

**HAND GESTURE CONTROLLED ROVER**

### SY B.Tech. Minor Project Implementation Report (SEM IV)

*submitted in partial fulfillment of the requirements for the award of the degree*

*of*

#### Bachelor of Technology

*in*

#### ELECTRONICS and TELECOMMUNICATION ENGINEERING BY

**Mayank Soni, [S2307-812]**

#### Tuhinansh Sharma, [S2306-1188] Pranav Khatavkar, [S2309-1397] Siddhant Mishra, [S2304-490]

**SCHOOL OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING**

#### MIT ACADEMY OF ENGINEERING, ALANDI (D), PUNE-412105 MAHARASHTRA (INDIA)

**MAY, 2024**



### CERTIFICATE

It is hereby certified that the work which is being presented in the SY B.Tech. Minor Project Design Report entitled ***HAND GESTURE CONTROLLED ROVER* ,** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Electronics and Telecommunication engineering** and submitted to the **School of Electronics and Telecommunication engineering of MIT Academy of Engineering, Alandi(D), Pune, Affiliated to Savitribai Phule Pune University (SPPU), Pune** is an authentic record of work carried out during an Academic Year 2023-2024, under the supervision of **Prof. Isaq Shikalgar, School of Electronics and Telecommunication Engineering.**

|  |  |  |
| --- | --- | --- |
| **Mayank Soni** | **202201090031** | **[S2307-812]** |
| **Tuhinansh Sharma** | **202201060010** | **[S2306-1188]** |
| **Pranav Khatavkar** | **202201090028** | **[S2309-1397]** |
| **Siddhant Mishra** | **202201040036** | **[S2304-490]** |

#### Date:18th June, 2024

|  |  |
| --- | --- |
| *Signature of Project Advisor* | *Signature of Dean* |
| **Prof. Isaq Shikalgar** | **Dr. Dipti Sakhare** |
| School of Electronics & Telecommunication Engineering, | School of Electronics & Telecommunication Engineering, |
| MIT Academy of Engineering, Alandi(D), Pune | MIT Academy of Engineering, Alandi(D), Pune |

**(STAMP/SEAL)**

|  |  |
| --- | --- |
| *Signature of Internal examiner/s* | *Signature of External examiner/s* |
| *Name………………………………* | *Name………………………………* |
| *Affiliation…………………………* | *Affiliation…………………………* |

### ACKNOWLEDGEMENT

We extend our heartfelt gratitude to our esteemed project advisor/guide, Prof. Isaq Shikalgar, whose unwavering encouragement and invaluable guidance have been instrumental in the successful completion of this project. His expertise and support have illuminated our path, enriching our learning experience significantly.

We also wish to express our sincere appreciation to the respected School Dean, Dr. Dipti Sakhare, for her consistent motivation and unwavering support throughout this endeavor. Her encouragement has been a driving force behind our efforts.

Furthermore, we acknowledge and thank all the esteemed staff and faculty members whose wealth of experience, insightful advice, and unceasing cooperation have played a pivotal role in shaping this project. Their contributions have been invaluable, and we are truly grateful for their guidance and support, which have been indispensable throughout this journey.

1.Mayank Soni 2.Tuhinansh Sharma 3.Pranav Khatavkar 4.Siddhant Mishra

### ABSTRACT

In today's dynamic era of technology, urban mobility constantly seeks innovative solutions to enhance efficiency and accessibility. One such groundbreaking advancement lies in the Hand Gesture Controlled Rover, a marvel of technology seamlessly integrating an ESP32 Wi-Fi module, motor driver L298 2A, and socket programming. This rover isn't just a technological feat; it's a transformative tool with diverse applications, including aiding individuals with disabilities.

Imagine a world where mobility knows no bounds, where individuals can navigate their surroundings with unprecedented ease and freedom. The Hand Gesture Controlled Rover embodies this vision, offering a revolutionary means of transportation for people of all abilities. Through intuitive hand gestures, users can command the rover, directing its movements and actions with precision and fluidity.

This rover isn't just about mobility; it's about empowerment. For individuals with disabilities, it represents a lifeline to independence and autonomy. Whether traversing urban streets or navigating indoor spaces, the Hand Gesture Controlled Rover provides a reliable and intuitive means of transportation, breaking down barriers and opening doors to new possibilities.

But the impact of this innovation extends far beyond individual empowerment. In urban settings, the Hand Gesture Controlled Rover offers a sustainable and efficient mode of transportation, reducing reliance on traditional vehicles and mitigating environmental impact. With its advanced technology and seamless integration into existing infrastructure, this rover is changing how we move in cities, making it easier for everyone to get around while taking care of our planet.

From bustling city streets to remote rural areas, the Hand Gesture Controlled Rover is revolutionizing the way we move and interact with our environment. It's more than just a vehicle; it's a symbol of progress, innovation, and inclusivity. As we look to the future, let us embrace the potential of this remarkable technology to create a world where mobility knows no limits, and everyone has the freedom to explore, engage, and thrive.

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **Fig No.** | **Headings Page no** | |
| **3.1.1** | **Block diagram of proposed design** | **10** |
| **3.2.1** | **Circuit Diagram ESP32 (Microcontroller) and Motor driver** | **12** |
| **4.1.1** | **Flowchart of software hardware interface** | **16** |
| **4.1.2** | **Flowchart of Server** | **17** |
| **4.1.3** | **Flowchart of client** | **18** |
| **4.3.1 to 4.3.6** | **Figures Showing Hand Gestures** | **21 -22** |
| **4.4.1** | **Hardware Image** | **23** |

|  |  |  |
| --- | --- | --- |
| **LIST OF TABLES** | | |
| **Table. No.** | **Table. Name** | **Page No.** |
| **2.1.1** | **Table for Paper 1** | **4** |
| **2.1.2** | **Table for Paper 2** | **5** |
| **2.1.3** | **Table for Paper 3** | **6** |
| **2.1.4** | **Table for Paper 4** | **7** |
| **2.1.5** | **Table for Paper 5** | **8** |
| **3.4.1** | **Bill of Materials** | **14** |

**CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acknowledgements | | | | i |
| Abstract | | | | ii |
| List of Figures | | | | iii |
| List of Tables | | | | iv |
| 1. | Introduction | | | 1 |
|  | 1.1 | Motivation for the project | |  |
|  | 1.2 | Problem Statement | |  |
|  | 1.3 | Objectives and Scope | |  |
|  | 1.4 | Organization of report | |  |
| 2. | Literature Survey | | | 4 |
|  | 2.1 | List of Referred Books and Papers | |  |
|  |  | 2.1.1 | Survey 1 |  |
|  |  | 2.1.2 | Survey 2 |  |
|  |  | 2.1.3 | Survey 3 |  |
|  |  | 2.1.4 | Survey 4 |  |
|  |  | 2.1.5 | Survey 5 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3. | System Design | | | 10 |
|  | 3.1 | Block diagram of Proposed System | |  |
|  | 3.2 | Circuit Design | |  |
|  | 3.3 | Hardware and Software Requirements | |  |
|  | 3.4 | Specifications | |  |
|  | 3.5 | Bill of Material | |  |
| 4. | Implementation and Results | | | 15 |
|  | 4.1 | | Algorithm and flowcharts |  |
|  | 4.2 | | Procedure and Setup |  |
|  | 4.3 | | Software Result |  |
|  | 4.4 | | Hardware Implementation |  |
|  | 4.5 | | Discussion |  |
| 5. | Conclusion and Future Scope | | | 24 |
| 6 | References | | | 26 |
| 7 | Appendix -1 (Code) | | | 28 |
| 8. | Appendix -2 | | | 36 |

## Chapter 1 Introduction

This chapter introduces the motivations, problem statement, and objectives that form the foundation of the project. It explores the potential of hand gesture recognition technology to revolutionize user interaction, particularly for individuals with disabilities, and outlines the goals and scope of developing a hand gesture-controlled rover system. By providing a comprehensive overview, this chapter sets the stage for understanding the project's significance and direction.

### Motivation

The increasing integration of smart technologies into various aspects of daily life has sparked a wave of innovation, promising to enhance efficiency, accessibility, and user experience across different domains. Similar to how smartphones revolutionized communication, the application of these technologies holds the potential to transform conventional systems into streamlined, user-friendly solutions.

Imagine controlling devices with simple hand gestures, empowering individuals, including those with disabilities, to interact with their environment effortlessly. Such innovations not only improve accessibility but also redefine the way we interact with technology, making it more intuitive and inclusive.

Our project is driven by the desire to harness the power of hand gesture recognition technology to create a seamless and intuitive user experience. By developing a hand gesture-controlled rover system, we aim to provide users, including individuals with disabilities, with an innovative means of navigation and interaction, regardless of their physical abilities.

Through this project, we seek to demonstrate the practical applications of hand gesture recognition technology in enhancing accessibility and user interaction for individuals with disabilities. By focusing on user-centric design principles and leveraging cutting-edge technologies, we aim to create a solution that not only meets the needs of individuals with disabilities but also offers a glimpse into the future of inclusive and empowering assistive technology.

