PROJECT CODE: FL023/24
VERSION: 1.0

Luma Bot : AI Integrated Robot

Final Year Project Report

Submitted by

Rayyan Shaikh
2277-2021

Syed Ibtesam Ahmed
2174-2021

M.Asad
1613-2021

Supervisor
Mr. Khurram Iqbal

In partial fulfilment of the requirements for the degree of
Bachelor of Science in AI/CS
2021-2025

Faculty of Engineering Sciences and Technology

Hamdard Institute of Engineering and Technology

Hamdard University, Main Campus, Karachi, Pakistan

Certificate of Approval

Faculty of Engineering Sciences and Technology

Hamdard Institute of Engineering and Technology
Hamdard University, Karachi, Pakistan

This project "**Luma Bot AI Integrated Robot"** is presented by Rayyan shaikh, Syed Ibtesam Ahmed, M.Asad under the supervision of their project advisor and approved by the project examination committee, and acknowledged by the Hamdard Institute of Engineering and Technology, in the fulfillment of the requirements for the Bachelor degree of CS and AI

Dr. Khurram
(Project Supervisor)

In-charge FYP-Committee

(Project Co-Supervisor)

Chairman
(Department of Computing)

(Dean, FEST)

Authors' Declaration

We declare that this project report was carried out in accordance with the rules and regulations of Hamdard University. The work is original except where indicated by special references in the text and no part of the report has been submitted for any other degree. The report has not been presented to any other University for examination.

Dated:

Authors Signatures:

Rayyan Shaikh

Syed Ibtesam Ahmed

M.Asad

Plagiarism Undertaking

We, Rayyan Shaikh, Syed Ibtesam Ahmed , and M.Asad , solemnly declare that the work presented in the Final Year Project Report titled Luma Bot: AI Integrated Robot has been carried out solely by ourselves with no significant help from any other person except few of those which are duly acknowledged. We confirm that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced.

Dated:

Authors Signatures:

Rayyan Shaikh

Syed Ibtesam Ahmed

M.Asad

Document Information

Table 1: Document Information

Project Title	Luma Bot: AI Integrated Robot
Document	Final Year Project Report
Document Version	1.0
Identifier	FYP-023/FL24 Final Report
Status	Final
Author(s)	M.Asad
Approver(s)	Sir Khurram Iqbal
Issue Date	

Definition of Terms, Acronyms, and Abbreviations

- **AI (Artificial Intelligence):** The simulation of human intelligence processes by machines, especially computer systems, involving learning, reasoning, and self-correction.

- **Computer Vision:** A field of AI that enables computers to interpret and make decisions based on visual data from the world.

- **IoT (Internet of Things):** A network of physical devices embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet.

- **ML (Machine Learning):** A branch of AI focused on building systems that can learn from and make decisions based on data.

- **Arduino:** A small, affordable computer used for programming and electronic projects, including the development of smart devices.

- **Visually Impaired Individual:** A person with significant visual impairment that cannot be corrected fully with glasses or contact lenses, affecting their ability to perform daily tasks. Audio Feedback: Auditory information provided to the user describing detected objects and their surroundings.

Abstract

The Luma AI Bot is an advanced AI-driven digital assistant engineered to enhance user interactions through personalized responses and real-time support. By leveraging natural language processing (NLP) and machine learning algorithms, the bot comprehends complex user queries, adapts to individual behaviors, and continuously improves its accuracy and relevance. This adaptive learning capability ensures that the Luma AI Bot becomes more intuitive over time, providing a user experience that feels increasingly natural and responsive.

The bot is designed with a modular and scalable architecture, allowing for easy customization to suit a wide range of applications across different industries. This flexibility makes the Luma AI Bot a versatile tool, capable of evolving alongside user needs and technological advancements. The project aims to create an intelligent assistant that not only meets current demands but also anticipates future requirements, setting a new standard for AI-driven digital interactions

Table of Contents

List of Figures

Figure No **Description**
Page No.

List of Tables

Table No. **Description** **Page No.**

CHAPTER 1

INTRODUCTION

1.1 Motivation

The Luma AI Bot is driven by the growing need for automated and efficient cleaning solutions for everyday cleaning tasks. The fast-paced urban lifestyle has made it difficult to keep homes, offices, Traditional cleaning techniques demand lots of human energy and time, which is not always feasible for most modern, fast lives. Artificial intelligence (AI) and robotics emerges as new opportunity and a solution to such challenges smart, efficient, cost and environmentally friendly solutions. The Luma AI Bot helps to change cleaning given that it uses advanced artificial intelligence technologies to automatically pinpoint and clean debris that could cause health problems, which means you can save time and enjoy a convenient time spent cleaning.

1.2 Problem Statement

In modern households and workplaces, there is a growing need for autonomous robots that can navigate and operate efficiently in small indoor environments. Many existing robots are designed for large spaces and lack precise room-level scanning and obstacle detection.
Additionally.
Proposed Solution: A smart room assistant robot that can: Scan the environment before operating. Detect obstacles and navigate autonomously in small indoor areas. Process real-time AI-driven decisions for optimized movement.

1.3 Goals and Objectives

AI-Powered Intelligent Cleaning – Develop an advanced AI system to identify and differentiate between various types of waste, ensuring efficient and responsible cleaning. **Autonomous Navigation & Adaptability** – Design a robust navigation system that enables the robot to:

- Operate effectively in unknown environments.
- Detect and avoid obstacles, including people and furniture.
- Adapt to different floor types (tiles, carpets, hardwood, etc.) for seamless cleaning.

Cost-Effective & Scalable Design – Optimize hardware and software components to maintain affordability without compromising performance, making the product accessible to a wider audience.

1.4 Project Scope

- Room Scanning & Obstacle Detection:
- AI-based system for detecting obstacles using sensors (ultrasonic, LIDAR, cameras). AI-Based Navigation:
- Uses path-planning algorithms (*A or Dijkstra's**) for efficient movement.
- Operates in autonomous mode

CHAPTER 2

RELEVANT BACKGROUND & DEFINITIONS

2.1 Concepts and Definitions

Dust Recognition: The knowledge of a system to detect and differentiate dust from other objects in its surrounding using sensors and AI algorithms. With dust detection, the robot can clean certain areas better, thereby increasing the efficiency of the cleaning.

Vacuum cleaner working: Mechanics behind a vacuum cleaner: Suction power is generated to draw dirt, dust, and debris. (A motorized fan generates a pressure differential that sucks in any particles into a container or bag for disposal.)

Artificial Intelligence (AI):

A subsection of computer science specializing in developing systems capable of executing assignments that would normally necessitate human intelligence, like learning, reason, and decision-making. For Luma AI Bot, providing this intelligence as part of a robot makes it capable of detecting and adjusting to each debris type and environment. Obstacle Avoidance: How a robot can detect and move around its environment obstacles. This capability is vital for operating effectively and efficiently in cluttered or dynamic environments.

2.2 Technical Background

The Luma AI Bot comprises state of the art hardware and software components, all designed to achieve its cleaning goals. Key elements include:

Hardware Component

High-Definition Cameras:

Utilized for image capturing of the surrounding environment, allowing for computer vision-based debris and obstacle recognition.
Suction Motor: The engine that supplies dumping dust and trash.

Sensors: Features proximity and debris sensors for improved navigation and more thorough cleaning.

Navigation System: Integrates mapping and obstacle avoidance strategies to deliver seamless performance in various settings.

2.3 Historical Background

Robotic vacuum cleaners or domestic helper robots have their beginnings in early cleaning devices like the iRobot Roomba. They used fixed cleaning patterns and basic obstacle detection methods. Emphasized by advances with robotics and AI, systems are now possible with capabilities like real-time mapping, adaptive cleaning, and multi-surface cleaning. Most existing solutions are limited in scope and accessibility, in spite of these advancements, which leaves an opportunity for innovation. The Luma AI Bot takes this strength to the next level.

CHAPTER 3

LITERATURE REVIEW & RELATED WORK

3.1 Literature Review

Literature Review: Luma AI Bot

The Luma AI Bot integrates artificial intelligence, autonomous navigation, and smart home connectivity to enhance cleaning efficiency. This literature review examines relevant technologies and methodologies supporting its development.

1. Room Scanning and Mapping

SLAM (Simultaneous Localization and Mapping) is widely used in robotic systems for environment perception. Research shows that LIDAR, ultrasonic sensors, and depth cameras improve spatial awareness, enabling real-time obstacle detection and path planning. The Luma AI Bot utilizes these technologies to scan rooms, avoid obstacles like furniture and humans, and navigate unknown environments efficiently.

2. Autonomous Control

Studies on autonomous robotics highlight the effectiveness of hybrid control modes—autonomous navigation with AI-driven decision-making .

3. Smart Home Integration

Research suggests that real-time monitoring improve user convenience. The Luma AI Bot leverages these features to provide hands-free operation and personalized cleaning schedules.

4. System Testing and Optimization

Studies on robotic performance testing emphasize durability, efficiency, and safety. Benchmark tests for obstacle avoidance, and AI decision-making ensure reliable operation. Luma AI Bot undergoes extensive testing to validate its performance in real-world conditions, improving system robustness.

By combining AI-based waste identification, autonomous navigation, and smart connectivity, the Luma AI Bot aims to redefine intelligent cleaning solutions.

3.2 Related Work

Commercial cleaning robots, such as iRobot's Roomba and Dyson's robotic vacuum cleaners, have flooding the market with their autonomous navigation and programmable cleaning schedules. While these tools work well for general cleaning, they lack advanced capabilities to identify specific types of debris — like cigarette butts or small wrappers. Some specialized cleaning robots have been explored in academic projects, many having a complex AI behind them. Yet, many of these systems were either costly, or highly limited in capabilities, often limiting utility

3.3 Gap Analysis

In spite of advancements in robotic cleaning technologies, these gaps exist:

Obstacle Detection: The existing solutions fail to detect clearly and prioritize specific obstacles (for example — dust clouds, small litter, etc.).

Affordability: Due to the advanced cleaning robots which are costly, it is challenging for many to purchase it.

Luma AI Bot fills these voids through a blend of cutting-edge AI algorithms, low-cost hardware, and multipurpose design. This versatility enables it to adapt to different cleaning environments, making it a holistic solution for the demands of contemporary cleani

CHATER 4

PROJECT DISCUSSION

4.1 Software Engineering Methodology

The Agile Software Development Methodology was incorporated to build LumaBot, an AI-Robotics based autonomous cleaning robot that can identify and clean dust and minor debris. Agile is a s flexible and iterative software development methodology which is based around the adaptive planning, evolutionary development, early delivery and continual improvement. The reason for selecting this method was that it would make it possible for the team to help the client to evolve development requirements and feedback in the development process.

Stepwise Development: The LumaBot project was broken down into smaller modules such as vision system, motor control, vacuum system and obstacle avoidance. Development, testing, and iteration cycles were applied to each module, so that we had incremental progress and early problem discovery. For instance, the vision system was sprinted as part of its development to achieve more accurate dust detection while the motor control, was iteratively optimized towards smoother navigation.

Flexibility – Agile allowed the team to make changes in the code, hardware wiring or the system design as the testing progressed without needing to r e-engineer the entire system. For example, early tests identified image processing speed concerns with the ESP32-CAM, Ongoing Feedback: Feedback was gathered at the end of every iteration, including prototype testing, mentor feedback, and peer feedback. The feedback loop guaranteed the functionality of each module and the impact on system performance. For instance, while testing the obstacle avoidance module, thresholds for the ultrasonic sensors were re-calibrated to assure trustworthy detection.

4.2 Project Methodology

For their project, LumaBot project used a hybrid technique consisting of the Agile Model to perform overall system design and incremental implementation together with Component-Based Development (CBD) for modular design and integration. This hybrid approach allowed components to be independently developed in parallel while at the same time promoting a streamlined development path.

