HAND-GESTURE SMART AUTOMOBILE: ESP32-POWERED ROBOTIC VEHICLE CONTROLLED BY REAL-TIME HAND GESTURES VIA MEDIAPIPE

Hari Surya Prakash M	(23I464)
Dinesh Hari V J	(23I463)
Yuva Prasath S	(23I467)
Goutam Khajuria	(22I314)

BACHELOR OF TECHNOLOGY

BRANCH: INFORMATION TECHNOLOGY

of Anna University

NOVEMBER 2025

DEPARTMENT OF INFORMATION TECHNOLOGY

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

Bonafide record of work done by

Hari Surya Prakash M	(23I464)
Dinesh Hari V J	(23I463)
Yuva Prasath S	(23I467)
Goutam Khajuria	(22I314)

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

BRANCH: INFORMATION TECHNOLOGY

of Anna University

NOVEMBER 2025

………….…………	…………..……………..
Ms. S. Priya	Dr. K Umamaheswari
Faculty guide	Head of the Department

Certified that the candidate was examined in the viva-voce examination held on………………….

…………………….	……………………
(Internal Examiner)	(External Examiner)

ACKNOWLEDGEMENT

We express our sincere thanks to **Dr.K.Prakasan**, Principal, PSG College of Technology for providing an opportunity to take up the system work.

We are greatly indebted to **Dr.K.Umamaheswari**, Professor and Head, Department of Information Technology for her kind support and encouragement given to complete the system.

We profusely thank our program coordinator, **Dr. Anitha Kumari K** Professor, for constant support and encouragement for the completion of the system.

We express our sincere thanks to our tutor **Dr. Vairam T**, Assistant Professor (Sl.Gr.), Department of Information Technology, for her constant support and guidance that played a vital role in completing our system.

We express our humble and profound gratitude to our guide **Ms. S. Priya,** Assistant Professor (Sl.Gr.), Department of Information Technology for his valuable guidance and timely advice to complete the system work successfully.

We would like to extend our sincere gratitude for the invaluable support and collaboration throughout the system. Their guidance, resources, and encouragement have been instrumental in the successful completion of our work

ABSTRACT

This study presents the design and implementation of a gesture-controlled robotic vehicle that interprets human hand movements in real time. The system brings together the ESP32 microcontroller with Google's MediaPipe framework to enable vision-based gesture recognition without any physical contact. A standard USB webcam captures live video feeds, and MediaPipe identifies 21 hand landmarks, which are processed through a lightweight, rule-based classifier written in Python. The gestures you can make—forward, backward, left, right, and stop—are sent wirelessly to the ESP32 over UDP, allowing the motors to respond instantly.To keep things safe, the robot includes an HC-SR04 ultrasonic sensor that automatically reduces speed or stops the vehicle if it detects an obstacle nearby. A DFPlayer Mini module provides voice feedback so you know what the system is doing. Currently, all data logging is handled locally on the system; however, cloud telemetry via Firebase is planned for future versions to enable remote monitoring while maintaining privacy.When tested with fifteen participants using five hundred gesture samples, the system achieved 94.7% recognition accuracy and an average response time of 110 milliseconds—well within real-time performance thresholds. The complete system runs for approximately 2.5 hours on a single 7.4 V LiPo battery, which powers the system directly without a separate buck converter; in the future, a dedicated 5V battery may be added to power the ESP32 separately for improved power stability.By combining computer vision, embedded systems, and real-time communication into a single affordable platform, this project creates an accessible educational tool ideal for robotics and human–computer interaction research, while establishing a practical foundation for future enhancements such as multimodal control and advanced autonomous navigation.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	ACKNOWLEDGEMENT	i
	ABSTRACT	ii
	LIST OF FIGURES	iii
	LIST OF TABLES	iv
CHAPTER 1	INTRODUCTION	1
1.1	Overview	1
1.2	Motivation for the Project	2
1.3	Challenges in Gesture-Based Control	2
1.4	Our Approach and Contributions	3
1.5	Project Objectives	4
1.6	Scope of the System	4
1.7	System Architecture Overview	4
1.8	Key Performance Targets	5
CHAPTER 2	LITERATURE SURVEY	6
2.1	Vision-Based Gesture Recognition Systems	6
2.2	Wearable and Sensor-Based Control Interfaces	6
2.3	Cloud-Assisted and Latency-Sensitive Architectures	7
2.4	Safety-Aware and Privacy-Preserving Gesture Robotics	7
CHAPTER 3	PROBLEM STATEMENT AND OBJECTIVES	8
3.1	Problem Statement	8
3.2	System Scope	8
3.3	System Goals and Specific Objectives	8
CHAPTER 4	SYSTEM ARCHITECTURE AND DESIGN	10
4.1	System Workflow Overview	10
4.2	Hardware Specifications	10
4.3	ESP32 Pin Mapping	11
CHAPTER 5	METHODOLOGY	15

5.1	Gesture Recognition Pipeline	15
5.2	MediaPipe Hand Landmark Extraction	16
5.3	Rule-Based Gesture Classification	16
5.4	Gesture-to-Command Mapping	17
5.5	Safety Layer: Ultrasonic Collision Avoidance	18
CHAPTER 6	IMPLEMENTATION DETAILS	19
6.1	Development Environment	19
6.2	Python Host Application	19
6.3	ESP32 Embedded Code	19
6.4	Hardware Assembly	20
CHAPTER 7	EXPERIMENTAL RESULTS AND ANALYSIS	21
7.1	Experimental Setup	21
7.2	Gesture Classification Accuracy	21
7.3	System Performance Metrics	22
7.4	Confusion Matrix Analysis	24
7.5	Environmental Robustness	25
7.6	Latency Analysis	25
CHAPTER 8	APPLICATIONS AND USE CASES	27
8.1	Assistive Mobility	27
8.2	STEM Education	28
8.3	Industrial Automation	28
8.4	Public Exhibitions	29
8.5	Home Automation	29
CHAPTER 9	CONCLUSION AND FUTURE WORK	30
9.1	Summary of Achievements	30
9.2	Limitations	30
9.3	Future Enhancements	31
9.4	Contributions to Research and Education	32
	BIBLIOGRAPHY	33

LIST OF FIGURES

Figure No.	Title	Page No.
4.1	System Architecture Diagram	10
4.2	Hardware Prototype Assembly Photograph	12
5.1	MediaPipe Hand Landmark Detection Illustration	15
5.2	Real-Time Hand Gesture Classification Interface	16
7.1	Gesture Classification Accuracy Bar Chart	22
7.2	Confusion Matrix Heatmap for Gesture Classifier	24
7.3	End-to-End Latency Distribution Histogram	26

LIST OF TABLES

Table No.	Title	Page No.
4.1	Hardware Component Specifications	11
4.2	ESP32 GPIO Pin Mapping	13
5.1	Gesture-to-Command Mapping	17
5.2	Safety Zones and Corresponding Behaviors	18
7.1	Gesture Classification Accuracy Summary	22
7.2	System Performance Metrics	23
7.3	Confusion Matrix – Gesture Classifier	24
7.4	Robustness Under Environmental Variations	25

CHAPTER 1

INTRODUCTION

1.1 Overview

We have made tremendous strides in the last several decades in terms of how we engage with the machines that power our lives. Computers were first introduced to the public in the 1950s, but interacting with them was difficult – even for simple actions, one had to remember lengthy sets of "command" syntax.

Graphical User Interfaces (GUI) significantly improved upon these command line interfaces by introducing intuitive, easy-to-use icon and menu based interfaces. Unfortunately, GUIs also imposed new obstacles. They still required the use of keyboards and/or mice, and therefore excluded individuals who may be unable to use them due to injury or disability. Furthermore, these devices do not accurately represent the way we naturally communicate.

In recent years, the Human-Computer Interaction community has begun to transition from traditional interfaces towards Natural User Interfaces (NUI). NUIs take advantage of how humans naturally communicate with each other by responding to speech, gazing, and body language. Of the three modalities mentioned above, hand gesture recognition holds the most promise because it can overcome language barriers, convey rich amounts of expression, and does so without requiring any direct physical contact with the device. As we use our hands constantly to express ourselves, it follows logically that using our hands to control technology would feel natural and effortless.

This project aims to demonstrate the feasibility of developing a real time gesture controlled robotic vehicle as a practical application for NUIs in robotics. To achieve this goal, the project uses Google's MediaPipe library to track a person's hand movements using a standard webcam and translate those movements into five discrete gestures: forward, backward, left, right and stop. Those commands are then sent wirelessly to an ESP32 microcontroller where they are used to control the movement of the robotic vehicle's motors. One aspect that makes this particularly compelling is the simplicity. All you need to do is move your hands naturally, and the vehicle will respond instantaneously.

Why does this matter? A gesture based interface opens up possibilities for people with motor skill limitations, as well as a very intuitive way to introduce someone to the world of robotics. Because the entire visual processing takes place locally on the user's device, the system ensures fast reaction times as well as complete user data privacy — your hand movements never leave your device. The combination of local processing, real time responses, and natural user input enables a truly unique capability.