Ultimately, our goal is to develop a hand gesture-controlled rover system that not only facilitates mobility and navigation but also promotes inclusivity, independence, and empowerment for all users, including those with disabilities. By pushing the boundaries of technology and design, we aspire to create a solution that enriches the lives of users and inspires further innovation in the field of assistive technology.

### Problem Statement

The task is to develop a hand gesture-controlled rover system that empowers users, including individuals with disabilities, to command and maneuver the rover through intuitive hand gestures. This system aims to enhance accessibility and user interaction, enabling individuals of all abilities to navigate their surroundings with ease and independence.

### Objective and Scope Objective

* + - To study socket programming (image processing).
    - To interface image processing with microcontroller (ESP32).
    - To interface motor driver with microcontroller.
    - To recognize the real time hand gestures using image processing and navigate the rover.

This project encompasses three primary objectives. Firstly, it seeks to explore socket programming techniques, with a focus on image processing. Secondly, the project aims to establish communication between image processing and a microcontroller, specifically the ESP32. Thirdly, it endeavors to interface a motor driver with the microcontroller. Lastly, the project aims to implement real-time hand gesture recognition using image processing and utilize this data to navigate the rover. By achieving these objectives, the project aims to create an integrated system capable of recognizing hand gestures in real-time and controlling the rover's movements accordingly, thus demonstrating the practical application of image processing techniques in rover navigation.

### Organization of report

Organization of the report provides a brief overview of the structure and contents of the report.

* + 1. Introduction
       - Motivation for the Project: Discusses the inspiration behind developing a hand gesture- controlled rover and its potential benefits, particularly for individuals with disabilities.
       - Problem Statement: Outlines the challenge of creating a system that allows users to control a rover using hand gestures.
       - Objectives and Scope: Defines the project's main goals, including studying socket programming, interfacing image processing with a microcontroller, and implementing real- time hand gesture recognition.
    2. Literature Survey
       - List of Referred Books and Papers: Summarizes key research papers and books that underpin the project's theoretical foundation.
       - Survey Summaries: Reviews various papers on gesture recognition and robot control, highlighting methodologies and findings relevant to the project.
    3. System Design
       - Block Diagram of Proposed System: Presents the overall architecture of the system, including the flow of information and control.
       - Circuit Design: Details the electrical schematics necessary for implementing the system.
       - Hardware and Software Requirements: Lists the components and software tools needed to build the system.
       - Specifications: Defines the technical requirements and constraints of the project.
       - Bill of Material: Enumerates all the hardware components required, along with their specifications.
    4. Implementation and Results
       - Algorithm and Flowcharts: Describes the algorithms and provides flowcharts illustrating the process of hand gesture recognition and rover control.
       - Procedure and Setup: Explains the steps taken to set up the hardware and software environments.
       - Simulation Result: Presents the outcomes from testing the system in a simulated environment.
       - Hardware Result: Showcases the results from real-world testing of the system.
       - Discussion: Analyzes the results, discussing the system's performance and potential improvements.
    5. Conclusion and Future Scope
       - Conclusion: Summarizes the project's achievements and the effectiveness of the hand gesture- controlled rover.
       - Future Scope: Suggests possible enhancements and future research directions to improve the system.
    6. References
       - Lists all the sources cited throughout the report, including books, papers, and online resources.
    7. Appendix - 1 (Pseudo Code)
       - Provides the pseudo code used in the development of the system, aiding in understanding the implementation details.
    8. Appendix - 2
       - Contains additional supporting information, diagrams, or data that complement the main text of the report.

## Chapter2 Literature Survey

This chapter presents a review of the books and research papers that form the foundation of the project. By examining existing literature, it aims to highlight the current advancements, methodologies, and technologies relevant to the system design, providing a contextual background and justification for the chosen approach.

### List of Referred Books and Papers

* + 1. Details of paper [1]: Table 2.1.1 provides details of referred research paper [1].

*Table 2.1.1: - Paper [1]*

|  |  |
| --- | --- |
| **Title** | Gesture based wireless robotic control using image processing |
| **Author name** | Gaurav Chauhan, Prasad Chaudhari |
| **Year** | 2015 |
| **Parameters** | HCI (Human computer interaction)  Zigbee |
| **Research Paper** | <https://ieeexplore.ieee.org/document/7449642> |
| **References** | G. Chanhan and P. Chandhari, "Gestures based wireless robotic control using image processing," *2015 5th Nirma University International Conference on Engineering (NUiCONE)*, Ahmedabad, India, 2015, pp. 1-7, doi: 10.1109/NUICONE.2015.7449642. |

### Summary of paper [1]:

Many people in this world are known to use gestures, a powerful communication medium amongst humans. They say actions speak louder than words. Communicating with gestures is a powerful method. Industrial as well as robots used for home purpose are often controlled by remote controllers. The use of gestures as a way to control can help a lot providing ease and can be more useful for the handicapped. Gestures provide a separate complementary modality to speech for expressing one's ideas. Furthermore, Image processing is one of the most effective methods used to process image signals. The main purpose of gesture recognition research is to identify a particular human gesture and convey information to the user pertaining to the individual gesture. Sign language recognition from hand motions or postures is an active area in gesture recognition research for Human Computer Interaction (HCI). This paper aims to brief the practical approach of robotics through an effective method of image processing using gestures as a mode of control.

* + 1. Details of paper [2]: Table 2.1.2 provides details of referred research paper [2].

*Table 2.1.2: - Paper [2]*

|  |  |
| --- | --- |
| **Title** | Towards Robust Ego-Centric Hand Gesture Analysis for Robot Control |
| **Author name** | Hongyong Song, Weijiang Feng, Naiyang Guan |
| **Year** | 2016 |
| **Parameters** | Hand detection and segmentation Fingertips detection |
| **Research Paper** | <https://ieeexplore.ieee.org/document/7888345> |
| **References** | H. Song, W. Feng, N. Guan, X. Huang and Z. Luo, "Towards robust ego-centric hand gesture analysis for robot control," *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*, Beijing, China, 2016, pp. 661-666, doi: 10.1109/SIPROCESS.2016.7888345. |

### Summary of paper [2]:

In this paper, a hand gesture-based control design is proposed for mobile robots. Mobile robots can move according to the control signals encoded by hand gestures. The gesture region is segmented from complicated backgrounds and the gestures are recognized by using some techniques such as image processing, image filtering processing, morphological image processing, image contour processing, etc. Then a template matching algorithm is proposed with the help of the invariant moment matching method to recognize the hand gestures. The recognition results are decoded as feedback information to control the mobile robots. Finally, some simulation results are provided to validate the proposed control algorithm.

* + 1. Details of paper [3]: Table 2.1.3 provides details of referred research paper [3].

*Table 2.1.3: - Paper [3]*

|  |  |
| --- | --- |
| **Title** | Hand Gesture Based Control Strategy for Mobile Robots |
| **Author name** | Hang Zhao, Jiangping Hu, Yuping Zhang, Hong Cheng |
| **Year** | 2017 |
| **Parameters** | Gesture recognition, Image processing, Template matching, Robot control. |
| **Research Paper** | <https://ieeexplore.ieee.org/document/7978217> |
| **References** | H. Zhao, J. Hu, Y. Zhang and H. Cheng, "Hand gesture based control strategy for mobile robots," *2017 29th Chinese Control And Decision Conference (CCDC)*, Chongqing, China, 2017, pp. 5868-5872, doi: 10.1109/CCDC.2017.7978217. |

### Summary of paper [3]:

In this paper, a hand gesture-based control design is proposed for mobile robots. Mobile robots can move according to the control signals encoded by hand gestures. The gesture region is segmented from complicated backgrounds and the gestures are recognized by using some techniques such as image processing, image filtering processing, morphological image processing, image contour processing, etc. Then a template matching algorithm is proposed with the help of the invariant moment matching method to recognize the hand gestures. The recognition results are decoded as feedback information to control the mobile robots. Finally, some simulation results are provided to validate the proposed control algorithm.

* + 1. Details of paper [4]: Table 2.1.4 provides details of referred research paper [4].

*Table 2.1.4: - Paper [4]*

|  |  |
| --- | --- |
| **Title** | Hand Gesture Recognition Using Convexity Hull Defects to Control an Industrial Robot |
| **Author name** | Srinivas Ganapathyraju |
| **Year** | 2013 |
| **Parameters** | Hand Gesture Recognition, Robotics, Control, Skin Detection, Convexity Hull. |
| **Research Paper** | <https://ieeexplore.ieee.org/document/6734047> |
| **References** | S. Ganapathyraju, "Hand gesture recognition using convexity hull defects to control an industrial robot," *2013 3rd International Conference on Instrumentation Control and Automation (ICA)*, Ungasan, Indonesia, 2013, pp. 63-67, doi: 10.1109/ICA.2013.6734047. |

### Summary of paper [4]:

This paper presents a method for hand gesture recognition to control a 6-axis industrial robot. The image is acquired by means of a web cam system, and undergoes several processing stages before a meaningful recognition can be achieved. Some of these stages include skin detection to effectively capture only the skin region of the hand, noise elimination, application of the convex hull algorithm to get the outline of the hand, and apply convexity hull defects algorithm to determine the finger count. Once the finger count has been determined, the information is transmitted via serial communication to the industrial robot controller. The end effector of the robot moves in four different directions based on the finger count input received from the hand gesture recognition module.