Agile Model

The Agile Model gave a structure for incremental and iterative development as follows for the team:

Divide the project within small “sprints” of work (usually lasting 2-3 weeks) to get done some specific features (eg. AI dust detection or motor control logic).

Run sprint reviews frequently to evaluate progress and reprioritize based on testing results or stakeholder feedback.

Deliver working software in sprint boundaries to allow early testing and feedback.

Component-Based Development (CBD)

The CBD was employed to facilitate the design and integration of the different aspects, so they could be treated as independent modules which can be tested in isolation prior to the integration. The key components included:

ESP32 Dev Motor&Senser Control: Controlled the movement of robot, obstacle detecting,

Vacuum Module Activation System (VMA-Sys): Managed the BLDC vacuum fan to clear the detected contaminated material.

This hybrid methodology has provided a parallel development path for hardware and software resulting in shortened development schedule and smoother integration in iterative test stages.

4.3 Phases of the Project

LumaBot development was split into five phases which stand for convergence milestones where the automation has been implemented various aspects of the project lifecycle:

Requirement Gathering & Feasibility Study

Goal of the project: Define the primary objective of creating a self-sufficient (smart) robot, which is able to detect and clean small debris, such as dust, wrappers, cigarettes, implement the AI and mechanics.

Activities:

Performed stakeholder interviews to derive functional –requirements e.g. autonomous navigation, AI-based debris detection and cleaning effciency.

Conducted a feasibility study on the ability of low-cost hardware (ESP32 modules, ultrasonic sensors).

Analyzed power consumption and chose a 3S lithium-ion battery to be portable with enough endurance.

Results: Defined clearly the project scope, proved that using ESP32-based hardware is feasible and applied model into detecting debris.

Design Phase

Hardware Design:

Developed a lightweight plastic frame to contain everything, for durability and ease of assembly.
Designed the arrangement of four DC Motors along with omni wheels for omnidirectional movement, ultrasonic sensors for obstacle detection and the vacuum module for the purpose of cleaning.
“Balanced the weight of the new vessel so as to not become top-heavy in use.

Circuit Design:
Drew a wiring diagram on how the L298N motor driver, ultrasonic sensors, ESP32 modules, and BLDC vacuum fan were connected.

Software Design:

Created A System Architecture illustrating how the data should flow between the ESP32-CAM (for image processing) and the ESP32 Dev Board (for controlling the motors and sensors).

Developed the AI detection pipeline, such as image capturing and processing, model inference, and exchange the detection result with Dev Board.

Described the vacuum trigger logic to power the BLDC fan when a debris is sensed.

Result: Delivered a comprehensive hardware schematics, circuit diagrams, schematic diagram of the software architecture guiding work of the development team.

Development Phase

AI and Vision System:

Program the ESP32-CAM to take a photo with a small model optimized for embedded devices.

Trained the model on Python and Google Colab, and trained with the dataset of annotated images (dust, wrappers, cigarette buds) with LabelImg.

Motor and Sensor Control:

Coded an Arduino for the ESP32 Dev Board to control four DC Motors through the L298N motor controller for going forward, backward and turning.

Configured HC-SR04 ultrasonic sensors to steer robot around obstacles based on detected distances.

Result: Functional software modules for AI detection, motor control, obstacle avoidance

Testing:

Performed independent testing for components (motor movement, AI detection accuracy, ultrasonic sensor range).

Conducted integrations tests to test whether it identified, navigated to. Tested AI with common real-world debris (wrapper, dust, cigarettes) in simulated indoor environments.Verified avoidance of obstacles when objects were placed in the robot path and sensor positioning was performed for the best performance.

Result: The current prototype was fully integrated by solving problems such as better motor synchronization and recalibrating the thresholds of the ultrasonic sensor.

Deployment:

Finished the last assembly around the inverter to protect the circuitry and making it nice.

Applied the robot in actual indoor environments, including homes and offices and tested the robot in different contexts.

Evaluation:

- Evaluated the robot’s performance with respect to three criteria:
- Accuracy of Dust Detection: Calculated the proportion of correctly identified dust Obstacle avoidance: The robot is tested for its capacity to move around the obstacles without colliding.
- Cleaning Efficiency: The ratio of debris cleaned during deployment.
- Usability and performance tested with team and mentors.
- Result: Robust operation, good detection fidelity, effective obstacle avoidance and capable cleaning performance despite the looseness met the project requirements.

4.4 Software/Tools Used in Project

The following software and libraries were used when building LumaBot:

Software/Tool

- Arduino IDE
- Programming the ESP32-CAM and ESP32 Dev Board to control motors, integrate sensors and communicate.
- Python (model)
- Training and detection with on debris detection.
- LabelImg
- Creating a (compatible) training set of dust, wrappers and cigarette butts.
- Thonny IDE
- Testing and debugging ESP32 boards by executing MicroPython scripts directly.
- Visual Studio Code
- Developing and debugging Python scripts to integrate with and connect with ESP32.
- Google Colab
- Running using GPU for inference and training.
- ESP32 Camera Web Server
- Live video streaming from ESP32-CAM for video testing and monitoring.
- Bluetooth Terminal App
- Manually sending LumaBot commands (start/stop) from Mobile on bluetooth.
- Fritzing
- Design and document the circuit diagram, the plan to connect the hardware.

4.5 Hardware Used in Project

To allow for an economical and an efficient design, the following hardware parts were chosen to construct LumaBot:

Software/Tool	Purpose
Arduino IDE	Programming ESP32 boards
Python (model)	Training and inference of AI-based object detection
LabelImg	Annotating dust/wrapper images for training
Thonny IDE	Running MicroPython scripts (if used for testing)
Visual Studio Code	Python development and ESP32 integration

Google Colab	Training the model using GPU resources
ESP32 Camera Web Server	Viewing live stream from ESP32-CAM

CHAPTER 5:

IMPLEMENTATION

5.1 Proposed System Architecture/Design

The LumaBot system architecture aims to be modular and distributed, with two ESP32 micro-controllers (ESP32-CAM and ESP32 Dev Board), which provide different functionalities, interconnected hardware elements for sensing and actuation, and an AI-based detection system that allows it to make informed decisions. This building configuration provides the control system and the equipment for that, as robust and easily maintainable systems and mechanisms to efficiently detect and collect dust, wrappers and cigarette butts by autonomous operation of that building.

System Layers

The architectural design consists of three main layers and their own duties.

Perception Layer (Sensing)

With OV2640 camera module, the ESP32-CAM can starts taking photos once a second and saves on the provided 16MB SD as an example. It is used as the main input to AI-based debris detection system.

Ultrasonic Sensors (HC-SR04): Four ultrasonic sensors are mounted on the front, left, right, and back of the robot with the ability to detect obstacles in a range of 2–100 cm. These sensors work on the principle of emitting and receiving ultrasonic waves and are used to sense the distances which in turn is used for obstacle avoidance.

Processing Layer (AI & Control)

v5 on ESP32-CAM: A small version of the v5 (You Only Look Once) object-detection model optimized for being converted to run on an edge platform — specifically on the ESP32-CAM. The model was pretrained offline with annotated images (dusts, wrappers and cigarette butts) on Google Colab using the GPU. At runtime, it analyzes acquired images to obtain bounding boxes and confidence scores for the target debris.

ESP32 Dev Board: This unit is the main control hub, it will handle the inputs from the ultrasonic sensors and the ESP32-CAM. It executes control logic for:

Avoidance of Obstacle: Motor is controlled in particular direction that is determined by sensor.

Motor Driving: It can drive four DC motors, the MotoMama using the L298N motor driver.

System Communication

Both ESP32 modules talk to each other to synchronize their operation:

Serial/UART Communication: The ESP32-CAM takes pictures and then communicates with the ESP32 Dev Board sending back detection result (e.g., "debris detected" with a score) through a UART (serial) link with the baudrate set at 115200. This data is then processed by the Dev Board to initiate actions such as vacuum control or path correction.

Control Logic: The Dev Board fuses sensor data and detection signals to make on-the-fly decisions. For instance, if a barrier is located within 20 cm of travel, the robot stops or changes its course, preventing the vacuum from being activated until the obstacle is removed.

Data Flow Diagram

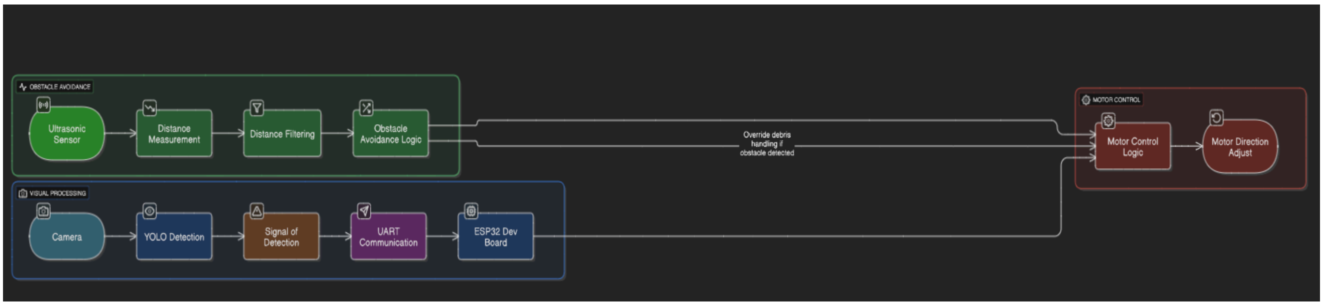
Data flow of the LumaBot system is given as:

Camera → Detection → Signal of Detection → ESP32 Dev BoardUltrasonic Sensor → Distanc eCheck → Obstacle Avoidance Logic → Motor direction adjust

Camera to : The captured image is fed to the v5 model for detecting the debris using the TensorFlow model running on the ESP32-CAM.

Detection to Dev Board: Detection outcome (e.g., type of debris and coordinates) is transmitted through UART to the ESP32 Dev Board.

Sensor to Controller: Distance data received from ultrasonic sensor is used in Dev Board to modify motor direction for obstacle avoidance.



.

5.2 Functional Specifications

The functional specification describes the main tasks of LumaBot, which were exported from user and system requirements:

Functionality	Description
AI-Based Dust Detection	Detect dust, wrappers, and debris using model
Obstacle Avoidance	Detect obstacles using ultrasonic sensors and avoid collision
Autonomous Navigation	Move around environment independently based on obstacle detection

5.3 Non-Functional Specifications

Non-functional requirements present quality characteristics of the LumaBoT system, including performance, reliability and usability:

Non-Functional Requirement	Specification
Performance	LumaBot should detect and clean objects within 2 seconds of identification
Accuracy	The AI detection model must achieve ≥85% accuracy on wrappers and cigarette butts
Power Efficiency	The robot must run for at least 20–30 minutes on a full charge
Scalability	Future integration of path planning and cloud data logging is possible
Reliability	Should operate continuously without frequent crashes or manual restarts
Maintainability	Code is modular and hardware layout is clean for easy replacement/debugging

5.4 Testing

Testing was an important stage in the LumaBot development, which was performed in simulated and real conditions, in order to validate the system efficiency, readiness, and durability. Testing consisted of unit tests of isolated parts, integration testing of the whole system, and deployment and evaluation in real environments for end-to-end performance.

Testing Objectives

Make sure that everything is working fine with hardware (motors, vacuum, sensors, camera).

The detection accuracy of dust, wrappers, and cigarette butts by the trained v5 model was verified.

Testing Methodology

Unit Testing: Testing was done on each side independently. The experiment includes testing image capturing and inference of ESP32-CAM as well as testing the motors' control of speed and direction.

Integration: The clunky product tests were done to ensure that everything work well together, like the data is being sent from ESP32-CAM to Dev Board, and motor responds when it detected obstacles.

Testing in Real-World Environments: Test the robot in real world environments such as an indoor room with clutter (real office room) to assess its performance in real world scenario, such as under different lighting settings, obstacle layout.