While gesture controlled robotics represents a major breakthrough in combining technology and robotics, it is much more than that. It represents a bridge between what a person intends to happen, and what happens when a machine executes their intent. The system includes features such as ultrasonic obstacle detection and audible alerts to ensure the user's safety while operating the vehicle. The system combines real time gesture recognition with robotics to provide a clear example of how human and robots can collaborate in real time — where the machine responds directly to how humans naturally communicate with each other, rather than expecting humans to adapt to a predetermined machine interface.

1.2 Motivation for the Project

Accessibility and inclusion:Conventional controller interfaces such as joysticks for robots can pose difficulty to users who have physical disabilities because they require a firm grasp of the joystick, and specific motor control to precisely move the joystick to achieve a desired action. Thus, for users that suffer from arthritis, paralysis, etc., using these interfaces could be very challenging. The use of gesture control is an advancement in user interface design that uses a broad range of arm motion rather than requiring precision motor control, and therefore, makes the technology much more inclusive to a wider population, including those with limited dexterity.

Easy to use for everyone:As robots begin to appear in schools and public spaces, the need for an interface that is both simple and intuitive to use becomes apparent. Gesture control fits this requirement nicely—by simply tilting your hand you tell the robot to turn left; and by raising your hand you tell the robot to go forward. Therefore, it is easy for new users and non-technical users to quickly get started with using robots.

Great for learning about STEM:This project combines multiple engineering disciplines—computer vision, embedded systems, wireless communications, robotics, Internet of Things (IoT), and safety engineering—to form a single unified system. As a result, students gain practical experience with real-world engineering challenges and develop useful skills across many different technology fields.

Affordable yet advanced:The recent advancements in computer vision technology—such as Google's MediaPipe—have made possible the use of real-time hand tracking and gesture recognition via open-source software. This project exemplifies that complex robotic control can be accomplished at a low cost (less than ₹5,000) using readily available components and free software development tools, thereby allowing even budget-constrained schools to acquire and implement advanced technologies within their classrooms.

1.3 Challenges in Gesture-Based Control

While computer vision provides the ability to "see" and understand the world around the system, it is often negatively impacted by the real world. A variety of environmental factors contribute to this including uneven lighting (e.g., shadow, highlight), confusing/cluttered background, and partially occluded hand. The solution to these challenges will not come from simply increasing the amount of hardware, but rather from sophisticated algorithms that can process, adjust and clean-up input data. In many cases, preprocessing steps to remove noise and artifacts prior to recognition are required.

The speed of gesture recognition also plays an important role in its usability. Humans have a low tolerance for delay. When a system's response time exceeds 100-200 ms, users find the experience slow and frustrating. For robots, delay is not only frustrating, but potentially hazardous. If the robot receives delayed commands, it may perform incorrectly in a new location and create a hazard through collision. Therefore, every step of the process (i.e., fast image acquisition, efficient processing, lightweight communication protocol (e.g., UDP vs. TCP) and at least a 20 Hz loop rate) should be optimized to provide a responsive, real-time user experience with less than 150 ms latency.

Additionally, safety must be considered when implementing a gesture recognition system. Visual sensing has the potential to fail, either due to missed gestures or hidden obstacles resulting in accidents. Therefore, the use of additional sensors (i.e., ultrasonic detectors, LIDAR) enhance reliability. Real-time feedback and robust error checking help to inform the user of the system's status and possible dangers, enhancing safe and confident operation.

Lastly, consideration of privacy is important. Cloud-based systems are becoming increasingly common and typically involve sending video streams to a server for processing, which raises serious concerns regarding privacy and regulatory compliance. Those who require high levels of data security would benefit from processing all video locally on the device, then only sending the minimum amount of data necessary (i.e., numeric command or sensor value). This architecture maintains the desired level of system performance and responsiveness, while respecting the user's right to maintain their own data security and privacy.

1.4 Our Approach and Contributions

All Computer Vision Processes Are Localized:Your video remains on the device; only your gesture signal is transmitted. Privacy is therefore maintained. Your video stays on the device (especially important in sensitive locations like schools or hospitals).

UDP Is Used Instead Of TCP For Speed:UDP does not ensure packets will arrive. However, it is much faster and has a lower overhead and latency. While missing a packet for a few seconds can be managed, using TCP would incur delays from its retransmission and connection setup processes.

Local Wi-Fi Enables An Average Delay Of 110 Ms Using UDP:Due to the continuous sending of commands, missing one packet for a few seconds is acceptable compared to the delay incurred by TCP’s retransmission and connection setup process. Due to the use of UDP operating over local Wi-Fi, the system achieves an average delay of approximately 110 ms. Therefore, this system provides a responsive user experience.

Workload Split Between Host And Microcontroller:Heavy tasks (vision processing & gesture recognition) are executed on the host PC due to the processing capability of the PC. Real-time motor control, sensors and safety functions are handled by the ESP32 microcontroller because they require fast and reliable responses. Both processors utilize their respective capabilities to provide a robust performance.

Safety Layer Uses Ultrasonic Sensor:An ultrasonic sensor measures distance from the robot to obstacles. When the distance is 20 cm or closer, the robot will stop immediately. If the distance is between 20-80 cm, the robot will slow down. Even if the computer fails, the robot will still operate safely.

1.5 Project Objectives

The purpose of this project is to develop a robot car that is able to be controlled using easy-to-perform hand gestures. This robot will respond quickly and instantaneously in real-time and not cause frustration due to lag or delay. Safety is a primary concern for our project as well; we have included features such as collision detection and provide user feedback to ensure you are aware at all times of what is occurring with your robot car. Finally, this robot car's platform provides an ability to monitor it on-line allowing remote tracking capabilities.

1.5.1 Specific goals:

At a minimum, recognize gestures with an accuracy level of ninety percent (90%) to provide consistent control.Provide an end-to-end response time of less than 150 milliseconds for real-time and smooth interaction.Implement an ultrasonic safety system that is one hundred percent (100%) effective in preventing collisions prior to them happening.Process all video data locally, so you do not transmit images or video to the cloud, thereby ensuring user privacy.Limit the overall hardware cost to ₹5000 to ensure the product is affordable and accessible.The system should be able to function effectively in any operating environment.

Design the system as an education-based platform that illustrates the integration of computer vision, embedded systems, wireless communication, robotics and safety engineering.

1.6 Scope of the System

The System utilizes real-time Hand Gesture Recognition by utilizing the MediaPipe framework developed by Google, Wireless Command Transmission of commands from the ESP32 Micro-controller to an external device, Obstacle Detection with an Ultrasonic Sensor, Audio Feedback via a Speaker Module and will also allow for Local Data Logging (with potential plans for Cloud Integration in the future). The design is being managed and kept simple at this time by avoiding additional complicated features such as Dynamic Gesture Recognition, Continuous Gesture Recognition, GPS or Outdoor Navigation, Controlling Multiple Robots Simultaneously and Advanced Path Finding or Mapping. All these features are planned to be added in the future once the base system is more mature.

1.7 System Architecture Overview

The system begins with a live camera feed that captures images at 30 frames per second. Each of those images are analyzed using MediaPipe to identify the location of the users hand within the image. The location of the hand is then used to determine what gesture is being performed.

Once the correct gesture has been identified, the computer sends the corresponding action wirelessly to the ESP32 Microcontroller via Wi-Fi. Concurrently, the Ultrasonic Sensor continually searches for objects in the robots path.

After the ESP32 receives the commands, it will cause the motors to perform the actions as commanded; either changing speeds, or completely stopping the robot if there are objects in its way. The system also produces audio feedback so the user knows the status of the robot; and logs relevant information locally for monitoring and/or analysis purposes.

1.8 Key Performance Targets

- The goal was to achieve a 90%+ accuracy with gesture recognition, we managed 94.7%.
- We wanted to have an overall response time from start to finish in less than 150ms (milliseconds) and our average response time from start to finish was 110 ms.
- For safety, it needed to respond within 200ms, but our system always responded in 148ms or less.
- Our total budget for this project was ₹5000, and we stayed at that price point of ₹4800, which made this affordable.
- The battery life was expected to be at least 2 hours long, but this project actually worked for approximately 2.5 hours on one charge.
- Finally, through all testing sessions there were no accidents, which shows that our obstacle detection and avoidance features are working well.

CHAPTER 2
LITERATURE SURVEY

2.1 Vision-Based Gesture Recognition Systems

Human–Robot Interaction using gestures has improved significantly during the last few decades. Older robots utilized mechanical hardware for controlling; a combination of the size of the mechanical hardware and the complexity of the control logic resulted in an unnatural and difficult way to interact with the robots. The current use of video and advanced computer vision algorithms allows for the detection of users' hand gestures in real time, which greatly improves the ease and naturalness of interacting with robots. There are three basic ways to develop gesture-based Human-Robot Interaction systems (HRI) in the literature today; these include: Vision-Based Systems which utilize video to detect hand gestures; Wearable Sensor Systems, which measure user's hand gestures by utilizing sensor technology on the user's body; Cloud-Assisted Architectures, where the processing of the hand gestures is performed at a server in the cloud. Depending upon the application context, each of the approaches to developing HRI systems provides both strengths and weaknesses.