* + 1. Details of paper [5]: Table 2.1.5 provides details of referred research paper [5].

*Table 2.1.5: - Paper [5]*

|  |  |
| --- | --- |
| **Title** | Designing and Implementation of a Wireless Gesture Controlled Robot for Disabled and Elderly People |
| **Author name** | Mahbuba Alam, Mohammad Abu Yousuf |
| **Year** | 2019 |
| **Parameters** | Gesture, image processing, wheelchair, Arduino, sensor, wirelessly, embedded system |
| **Research Paper** | <https://ieeexplore.ieee.org/document/8679290> |
| **References** | M. Alam and M. A. Yousuf, "Designing and Implementation of a Wireless Gesture Controlled Robot for Disabled and Elderly People," *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Cox'sBazar, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ECACE.2019.8679290. |

### Summary of paper [5]:

Gesture controlled robot should perform its job with the embedded system, a combination of hardware and software that is designed for performing dedicated tasks for specific users which can be controlled by hand gestures. In this paper, we propose a gesture-controlled robot for physically challenged and elderly people. The robot uses motion sensors to recognize five gestures of hand. These five hand gestures are employed to control five directions such as stop or steady, forward, backward, left and right. In this paper, the gesture has been recognized by motion sensors and without any image processing to keep the system simple & efficient. The prototype device can be modified into a wheelchair, trolley bed or any physical platform consisting of more than two wheels. An Arduino Nano v3 embedded system with accelerometer gyroscope held by the user and the signals will be transferred wirelessly to the motors of the platform intended to move it as per the users’ hand gestures in order to carry themselves in their surrounding environment. Machine learning is used to classify hand gestures accurately. The result shows that the classification accuracy is near to 94% (93.8%).

### Conclusive summary of literature survey:

The five papers collectively highlight advancements in gesture-based control systems for robotics, showcasing their potential to provide intuitive and accessible interfaces. Gesture recognition, enhanced by image processing techniques, offers significant benefits, particularly for individuals with disabilities, by enabling an alternative mode of interaction for controlling robots.

The papers explore various methodologies and applications, such as mobile robots controlled by hand gestures through image processing and template matching algorithms. One paper details the control of a 6-axis industrial robot via hand gestures, using stages like skin detection and convex hull algorithms to interpret finger counts, demonstrating the practicality and effectiveness of these systems in different robotic contexts.

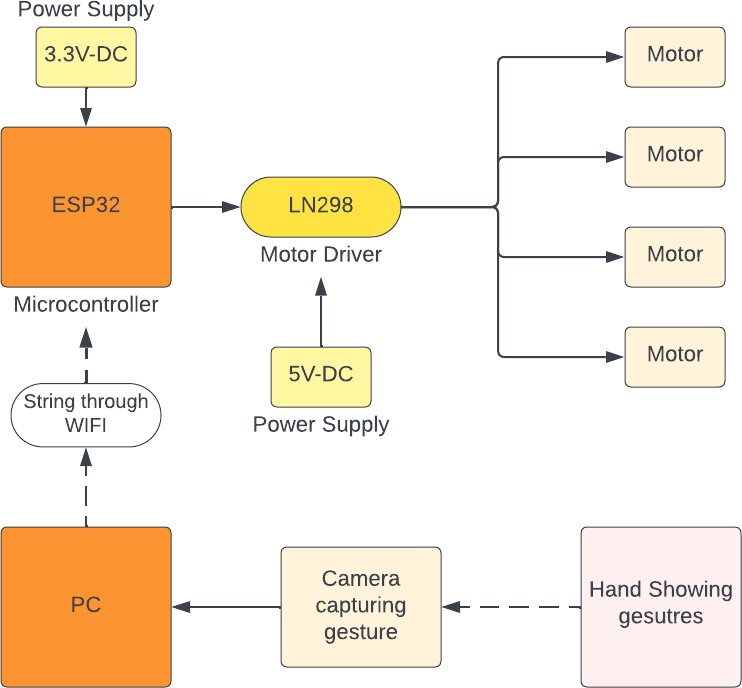
Additionally, a paper presents a gesture-controlled robot designed for physically challenged and elderly individuals, using motion sensors to simplify the system. This approach highlights the versatility of gesture recognition technology, showing its adaptability to various user needs and contexts. Collectively, these studies illustrate the potential of gesture recognition to transform human- robot interaction, promoting inclusivity and ease of use.

## Chapter 3 System Design

This chapter deals with the architecture and components of the proposed system, outlining its design through block diagrams, circuit schematics, and the selection of both hardware and software elements. It provides a detailed examination of how the various components, including the ESP- WROOM32 microcontroller and motor driver, are integrated to create a responsive system for gesture recognition and motor control. Through this comprehensive overview, the chapter aims to present a clear understanding of the system's functionality and the rationale behind the chosen design and components. In reference to research paper [1] and research paper [5].

### Block diagram of proposed system

In Fig 3.1.1, the block diagram showing basics of system-flow explains how the system works. The working is as follows: The ESP-WROOM32, a highly versatile microcontroller embedded within a single integrated circuit, serves as an important component in this setup. Its function is to detect and receive incoming string data after the gesture being processed by an ML model for image processing. This module will be seamlessly connected to the Motor driver L298 2A. The ESP-WROOM32, equipped with CPUs, memory, and customizable input/output peripherals, operates based on code instructions, provided by the programmer. This setup, strategically programmed or operated based upon socket programming and embedded c code for microcontroller, aims to efficiently recognize hand gestures and process the respective data transmission over Wi-Fi. As gesture gets detected via camera (embedded in PC), string is sent to microcontroller ESP32 via socket programming. Subsequently, the ESP32 processes this information, leveraging the programmed parameters, and generates output. This output can be conveniently accessed by motor driver and consequently the operation on motors is tuned.



***Fig 3.1.1: - Block diagram showing basics of system-flow***

### Circuit Diagram:

The Fig 3.2.1, gives overview of the circuit diagram:

* + - ESP32 Microcontroller: Controls the motors.
    - L298N Motor Driver: Drives the four DC motors.
    - Motors: Two motors on each side (Right and Left).
    - Power Supply: 7-12V DC for the motor driver and motors. The connections are as follows:

#### Power Supply:

* + - 7-12V DC to the L298N's +12V and GND.

#### ESP32 to L298N:

**Motor A (Right Motors):**

* + - IN1 to GPIO 5.
    - IN2 to GPIO 18.
    - ENA (PWM) to GPIO 22.

#### Motor B (Left Motors):

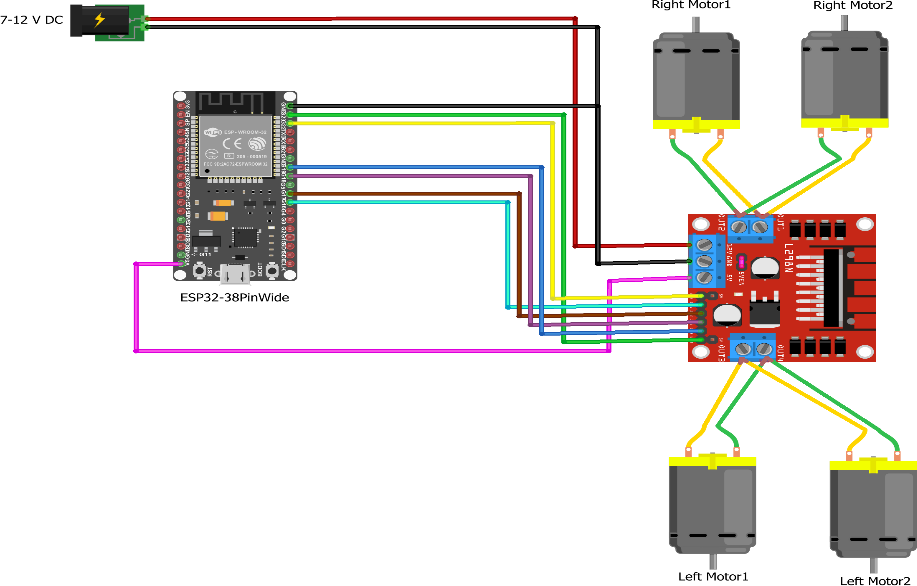
* + - IN3 to GPIO 19.
    - IN4 to GPIO 21.
    - ENB (PWM) to GPIO 23.

#### Power and Ground:

* + - ESP32's 5V and GND to L298N's 5V and GND.

#### Motor Connections:

* + - Right Motors to OUT1 and OUT2.
    - Left Motors to OUT3 and OUT4.



ESP WROOM 32

***Fig 3.2.1: - Circuit Diagram of ESP WROOM 32 and Motor Driver***

### Selection of components and mechanisms for designed project

The careful selection process for the hardware components and mechanisms utilized in the design of the hand gesture-controlled rover system is very crucial.