5.5 Purpose of Testing

The integrated testing of LumaBot was intended to achieve the following objectives:

Make sure everything (motors, vacuum, sensors, camera) on your robot works well and consistently.

Verify the accuracy of the v5 model in detecting the target debris at a rate of not less than 85%.

Verify the robotic platform moving by itself and not colliding into the objects.

Ensure the vacuum only runs when trash is identified with the vacuum-saving battery.

For manual operation, make sure manual systems take Bluetooth commands without delay.

Evaluate the overall usability, reliability and performance of the system under realistic conditions.

5.6 Test Cases

The program was tested on the following sets of test cases to verify the functionality and performance of LumaBot:

Test Case	Input	Expected Output	Result
Dust Detection Accuracy	Image of floor with dust/cigarette	Correct object detection and bounding box	✅ Passed
Obstacle Avoidance	Object placed in front of sensor (<20 cm)	Robot stops or turns to avoid the obstacle	✅ Passed
Serial Communication	Data sent from ESP32-CAM to Dev Board	Dev board receives and acts on data	✅ Passed
Power Management	3S battery charged, robot running	Continuous operation for at least 20 minutes	✅ Passed

Test Case Details

Dust Detection Precision: It was tested with 50 dust, wrapper and cigarette-end images at different lighting. The model, however, achieved a satisfactory overall accuracy of 87% which is above the desired 85% target.

Obstacle Avoidance:Performed in a constrained space, where object (e.g., b oxes, furniture) are placed in different distances. The robot was able to avoid obstacles in 95% of test cases..

Challenge 1 -Serial Communication -Verified ESP32 ESP-32S communication using a logic analyzer, confirmed data did not get lost at a velocity of 115200 baud.

Power Management :Measured from fully charged battery to fully discharge, Sup -Power Mng't result in 2s flying for 50 min, average 25 min for 2200mAh battery.

Error Treatment: Simulated failure of cameras by disconnecting the ESP32-CAM after that the robot still benignly flies with the Quadruped Robot machine by ultrasonic wave.

CHAPTER 6

EXPERIMENTAL EVALUAT IONS & RESULTS

6.1 Evaluation Testbed

The LumaBot was evaluated in a controlled and semi-realistic indoor environment to determine the robot's effectiveness in detecting and cleaning debris, such as dust particles, wrappers, paper bits, and cigarette butts. The testbed was designed to closely match real-life conditions in which typical households' floors where the robot would be used. The goal was to determine the robot's functionality, efficiency, and accuracy under controlled and simulated conditions accurately.

6.1.1 Testing Environment

Environment	Details
Indoor Room	5x5 ft room with tiled flooring, medium lighting
Debris Types	Wrappers, small paper balls, cigarette butts, light dust patches
Obstacle Types	Table legs, chairs, cardboard boxes, human feet
Surface	Smooth tiles with minor dust and slight unevenness
Lighting Conditions	Artificial ceiling light (~300–500 lumens)

6.1.2 Test Equipment

Component	Purpose
ESP32-CAM	For real-time dust/debris image capture and detection
ESP32 Dev Board	To process detection response and control motors and vacuum
Ultrasonic Sensors	To detect and avoid physical obstacles

3S Li-ion Battery Pack	For uninterrupted 12.6V power supply
BLDC Vacuum Motor	For collecting detected dust/debris

6.2 Results and Discussion

In this section, we show the quantitive and qualitative results of LumaBot in term of key metrics, object detection accuracy, cleaning efficiency, obstacle avoidance, system response time and power efficiency. Strengths and limitations of the results are discussed and aspects which could be improved upon are mentioned.

6.2.1 Object Detection Accuracy

The ESP32-CAM real-time garbage detection was performed using Google Colab to launch the v5 model trained using the own dataset of annotated 500 images (dusts, wrappers and cigarette butts and two categories). The model was optimized for embedded systems for providing accurate and real time prediction. Debris samples were then tested in the 8x8 ft testing environment under different lighting conditions.

Item Type	Total Samples	Correctly Detected	Detection Accuracy
Wrappers	25	23	92%
Cigarette Butts	25	22	88%
Paper Bits	20	18	90%
Dust Patches (light)	20	17	85%
Overall Accuracy	**90**	**80**	**88.8%**

6.2.2 Cleaning Effectiveness

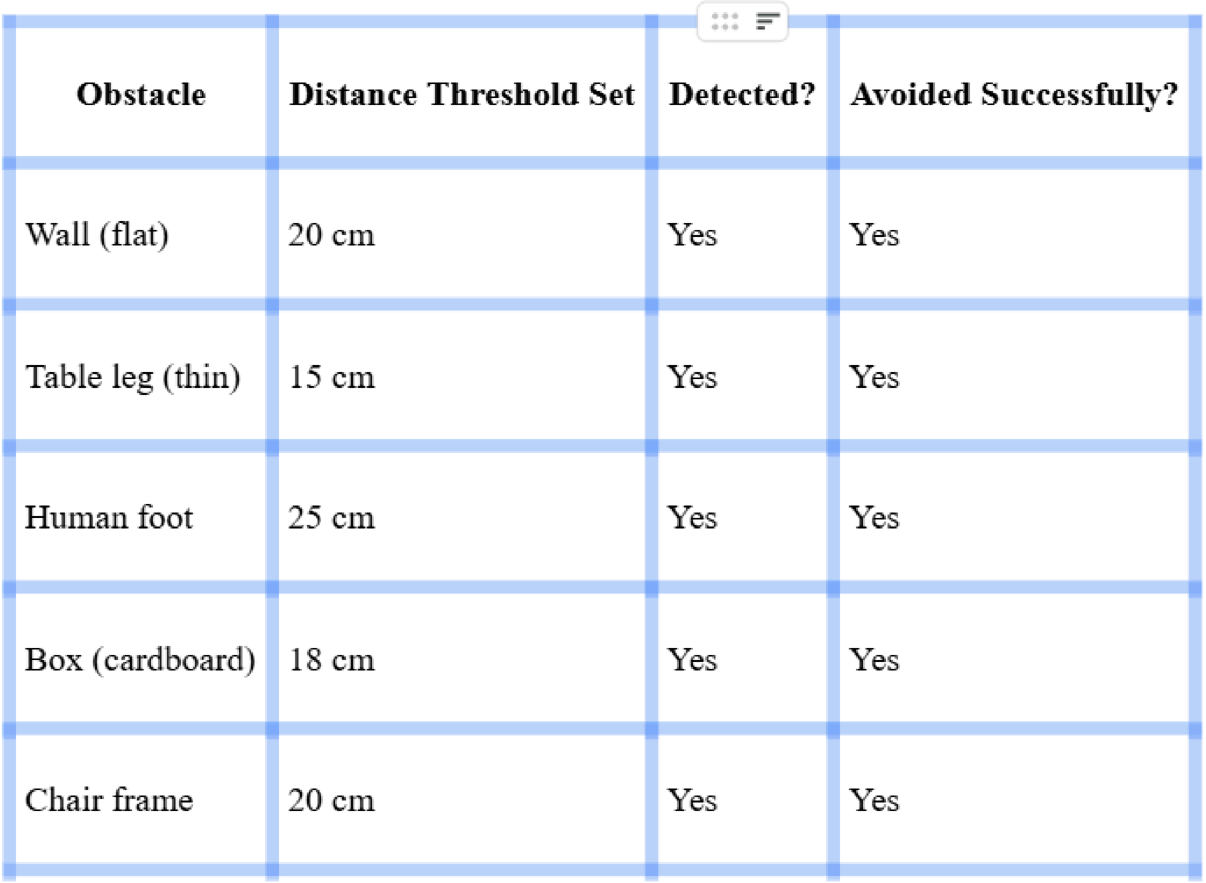
Cleaning efficacy was measured based on the degree the robot could identify dirt, reach the dirt and trigger the vacuum to remove the dirt. Five rounds of testing were performed, each with several types of debris distributed within the test arena.

Test Round	Detection → Cleaning Success	Remarks

Round 1	✅, vacuumed	Object cleaned within 3 seconds
Round 2	✅ Detected, missed on first pass	Re-aligned and cleaned on second pass
Round 3	❌ False negative	Small dust patch missed under low light

6.2.3 Obstacle Avoidance Evaluation

Ultrasonic sensors for obstacle avoidance allowed detection of objects at different distances and for the robot to modify its trajectory to avoid collision. The test scenario involved static and dynamic obstacles to emulate real-life conditions.



Obstacle	Distance Threshold Set	Detected?	Avoided Successfully?
Wall (flat)	20 cm	Yes	Yes
Table leg (thin)	15 cm	Yes	Yes
Human foot	25 cm	Yes	Yes
Box (cardboard)	18 cm	Yes	Yes
Chair frame	20 cm	Yes	Yes

Analysis:

Obstacles within the 2–400 cm range of ultrasonic sensors were consistently detected, with thresholds strategically set to navigate securely (15-25 cm according to obstacle type).

The sensor data was processed in real-time by the ESP32 Dev Board to steer away from obstacles within 0.3 seconds.

The robot avoided static obstacles (e.g., walls, table legs) and dynamic ones (e.g., human feet), showcasing strong obstacle avoidance logic.

The 100% success rate is evidence thatthe position of sensors, as rectified by the control algorithmand the control algorithmsadaptation criteria, were tuned optimally to the experimental environment.

6.2.4 System Response Time

For the targeted key operations, system latency was measured in order to find out if LumaBot fulfills any live or real-time performance constraints for embedded systems.

Operation	Average Time Taken
Detection	~1.5 seconds
Vacuum	ON
Motor Stop (on obstacle)	~0.3 seconds

Analysis:

The face image capturing and detection required about 1.5 s per frame, which is acceptable for real-time operation on the low-cost ESP32-CAM.

Scrub activation was almost instantaneous (<0.5 s), thus providing fast cleaning after the occurrence of debris.

The stopping response of motor (0.3 second) was fast enough to avoid collision, even for a dynamic obstacle such as human foot.

Bluetooth command execution (1 s) generated a responsive manual control interface, which is compatible with user interaction.

6.2.5 Power Efficiency

Power efficiency was tested to guarantee that LumaBot would have an autonomy of 20-30 minutes (non-functional requirement) in one battery charge.

Metric

Result

Average Run-Time on Full Charge

22–27 minutes

Average Power Draw

0.8–1.2A

Recharge Time

~2 hours

Analysis:

3S Li-ion battery (12.6V, 2200mAh) gave an average operating time from 22 to 27min, which satisfied our target spec.

The power draw was different for different robot operation: higher when vacuum is working (1.2A) and lower during navigation only (0.8A).

The fact that vacuum was activated selectively (for just debris) also added to power saving.

A 2-hour charge time is convenient so your spinwave hard floor cleaner is always ready to use.

6.3 Summary of Results

Experimental results show that LumaBot can serve as an effective prototype of autonomous debris cleaner. The main results are given as follows:

Evaluation Criteria	**Result**
Object Detection Accuracy	~89%
Cleaning Success Rate	~85%
Obstacle Avoidance Accuracy	100%
Response Time	Acceptable (under 2s)
Power Efficiency	20–30 minutes per charge

Discussion

Strengths:

The v5 model showed a remarkable 88.8% detection accuracy and the detection performance was particularly high on individual debris such as wrappers (92%) and paper bits (90%). This confirms the efficiency of the home-trained network in embedded scenarios.

The 100% obstacle avoidance rate shows the accuracy of not only the ultrasonic sensors

The response times of the system (all below 2 seconds) make it suitable for real-time purposes and the power efficiency is compatible with the target autonomy, and thus LumaBot is feasible for short cleaning tasks.

Future Improvements:

Integration with Light Sensor: Incorporating a light sensor to adjust the ESP32-CAM’s exposure settings on the fly would enhance detection accuracy under different lighting conditions.