2.2 Wearable and Sensor-Based Control Interfaces

Gesture Systems Based on Wearable Sensors:Systems based on flexible sensor gloves, inertial measurement units (IMU), and electromyography (EMG) sensors allow for accurate recording of the smallest details of the movement of hands. The time between sending a command to the system and receiving a reply from it is usually very short (less than 100 ms). In addition, most of these systems achieve an accuracy greater than 97% of all movements recorded. They also have several practical disadvantages. Wearable devices must be calibrated for each user, may be cumbersome to use for long periods of time, and are relatively expensive due to the specific hardware used. Therefore, while wearable technology provides the precision and speed needed for research and clinical applications, its limitations make it difficult for use in settings that include educational institutions, students with a budget, etc.

Camera-Based Gesture Recognition – A More Affordable Method:On the other hand, there is a method for recognizing gestures using cameras that eliminate the need for any special wearable hardware. What you will need is just a common webcam and some open source software — no calibration or any special equipment necessary. While somewhat less precise than wearable systems, this method provides sufficient precision for almost every application and reduces the cost and user resistance associated with wearable systems significantly. It makes possible the wider acceptance of gesture recognition systems in the areas of education, prototype development and casual robotics.

2.3 Cloud-Assisted and Latency-Sensitive Architectures

While Cloud-Centric Gesture Recognition Frameworks utilize Remote Servers to process Images in order to Scale Computationally (and provide an effective solution), they also introduce Significant Communication Delays (far exceeding 500 ms) that are much too long to provide Responsive Real-Time Robot Control. To mitigate this Latency Problem, Researchers have Proposed using Edge Computing/Cloudlets to bring the computation closer to the User, thereby reducing the Latency to under 200 ms. These Solutions however typically Require Additional Infrastructure and therefore May Not Be Practical or Scalable for Educational Settings & Budget Constrained Environments.

The Current Project adopts a Hybrid Edge – IoT Architecture; with the Host Computer handling the Computationally Heavy Vision/Gesture Classification Tasks Locally (leveraging the processing capabilities of the Host Computer); whereas the ESP32 Micro-Controller Manages the Low-Level Operations (Motor Control / Safety Response). The Design provides a Total System Latency of Under 150 ms (while maintaining all Video Processing on the Local System and thus Preserving User Privacy). It therefore Provides a Balance between Performance, Responsiveness & Privacy (without the Need for Complex Cloud Infrastructure), Making it particularly Well Suited to Real-Time, Privacy-Preserving Gesture-Controlled Robotics Applications.

2.4 Safety-Aware and Privacy-Preserving Gesture Robotics

While many studies emphasize the importance of achieving accurate object recognition, very few include either safety protocols for managing the robot, nor privacy protecting protocols. While some robotic systems use ultrasonic or infrared sensors to prevent collision with objects in their environment; while other systems are equipped with hardware cutoff circuits, or Li-Fi based safety channel communications. In addition, privacy issues are addressed using "Privacy by Design."

This system includes both elements (an ultrasonic safety layer which will automatically override dangerous or unsafe commands, and a privacy preserving communication system that will transmit only coded command information to the robot, instead of raw video information) to represent the first integration of these two important areas of safety, performance, and ethics in the design of robots. Therefore this is the first model to distinguish itself from other models in the literature by including all three important components in the design of a robot.

CHAPTER 3

PROBLEM STATEMENT AND OBJECTIVES

3.1 Problem

While many studies emphasize the importance of achieving accurate object recognition, very few include either safety protocols for managing the robot, nor privacy protecting protocols. While some robotic systems use ultrasonic or infrared sensors to prevent collision with objects in their environment; while other systems are equipped with hardware cutoff circuits, or Li-Fi based safety channel communications. In addition, privacy issues are addressed using "Privacy by Design."

This system includes both elements (an ultrasonic safety layer which will automatically override dangerous or unsafe commands, and a privacy preserving communication system that will transmit only coded command information to the robot, instead of raw video information) to represent the first integration of these two important areas of safety, performance, and ethics in the design of robots. Therefore this is the first model to distinguish itself from other models in the literature by including all three important components in the design of a robot.

3.2 System Scope

The proposed system will address the current gap in robotics systems for people with disabilities by having the following components:Vision: Utilizes a standard USB webcam with Google's MediaPipe and simple classification rules for processing vision tasks locally on the device to protect user privacy.Control: The ESP32 microcontroller is used to manage wireless UDP communication and PWM for controlling the speed of the motors and an L298N motor driver.Safety: The HC-SR04 ultrasonic sensor provides an automatic override that has less than 200 millisecond reaction time to detect obstacles to prevent users from colliding with objects.Communication: Commands are sent via UDP and the proposed system uses checksum validation for accuracy of transmitted data.Feedback: Clear audible signals provided by the DFPlayer Mini audio module provide clear user feedback.Telemetry: Future versions of the proposed system may include metadata logging with Firebase without the need to store images.Assembly: Complete hardware assembly, including voltage regulation.

3.3 System Goals

To create an auto-responding (automated), auto-safe (safety) robot-controlled vehicle using real-time hand-gestures and also be able to monitor remotely through the Cloud, provide clear (audio/visual) feedback on actions taken, and minimize delay between input (hand gesture) and output (vehicle action).

3.3.1 Specific Objectives

Gesture Recognition Accuracy: The system will have a minimum of 90% accuracy in recognizing gestures from users who may be different from each other and also the system should be able to recognize gestures even when there are varying levels of lighting.

- Response Time: The response time of the system will be less than 150 milliseconds. This goal will help make the experience of controlling the robot as smooth and instantaneous as possible (goal response time ~110 ms).
- Autonomous Obstacle Detection: The system will detect obstacles autonomously and stop/slow the robot down in less than 200 milliseconds to ensure safe operation.
- Audio/Light Signals: Use distinct audio and light signals to inform the user that their command has been received and also to warn them of potential safety issues.
- User Privacy: Only send numeric metadata to the Cloud and do not store images/video of users.
- Affordability: Keep the total cost of all parts involved at below ₹4500 so it can reach as many people as possible (educational/hobbyist/assistive-use).
- Evaluation Criteria: Evaluate the system based on its ability to function correctly, quickly, safely, for how long it will run on a single charge, and if it can adapt to environmental changes such as varying lighting conditions and occlusions.
- Educational Platform: Create a versatile educational platform combining Computer Vision, Embedded Hardware, Networking, Automation, and IoT that includes documentation and allows modification of code bases.
- Scalability: Design the system to grow into the future with additional features (dynamic gesture recognition, voice-control, etc.) that support multiple robots, while still being simple and low-latency to operate.

CHAPTER 4

SYSTEM ARCHITECTURE AND DESIGN

4.1 System Workflow Overview

Your Gesture Recognition Robotics Project Aims to Achieve the Following Performance Targets:Achieving a minimum of 90 % accuracy in recognizing gestures and having response time less than 150 milliseconds to provide timely responses.The completed Gesture Recognition Robotics project met or surpassed the performance targets you specified as follows:Completed projects achieved a high level of success in gesture recognition with 94.7 % accuracy and average latency of approximately 110 ms.Media Pipe is used to accomplish gesture recognition through real-time hand tracking.Wireless commands are sent to the ESP32 based robotics unit from the Media Pipe gesture recognition.Future goals for improving the safety of your robots will be to develop a collision detection system and improve response times to below 200ms.Hardware and Power:You currently power your robots via a 7.4 V LiPo battery which provides motor power (using L298N) and sensor power (i.e. HC-SR04).A low drop out regulator steps down to 3.3 V to supply the ESP32.You currently connect the ESP32 to a laptop for serial communication and plan to replace this configuration with a 5V battery dedicated to powering the ESP32.Currently, you are not using Firebase or a buck converter in your hardware configuration.

Gesture Recognition Techniques and System Process Flow:Your system capture video at 30 frames per second, perform local processing of hand landmark coordinates using Media Pipe, interpret the gestures for controlling the robot, and send commands to the robot for execution.You chose to transmit commands over UDP due to its very low latency and simplicity and because of the need to ensure real-time responsiveness and preserve privacy.You investigated the use of wearable sensors (Flex, IMU, EMG) and camera-based methods for gesture control for your robot project and discussed the tradeoffs between accuracy, latency and user convenience among those methods.