The components selected used for project are:

* + - * ESP-WROOM32
      * Motor Driver L298 2A
      * Jumper wires
      * 3.7V 3800mAh Rechargeable LiPo battery
      * 100 RPM BO Motors

The ESP-WROOM32, a highly versatile microcontroller embedded within a single integrated circuit, serves as an important component in this setup. Its function is to detect and receive incoming string data after the gesture being processed by an ML model for image processing. This module will be seamlessly connected to the Motor driver L298 2A. The ESP-WROOM32, equipped with CPUs, memory, and customizable input/output peripherals, operates based on code instructions, provided by the programmer. This setup, strategically programmed or operated based upon socket programming and embedded c code for microcontroller, aims to efficiently recognize hand gestures and process the respective data transmission over Wi-Fi. As gesture gets detected via camera (embedded in PC), string

is sent to microcontroller ESP32 via socket programming. Subsequently, the ESP32 processes this information, leveraging the programmed parameters, and generates output. This output can be conveniently accessed by motor driver and consequently the operation on motors is tuned.

This image processing model and data processing through the ESP32 presents an opportunity to seamlessly operate the rover and enables Human-Computer interaction, referred through paper [1] and [6]. This ultimately enhances operational efficiency, provides real-time insights into system operation, and enables better interaction and service optimization (if needed).

### Software Requirements

To ensure successful development and implementation of the hand gesture-controlled rover system, specific software tools are essential. The required software tools are:

1.Jupyter Notebook IDE 2.Arduino IDE

### Hardware Requirements

Building the hand gesture-controlled rover system necessitates key hardware components. The essential hardware elements crucial for constructing and operating the system effectively are as follows:

1. Motor Driver L298 2A
2. ESP-WROOM32
3. Jumper wires
4. 3.7V 3800mAh Rechargeable LiPo battery
5. 100 RPM BO Motors

### 3.4 Bill of Material:

Table 3.4.1 provides a breakdown of the hardware components required for the project, along with their respective prices. The total cost analysis offers insight into the budget required for implementing the hand gesture-controlled rover system.

This bill of materials outlines the essential components necessary for constructing the rover, including the ESP-WROOM32 microcontroller, motor driver, jumper wires, 100 rpm BO motor with wheels, and miscellaneous items such as the battery and chassis. The prices listed reflect the current market rates for each component, contributing to the total cost of the project.

By assessing the cost analysis, the budget could be estimated accordingly and ensure the availability of all required components for successful implementation.

*Table 3.4.1: - Cost of the project*

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Component name** | **Price (in rupees)** |
| **1** | ESP-WROOM32 | **549/-** |
| **2** | Motor driver | **284/-** |
| **3** | Jumper wires | **20/-** |
| **4** | 100 rpm BO motor + wheels | **449/-** |
| **5** | Miscellaneous (3.7V 3800mAh battery, chassis, battery socket) | **400/-** |
|  | **Total Cost** | **1702/-** |

## Chapter 4 Implementation and Results

This chapter provides a detailed account of the project's implementation, including the algorithms, flowcharts, software development, hardware setup, and the results obtained from simulations and real-world testing. By presenting these elements, it aims to showcase the practical execution and outcomes of the project.

### Algorithm and flowcharts:

In Fig 4.1.1, Flowchart of software hardware interface is being shown. In reference to paper [10], [1] and [5], the brief description of process flow is explained below:

#### Hand Gesture Controlled Vehicle Using ESP32:

1. **Start Process:**
   * System initiation.

#### Hand Gesture Detection:

* + Capture hand gestures via camera.

#### Gesture Mode Identification:

* + Analyze and identify the gesture mode.

#### Action Determination:

* + **Forward Gesture:** Command to move forward.
  + **Backward Gesture:** Command to move backward.
  + **Stop Gesture:** Command to stop.
  + **Left Gesture:** Command to turn left.
  + **Right Gesture:** Command to turn right.

#### Command Communication via Socket Programming:

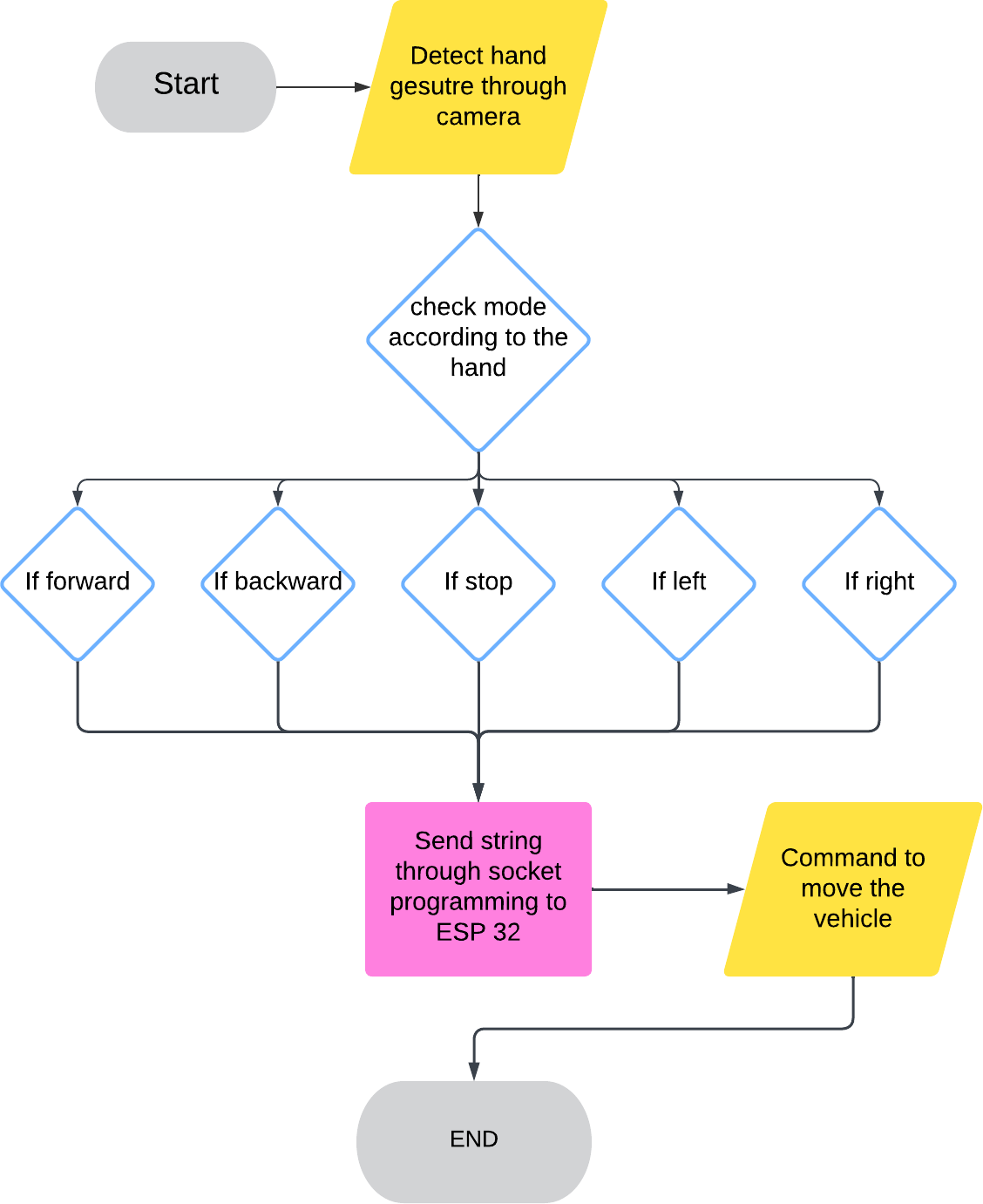
* + Generate string command based on gesture processed.
  + Send command to ESP32 using socket programming.

#### ESP32 Command Execution:

* + ESP32 receives and executes the vehicle movement command.