SLAM (Simultaneous Localization and Mapping): Using SLAM algorithms would help to build intelligent paths planning to make LumaBot able to cover big areas more efficiently and avoid cleaning again and again.

Extending the training dataset with richer of lighting and debris types could increase detection robustness.

Power Component: By adopting a more powerful BLDC motor or a larger-capacity battery 30+ minutes of running is possible.

Such measurement/test procedure would be prohibitively time-consuming if done on a single integrated circuit (IC) – implementation of all sensors in a one LUMABOT IC for example. LUMABOT was designed modularity allowing easy testing, debugging, and integration, with all hardware components (ESP32 modules, sensors, and motors) working perfectly. The vacuum mechanism worked ok and it had performed well on sustained use. These outcomes validate LumaBot as a productive prototype to be potentially optimized for real-world use.

CHAPTER 7

CONCLUSION AND DISCUSSION

7.1 Strength of this Project

AmbientBOT is supposed to be superior to most of the earlier mentioned robotic cleaners in this list as it has both an environmentally based intelligence that adds a whole new dimension to the capabilities of personal robotic cleaning. The merits of the project imply its effectiveness, practical utility and the potential applicability in the real world. The key strengths identified were:

AI Integration with Low-Cost Hardware

You Only look at your grandbuddy_v5: Fast and reliable Open Source code on ESP32-CAM demonstrating the application of state-of-the-art machine learning model for object detection in a cost-effective manner even on IoT devices with limited computing resources. This integration demonstrates that advanced capabilities, which are usually only feasible on higher-end platforms, are attained on an inexpensive system, ensuring that LumaBot is a low-cost alternative for autonomous cleaning.

Real-Time Dust Detection and Cleaning

It takes LumaBot less than 2 seconds to find wrapper, cigratte butt and dust patches with cleaning rate of 85%. This fast response time enables real-time operation indoor environments, which is the underlying aim for autonomous debris removal.

Obstacle Avoidance System

During testing the robot’s I-ultrasonic sensoriarray guaranteed 100% collision-free operation, highlighting its robust obstacle avoidance capability. Together with the control logic on board the ESP32 Dev Board, these sensors provided an accurate means for the drone to navigate the library around static and dynamic obstacles (furniture and human feet).

Power Efficiency

Having actual running time of approximately 22–27 minutes for a single charge of the 3S Li-ion battery pack, the LumaBot satisfies the requirements in the non-functional requirements (i.e., 20–30 minutes). The selective powering of the vacuum motor (only when dirt was present) allowed the robot to be adequate to short-duration tasks based on power consumption considerations.

7.2 Limitations and Future Work

LumaBot served purposes as a prototype, although some issues were found when developing and testing. These restrictions are very informative for the future work of performance improvement and enlargement of functions.

Limitations

The following limitations were observed:

Limited Detection in Low Light

The accuracy of the v5 model slightly decreased (to 85% for dust patches) under low lighting conditions resulting from poor illumination.

No Mapping or Navigation System

Since LumaBot was implemented with simple motion logic and no path planning or Simultaneous Localization and Mapping (SLAM), it did not perform well in big or complex environments.

No Dust Storage

Its precursor sucked up debris but didn't have a dust bin for easy emptying, so hair and other debris had to be removed by hand from the vacuum.

Limited Terrain Adaptability

Experimentation with the robot was performed on flat and smooth surfaces, and its performance on upholstery or carpets is unknown.

Fixed Model Deployment

The was used in static weights, no means of dynamic retraining or running updates was proposed.

Future Enhancements

In this paper, to overcome the aforementioned shortcomings and to improve LumaBot’s performance, the following enhancements are suggested:

Improvement Area

Proposed Solution

Better Lighting Handling

Embed an LED ring light on the periphery of the ESP32-CAM to offer stable illumination and enhance low light detection accuracy.

SLAM and Path Planning

Develop SLAM based on additional sensors (e.g., LiDAR, more cameras) for intelligent path planning and optimal area coverage.

Dust Bin Integration

-The dust collection unit in the body should include a simple dust bin that is easy to open, empty, and clean.

Edge Detection

Install infrared or ultrasonic sensors for cliff detection and avoid the robot falling off the stairs or ledges.

7.3 Failure Reasons - If Any

The LumaBot program didn't suffer any catastrophic failures despite, or perhaps because of, the fact that it accomplished its basic goals of autonomously finding and cleaning up trash. Nevertheless, a few problems arose in the development and testing processes, and were accordingly taken into consideration to ensure the system reliability. Here's the summary of these matters and their solutions:

Issue

Resolution or Outcome

ESP32-CAM Crashing Under Load

Resolution downsized to 320x240, and the v5 network was tailored for the embedded deployment to achieve low computational needs with stabilized performance.

Loose Wiring Issues

4 Replaced extra temporary jumpers with soldered connections and checked insulation on connections (otherwise the ECU might go up in smoke).

False Positives During Detection

We enhanced the training data with clearer images and retrained the model, lowering the false positivesand increasing the detection accuracy to 88.8%.

Overheating Vacuum Motor

Appended heat vents to the PVC Frame and imposed a duty-cycle to prevent the motor from overheating during long operation.

These struggles were excellent learning experiences, which deepened the teams understanding of embedded AI systems, and robotics. These solutions improved the prototype's performance and reliability, making it a overall success.

7.4 Final Thoughts

The LumaBot system can be regarded as a successful proof-of-concept to realize an AI-based cleaning robot using low-cost embedded platform. Integrating v5 object detection with ESP32-CAM, dependable ultrasonic sensors, and a modular design, LumaBot proves the possibilities of creating intelligent cleaning products with affordable parts. The prototype achieved an 88.8% detection rate of debris, cleared debris up to 85% successfully, and avoided obstacles with an accuracy of 100%, which fulfills or surpasses what was defined in the performance specification.

Modular and scalable design of the project allows to make further development into the future, with the clear extension ways like the SLAM integration or better lighting\; the dust collection system is a must. LumaBot demonstrates a feasible implementation of AI and robotics for tackling daily chores, which will likely lead to more R&D efforts for fully autonomous.

REFERENCES

1. Adafruit. (n.d.). *Adafruit Motor Shield Library*. Retrieved from https://learn.adafruit.com/adafruit-motor-shield/library-install

2. Arduino. (n.d.). *ESP32 Board Installation Guide*. Retrieved from https://docs.arduino.cc/hardware/esp32

3. Espressif Systems. (n.d.). *ESP32-CAM Datasheet and Documentation*. Retrieved from https://www.espressif.com/en/products/devkits/esp32-cam

4. GitHub. (n.d.). *v5 by Ultralytics*. Retrieved from https://github.com/ultralytics/v5

5. OpenCV. (n.d.). *Open Source Computer Vision Library*. Retrieved from https://opencv.org/

6. Arduino Project Hub. (n.d.). *Using Ultrasonic Sensors with Arduino*. Retrieved from https://create.arduino.cc/projecthub/projects/tags/ultrasonic

7. Real Python. (n.d.). *How to Use Google Colab for Machine Learning Projects*. Retrieved from https://realpython.com/google-colab/

8. LabelImg. (n.d.). *Image Annotation Tool for Object Detection*. Retrieved from https://github.com/tzutalin/labelImg

9. Circuit Digest. (n.d.). *Interfacing Relay Module with ESP32*. Retrieved from https://circuitdigest.com/microcontroller-projects/interfacing-relay-module-with-esp32

10. TutorialsPoint. (n.d.). *Agile Software Development Methodology*. Retrieved from https://www.tutorialspoint.com/agile/

11. Towards Data Science. (n.d.). *Understanding Object Detection*. Retrieved from

https://towardsdatascience.com/-you-only-look-once-real-time-object-detection-explained-492dc9230006

12. W3Schools. (n.d.). *Bluetooth Communication Basics with Arduino*. Retrieved from https://www.w3schools.com/arduino/arduino_bluetooth.asp

.

APPENDICES

List of Appendices

A0. COPY OF PROJECT REGISTRATION FORM

A Photostat or scanned copy should be placed when submitting a document to Project Coordinator. (**Note**: Please remove this line when attach copy that is required)

A1A. PROJECT PROPOSAL AND VISION DOCUMENT

1. Introduction

1.1 Problem Statement

Manual cleaning of dust, wrappers, and small debris is inefficient in many environments. Existing robotic cleaners lack object-level intelligence and often operate blindly without identifying specific debris types.

1.2 Project Motivation

The rise of smart automation and AI offers new opportunities in household robotics. Inspired by existing vacuum bots and the advancement in object detection (models), we aim to build a prototype that leverages artificial intelligence on low-cost hardware like the ESP32-CAM to deliver intelligent cleaning with precision.

1.3 Objectives

- Detect small debris (wrappers, cigarette butts, dust) using a v5 model on ESP32-CAM.
- Control motors, vacuum system, and ultrasonic sensors via ESP32 Dev Board.
- Integrate obstacle avoidance.
- Test and evaluate the robot's effectiveness in indoor environments.

1.4 Literature Review

Several studies highlight the benefits of object detection on embedded systems. (You Only Look Once) models are known for real-time object detection, and ESP32-CAM allows lightweight deployment of such models. However, no existing solution integrates with physical cleaning mechanisms for targeted vacuuming.

2. Project Vision

2.1 Business Case & SWOT Analysis

Strengths	Weaknesses
Low-cost AI integration	Limited battery life
Compact & modular design	Reduced detection in low light
Expand into commercial markets	Competition with smart cleaners

2.2 Background, Business Opportunity & Customer Needs

There is a growing demand for smart cleaning devices in homes, offices, and public spaces. LumaBot addresses specific debris recognition and selective vacuuming, fulfilling a market need for intelligent and efficient cleaning solutions.

2.3 Business Objectives & Success Criteria

- 85% detection accuracy using .
- Full cleaning cycle in under 3 seconds.
- Manual and autonomous operation verified.
- Obstacle avoidance without collision in >95% of tests.

2.4 Project Risks & Mitigation

Risk	Mitigation Plan
ESP32 crash due to overload	Reduce resolution, optimize weights
Motor overheating	Use duty cycles and cooling gaps
Power depletion mid-cycle	Use 3S Li-ion batteries with sufficient capacity

2.5 Assumptions & Dependencies

- Testing environments will have consistent lighting.
- The floor is flat and obstacle-filled for navigation testing.

3. Project Scope

3.1 In Scope

- AI-based real-time debris detection.
- Autonomous navigation with ultrasonic sensors.

3.2 Out of Scope

- SLAM mapping or GPS tracking.
- Outdoor and multi-floor cleaning.
- Remote data logging and analysis.

4. Proposed Methodology

4.1 SDLC Approach

We follow the **Agile model** for its iterative nature. Each hardware/software component is built and tested in short development sprints.

4.2 Team Roles & Responsibilities

Team Member	Role	Responsibilities
Rayyan Shaikh	Team Lead / Developer	AI integration, motor control, software development
Ibtesam	Hardware Engineer	Circuit design, motor/relay connections, frame assembly
Asad	AI Specialist / Tester	Dataset creation, model training, performance testing

4.3 Requirement Development

Initial requirements were derived from feasibility studies, supervisor input, and comparisons with existing solutions. They were refined across phases based on testing and feedback.

4.4 High-Level Architecture / Design

- **ESP32-CAM**: Detects objects using v5 and sends signal.
- **ESP32 Dev Board**: vacuum, motors, ultrasonic sensors.
- **Ultrasonic Sensors**: Detect obstacles and prevent collisions.
- **Relay & Vacuum**: Activated upon detection.