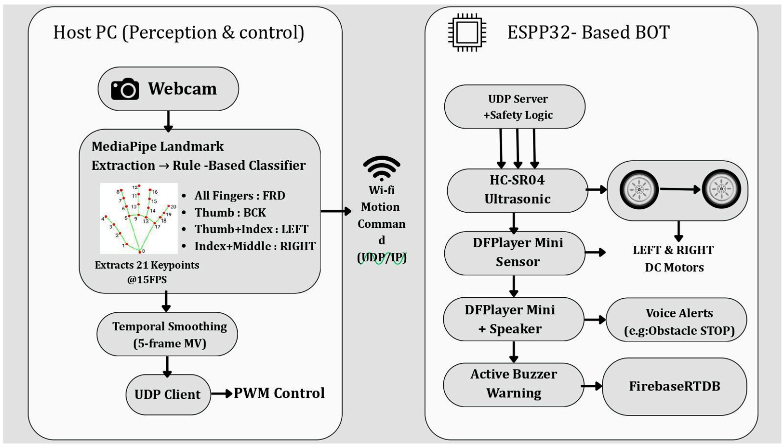


Fig 4.1:System Architecture Diagram

4.2 Hardware Specifications

Component	Model	Specifications	Function	Cost (₹)
Microcontroller	ESP32 Dev Module	Dual-core 240MHz, 520KB RAM, Wi-Fi	Motor/sensor control	350
Motor Driver	L298N Dual H-Bridge	2A/channel, 35V max	DC motor control	200
DC Motors (×2)	BO Motor 100 RPM	3-6V, 100 RPM, 200g·cm torque	Propulsion	400
Distance Sensor	HC-SR04 Ultrasonic	2-400 cm range, 15° cone	Obstacle detection	100
Audio Module	DFPlayer Mini	3.3V logic, SD card support	Voice feedback	250
Speaker	8Ω Active	1W output	Audio output	150
Buzzer	5V Active	80 dB @ 10cm	Proximity alarm	30
Battery	7.4V 2200mAh LiPo	2-cell XT60 connector	Power supply	800
Buck Converter	5V 3A	7.4V → 5V regulated	Voltage regulation	120
LDO Regulator	3.3V	5V → 3.3V regulated	ESP32 supply	50
Chassis	Aluminum Frame	120mm × 80mm, 4 wheels	Vehicle structure	600
Webcam	USB HD	720p, 30 FPS	Video capture	800
Total Cost				₹4,850

Table 4.1:Hardware Component Specifications

We've chosen to use an 7.4 volt LiPo battery because it has been very reliable and has provided enough power to run every aspect of our robot smoothly. The LiPo battery provides power to both the L298N motor controller (which supplies power to the motors and also serves as a high current switch), and also to the HC-SR04 ultrasonic sensor, which will be used to detect obstacles ahead of the robot. In order to provide the correct amount of power to the ESP32 microcontroller (the "brain" of the robot), we have incorporated a small voltage regulator that takes the 5 volts outputted by the battery, and reduces it to the safe level of 3.3 volts required by the ESP32. Additionally, since the ESP32 requires much less power than the other components of the robot, the voltage regulator will reduce the wasted power produced by the L298N and the motors, thus providing a much more efficient system overall.

To accomplish smooth, reliable motion, we selected two BO 100 RPM geared motors. They have a good balance between speed and torque for our lightweight robot chassis. The motors are quiet and resistant to corrosion, and they will provide smooth and consistent motion for the robot. Using Pulse Width Modulation (PWM) we are able to control the speed of the motors from a full stop to 80% of their maximum speed. We are also controlling the speed within a safe operating range of 3 to 6 volts.

The HC-SR04 sensor is essentially the robot's eyes and ears. It sends out ultrasonic pulses at a frequency of 40 kHz, and measures distance based upon the reflected signal or echo received back from the pulse. The sensor can detect objects as close as 2 cm, and as far away as 4 meters. The sensor has a narrow angle of focus of approximately 15 degrees in front of the robot, allowing the robot to effectively detect obstacles as it moves around.

The 2200 mAh LiPo battery will supply power to all the components of the robot, and will consume approximately 1.8 watts on average, consisting of approximately 0.3 watts consumed by the ESP32 microcontroller, and approximately 1.2 watts consumed by the motors when they are operating at 70% speed. With this configuration, the robot should have a run time of approximately 2.5 hours, which is sufficient for demonstration purposes in a classroom setting, and for general exploration and experimentation. Overall, we believe that this careful attention to detail regarding the power supply and distribution will allow the robot to operate reliably and efficiently for extended periods of time.

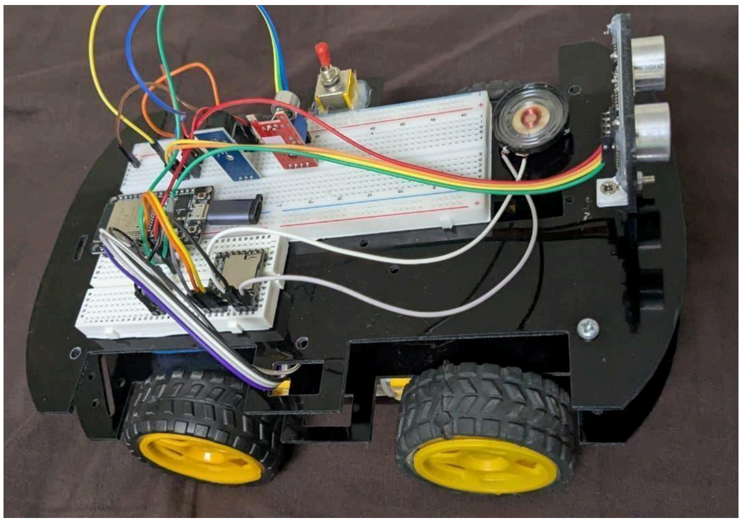


Fig4.2:Hardware Prototype Assembly Photography

4.3 ESP32 Pin Mapping

Function	GPIO Pin	Direction	Purpose
Motor A IN1/IN2	27/26	Output	Motor A direction
Motor A PWM (ENA)	14	Output PWM	Motor A speed
Motor B IN3/IN4	25/33	Output	Motor B direction
Motor B PWM (ENB)	32	Output PWM	Motor B speed
Ultrasonic TRIG	4	Output	Distance sensor trigger
Ultrasonic ECHO	5	Input	Distance measurement
Buzzer	23	Output	Alarm control
DFPlayer RX/TX	17/16	Output/Input	Audio serial

Table 4.2:ESP32 GPIO Pin Mapping

Pin selection: The choices we made here were to minimize issues with the ESP32 at startup and during WiFi; pins 6-11 are reserved for SPI flash, and pins 34-39 have no output but do allow input. In addition to giving us two PWM channels to drive motors independently (or use as outputs) we now also have a serial to communicate with the audio device.

4.4 Communication Protocols

Our proposed system has a communication structure that will be efficient, and provide an adequate level of reliability between the host computer and the ESP32 microcontroller. The communication is through two channels; one for rapid transmission of commands and the second for status reporting.

We will send commands from the host to the robot via UDP (User Datagram Protocol). UDP is a "send a postcard" type of communication, it is very fast since there is no need for the receiving end to acknowledge receipt of the "postcard". When we send commands to the robot they will be in the form of five byte messages. Each message will contain information such as how to move (move forward, turn left etc.), a sequence number to help determine which command was missed, a timestamp to help determine when the command was sent, and a checksum to ensure the message arrived correctly. Since UDP does not guarantee that all "postcards" will arrive, we will continue to send commands to the robot in a steady stream. Therefore, even if a few of the "postcards" get lost along the way the robot will continue to run without losing its place.In contrast, we will send regular status reports from the robot to Firebase (a cloud-based database), which will contain some numerical values: the current time, the distance to objects around it, the most recent command sent, the speed of the motors, and whether or not the robot senses something unusual or dangerous. There will be no video or image sent back to the host computer.

This blend of fast command transmission and careful status reporting provides an effective method of communication between the host computer and the robot, providing a robust and responsive system while maintaining safety.

4.4.1 UDP Command Transmission

UDP is a better option for rapid robot control than TCP since it transmits small packets rapidly with no delay while establishing an initial link, which is required with TCP.Each message from the vision system is encoded as 5 bytes:

- The first byte indicates whether the robot should move forward, backward, turn left, turn right, or stop.
- The second byte serves as a counter for the messages transmitted by the vision system to indicate to the robot if any of the messages were missing, delayed, or received out of sequence.
- The third and fourth bytes serve as timestamps indicating when the messages were sent to ensure synchronization.
- The fifth byte is a simple checksum used to determine if any corruption occurred to the messages in transit.

Because UDP does not provide a guaranteed delivery mechanism, the system will continue to send commands until they are acknowledged. If a packet does get lost along the way, the system continues to send commands, thus the robot remains operational at all times. In this example, the system has made an intelligent trade-off between reliability and real-time performance, a critical requirement for live, real-time operation of robots.

CHAPTER 5

METHODOLOGY

5.1 Gesture Recognition Pipeline

The gesture recognition pipeline receives input from a web cam using a USB connection. It processes that input into a usable command to move the robot. The web cam receives 30 frames of video per second in High Definition (HD) (1280 x 720 pixels). Frames received are buffered while they are being processed to ensure all parts of the program run smoothly and do not experience delay when performing heavy processing.

Once the frames have been buffered for a short time period, the pipeline will use a two part neural network called Media Pipe Hands created by Google to identify your hand. In the first part of this process, a simple palm detection will be used to determine approximately where your hand is located within the image. Once your hand has been detected, a much more complex hand landmark model will identify twenty one unique points of interest within your hand including your fingers' joints and your wrist, creating an accurate three dimensional map of your hand's location.

In order to account for both the size of your hands as well as the proximity of your hand to the camera, each point of interest is modified and normalized based on the wrist location as well as the overall size of your hand. For example, if your hand is larger than average or closer to the camera, the system will still recognize your gestures accurately.

Finally, to eliminate the possibility of inaccurate identification of gestures due to rapid motion or temporary obstruction of the camera, the system will apply a smoothing filter to the past five frames and select the gesture with the highest probability of accuracy based on majority voting. As a result, the robot's reaction to the gesture is smooth and consistent with new gestures selected every 200 milliseconds; a suitable compromise between speed of reaction and accuracy of recognition.