#### End Process:

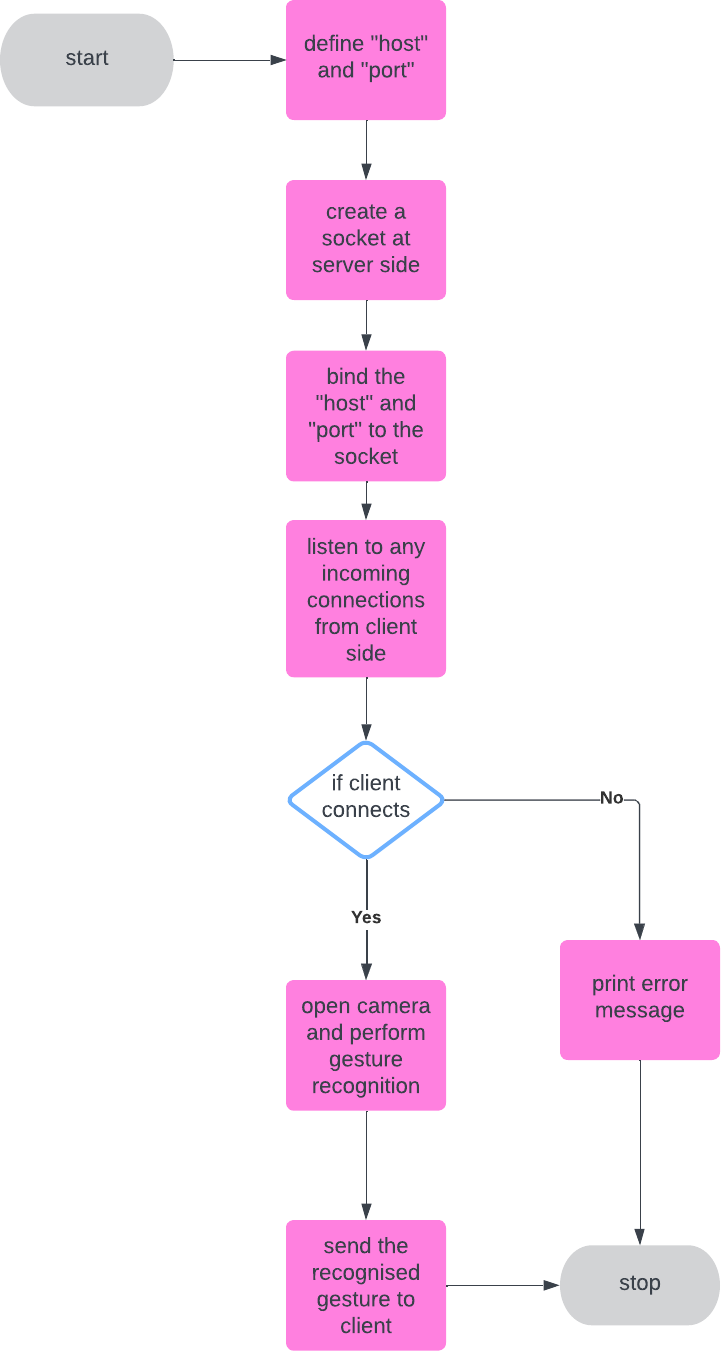
* + System awaits the next hand gesture.



*Figure 4.1.1: - Flowchart of software hardware interface*

#### Flowcharts of software setup:

The provided flowchart in Fig 4.1.2, outlines the steps for a server-side application that involves setting up a server, recognizing gestures using a camera, and communicating with a client. Here’s a concise explanation of each step in the flowchart:



#### Start:

* + - The process begins.

#### Define "host" and "port":

* + - The server's IP address (host) and port number are specified. These will be used for establishing the server.

#### Create a socket at the server side:

* + - A socket is created to enable communication over the network.

#### Bind the "host" and "port" to the socket:

* + - The socket is bound to the defined host and port, associating the socket with a specific network interface and port

*Fig 4.1.2: - Flowchart of Server*

#### Listen to any incoming connections from the client side:

* + - The server listens for incoming connection requests from clients.

#### If client connects:

* + - The server checks if a client has connected.

#### Yes:

* + - If a client connects, the server proceeds to the next step.

#### No:

* + - If no client connects, an error message is printed, and the process stops.

#### Open camera and perform gesture recognition:

* + - The server opens the camera and starts recognizing gestures using the camera feed.

#### Send the recognized gesture to the client:

* The recognized gesture is sent to the connected client.

#### Stop:

* The process ends.

Once a connection is established, the server opens a camera, performs gesture recognition, and sends the recognized gestures to the client. If no client connects, an error message is displayed, and the process terminates.

The provided flowchart in Fig 4.1.2, outlines the steps for a client-side process to connect to a server and control a rover based on commands received from the server. Here's a detailed explanation of each step:

#### Start:

* + The process begins here.

#### Define "host" and "port" as server:

* + The client defines the server's address (host) and the port number it will use to connect.

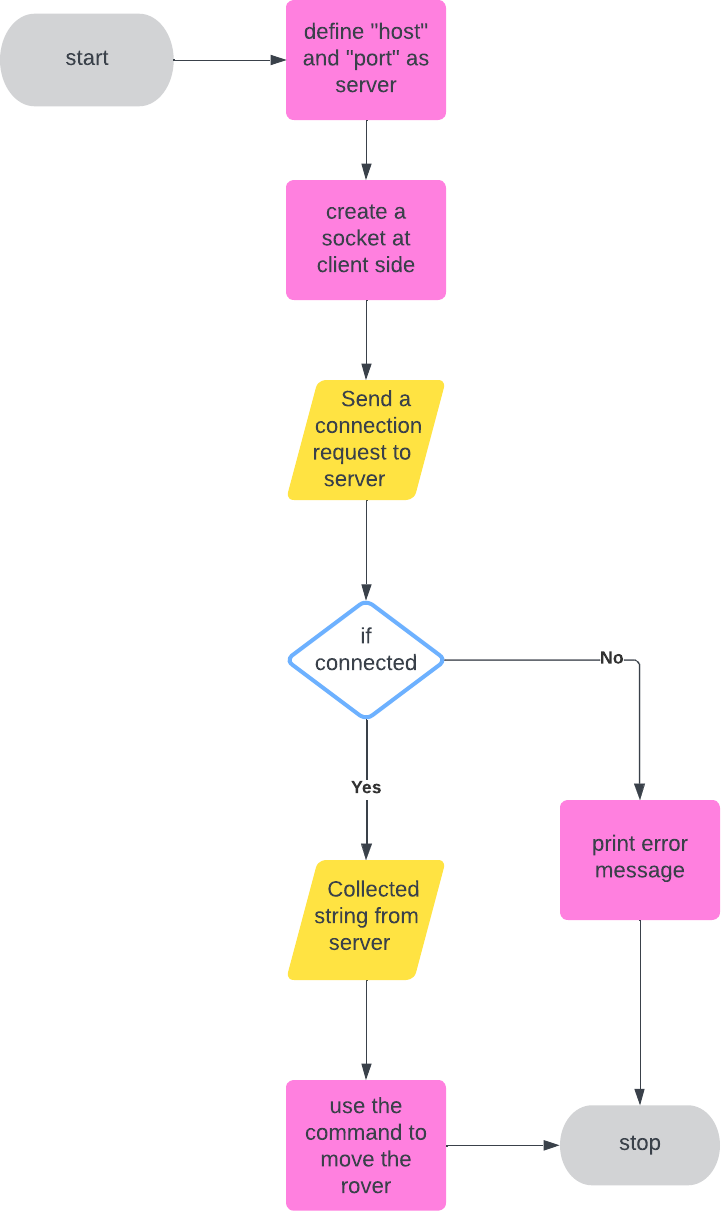
#### Create a socket at client side:

* + The client creates a socket, which is an endpoint for sending and receiving data.

#### Send a connection request to server:

* + The client sends a request to the server to establish a connection.

#### If connected (decision point):



*Fig 4.1.3: - Flowchart of Client*

* + The client checks if the connection to the server was successful.

#### Yes:

* + If the connection is successful, proceed to the next step.

#### No:

If the connection fails, print an error message and stop the process.

#### Collected string from server:

* If connected, the client collects the command string sent by the server.

#### Use the command to move the rover:

* The client uses the received command to control the rover's movement.

#### Stop:

* The process ends here.

This is how the client connects to a server, retrieves commands, and then uses those commands to control a rover, with error handling if the connection fails.

### Procedure and Setup

System design, software development, system calibration and testing are explained in detail to overview the flow of project development.

#### System Design

The provided circuit diagram demonstrates how to control a four-motor robotic rover using an ESP32 microcontroller and an L298N motor driver module. The circuit includes the ESP32 as the central component that processes commands and controls the motors, the L298N motor driver to manage the speed and direction of the four DC motors (two on the right and two on the left), and a 7- 12V DC power supply to power the motor driver and motors. The ESP32 is connected to the L298N motor driver via several GPIO pins: Motor A (Right Motors) control pins (IN1 to GPIO 5, IN2 to GPIO 18, ENA to GPIO 22) and Motor B (Left Motors) control pins (IN3 to GPIO 19, IN4 to GPIO 21, ENB to GPIO 23). The 5V and GND pins of the ESP32 are connected to the corresponding pins on the L298N motor driver. The motors are connected to the output terminals of the L298N: Right Motors to OUT1 and OUT2, and Left Motors to OUT3 and OUT4. This setup allows the ESP32 to control the direction and speed of the motors, enabling the robotic rover to move forward, backward, and turn left or right based on the received commands.

#### Software Development

The development environment is set up by installing the Arduino IDE and configuring it to support the ESP32, including adding the ESP32 board package. Moreover, the client-server communication for string transfer past image processing is being executed on Jupyter Notebook IDE which supports python coding environment, for image processing, referred through paper [2], [7] and [10]. The program written for the ESP32 includes string identification logic, which reads input string from the command sent by, after processing the gesture and subsequently actuates the motor driver to actuate the motors accordingly and rotate them in respective direction. The provided code in Appendix 1, sets up a system for controlling a four-motor robotic rover using gesture recognition. The server- side code uses a webcam and a pre-trained Keras model to recognize hand gestures. It captures video frames, processes them to identify gestures, and sends the recognized commands to a connected client via a socket.

