. 2015.

[13] A. Cheong, M. Lau, E. Foo, J. Hedley, and J. W. Bo, "Development of a robotic waiter system," *IFAC-PapersOnLine*, vol. 49, no. 21, pp. 681–686, 2016.

[14] M. Jahromi, F. Goli, H. Emamiyeh, and V. Noei, "Design of a waiter robot and automatic cleaning and disinfection table," *Turkish J. Comput. Math. Educ.*, vol. 12, pp. 1869–1874, Jul. 2021.

[15] W. D. Freeman, D. K. Sanghavi, M. S. Sarab, M. S. Kindred, E. M. Dieck, S. M. Brown, T. Szambelan, J. Doty, B. Ball, H. M. Felix, J. C. Dove, J. M. Mallea, C. Soares, and L. V. Simon, "Robotics in simulated COVID-19 patient room for health care worker effector tasks: Preliminary, feasibility experiments," *Mayo Clinic Proceedings: Innov., Quality Outcomes*, vol. 5, no. 1, pp. 161–170, Feb. 2021.

[16] J. Kim, "Distributed herding of multiple robots in cluttered environments," *Robot. Auto. Syst.*, vol. 146, Dec. 2021, Art. no. 103889.

[17] X. Zhang, M. S. Balaji, and Y. Jiang, "Robots at your service: Value facilitation and value co-creation in restaurants," *Int. J. Contemp. Hospitality Manage.*, vol. 34, no. 5, pp. 2004–2025, Apr. 2022.

[18] A. Jahangir Moshayedi, K. S. Reza, A. Sohail Khan, and A. Nawaz, "Integrating virtual reality and robotic operation system (ROS) for AGV navigation," *EAI Endorsed Trans. AI Robot.*, vol. 2, no. 1, p. e3, Apr. 2023.

[19] R. Alami, M. Warnier, and M. Gharbi, "Towards socially aware robots: A survey of the state of the art," *Robotics Auton. Syst.*, vol. 124, 2020, Art. no. 103286.

[20] F. Galasso, D. L. Rizzini, F. Oleari, and S. Caselli, "Efficient calibration of four wheel industrial AGVs," *Robot. Comput.-Integr. Manuf.*, vol. 57, pp. 116–128, Jun. 2019.

[21] I. Choi, H. Shim, and N. Chang, "Low-power color TFT LCD display for hand-held embedded systems," in *Proc. Int. Symp. Low Power Electron. Design*, 2002, pp. 112–117.

[22] C. Li, J. Park, H. Kim, and D. Chrysostomou, "How can i help you? An intelligent virtual assistant for industrial robots," in *Proc. Companion ACM/IEEE Int. Conf. Human-Robot Interact.*, Mar. 2021, pp. 220–224.

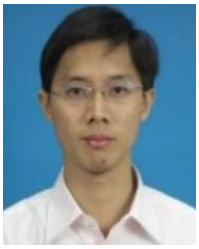
[23] S. E. Mathe, A. C. Pamarthy, H. K. Kondaveeti, and S. Vappangi, "A review on raspberry Pi and its robotic applications," in *Proc. 2nd Int. Conf. Artif. Intell. Signal Process. (AISP)*, Feb. 2022, pp. 1–6.

[24] J.-Y. Jang, S.-J. Yoon, and C.-H. Lin, "Automated guided vehicle (AGV) driving system using vision sensor and color code," *Electronics*, vol. 12, no. 6, p. 1415, Mar. 2023.