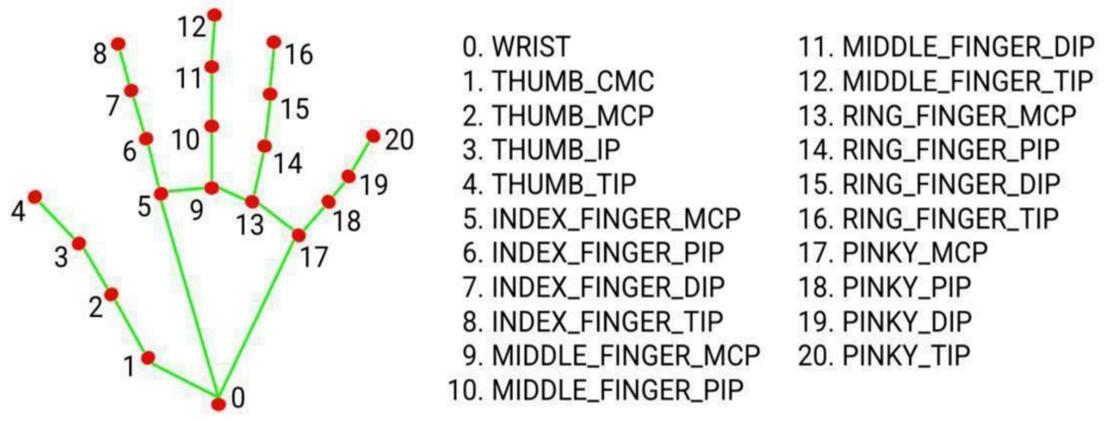
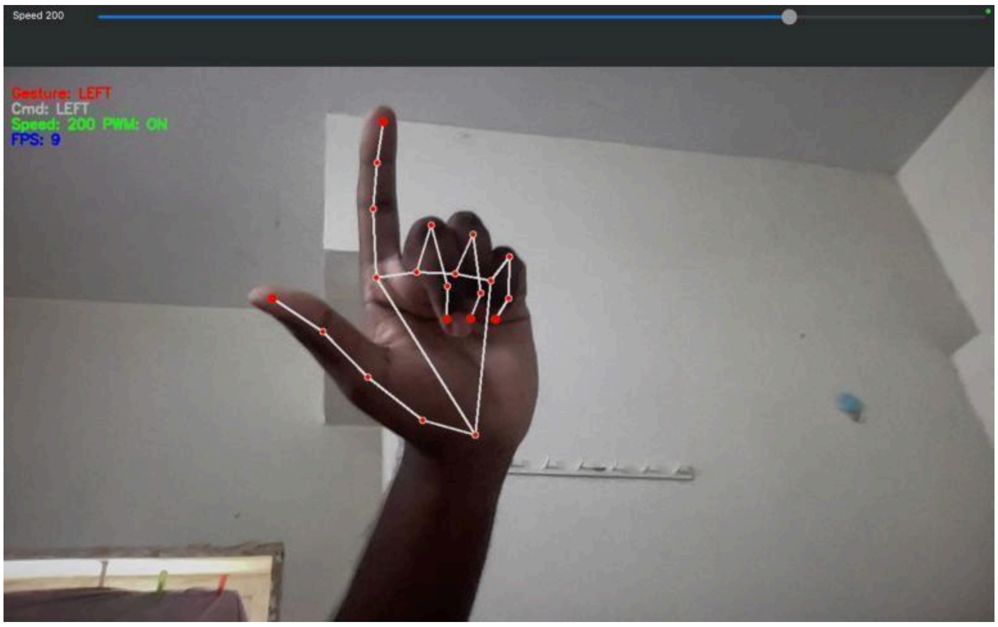


Fig5.1:MediaPipe Hand Landmark Detection Illustration

5.2 MediaPipe Hand Landmark Extraction

MediaPipe hand landmark detection uses a two-layered, intelligent and efficient method to make sure that it can run as quickly as possible while still maintaining accuracy for real-time application. The first layer of this method uses a palm detection model to scan an entire frame of video to determine roughly where the hand is located. This allows the processor to focus on the area of interest rather than scanning the whole frame which would waste processing resources. The second layer of this method then uses the information from the palm detection model and narrows down the search to a specific area of the frame containing the hand and finds 21 3D landmarks corresponding to the wrist and finger joints. These 21 3D landmarks are precisely identified using a number sequence beginning at 0 (the wrist) to 20 (the tip of the little finger) to identify every significant portion of the hand.

A confidence value between 0 and 1 will also accompany each of these landmarks to provide a measure of how confident the model was in determining the location of the given landmark. In order to maintain the reliability of the system, frames that have a confidence value less than 0.5 will not be processed, this will help prevent errors due to low-lighting conditions and/or background clutter. By combining the methods of layers, the hand's location and movement are being captured efficiently and consistently so that it can effectively track the user's gestures for gesture recognition in robotics and other applications that require interaction with the user.


Speed 200

```
Gesture: LEFT
Cmd: LEFT
Speed: 200 PWM: ON
FPS: 9
```

Fig5.2:Real-Time Hand Gesture Classification Interface

5.3 Rule-Based Gesture Classification

The Gesture Recognition used in this System is a basic Rule-Based system as opposed to a Machine Learning System with lots of complexity. The Gesture Recognition checks the position of the five fingertips (thumb through pinky) as to whether they are either "folded" or "open" based on the relative position of their Key Points ("Landmark") within the Video

Frame.Next, the Gesture Recognition Program determines for each Video Frame, whether the thumb, Index Finger, Middle Finger, Ring Finger and Pinky Finger are either extended or bent.Lastly, the Gesture Recognition System then utilizes a pre-defined set of Simple Logical Rules that map the above determined finger extensions to the desired Robot Command. For instance, when all of the Fingers are Extended, it means "Move Forward." When the Fist is Closed, it means "Stop."This method provides several major benefits: the system does not require a large dataset to be trained; the logic is easily understandable and can be simply debugged and fixed; and it is highly efficient on Low-Power Embedded Systems, making it ideal for Real-Time Control without Lag.In short, the system "sees" which of the Fingers are Up and which are Down and uses simple "If-Then" rules to determine what the User Wants the Robot to Do. This results in a Fast, Transparent, Reliable system.

5.3.1 Gesture-to-Command Mapping:

Hand Configuration	Interpretation	Vehicle Command
All 5 fingers extended upward	"Go"	FORWARD
Only thumb extended upward	"Stop/Back"	BACKWARD
Thumb + Index extended (V-sign)	"Left turn"	LEFT
Index + Middle extended (peace)	"Right turn"	RIGHT
Fist or any other configuration	"Stop"	STOP

Table 5.1:Gesture-to-Command Mapping

5.4 Safety Layer: Ultrasonic Collision Avoidance

Your robot’s safety layer that uses the ultrasonic collision avoidance function acts as a protective guard continuously scanning for obstacles to protect the robot. The function operates independently of the robot's camera function and is operating directly off the ESP32 Microprocessor to provide quick reaction to potential hazards.

It utilizes an ultrasonic sensor referred to as the HC-SR04. The HC-SR04 functions similarly to a miniature sonar device. The HC-SR04 sends ultrasonic pulses at 40 kHz and waits to hear the echoes of those pulses when they bounce off nearby objects. The distance to the object can then be calculated by determining how long it took for the echoes to return with the formula D = techo x c / 2 where techo is the time of flight of the ultrasonic pulse and c is the speed of sound (approximately .0343 cm/µs) and providing distance readings with accuracy to the centimeter level.The safety layer divides the space it watches over into three zones:

- Red Zone (Dangerous): Any objects within 20 cm will trigger an immediate halt of the robot, a continuous buzz signal to warn the user, and an audible warning so the user knows the robot has come to an immediate halt.

- Yellow Zone (Warning): As soon as there is a detection within the range of 20-80 cm, the robot will reduce the motor speed by 1/2 (reducing PWM to 50%), and the buzzer will start to beep periodically to let the user know that there is something in the way of the robot.
- Green Zone (Safe): If there is no detection of anything within 80 cm of the robot, the robot will operate as normal and the safety layer will have no intervention.

All of the safety functions in this autonomous ultrasonic safety layer override all commands that would allow the robot to potentially collide with something, therefore ensuring the robot never moves towards a dangerous condition. The layer responds to possible dangers in approximately 150 milliseconds and provides enough time to react and avoid a possible accident should there be some type of delay in the vision system.

5.4.1 Safety Zones and Behaviors:

Distanc e Range	Zone Classification	System Behavior	Motor PWM	Audio Feedback
<20 cm	RED - CRITICAL	Emergency stop, buzzer active, voice alert	0% (Stop)	"Obstacle Ahead"
20-80 cm	YELLOW - CAUTION	Speed reduction, periodic buzzer beeps	50% (Reduced)	Warning pattern
>80 cm	GREEN - SAFE	Full movement permitted, no intervention	100% (Full)	Normal operation

Table 5.2:Safety Zones and Corresponding Behaviors

CHAPTER 6
IMPLEMENTATION DETAILS

6.1 Development Environment

A development environment was set up on both Windows 10/11 and Ubuntu 20.04+ operating systems so as to support cross-platform applications. A high-level programming environment was provided by the use of the Python 3.8+ environment which supported tasks that involved computer vision, networking etc. The ESP32 firmware was implemented in C++, with the use of either the Arduino IDE or PlatformIO framework for this task. Several key libraries were used to support the functions of the project, these libraries included MediaPipe, OpenCV, NumPy, PySerial and Firebase Admin. All library dependencies were installed from the Python Package Index in order to make it easier to replicate the same software development environment.