On the client side, an ESP32 microcontroller connects to Wi-Fi and establishes a TCP connection with the server. It listens for gesture commands sent by the server and controls the rover's motors accordingly. The ESP32 uses GPIO pins and PWM signals to move the rover in response to commands such as "forward," "backward," "left," "right," and "stop." This setup allows real-time remote control of the rover using hand gestures.

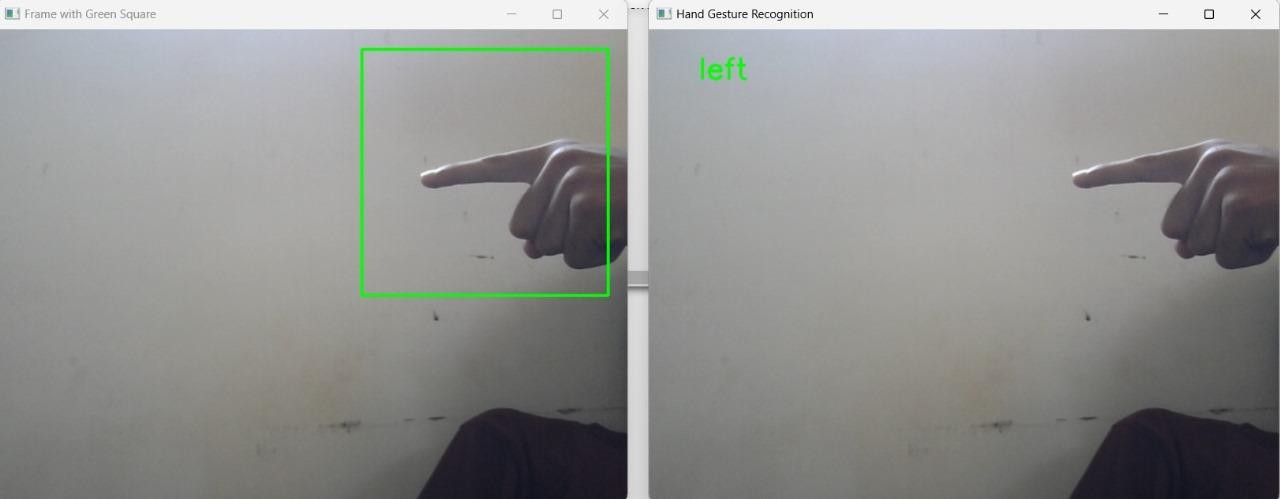
#### System Calibration and Testing

System calibration and testing are crucial for ensuring the accuracy and reliability of the project. In reference with paper [1], [5] and [3], the ML model is calibrated by adjusting its sensitivity towards various hand gestures to ensure accurate detection, and it is tested under various conditions to minimize false processing. For the ESP32 module, it is essential to ensure it has its drivers installed in the PC, pinout verification is also necessary to ensure the number of GPIOs in the respective controller used. The operating voltages of controller and motor driver must be checked to avoid shot- circuit of internal circuitry of the components. The algorithm section 4.1, outline the steps for both the server-side and client-side processes involved in controlling a four-motor robotic rover. On the server side, the process starts by defining the server's IP address and port, creating a socket, binding the socket to the specified host and port, and listening for incoming connections. Upon a client connection, the server opens the camera, performs gesture recognition, and sends the recognized gesture commands to the client. If no client connects, an error message is printed, and the process stops. On the client side, the process begins by defining the server's address and port, creating a socket, and sending a connection request to the server. If the connection is successful, the client collects the command string sent by the server and uses it to control the rover's movement. If the connection fails, an error message is printed, and the process stops. This combination of server-side gesture recognition and client-side command execution allows for remote control of the rover based on the recognized gestures.

### Software Result

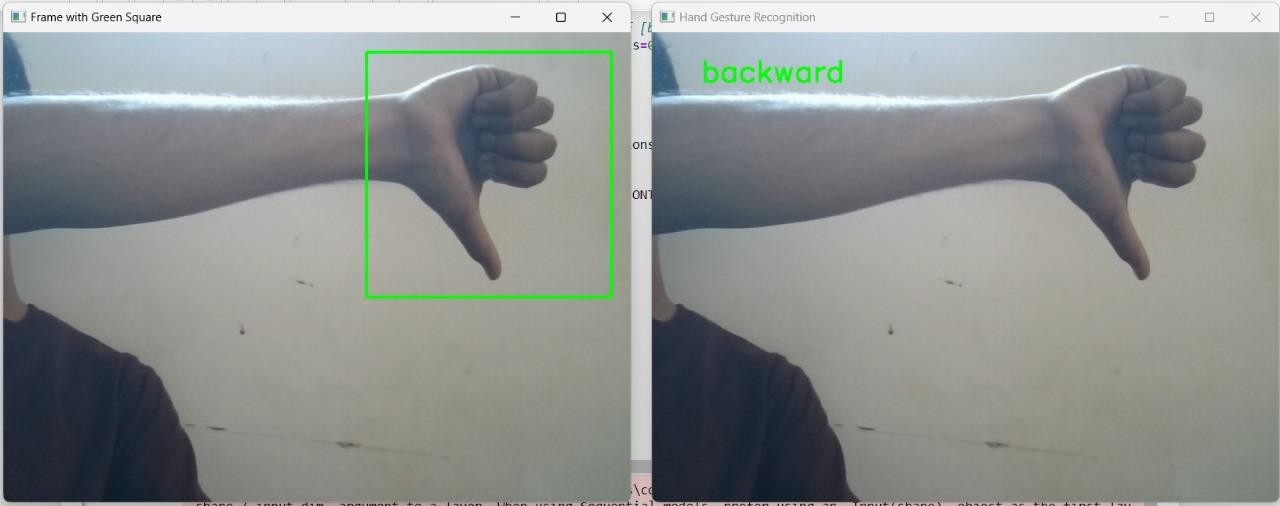
Software results upon the successful accomplishment of image processing (Hand gesture recognition model) and transmission of the respective associated string to each gesture is being shown below:

In Fig 4.3.1, the gesture displayed is chosen to transmit string “left” to the client via server via socket programming, to the rover (robotic vehicle) in left direction.



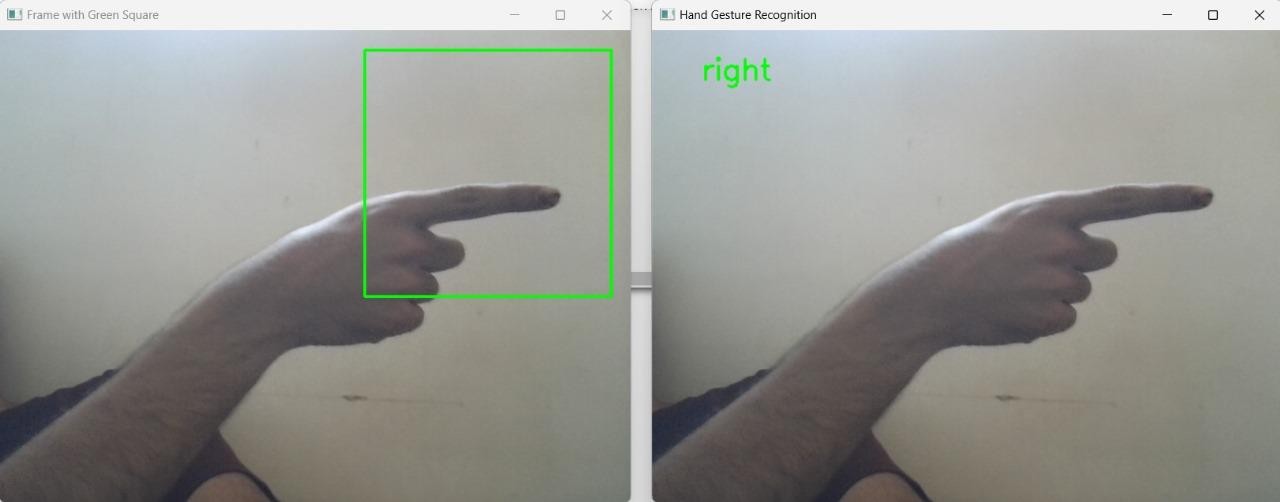
***Fig 4.3.1: -Gesture command for left motion of rover***

In Fig 4.3.2, the gesture displayed is chosen to transmit string “backward” to the client via server via socket programming, to mobilize the rover (robotic vehicle) in backward direction.