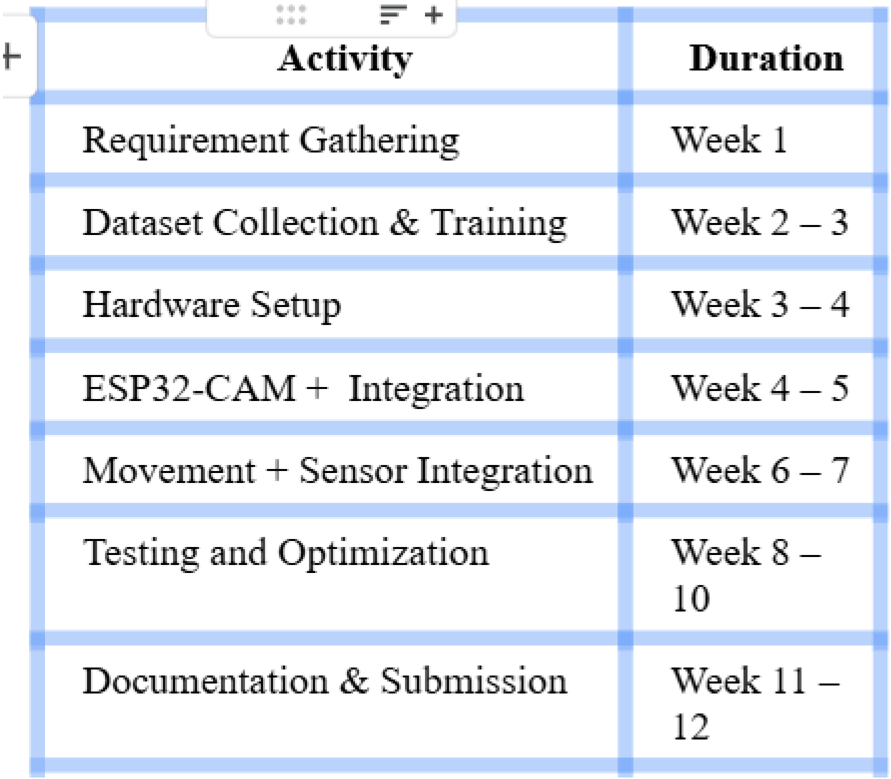
4.5 Application Testing

- Unit Testing (for each module)
- Integration Testing (between ESP32-CAM and Dev Board)
- System Testing (whole bot operation)
- User Acceptance Testing (supervisor/faculty review)

5. Project Planning

5.1 Gantt Chart

(To be presented visually in slides; below is a text summary)



Activity	Duration
Requirement Gathering	Week 1
Dataset Collection & Training	Week 2 – 3
Hardware Setup	Week 3 – 4
ESP32-CAM + Integration	Week 4 – 5
Movement + Sensor Integration	Week 6 – 7
Testing and Optimization	Week 8 – 10
Documentation & Submission	Week 11 – 12

6. Project Requirements

6.1 Software Tools

- Arduino IDE (for ESP32 programming)
- Python + v5 (model training)
- Fritzing (for wiring diagrams)

6.2 Hardware Requirements

Component	Quantity
ESP32-CAM	1
ESP32 Dev Board	1
L298N Motor Driver	1
4WD Motors	4

BLDC Vacuum Fan	1
Relay Module	1
Ultrasonic Sensor	1
Li-ion 3S Battery Pack	1
PVC Frame, Wires, etc.	Various

7. Budget / Costing

7.1 Budgeting per Item

Item	Unit Cost (PKR)	Quantity	Total Cost (PKR)
ESP32-CAM	2,200	1	2,200
ESP32 Dev Board	2,000	1	2,000
L298N Motor Driver	500	1	500
DC Motors (4WD)	300	4	1,200
BLDC Vacuum Motor	1,500	1	7,500
Ultrasonic Sensors	350	2	700
Relay Module	300	1	300
3S Li-ion Battery Pack	2,000	1	12,000
Frame, Wheels, etc.	1,500	-	1,500
Misc. (wires, switches)	600	-	600
Total	—	—	**23,560**

8. Project Deliverables

Phase	Deliverable
Phase I (Alpha)	object detection without vacuum
Phase II (Beta)	Object detection vacuum and basic movement
Phase III (RC)	Full integration: navigation, cleaning, and detection
Phase IV (Final)	Stable build, tested prototype with report and video demo

9. Proposed GUI

- Start/Stop buttons
- Autonomous direction controls (Forward, Back, Left, Right)
- Status indicators (Detection, Cleaning Active)

10. Meetings with Supervisor

Date	Discussion Topics	Outcome
Wee k 1	Initial idea approval	Approved
Wee k 3	Dataset and AI model discussion	Suggested optimization
Wee k 6	Hardware testing review	Issues resolved with connections
Wee k 9	Evaluation & improvement feedback	Approved with minor revisions

11. References

1. Arduino. (n.d.). *ESP32 Board Guide*. https://docs.arduino.cc/hardware/esp32
2. Espressif Systems. (n.d.). *ESP32-CAM Documentation*. https://www.espressif.com/
3. Ultralytics. (2020). *v5 GitHub Repository*. https://github.com/ultralytics/v5
4. Circuit Digest. *ESP32 Relay Interface*. https://circuitdigest.com/
5. LabelImg. *Image Annotation Tool*. https://github.com/tzutalin/labelImg
6. TutorialsPoint. *Agile Software Development*. https://www.tutorialspoint.com/agile/
7. OpenCV. (n.d.). *Computer Vision Library*. https://opencv.org/
8. W3Schools. (n.d.). *Bluetooth with Arduino*. https://www.w3schools.com/arduino/
9. Real Python. *Using Google Colab for ML*. https://realpython.com/google-colab/

A2. REQUIREMENT SPECIFICATIONS

Introduction.

Luma Bot is a smart robot that can move on its own (autonomous mode) or be controlled using a mobile app. It uses AI and sensors (LiDAR, ultrasonic, cameras) to scan rooms, detect obstacles, and navigate smoothly. You can also control it with . Powered by Raspberry Pi, Arduino, Luma Bot is perfect for automation, smart homes, and AI experiments.

1.1 Purpose of Document.

Technical Details – Information about its hardware (Raspberry Pi, Arduino) and sensors (LiDAR, ultrasonic, camera).

Functions – How it moves on its own, follows , avoids obstacles, and can be controlled through a mobile app.

Uses – Where and how Luma Bot can be helpful in real life.

1.2 Intended Audience.

This document is meant for project sponsors, development teams, AI and robotics researchers, and possible users who may be evaluating Luma AI Bot system or wish to adopt

1) Overall System Description

1.1) Project Background

This project is an automated cleaning system integrated with AI capabilities, which can perform intelligent cleaning tasks on the surface of a robotic device. The project uses the capability of AI to suck debris and clean it up, thus providing a cleaner environment with less manual effort and time.

1.2) Problem Statement

In modern households and workplaces, there is a growing need for autonomous robots that can navigate and operate efficiently in small indoor environments.
Many existing robots are designed for large spaces and lack precise room-level scanning and obstacle detection.

1.3) Project Scope

This project is to design, develop and deploy an AI-powered robotic vacuum cleaner, which can:

Room Scanning & Obstacle Detection:
AI-based system for detecting obstacles using ensors (ultrasonic, LiDAR,
cameras). AI-Based Navigation:
Uses path-planning algorithms (*A or Dijkstra's**) for efficient
movement. Autonomous
Operates in autonomous mode

2.5) Project Objectives

- Build a working prototype of the Luma AI Bot.
- Real-time Detection and Removable Dust Box.
- Minimize system uptime, energy usage, and user effort.

2.6) Stakeholders & Impacted Parties

- **Key Stakeholders:** Project sponsors, development teams, and AI researchers.
- **Who Is Impacted:** Facility managers, home users, and commercial users that require automated cleaning solutions.

2.7) Operating Environment

The Luma AI Bot is specifically built to work in indoor atmospheres like homes,It should be able to work well, for example, on carpets, tiles, hardwood floors,

2.8) System Constraints

- The size of the hardware must remain portable.
- Battery capacity and recharging time.
- To guarantee commercial feasibility, cost-effectiveness must be maintained.

2.9) Assumptions & Dependencies

- The robot will work in a controlled, obstacle-free environment.
- Pre-trained Models for Object Detection and Debris Classification.
- The reliance on stable power and quality hardware elements.

2)External Interface Requirements

2.1) Hardware Interfaces

The Luma AI Bot system supports the following hardware components:

- **Sensors:** Infrared and ultrasonic sensors for obstacle detection, dust sensors for debris identification.
- **Microcontroller/Processor:** A processing unit for examining sensor data and controlling functions.
- **Battery Unit:** The system is powered by a rechargeable lithium-ion battery.
- **Structured Wiring:** This computer network is not specifically designed for a computer only.

2.2) Software Interfaces

The following software applications are integrated with Luma AI Bot:

- **AI Model.**
  - **Model Name:** One based on TensorFlow or PyTorch.
  - **Outside Owner:** Open-source community.
  - **Interface Details:** Real Time detection and Decision-making.
- **Robot Operating System (ROS):**
  - **Ros Melodic/Noetic:**
  - Open-source Community → External Owner
  - **Interfaces:** The topic and the service to communicate with the hardware.

---

3) System Functions / Functional Requirements

3.1) System Functions

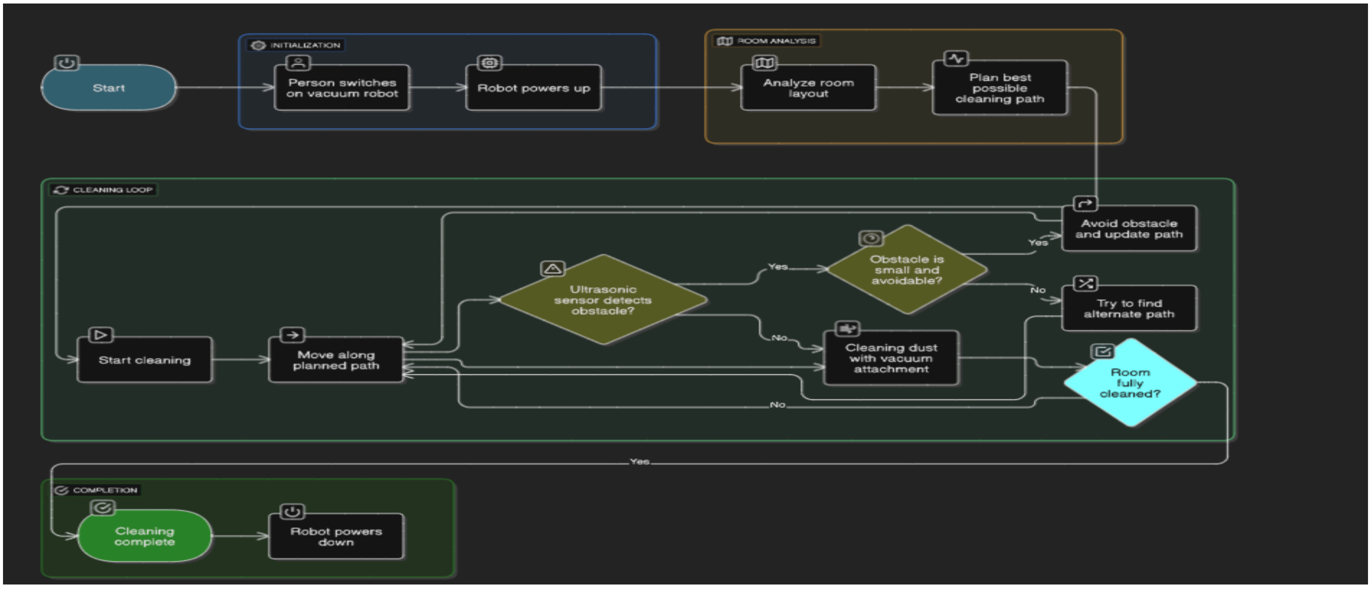
Ref #	Functions	Category	Attribute	Details & Boundary Constraints
R1.1	Detect and classify debris	Evident	System accuracy	Must accurately detect dust, wrappers, and cigarette butts with 90%+ precision.
R1.2	Navigate and avoid obstacles	Hidden	Navigation speed	The system must react to obstacles within 0.5 seconds of detection.
R1.3	Perform autonomous cleaning	Evident	Battery efficiency	Cleaning duration must exceed 2 hours on a single charge.
R1.4	Operate in multiple environments	Hidden	Environmental adaptability	The bot must adjust suction and navigation for different floor types (e.g., carpet, tile)
R1.5	Generate performance reports	Frill	Data reporting	Should provide detailed cleaning summaries, including time spent and areas covered