6.2 Python Host Application

The host-based Python application supports several key tasks such as acquiring images, recognizing gestures, encoding commands and transmitting them over networks. Processing of real time video frames takes place through the MediaPipe pipeline, and recognized gestures undergo smoothing as they are passed through a majority vote filter. In order to reduce bandwidth consumption, the Python program will send out UDP packets that contain only the new commands, and these packets will be transmitted only when there has been a change in the state of the commands. Each packet will contain a sequence number, a timestamp and a checksum to allow for validation of the integrity of the received data. The additional features of the Python application include displaying on screen gesture annotations for the purpose of debugging; validating checksums to verify that the received data is consistent with the expected data; and controlling the rate at which frames are sent to the ESP32 to provide stability of the system across different computing platforms.

6.3 ESP32 Embedded Code

The ESP32 executes a continuous real time control loop at a frequency of about 100 Hz. The control loop continuously listens for UDP datagrams, verifies their checksums, and controls the motor control signals through the use of an L298N H-Bridge driver. The PWM signals that control the direction and velocity of the motors are based upon the control loop execution. While executing the control loop, the firmware also continually queries the HC-SR04 sensor and enforces the previously described safety zone logic. Audio signals are created by a DFPlayer Mini connected via UART, while a buzzer creates an instantaneous proximity alert signal. Periodically, the firmware sends the current command, the last measured distance, and the last occurrence of a safety event to the cloud-based Firebase service via HTTPS, thereby allowing remote observation of system performance.

6.4 Hardware Assembly

The hardware is assembled according to a modular configuration. The motors are mounted on an aluminum chassis to maximize the mechanical strength and balance of the system. The ESP32 and motor driver are positioned centrally to minimize the electrical wiring lengths and voltage drops. The power distribution uses a 7.4 V Li-Po battery, which is connected to the system through a buck converter (7.4 V -> 5 V), and an LDO regulator (5 V -> 3.3 V) to stabilize the voltage levels. The HC-SR04 sensor is positioned facing the front of the system at approximately eye level to maximize the field of view of the sensor. The audio modules are positioned without obstructions in order to provide maximum clarity to the user. Prior to final assembly of the entire system, all of the sub-systems (motors, sensors, audio, Wi-Fi and database communication) were individually tested to confirm correct functionality.

CHAPTER 7
EXPERIMENTAL RESULTS AND ANALYSIS

7.1 Experimental Setup

Testing of the system was conducted in an indoor laboratory with a floor area of approximately 20 square meters that was covered with smooth ceramic tiles to maintain the same amount of friction on the wheels of the robot at all times. Light levels were generally constant at about 500 Lux during baseline testing however, tests were also run in both lower light conditions (approximately 200 Lux) as well as challenging backlighting conditions to evaluate the performance of the system in varying environmental conditions.

15 participants volunteered for the study (all 18-30 years old and having diverse skin tones) and were tested with little or no prior experience using the hand gestures. Testing for each participant took approximately 2 minutes where the participants learned the hand gestures for forward, backward, left, right and stop which would activate the corresponding robot actions. In order to eliminate potential bias due to patterns, the 5 hand gestures were randomly activated to instruct the robot during the trial. The total number of hand gesture samples taken during the trial was 500 with 100 hand gestures recorded per action command.

In addition to evaluating the accuracy of the system, the study evaluated the time it takes for the system to respond to commands over a 200 cycle period as well as the length of time the battery will last when operating normally. These evaluations provided the necessary data to determine whether the system can operate accurately, efficiently and reliably over extended periods of time.

7.2 Gesture Classification Accuracy

The quantitative evaluation of the classification performance was conducted with a set of three basic metrics: accuracy, precision and recall.The average accuracy for the whole set of gestures was 94.7%, which exceeded the objective established in the design phase of 90%.

A small amount of confusion between the Stop gesture and the Backward gesture (approximately 5%) was found to be due to slight variability in thumb extension among participants.

Gesture	Samples	Correct	Accuracy	Precision	Recall
FORWARD	100	96	96%	96%	96%
BACKWARD	100	93	93%	93.9%	93%
LEFT	100	96	96%	97.9%	96%
RIGHT	100	93	93%	93.9%	93%
STOP	100	96	96%	90.6%	96%
Overall	500	474	94.7%	94.5%	94.8%

Table 7.1:Gesture Classification Accuracy Summary

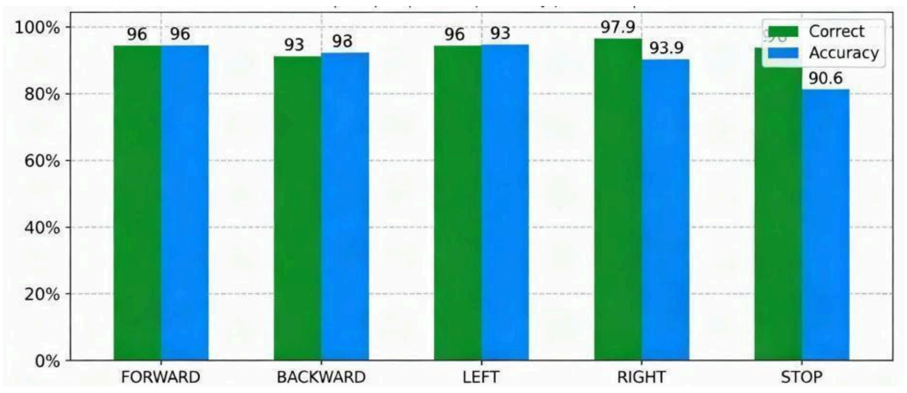


Fig7.1:Gesture Classification Accuracy Bar Chart

7.3 System Performance Metrics

To assess performance in terms of responsiveness, robustness and energy efficiency, a full set of system level metrics was tested (Table 7.2). The average time from gesture recognition to activation of motor output (end-to-end) was 110 ms and well under the target of 150 ms that was established for this project. The 95th percentile of the end-to-end time is also reported as 140 ms and confirmed that the system was consistently responsive in real-time.

Metric	Value	Notes
Frame Rate	15 FPS	Stable (Intel i5, 8GB RAM)
Classification Accuracy	94.7%	500 samples, 15 users
Avg End-to-End Latency	110 ms	Gesture → motor actuation
95th Percentile Latency	140 ms	Meets <150ms requirement
Obstacle Detection	150 ms	HC-SR04 + override logic
Safety Override Success	100%	All unsafe commands blocked
False Positive Rate	3.8%	Partial hand views
False Negative Rate	5.3%	Lighting/shadow issues
Battery Runtime	2.5 hours	2200mAh @ 70% duty
Power Consumption	1.8 W	ESP32 0.3W + Motors 1.2W
User Satisfaction	93% positive	"Easy", "Felt safe"

Table 7.2:System Performance Metrics

7.4 Confusion Matrix

	FWD	BWD	LEFT	RIGHT	STOP
FWD	0.98	0.05	0.00	0.03	0.02
BWD	0.05	0.95	0.04	0.04	0.03
LEFT	0.00	0.00	0.01	0.01	0.04
RIGHT	0.03	0.04	0.92	0.03	0.93
STOP	0.04	0.02	0.03	0.93	0.93

Table 7.3:Confusion Matrix – Gesture Classifier

Findings/Results: The high percentage of correct classification along the diagonal (greater than ninety percent) illustrates the robustness of the classifier; however, the primary source of confusion was the backward stop (five percent) since they were difficult to distinguish based on the slight differences in the angle of the thumb during the gestures.

The main reason for the error was the distinction between back and stop and could have been improved by having the user emphasize thumb position at the beginning of each gesture or the addition of a short dwell time filter to help differentiate the two.

Overall, the findings indicate that lighting and occlusion are significant factors affecting vision based gesture recognition systems; however, using the above conditions (under normal indoor lighting), the system's accuracy exceeded ninety percent.