[25] S. Zhou, G. Cheng, Q. Meng, H. Lin, Z. Du, and F. Wang, "Development of multi-sensor information fusion and AGV navigation system," in *Proc. IEEE 4th Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, vol. 1, Jun. 2020, pp. 2043–2046.
[26] M. F. Aqillah, R. Mardiati, and A. E. Setiawan, "Prototype of robot movement navigation system using pixy camera (CMUCAM 5)," in *Proc. 8th Int. Conf. Wireless Telematics (ICWT)*, Jul. 2022, pp. 1–6.
[27] A. E. Setiawan, A. Rusdinar, S. Rizal, R. Mardiati, and E. A. Zaki Hamidi, "Design of multi robot AGV prototype maneuver control based on inverted camera," in *Proc. 16th Int. Conf. Telecommun. Syst., Services, Appl. (TSSA)*, Oct. 2022, pp. 1–5.
[28] J. F. Gorostiza, J. Morales, and J. Ruiz-del-Solar, "Survey of self-repair in robotics: A review from the cybernetic point of view," *Robot. Auton. Syst.*, vol. 133, Oct. 2020, Art. no. 103634.
[29] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Int. J. Robot. Res.*, vol. 3, no. 1, pp. 43–53, 2016.
[30] A. J. Moshayedi, A. S. Roy, L. Liao, and S. Li, "Raspberry Pi SCADA zonal based system for agricultural plant monitoring," in *Proc. 6th Int. Conf. Inf. Sci. Control Eng. (ICISCE)*, Dec. 2019, pp. 427–433.
[31] S. Chowdhury, R. Rahmani, H. N. Saha, and D. Taniar, "Predictive maintenance for effective asset management in industry 4.0," *Proc. Manuf.*, vol. 11, pp. 940–947, 2017.
[32] I. Nevliudov, S. Maksymova, O. Klymenko, and M. Bilousov, "Development of a mobile robot prototype with an interactive control system," *Syst. Manag., Navigat. Commun.*, vol. 3, no. 73, pp. 128–133, 2023.
[33] A. J. Moshayedi, J. Li, N. Sina, X. Chen, L. Liao, M. Gheisari, and X. Xie, "Simulation and validation of optimized PID controller in AGV (automated guided vehicles) model using PSO and BAS algorithms," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–22, Nov. 2022.
[34] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Eng. Sci. Technol., Int. J.*, vol. 40, Apr. 2023, Art. no. 101343.
[35] S. Levine and D. Shah, "Learning robotic navigation from experience: Principles, methods and recent results," *Phil. Trans. Roy. Soc. B: Biol. Sci.*, vol. 378, no. 1869, Jan. 2023, Art. no. 20210447.
[36] E. S. Ghith and F. A. A. Tolba, "Tuning PID controllers based on hybrid arithmetic optimization algorithm and artificial gorilla troop optimization for micro-robotics systems," *IEEE Access*, vol. 11, pp. 27138–27154, 2023.

ATA JAHANGIR MOSHAYEDI (Member, IEEE) received the Ph.D. degree in electronic science from Savitribai Phule Pune University, India. He is currently an Associate Professor with Jiangxi University of Science and Technology, China. He is a member of the editorial team of various conferences and published various articles in journals, two books published, and owns two patents. His research interests include robotics and automation/sensor modeling/bio-inspired robots, mobile robot olfaction/plume tracking, embedded systems/machine vision-based systems/virtual reality, and machine vision/artificial intelligence. He is a member of different scientific societies, such as IEEE, ACM, the Instrument Society of India, a Life Member, and a Lifetime Member of the Speed Society, India.

ATANU SHUVAM ROY received the bachelor's degree in computer science and engineering from Jiangxi University of Science and Technology. He is currently pursuing the M.Tech. degree in computer science and engineering with Indian Institute of Technology Kanpur. His research interests include embedded systems and the Internet of Things (IoT).



LIEFA LIAO received the B.E. degree in computer science and technology from Central South University, Changsha, China, in 1997, the M.E. degree in automatic control engineering from Jiangxi University of Science and Technology, Ganzhou, China, in 2003, and the Ph.D. degree in system management science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2011. He is currently the Dean and a Graduate Tutor with the School of Information Engineering, Jiangxi University of Science and Technology. His research interests include artificial intelligence and neural networks.

AMIR SOHAIL KHAN received the B.S. degree in computer science from Jiangxi University of Science and Technology, Ganzhou, Jiangxi, China. He is focusing on embedded systems, deep learning, and robotics in computer vision and sensor systems in order to facilitate solutions for smarter cities of the future.

AMIN KOLAHDOOZ is currently a Senior Lecturer in design with the School of Engineering and Sustainable Development, De Montfort University. His academic focus spans manufacturing materials processes, materials science, and the utilization of virtual reality (VR) and augmented reality (AR) in the manufacturing sector. Proficient in finite element modeling, robotics, and optimization methods, he leverages these tools to address intricate design and production challenges effectively. His significant involvement in multifaceted research projects underscores a steadfast commitment to innovation. He was honored with a fellow (FHEA) designation, acknowledging his adherence to the U.K. Professional Standards Framework for exceptional teaching and learning support within higher education.

ALI EFTEKHARI (Member, IEEE) received the M.Sc. and Ph.D. degrees in applied mechanical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2005 and 2011, respectively. He is currently a Researcher and a Faculty Member with the Mechanical Engineering Department, Islamic Azad University, Khomeinishahr Branch, Isfahan.