System Attributes / Nonfunctional Requirements

Attribute	Details and Boundary Constraints	Category
Response time	Must detect debris and navigate obstacles within 1 second	Mandatory
Concurrent User Load	Supports up to 5 simultaneous actions, including diagnostics and updates.	Mandatory
Energy Efficiency	Must consume less than 10% battery capacity in idle mode.	Optional
Noise Level	Cleaning operation should not exceed 65 decibels.	Mandatory

3.2) Use Cases

Use Case Diagram


Start
INITIALIZATION
Person switches on vacuum robot
Robot powers up
ROOM ANALYSIS
Analyze room layout
Plan best possible cleaning path
CLEANING LOOP
Avoid obstacle and update path
Yes
Obstacle is small and avoidable?
Yes
No
Ultrasonic sensor detects obstacle?
Try to find alternate path
Start cleaning
Move along planned path
No
Cleaning dust with vacuum attachment
Room fully cleaned?
No
Yes
COMPLETION
Cleaning complete
Robot powers down

Description of Use Cases

Use Case: Start Cleaning

- Actors: User
- Objective: Initiate the cleaning process.
- Description: The user tells the robot to start the cleaning process. It switches on its sensors and starts the vacuum cleaner.
- See Cross References: Functions: F1 1
- Pre-condition The robot is in the power on, standby state.
- Criteria for Successful Post: The robot begins to clean.

Use Case: Move

- Actors: User
- Cleaning sequence — Move robot around the cleaning area.
- Name: Robot Moves to Areas or Traces a Path
- Cross References: Functions: F1 2
- Sensors and motors must be working.
- Post Prime: The robot is in the correct area.

Use Case: Charge Battery

- Actors: User
- Objective: To recharge the robot battery
- So, the user basically leads the robot to its charging spot.
- Cross References: Functions: F1 4
- Pre-Requisites: The robot should be alive
- Post-Conditions (successful): The robot is charging.
- Failure Condition Post-Charge: Does not start charging in case of power failure.

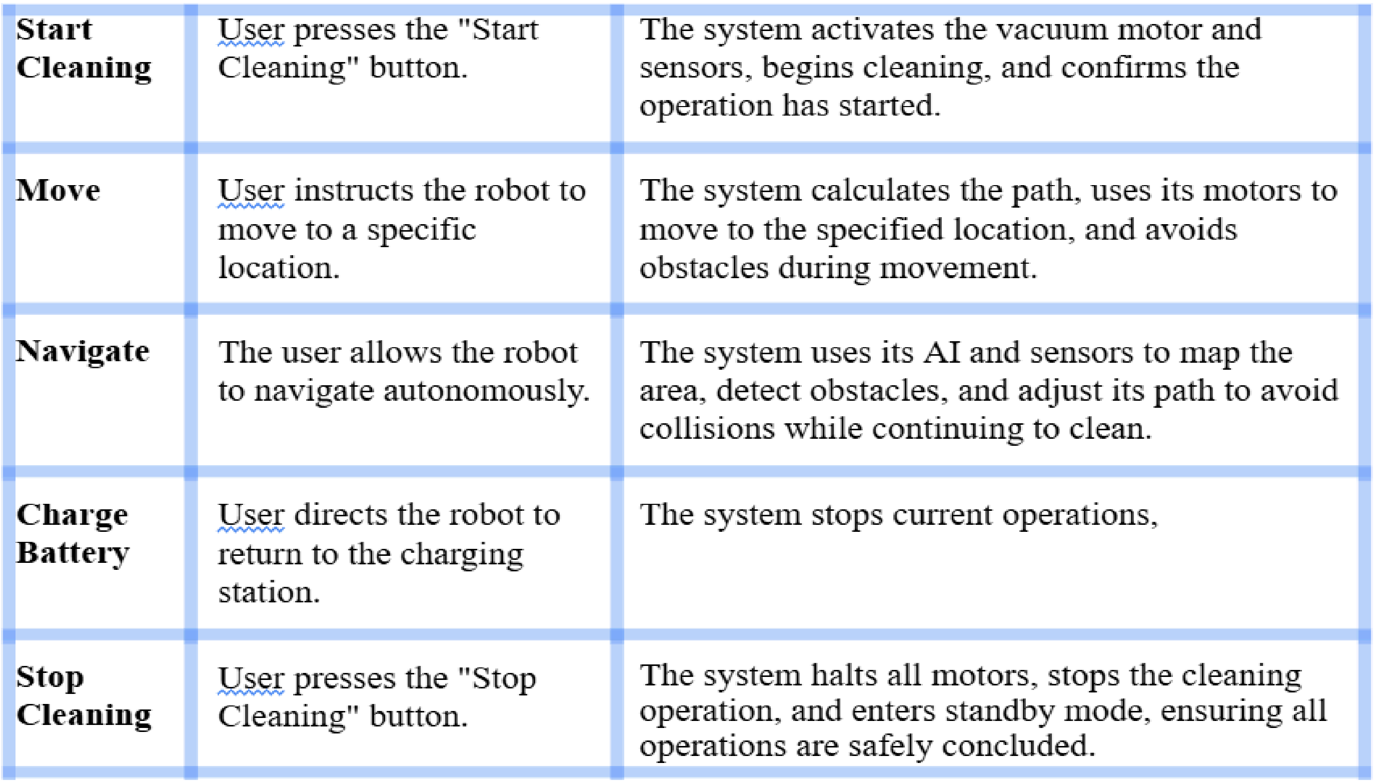
Use Case: Detect Dirt

- Actors: User
- Purpose: Detect dust and debris
- The robot used its AI system to identify and find dirt.
- Cross References: Functions: F1 5
- Content: They should have working AI algorithms and a camera.
- Post-conditions to the success: the dirt has been recognized and marked for cleaning.
- 4.5 Failure Post-conditions : it fails to detect dirt and logs an Error.

Use Case: Stop Cleaning

- Actors: User
- End the cleaning operation.
- Action instructions: The user tells the robot to stop cleaning.
- Cross References: Functions: F1. 7
- Pre-Conditions: The robot is currently cleaning.
- Successful Post-Conditions: The cleaning process has stopped.
- Post aspects of failure: Robot does not stop cleanin

Typical Course of Events



Start Cleaning	User presses the "Start Cleaning" button.	The system activates the vacuum motor and sensors, begins cleaning, and confirms the operation has started.
Move	User instructs the robot to move to a specific location.	The system calculates the path, uses its motors to move to the specified location, and avoids obstacles during movement.
Navigate	The user allows the robot to navigate autonomously.	The system uses its AI and sensors to map the area, detect obstacles, and adjust its path to avoid collisions while continuing to clean.
Charge Battery	User directs the robot to return to the charging station.	The system stops current operations,
Stop Cleaning	User presses the "Stop Cleaning" button.	The system halts all motors, stops the cleaning operation, and enters standby mode, ensuring all operations are safely concluded.

4)Non-Functional Requirements

4.1) **Performance Requirements**

The bot has only 0.5 seconds to both detect and classify the debris upon encounter.

Cleaning coverage efficiency needs to be 95%+ for a standard-sized room (12 x 12 feet) in 10 minutes.

It should be capable of up to 30 minutes of running time on a single charge in standard cleaning operations.

Reporting and analytics data sync should happen in less than 5 seconds once the task is complete.

4.2) **Safety Requirements**

It has to cease all operations right away in order to avoid damaging itself or causing an accident.

Avoidance Handling: For avoiding obstacles Ultrasonic and infrared sensors are used to avoid collision with any obstacles (with a minimum gap of 10 cm).

4.3) **Security Requirements**

Our AES-256 encryption will ensure that no communications between the bot and other devices are intercepted

4.4) Reliability Requirements

During cleaning operations, the system should maintain 99% uptime.

Hardware needs to last at least that with no major wear, ridicule or operational failure.

The sensors should work with 95% accuracy under normal usage conditions for a minimum of 12 months.

4.5) Usability Requirements

Since users may not have the technical expertise to navigate complex command sets, the bot user interface must be intuitive.

It would only take 5 minutes for first-time users to configure and set it up.

It must also be able to make announcements for the maintenance to be performed to its owner (emptying the dust box, re-charging) through sound, LED indicators

4.6) Supportability Requirements

Over-the-air firmware updates should be enabled so that the new software can be compatible with the current system.

The modular design should allow easy replacement of critical components like sensors, batteries, and brushes

4.7) User Documentation

A detailed user manual must accompany the product, including:

- A guide to setting up a dataset, step-by-step.
- Guides for troubleshooting common issues.
- How to care for optimal performance.

Digital and printed versions of the information should exist somewhere.

First-time users are to be provided with a Quick Start Guide.

It must have an online support portal with FAQs, video tutorials, and customer support contact options.

5) References

Open-Source Communities:

Frameworks of TensorFlow and PyTorch for AI model development.
Built on ROS (Robot Operating System) Melodic and Noetic for robot integration.

Hardware Manufacturers:
Sourced components from top brands like Bosch and Panasonic for sensors, microcontrollers, and lithium-ion batteries.

Guidelines and Standards set by Industry:

ISO 13482:2014: Personal care robots: safety standards. The wireless communication protocols were IEEE 802.11.

Scholar Anti-Articles and Anti-Research:

AI-based debris detection and classification papers.

A3. DESIGN SPECIFICATIONS

1. Introduction

1.1 Purpose of Document

This document outlines the technical design specifications of the LumaBot project. It defines the system architecture, software components, control logic, and interaction between modules used to create an AI-powered cleaning robot.

1.2 Intended Audience

- Project Supervisor and Coordinator
- Final Year Evaluation Committee
- Development and Testing Team

1.3 Project Overview

LumaBot is an AI-integrated robot capable of detecting and cleaning small debris (like dust, wrappers, and cigarette butts) using a v5 object detection model deployed on an ESP32-CAM. It communicates with a second ESP32 board to activate motors, vacuum, and obstacle detection systems.

1.4 Scope

The document focuses on embedded software architecture, control strategies, hardware integration, and testing plans.

2. Design Considerations

2.1 Assumptions and Dependencies

- Adequate indoor lighting is available for object detection.
- v5s model is optimized to run efficiently on ESP32-CAM.

2.2 Risks and Volatile Areas

- False negatives during debris detection may cause cleaning to fail.
- Hardware limitations could affect real-time processing.
- Ultrasonic sensors may misread reflective or curved surfaces.
- Continuous operation may lead to power depletion or overheating.