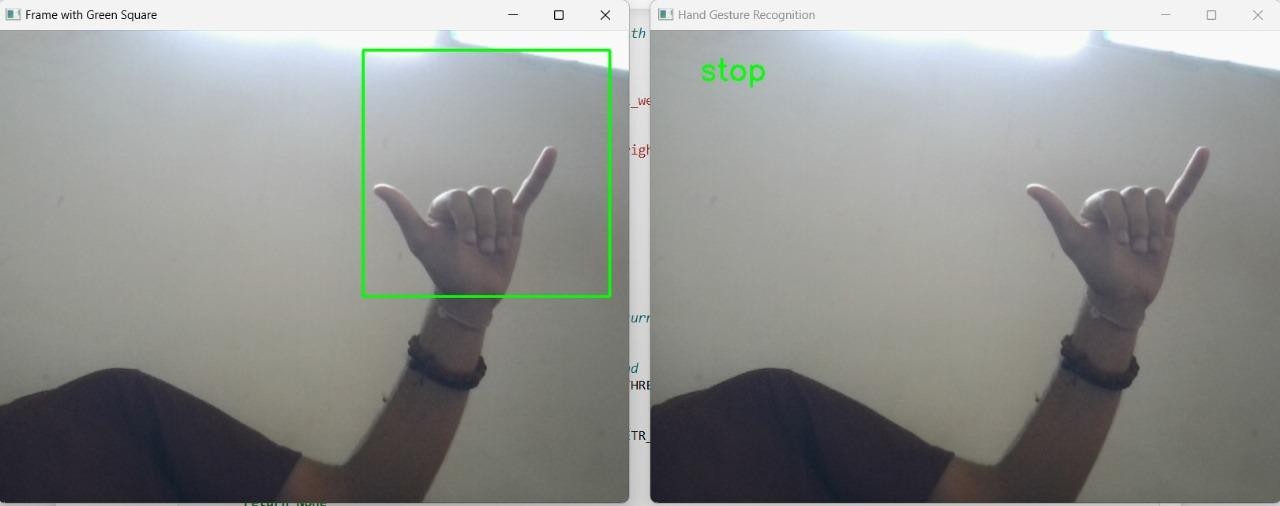
***Fig 4.3.2: - Gesture command for backward motion of rover***

In Fig 4.3.3, the gesture displayed is chosen to transmit string “right” to the client via server via socket programming, to mobilize the rover (robotic vehicle) in right direction.



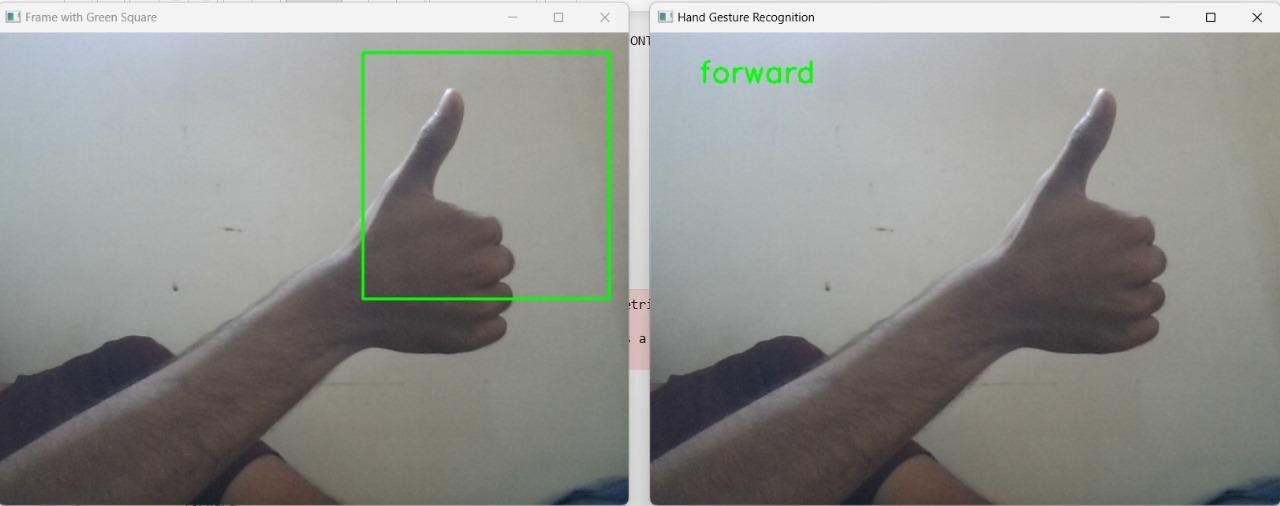
***Fig 4.3.3: - Gesture command for right motion of rover***

In Fig 4.3.4, the gesture displayed is chosen to transmit string “stop” to the client via server via socket programming, to demobilize make the rover (robotic vehicle).



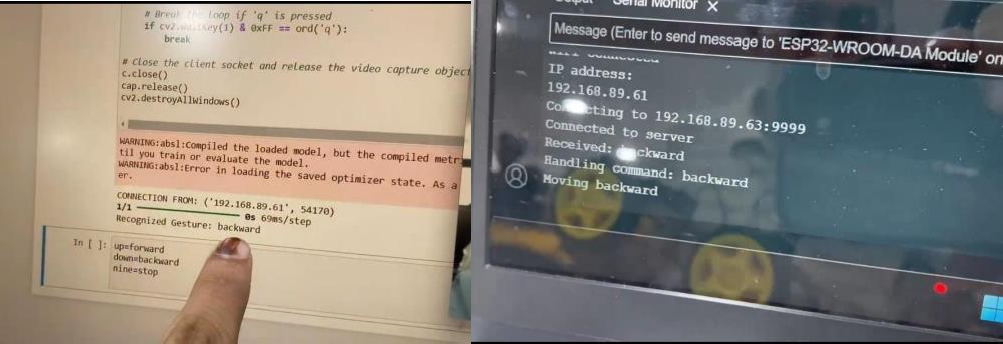
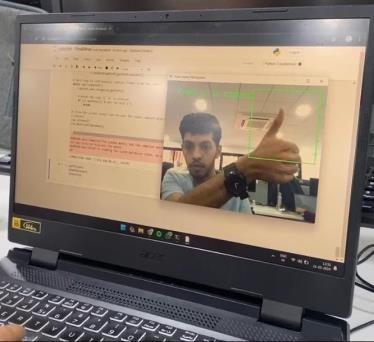
***Fig 4.3.4: - Gesture command for stopping the rover***

In Fig 4.3.5, the gesture displayed is chosen to transmit string “forward” to the client via server via socket programming, to mobilize the rover (robotic vehicle) in forward direction.



***Fig 4.3.5: - Gesture command for forward motion of rover***

Fig 4.3.6, shows an example of ‘backward’- gesture command transmission from server to client to control the rover.



***Fig 4.3.6: - ‘backward’- gesture command transmission from server to client***

### Hardware Implementation

The final presentation of the outcomes and findings obtained from the practical implementation of the hardware components in the hand gesture-controlled rover system, refer Fig 4.4.1.

We have observed that the rover moves correctly when it receives commands such as “Forward,” “Backward,” “Right,” “Left,” and “Stop” from the server. For “Left” and “Right” movements, the rover follows the command for a set period before waiting for the next command. The input lag between command reception and movement execution is minimal, ensuring the rover operates under ideal conditions.



***Fig 4.4.1: -Hand gesture-controlled rover (Hardware)***

### Discussion

The integration of the ESP-WROOM32 microcontroller with the ML model for image processing and the Motor Driver L298 2A has proven to be effective in creating a responsive hand gesture-controlled rover system. The ESP-WROOM32 detects and processes incoming string data from gesture recognition, which is transmitted via socket programming. This microcontroller, with its CPUs, memory, and customizable input/output peripherals, processes the gesture data and generates commands for the motor driver over Wi-Fi.

Real-time hand gesture detection via a camera, followed by the transmission of commands to the ESP32, ensures accurate and efficient control of the rover. The ESP32 processes these commands and outputs the necessary signals to the motor driver, enabling precise movements of the rover. This setup enhances Human-Computer interaction, operational efficiency, and provides real-time insights into system performance.

Overall, this system demonstrates the potential of combining image processing with microcontroller- based control to create intuitive and accessible assistive technologies, improving user interaction and inclusivity.

**Chapter 5**

# Conclusion

This chapter summarizes the findings, results, and overall achievements of the project. It reflects on the objectives outlined in the beginning, evaluates the success of the implementation, and discusses the potential impact and future developments of the hand gesture-controlled rover system. Conclusion:

In this project, we successfully developed a hand gesture-controlled rover system using the ESP- WROOM32 microcontroller and Motor Driver L298 2A, integrated with an ML model for image processing. The system efficiently detects and processes hand gestures in real-time, transmitting commands via socket programming to control the rover's movements. Major reference by research paper [1] and [5].

The implementation demonstrated the feasibility of using hand gesture recognition for intuitive and accessible control of a robotic vehicle. The ESP-WROOM32 microcontroller played a crucial role in processing the received data and executing the corresponding commands, ensuring seamless and responsive control.

The project achieved its primary objectives, including studying socket programming and image processing algorithm, interfacing the ESP32 with image processing and motor driver components, and successfully implementing real-time hand gesture recognition. The results showed significant potential for enhancing accessibility and user interaction, particularly benefiting individuals with disabilities. In conclusion, this project highlights the potential of integrating advanced microcontroller technology with machine learning and image processing to create innovative assistive solutions. Future work could explore further optimization of gesture recognition accuracy, integration with additional sensors for enhanced functionality, and potential applications in various fields requiring intuitive control systems.