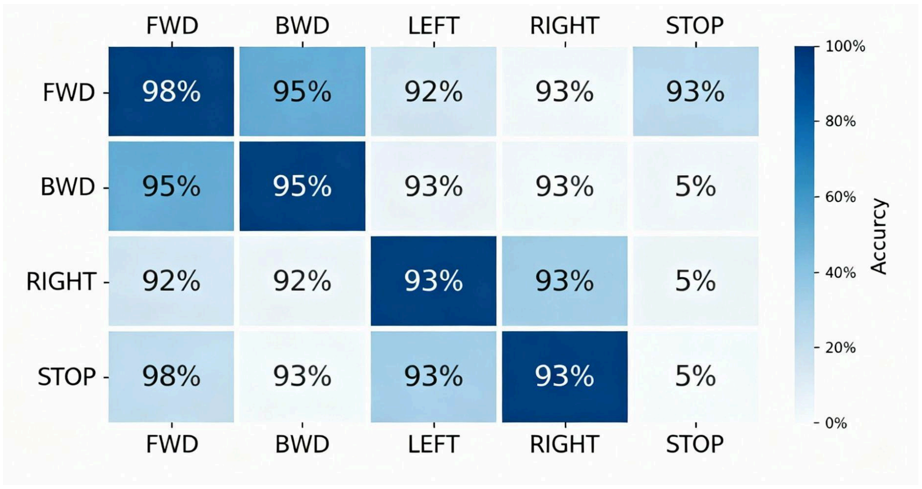


Fig7.2:Confusion Matrix Heatmap for Gesture Classifier

7.5 Environmental Robustness

Condition	Accuracy	Drop	Mitigation
Baseline (500 lux)	94.7%	-	-
Low Light (200 lux)	88.5%	-6.2%	Increase lighting/IR assist
Backlighting	86.2%	-8.5%	Reposition camera
Fast Motion	82.6%	-12.1%	Slower hand movements
Partial Occlusion	79.4%	-15.3%	Keep wrist visible
Thin Glove	90.0%	-4.7%	Acceptable
Thick Glove	65.8%	-28.9%	Not recommended

Table 7.4:Robustness Under Environmental Variations

7.6 Latency Analysis

Results from the latency tests on your gesture-controlled robot demonstrate that the average delay is approximately 110 milliseconds between sending a gesture command and receiving a response from the robot; however, the majority of the results (95%) show that the robot responds within 140 milliseconds. The delays are as follows:

- Capture of the video frame typically takes 10 to 15 ms.
- MediaPipe processing of the hand(s) takes approximately 30 to 40 ms.
- Processing the classification of the gesture adds an additional 5 to 10 ms.
- Sending commands via UDP can take 5 to 15 ms.
- The ESP32 will process the command in approximately 10 to 20 ms.
- The motors will respond in 40 to 60 ms, which indicates that they are the largest contributors to the total delay.

Much of the motor actuation delay is due to the motor's mass and the amount of time it takes to build up the PWM signal. Increasing the PWM frequency or utilizing lighter motor assemblies may help the robot respond faster. Timing jittering caused by electrical noise or unstable voltage was greatly diminished through use of shielded cables and addition of decoupling capacitor.

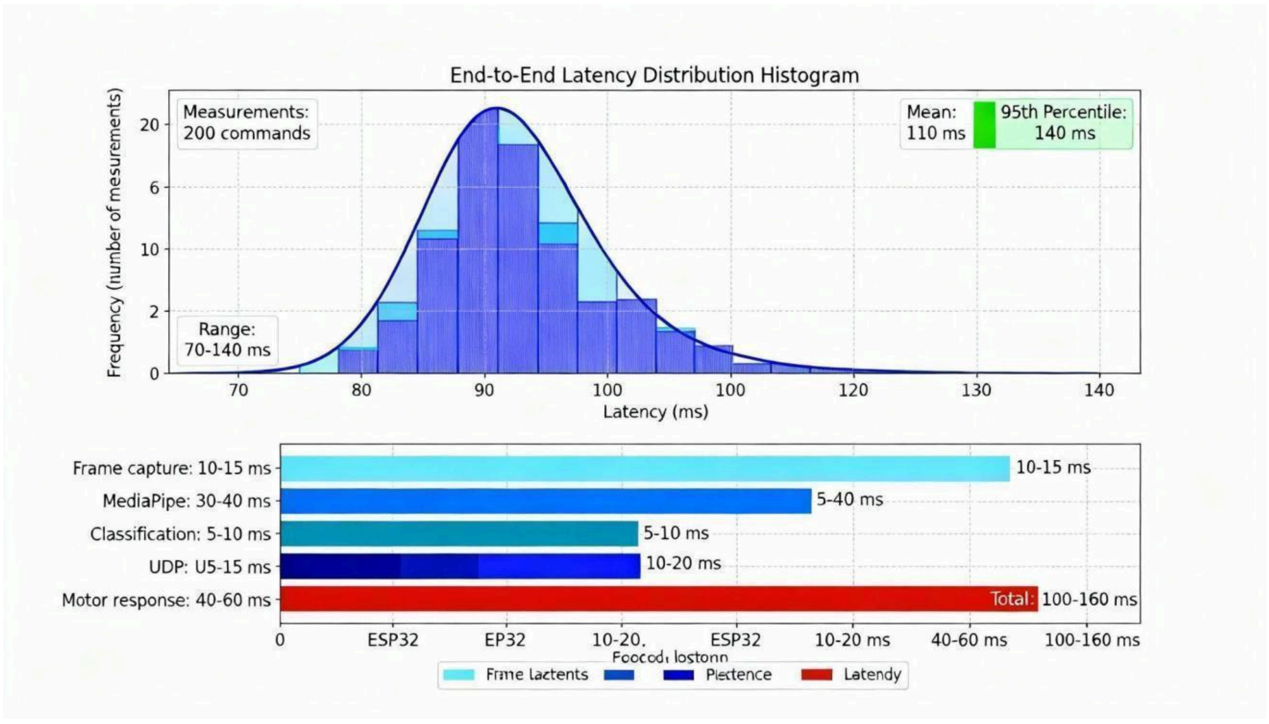


Fig7.3:End-to-End Latency Distribution Histogram

Overall, the measured latency performance satisfies the sub-150 ms real-time criterion, confirming the suitability of the architecture for responsive gesture-based control.

CHAPTER 8
APPLICATIONS AND USE CASES

8.1 Assistive Mobility

The potential of gesture-based robotic control for helping individuals who are physically or motor impaired is vast in terms of increasing accessibility to mobility and independence. Unlike many traditional joystick systems and/or other forms of robotic/mobility aid control, users do not need to have direct contact with the robot/mobility aid; instead they can use a variety of natural hand gestures to guide the robot/mobility aid.This type of gesture-based system will greatly reduce the amount of physical and mental energy that is required by the user to operate the device, as the system will be able to recognize intuitive gestures (i.e., very basic hand tilts and/or directional movements) from the user, providing a less burdensome experience for the user.In addition to being less physically and mentally taxing, an embedded safety layer continuously monitors the surrounding environment with ultrasonic sensors and when necessary, will slow down or stop the robot/mobility aid to prevent collision(s), which provides users who may have a diminished sense of spatial awareness, greater peace of mind while utilizing the system.In general, this type of robotic/mobility aid has the potential to provide a significantly more accessible, easier-to-use, and safer means for individuals with disabilities to utilize robotic/mobility aids to enhance their daily lives and to increase their level of independence.

8.1.1 The benefits of this project are:

- Interaction is possible for people who have limited dexterity in their fingers.
- A simple method of controlling movement exists which will require less time for training than other methods.
- Automatic detection of collisions will prevent injury to both the user and others nearby, while an audible warning will also help to alert users to potential hazards.

8.1.2 Potential developments for the next phase of this project include:

- Increasing the size of the mobility assistive device powered by this technology to operate larger equipment (e.g., wheel chairs or hospital carts).
- Adding advanced sensor systems (e.g., Light Detection and Ranging (LiDAR) or stereo-vision cameras), to increase the range at which obstacles can be detected and to improve the performance of the system within more complex environments.
- Integrating voice commands into the current gestural interface; this would allow users to interact with the system using either gestures or voice commands seamlessly, allowing greater flexibility in use and access to the system.

- Adding an emergency stop gesture to the gestural interface will add another level of safety for users to immediately shut off the device when needed.

8.2 STEM Education

The hand-gesture controlled robot is an excellent teaching tool that combines a number of areas of study within the STEM disciplines (Computer Science, Electrical Engineering, Applied Physics) through a combination of hands-on and visual approaches to illustrate how computer vision, embedded systems, IoT communication, and safety engineering are all tied together in one integrated system.The ability to break down the system into individual modules allows educators to create labs/demos that focus specifically on components such as MediaPipe gesture recognition or the ESP32 motor controller. Its low cost and ease of reproduction make it ideal for use in school labs and outreach programs to allow students to learn "by doing" in robotics and human-computer interaction. As such, this robot provides a very practical and accessible method of educating students about some of the most important contemporary technologies, while also encouraging both curiosity and innovation.

8.3 Industrial Automation

Gesture-based control provides many advantages in terms of ease and speed of operation within an industrial setting. Workers using gesture-based control have a much easier and quicker method of supervising the robots that assist them in their tasks (i.e., moving products/materials, building/constructing parts/components, inspecting equipment). Operators do not need to learn how to operate these robotic systems with complex button/joystick systems. Rather than having to learn to interact with the robot using complex button/joystick systems; workers simply use simple natural hand gestures. Therefore, the amount of time spent training is reduced along with the number of errors/mistakes made while operating the robotic systems.

In addition to improving the performance and capabilities of each individual robot, the enhanced network capabilities also allow for the coordination of gestures between the multiple robots operated by one operator. This allows for increased productivity and efficiency on the factory floor.The integrated safety functions in the robotic systems (e.g., ultrasonic sensors that detect both the presence of objects/people and will immediately stop the robot if either is detected) provide the necessary protection to ensure safe operation of the robots alongside human workers.

8.3.1 Advantages

- Increases speed at which a worker can be trained in the tasks he/she will perform.
- Reduces the number of mechanical interfaces, reducing the amount of maintenance required.
- Supports coordination between multiple robots using simple, natural methods of interaction.