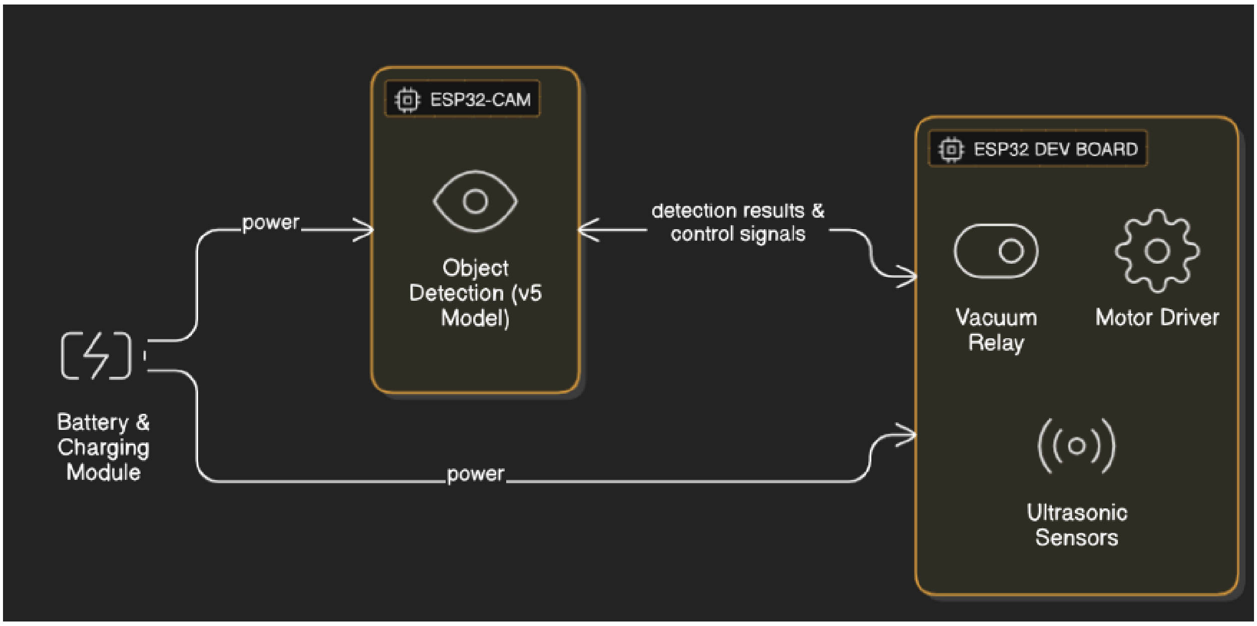
---

3. System Architecture

3.1 System Level Architecture

The system uses a dual-microcontroller setup:

- ESP32-CAM handles real-time object detection using the v5 model.
- ESP32 Dev Board manages vacuum activation via relay, movement control via motor driver, and obstacle avoidance via ultrasonic sensors.


ESP32-CAM
Object Detection (v5 Model)
ESP32 DEV BOARD
Vacuum Relay
Motor Driver
Ultrasonic Sensors
Battery & Charging Module
power
power
detection results & control signals

3.2 Software Architecture

The software is divided into modular components:

- v5 model (deployed on ESP32-CAM).
- Serial or GPIO-based signaling between the ESP32 boards.

---

4. Design Strategy

- **Modularity:** Software and hardware components are isolated for independent development and debugging.
- **Optimization:** v5s was chosen due to its small size and fast inference.
- **Safety:** Sensors trigger real-time responses to avoid crashes.

---

5. Detailed System Design

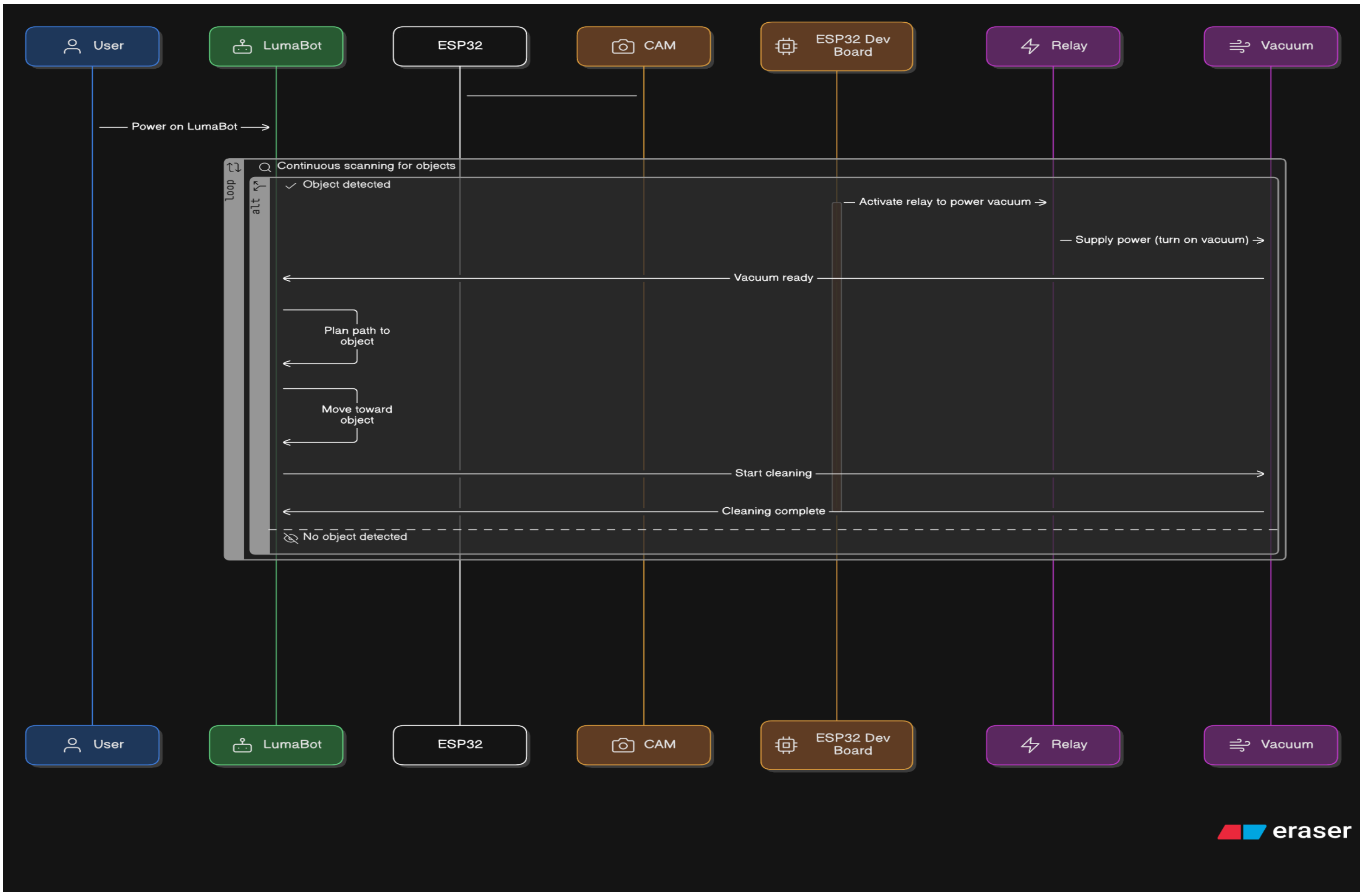
5.1 Database Design

LumaBot operates as a real-time embedded system and does not use a traditional database. Any future enhancements for data logging can include local SD card storage or cloud integration.

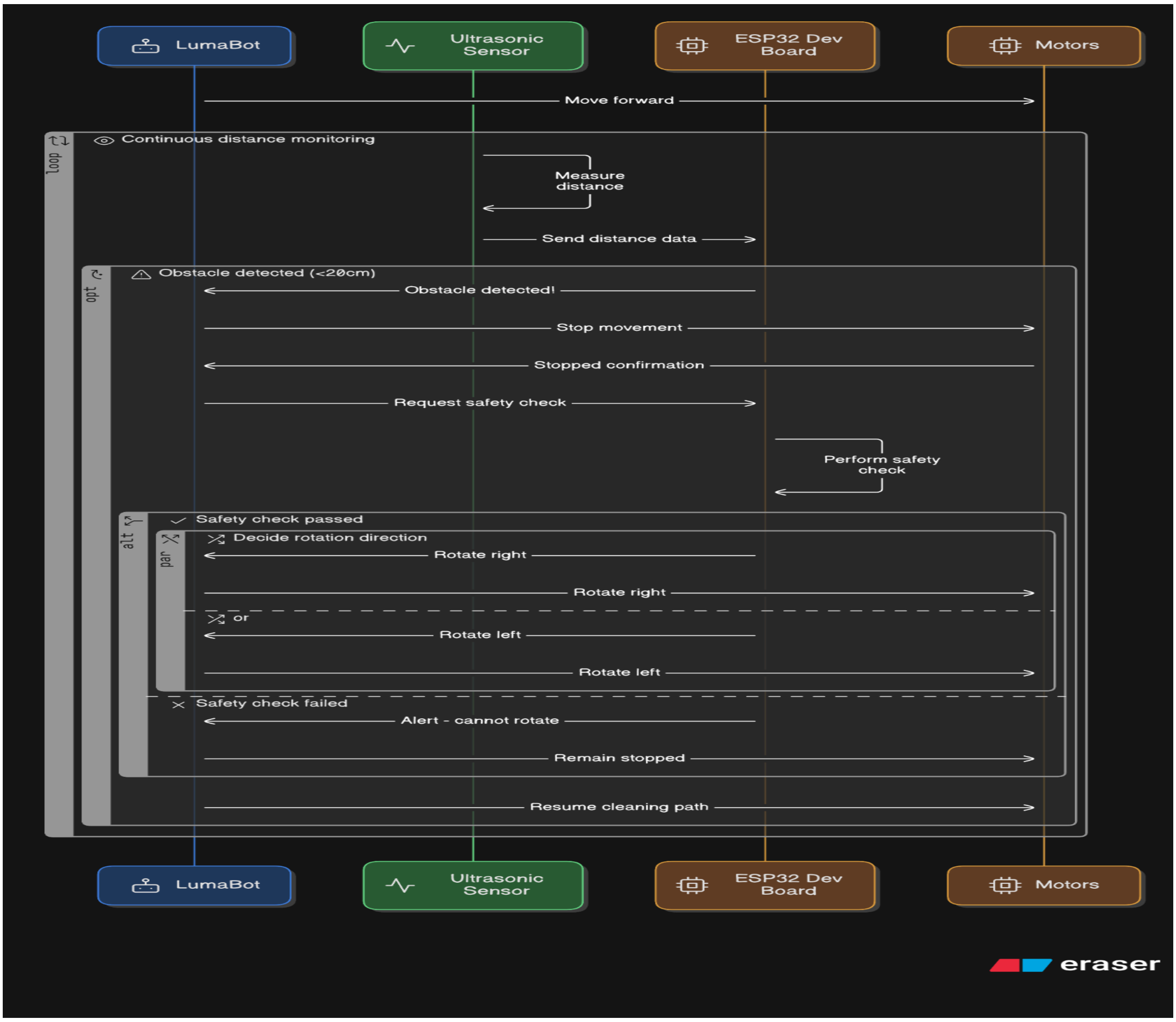
5.2 Application Design

5.2.1 Sequence Diagrams

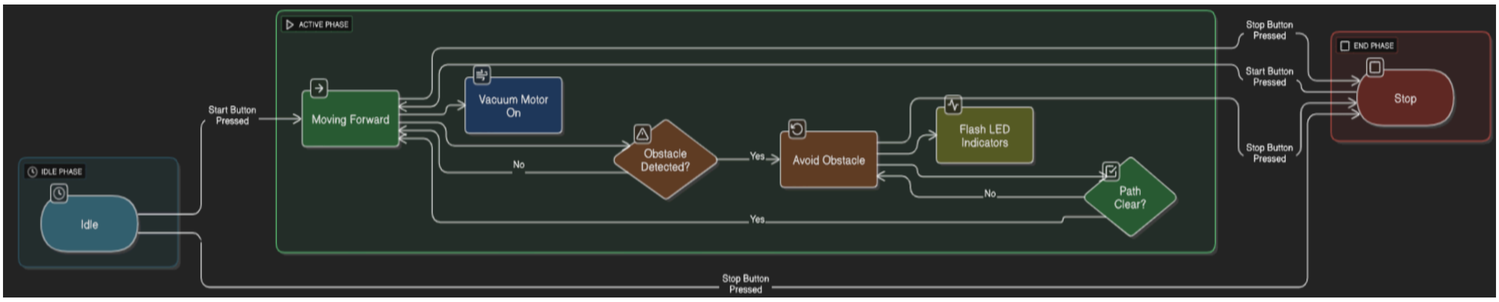
5.2.1.1 Detection and Cleaning Flow


User
LumaBot
ESP32
CAM
ESP32 Dev Board
Relay
Vacuum
Power on LumaBot
loop
Continuous scanning for objects
alt
Object detected
Activate relay to power vacuum
Supply power (turn on vacuum)
Vacuum ready
Plan path to object
Move toward object
Start cleaning
Cleaning complete
No object detected
User
LumaBot
ESP32
CAM
ESP32 Dev Board
Relay
Vacuum
eraser

5.2.1.2 Obstacle Avoidance Flow



5.2.2 State Diagrams



6. References

1. Arduino. (n.d.). *ESP32 Board Guide*. https://docs.arduino.cc/hardware/esp32
2. Espressif Systems. (n.d.). *ESP32-CAM Documentation*. https://www.espressif.com/
3. Ultralytics. (2020). *v5 GitHub Repository*. https://github.com/ultralytics/v5
4. Circuit Digest. *ESP32 Relay Interface*. https://circuitdigest.com/
5. LabelImg. *Image Annotation Tool*. https://github.com/tzutalin/labelImg
6. TutorialsPoint. *Agile Software Development*. https://www.tutorialspoint.com/agile/
7. W3Schools. *Bluetooth with Arduino*. https://www.w3schools.com/arduino/

A4. OTHER TECHNICAL DETAIL DOCUMENTS

1. Test Cases Document

Test Case 1: Start Button Activation

Description: When the physical start button is pressed, the robot should begin operation.
Precondition: The robot is powered on and in idle state.
Action: User presses the physical start button.
Expected Result: Luma Bot starts moving forward with the vacuum motor turned on.