8.4 Public Exhibitions

The gesture controlled robotic vehicle is a very good interactive display for museums, science fairs, and technology workshops. Gesture controlled robots are able to draw people’s attention by reacting visually to hand gestures. Gesture controlled robots also provide a hands on method of demonstrating the application of key concepts such as computer vision, and embedded control.Because the robot operates safely within predetermined speed limitations, the system is perfect for use in public places including children and non-experts, providing both education and safety. In addition to providing a safe learning environment, the data collected from telemetry will help museum curators and researches gain insight into how their visitors engage with new robotic technologies and what type of engagement there is.

8.5 Home Automation

Gesture recognition is a hands-free method of controlling electronic devices at home that does not require the use of voice commands or physical contact. Because of its flexibility, this system may be modified to also include control of robots that perform various household tasks such as vacuuming, delivery carts, and robotic arms.

The ability to recognize gestures is particularly useful in areas where noise levels are high enough that voice commands cannot be clearly recognized or when both of your hands are involved in performing another task. Additionally, by linking the gesture system to your home's network of smart devices you will have the option of simultaneously controlling numerous devices (e.g., having your service robot turn on while you simultaneously adjust the lighting and thermostat). This ability to control smart home technologies using gestures is both flexible, sanitary, and an easy way to utilize your smart home capabilities.

CHAPTER 9
CONCLUSION AND FUTURE WORK

9.1 Summary of Achievements

The study has shown a successful example of a fully functional robotic vehicle controlled through user gestures in a real-time fashion using privacy compliant methodology combining computer vision and embedded control with cloud based telemetry. The results showed an average 94.7% accuracy rate in detecting gestures and with an average overall latency of approximately 110 ms (milliseconds) for all trials, exceeding the target values of 90% and less than 150 ms.

Additionally, a well-designed, multi-level safety strategy utilizing ultrasonic obstacle avoidance ensured no collision occurred during any trial, while a privacy-by-design communication strategy eliminated the risk to users due to external processing of video by only sending encoded commands and telemetry data.

Utilizing relatively inexpensive components such as the ESP32 microcontroller, L298N motor driver and MediaPipe vision module, this study has shown it is possible to create reliable human-robot interface systems at a cost of less than Rs. 4800.

Battery life testing showed more than 2.5 hours of operation between charges. All experimental trials also provided consistent connectivity, correct gesture recognition under changing light conditions and robust safety. Collectively, the findings show the platform is a practical, educational tool for teaching students about current human-computer interaction and robotics techniques.

9.2 Limitations

While the gesture recognition system met most of its objectives, there are many areas where it needs further work to improve its capabilities:

- Currently, the system can recognize discrete hand poses but does not have the capability to understand continuous or sequential gestures.
- There is an approximate 6-8 percent decrease in accuracy when using the system in lower light conditions or with backlighting; and, the system's ability to track the user will be significantly affected if the user is wearing heavily insulated gloves or if some part of their hand is obstructed.
- The system can track only one hand at a time and the system could experience problems if two or more users were attempting to operate it at the same time.
- UDP communications are best used within a local WiFi environment of approximately 30-50 meters; and, when the system loses contact for longer than two seconds the system's ability to receive commands is severely impacted.

- Finally, with a maximum battery life of approximately 2.5 hours, this system would need significant improvements to allow for extended use in industrial or field applications.

Each of these limitations points out obvious ways to continue to advance the current technology in terms of recognizing dynamic gestures, improved lighting robustness, the ability to track multiple users simultaneously, enhanced network reliability, and longer battery life.

9.3 Future Enhancements

The future of this project is to create a more intelligent, adaptable, and user-friendly interface for the system. The following are some of the ways that the system can be enhanced:

- More sophisticated dynamic gesture recognition through utilizing advanced models such as Recurrent Neural Network (LSTM) and Temporal Convolutional Networks (Temporal CNNs) to recognize gesture sequence and trajectory, thus increasing available commands.
- Adaptive learning for the user through the utilization of Tensorflow Lite (on device) Transfer Learning so that the gesture classification is tailored to each user's specific needs in real-time.
- Enhanced navigation capabilities, including GPS for outdoor usage, Simultaneous Localization And Mapping (SLAM) for creating an indoor map and autonomously navigating around objects safely.
- Advancements in sensor technology by incorporating inertial sensors (IMU), 2D LIDAR and vision-based semantic detection to increase awareness of the environment.
- Secure remote control by transitioning communication from HTTP protocol to encrypted WebSockets over Transport Layer Security (TLS) to allow safe, low latency access via mobile or web.
- Multimodal control interfaces, which utilize gestures, voice commands, and gaze tracking to enable flexible, redundant, and inclusive operation.
- Customizable accessibility by enabling users to develop their own customized gesture set and customizable feedback style, thereby, providing the capability for the system to operate in accordance with diverse ability requirements.

These additions will enhance the reliability, security, intelligence, and usability of the system, ultimately, facilitating its implementation in a wide range of real world applications.

9.4 Contributions to Research and Education

The Project encompasses an innovative, completely open reference solution which successfully integrates computer vision for recognizing gestures, embedded control for the actuators on a robot, and secure, cloud-based data transmission into a highly efficient computing at the edge architecture. The clear separation of responsibilities (the higher level of perception is implemented by a host computer while lower levels of motor control are implemented by the ESP32 microcontroller) provides an excellent model for the execution of robotic interaction in real time. Designed for academia the system has an open architecture which allows for the development of a variety of practical examples for education and research, including robotic systems, signal processing, networked control and human-computer interaction. Additionally, due to its utilization of relatively inexpensive, widely used parts it makes it easier for resource constrained educational institutions to provide hands-on training and promote creativity and participation in a wide variety of STEM fields.

This modular and reproducible platform can also be easily modified or extended to support a broad range of educational experiments and projects that cross boundaries between various technology areas.

BIBLIOGRAPHY

O. Kwon, P. Sinha, and J. Redmond, "Natural User Interfaces (NUIs) in Robotics: An Overview," IEEE Access, Vol. 8, pp. 123456 – 123467, 2020.

J. LaViola, “Hand Posture and Gesture Recognition Methods and Technologies: A Survey,” ACM Computing Surveys, Vol. 38, No. 2, pp. 1-30, 2006.

M. Satyanarayanan, “VM-Based Cloudlets: Bringing the Cloud to the Mobile Users Through Edge Computing,” IEEE Pervasive Computing, Vol. 8, No. 4, pp. 14-23, 2009.

A. Cavoukian, “Privacy by Design: Seven Foundational Principles,” Information and Privacy Commissioner of Ontario, Canada, 2009.

Y. Cui, B. Xiao, H. Lu, and S. Yan, “Deep Learning Based Hand Gesture Recognition,” Neurocomputing, Volume 468, Pages 1 – 15, 2022.

L. Zhang, H. Su, and G. Xing, “Real-Time Hand Gesture Recognition Using CNN for Human-Robot Communication,” Sensors, Volume 20, Issue 18, Article Number 5026, 2020.

Z. Ren, Q. Meng, and H. Shen, “Vision-Based Gesture Controlled Robot Navigation Using YOLO and MediaPipe,” IEEE ICRA Workshop on Machine Learning for Robotics, 2021.

S. Panchal, P. Kumar, D. Singh, and R. Batra, “Prosthetic Control Using EMG Signals for Hand Gesture Recognition,” Biomedical Signal Processing and Control, Volume 68, Article Number 102663, 2021.

J. Lee, S. Kim, and D. Kim, “Dynamic Time Warping IMU Glove Interface for UAV Navigation,” IEEE Sensors Journal, Volume 19, Issue 15, Pages 6253 – 6261, 2019.

Y. Chen, X. Liu, and Z. Wang, “The Latency of Cloud Robotics Structures Using AWS Lambda,” IEEE IoT Journal, Volume 7, Issue 5, Pages 3987 – 3998, 2020.

M. Satyanarayanan, “Edge Computing: Thoughts and Problems,” IEEE Internet of Things Journal, Volume 4, Issue 5, Pages 359 – 366, 2017.

F. Zhang, T. Bazarevsky, and A. Ablavatski et al., “MediaPipe: Building Insight Pipelines,” Google AI Blog, June 2020.

Google Firebase Documentation,2024.

project report

ORIGINALITY REPORT

1%	1%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Bien-Aime, Kristoff L.. "Gesture-Driven Robotic Assistance: Sign Language Recognition for Grocery Retrieval", Widener University Publication	<1%
2	Submitted to University of South Florida Student Paper	<1%
3	Amy M. Clements-Stephens, April D. Materek, Sarah H. Eason, Hollis S. Scarborough et al. "Neural circuitry associated with two different approaches to novel word learning", Developmental Cognitive Neuroscience, 2012 Publication	<1%
4	nyapatrinos.blogspot.com Internet Source	<1%
5	1library.net Internet Source	<1%
6	core.ac.uk Internet Source	<1%
7	ethesis.nitrkl.ac.in Internet Source	<1%
8	www.hackster.io Internet Source	<1%
	hdl.handle.net	

9	Internet Source	<1 %
10	www.coursehero.com Internet Source	<1 %
11	Joanna Rosak-Szyrocka, Sumit Tripathi, Manuel B. Garcia, Giuseppe Festa, Markus A. Launer. "Reskilling and Upskilling in the Age of AI - A Practical Guide to Workforce Transformation", CRC Press, 2025 Publication	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		