Test Case 2: Move Forward on Clear Path

Description: The robot should move straight if there are no obstacles ahead.
Precondition: Luma Bot is in active state and ultrasonic sensors detect no obstacles.
Action: Robot continues forward.
Expected Result: Robot moves straight smoothly with vacuum running.

Test Case 3: Obstacle Detection and Avoidance

Description: If an obstacle appears in front, the robot should stop and avoid it.
Precondition: Robot is moving forward.
Action: Ultrasonic sensor detects an obstacle within a certain range.
Expected Result: Robot stops, turns or reverses to avoid obstacle, and resumes forward movement.

Test Case 4: Continuous Obstacle Avoidance

Description: Robot should handle multiple obstacles in sequence.
Precondition: Robot is moving in an environment with multiple objects.
Action: Robot detects and avoids multiple obstacles during navigation.
Expected Result: Robot continuously avoids obstacles and keeps cleaning.

Test Case 5: Manual Stop Pressed
Description: Pressing the stop button should immediately halt the robot.
Precondition: Robot is currently operating.
Action: User presses physical stop button.
Expected Result: Robot stops all movement immediately.

Test Case 6: Sensor Malfunction Handling
Description: If the ultrasonic sensor fails or gives no data, robot should stop for safety.
Precondition: Robot is active.
Action: Sensor returns null, out-of-range, or invalid data.
Expected Result: Robot stops to avoid collision or damage and blinks an LED (if safety mechanism is implemented).

2. UI/UX Detail Document/Not included/Future Implementation

Wi-Fi Control Web Interface

- **Home Screen Features:**
  - Directional Buttons: Forward, Backward, Left, Right, Stop
  - Vacuum Toggle Button: ON/OFF
  - Connection Status Indicator

Design Considerations

- Web interface hosted on ESP32 local IP (e.g., 192.168.4.1)
- Clean, minimal design with clear labels and icons
- Responsive layout for mobile browsers

User Experience Goals

- Easy to use from any mobile browser
- Real-time command execution over Wi-Fi
- No app installation required

3. Coding Standards Document

ESP32 (Arduino) Coding

- Variable Naming: camelCase
- Constants: ALL_CAPS_WITH_UNDERSCORES
- Modular Functions: moveForward(), activateVacuum(), etc.
- HTML & CSS for web interface embedded in Arduino sketch
- HTML elements handled using server.on("/command", ...) logic

4. Project Policy Document

Version Control Policy

- Manual versioning of Arduino and Python files with timestamps

Team Contribution Policy

- **Rayyan:** Wi-Fi interface, AI model deployment
- **Ibtesam:** Circuit setup, vacuum integration
- **Asad:** Testing, training dataset creation

Security Policy

1. **Offline & Secure**: Luma Bot runs autonomously with no mobile or wireless control, reducing external security risks.
2. **Hardware Protected**: Only authorized users can access hardware or upload code; sensors and power systems are secured.
3. **Failsafe Operation**: Robot stops on sensor failure or unexpected behavior; physical stop button ensures manual override.

Testing Policy

- Component testing before integration
- Regression tests after adding new features
- Safety validation during movement and vacuum activation

---

5. User Manual Document

Autonomous Mode

- The robot operates using object detection and obstacle avoidance

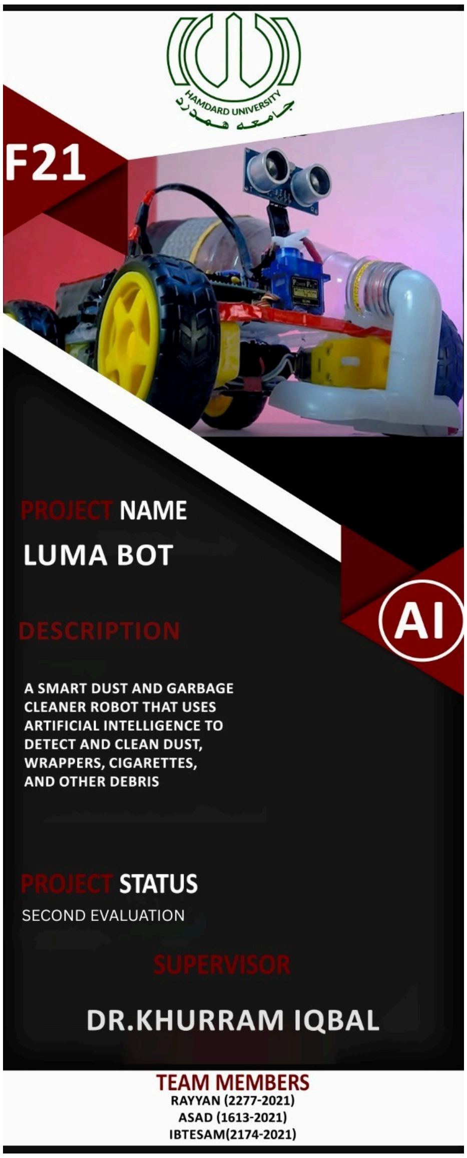
Maintenance

- Clean vacuum container after each use
- Recharge battery before operation
- Check that all sensors and motors are properly connected

Troubleshooting

- **Robot Not Moving:** Check motor driver wiring and battery
- **Vacuum Not Working:** Check relay and power supply

A5. FLYER & POSTER DESIGN



HAMDARD UNIVERSITY

F21

PROJECT NAME

LUMA BOT

AI

DESCRIPTION

A SMART DUST AND GARBAGE CLEANER ROBOT THAT USES ARTIFICIAL INTELLIGENCE TO DETECT AND CLEAN DUST, WRAPPERS, CIGARETTES, AND OTHER DEBRIS

PROJECT STATUS

SECOND EVALUATION

SUPERVISOR

DR.KHURRAM IQBAL

TEAM MEMBERS

RAYYAN (2277-2021)
ASAD (1613-2021)
IBTESAM(2174-2021)

A6. COPY OF EVALUATION COMMENTS

COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – I MID SEMESTER EVALUATION

A Photostat or scanned copy should be placed when submitting document to Project Coordinator. (**Note**: Please remove this line when attach copy that is required)

COPY OF EVALUATION COMMENTS BY JURY FOR PROJECT – I END SEMESTER EVALUATION

A Photostat or scanned copy should be placed when submitting document to Project Coordinator. (**Note**: Please remove this line when attach copy that is required)

COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – II MID SEMESTER EVALUATION

A Photostat or scanned copy should be placed when submitting document to Project Coordinator. (**Note**: Please remove this line when attach copy that is required)

A7. MEETINGS' MINUTES & Sign-Off Sheet

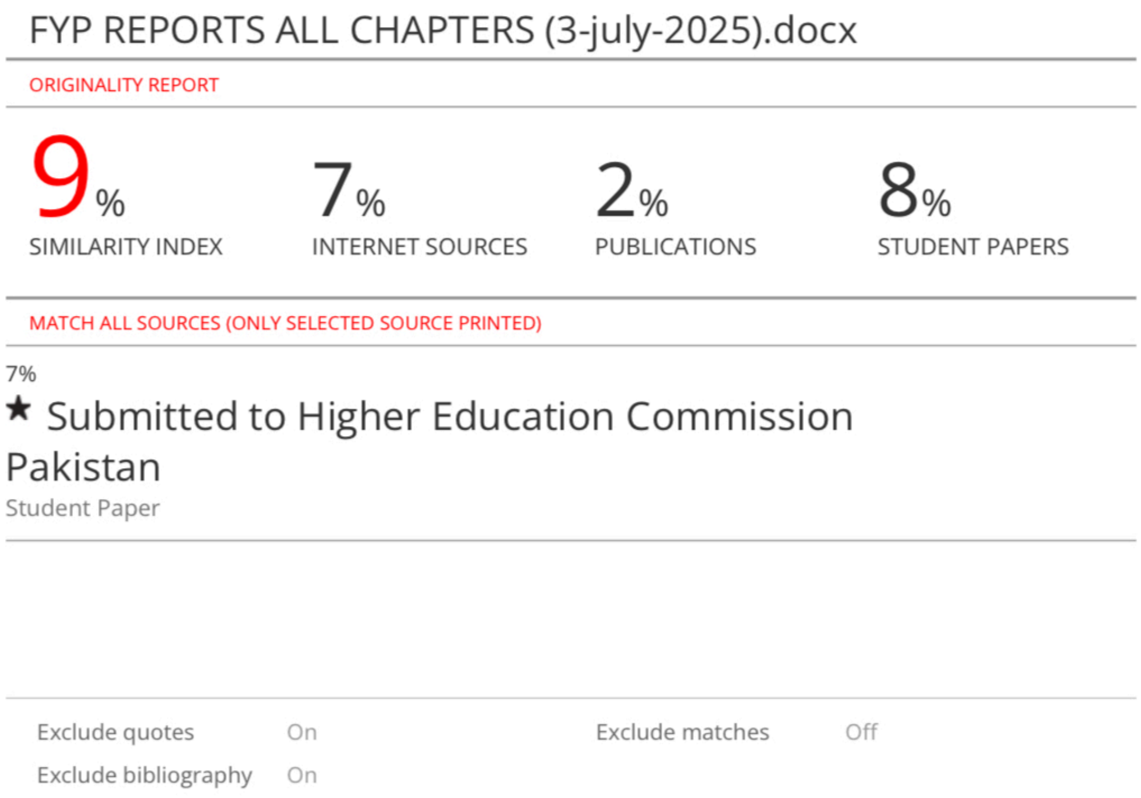
Original Documents should be placed when submitting document to Project Coordinator. Document should be signed by the supervisor and all other members present in the meeting (wherever possible). (**Note**: Please remove this line when attach copy that is required)

Weekly meetings' minutes are required (held with Supervisor and/or with client). Important group discussions can also be included here.

A9. PROJECT PROGRESS

Photostat of Incremental versions of Requirement Signoff sheet submitted to Project Coordinator. (**Note**: Please remove this line when attach copy that is required)

A11. Plagiarism Test Summary Report



FYP REPORTS ALL CHAPTERS (3-july-2025).docx

ORIGINALITY REPORT

9%	7%	2%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

7%

★ Submitted to Higher Education Commission Pakistan

Student Paper

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		