**Future Scope**

The future scope of this research includes several promising advancements and applications. Integrating Internet of Things (IoT) features could enhance connectivity, enabling remote monitoring and control for greater user independence. Adapting the system for real-world applications involves integrating the prototype with actual wheelchairs, ensuring reliability, safety, and user-friendliness. Additionally, incorporating health monitoring sensors to track vital signs such as heart rate and blood pressure could provide caregivers with real-time health data, improving patient care.

Leveraging advanced machine learning techniques can enhance gesture recognition accuracy, making the system more intuitive and responsive. Beyond its primary focus, this technology has potential applications in tourism, short-distance travel, and military contexts. Collaborations with industry partners will be crucial for scaling up and commercializing the prototype, facilitating widespread adoption.

Continuous refinement through trial and error will improve system performance and user experience, addressing diverse needs. Integrating artificial intelligence (AI) for predictive analytics and personalized experiences could revolutionize user interaction, adapting responses to individual preferences. Exploring renewable energy sources to power the vehicles can make the system more sustainable, aligning with global carbon reduction efforts.

Finally, expanding the system's capabilities to include multi-modal interaction—such as voice commands and eye-tracking—could provide a more inclusive solution for users with varying abilities. These additional modes could work in conjunction with hand gestures, offering a robust and versatile control system. By continuously integrating emerging technologies, this research can pave the way for innovative applications and improved quality of life for users across different sectors.

# References

1. G. Chanhan and P. Chandhari, "Gestures based wireless robotic control using image processing," *2015 5th Nirma University International Conference on Engineering (NUiCONE)*, Ahmedabad, India, 2015, pp. 1-7, doi: 10.1109/NUICONE.2015.7449642.
2. S. Ganapathyraju, "Hand gesture recognition using convexity hull defects to control an industrial robot," 2013 3rd International Conference on Instrumentation Control and Automation (ICA), Ungasan, Indonesia, 2013, pp. 63-67, doi: 10.1109/ICA.2013.6734047.
3. H. Zhao, J. Hu, Y. Zhang and H. Cheng, "Hand gesture based control strategy for mobile robots," 2017 29th Chinese Control And Decision Conference (CCDC), Chongqing, China, 2017, pp. 5868- 5872, doi: 10.1109/CCDC.2017.7978217.
4. H. Song, W. Feng, N. Guan, X. Huang and Z. Luo, "Towards robust ego-centric hand gesture analysis for robot control," *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*, Beijing, China, 2016, pp. 661-666, doi: 10.1109/SIPROCESS.2016.7888345.
5. J. L. Raheja, G. A. Rajsekhar and A. Chaudhary, "Controlling a remotely located robot using hand gestures in real time: A DSP implementation," *2016 5th International Conference on Wireless Networks and Embedded Systems (WECON)*, Rajpura, India, 2016, pp. 1-5, doi: 10.1109/WECON.2016.7993420.
6. S. Ikegami, C. Premachandra, B. H. Sudantha and S. Sumathipala, "A Study on Mobile Robot Control by Hand gesture Detection," 2018 3rd International Conference on Information Technology Research (ICITR), Moratuwa, Sri Lanka, 2018, pp. 1-5, doi: 10.1109/ICITR.2018.8736135.
7. S. T. Kebir, M. Bouhedda, S. Mekaoui, M. Guesmi and A. Douakh, "Gesture control of mobile robot based arduino microcontroller," 2016 8th International Conference on Modelling, Identification and Control (ICMIC), Algiers, Algeria, 2016, pp. 1081-1085, doi: 10.1109/ICMIC.2016.7804273.
8. A. Pasarica, C. Miron, D. Arotaritei, G. Andruseac, H. Costin and C. Rotariu, "Remote control of a robotic platform based on hand gesture recognition," 2017 E-Health and Bioengineering Conference (EHB), Sinaia, Romania, 2017, pp. 643-646, doi: 10.1109/EHB.2017.7995506.
9. B. M. Rianu, A. Kitagawa, W. Dirgantara and Subairi, "Hand Gesture Recognition-Based Control System Using Computer Vision for Transporter Robot," 2023 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), Bandung, Indonesia, 2023, pp. 242-247, doi: 10.1109/ICRAMET60171.2023.10366557.
10. M. Alam and M. A. Yousuf, "Designing and Implementation of a Wireless Gesture Controlled Robot for Disabled and Elderly People," 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ECACE.2019.8679290.

## Appendix -1

The code for key algorithms and processes implemented in the project, offering a detailed reference for understanding the underlying logic and implementation steps is provided below:

### Implementation code:

This Python script demonstrates the integration of OpenCV for image processing, Keras for gesture recognition using a pre-trained model, and socket programming for communication with the rover's microcontroller. The provided C++ code is associated with ESP32 and client-side interface.

#### #START OF CODE: Socket and gesture recognition.

import cv2

import numpy as np

from keras.models import load\_model import socket

# Global variable for background subtraction (initialize with None) bg = None

model = load\_model('C:\\Users\\prana\\Desktop\\Minor\\model\_weights.h5') # Define the hand gesture labels

gesture\_labels = ['backward','left','right','stop','forward']

def segment(image, threshold=25): global bg

# Initialize the background if not set if bg is None:

bg = image.copy().astype("float") return None

# Find the absolute difference between background and current frame diff = cv2.absdiff(bg.astype("uint8"), image)

# Threshold the diff image so that we get the foreground

thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH\_BINARY)[1]

# Get the contours in the thresholded image

(cnts, \_) = cv2.findContours(thresholded.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Return None if no contours detected if len(cnts) == 0:

return None else:

# Based on contour area, get the maximum contour which is the hand segmented = max(cnts, key=cv2.contourArea)

return (thresholded, segmented)

# Initialize video capture from webcam cap = cv2.VideoCapture(0)

square\_size = 250

square\_margin = 20

square\_x = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)) - square\_size - square\_margin square\_y = square\_margin

# take the server name and port name host = '0.0.0.0'

port = 9999

# create a socket at server side # using TCP / IP protocol

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# bind the socket with server # and port number

s.bind(('', port)) s.listen(1)

# Accept a client connection c, addr = s.accept()

print("CONNECTION FROM:", str(addr))

# Variable to store the recognized gesture recognized\_gesture = None

def capture\_and\_recognize\_gesture(): global recognized\_gesture

# Read a frame from the camera ret, frame = cap.read()

if not ret: return

# Flip the frame horizontally for better user experience frame = cv2.flip(frame, 1)

# green frame

square\_top\_left = (square\_x, square\_y)

square\_bottom\_right = (square\_x + square\_size, square\_y + square\_size) cv2.rectangle(frame, square\_top\_left, square\_bottom\_right, (0, 255, 0), 2)

cv2.putText(frame, "Press 'c' to capture", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255,

0), 2)

cv2.imshow('Hand Gesture Recognition', frame)

key = cv2.waitKey(1) & 0xFF # Check if 'c' key is pressed

if key == ord('c'):

# Extract the region of interest (ROI) defined by the green square

roi = frame[square\_y:square\_y + square\_size, square\_x:square\_x + square\_size]

# Convert the ROI to grayscale for background subtraction gray = cv2.cvtColor(roi, cv2.COLOR\_BGR2GRAY)

# Call the segment function to get the segmented hand region hand\_segment = segment(gray)

# Process the segmented hand region for recognition if hand\_segment is not None:

thresholded, segmented = hand\_segment

# Resize the segmented hand region to match model input size and convert to RGB processed\_hand\_rgb = cv2.resize(thresholded, (48, 48))

processed\_hand\_rgb = cv2.cvtColor(processed\_hand\_rgb, cv2.COLOR\_GRAY2RGB)

# Normalize pixel values

processed\_hand\_rgb = processed\_hand\_rgb / 255.0

# Add batch dimension (model expects input shape of [batch\_size, height, width, channels]) input\_data = np.expand\_dims(processed\_hand\_rgb, axis=0)

# Perform inference with the model predictions = model.predict(input\_data)

# Get the predicted gesture label

recognized\_gesture = gesture\_labels[np.argmax(predictions)] print("Recognized Gesture:", recognized\_gesture)

# Send the recognized gesture to the client c.send(recognized\_gesture.encode())

# Main loop to continuously capture frames from the camera while cap.isOpened():

capture\_and\_recognize\_gesture()

# Break the loop if 'q' is pressed

if cv2.waitKey(1) & 0xFF == ord('q'): break

# Close the client socket and release the video capture object c.close()

cap.release() cv2.destroyAllWindows()

#### #START OF CODE: ESP32 and Client interface.

#include <WiFi.h>

// Replace with your network credentials const char\* ssid = "Avi's M34";

const char\* password = "Tj@292014";

// Server IP and port

const char\* host = "192.168.89.63"; // Ensure this matches your server's IP address const uint16\_t port = 9999;

WiFiClient client;

// Define motor control pins const int motorA\_in1 = 5; const int motorA\_in2 = 18;

const int motorA\_ena = 22; // PWM pin for Motor A

const int motorB\_in3 = 19; const int motorB\_in4 = 21;

const int motorB\_enb = 23; // PWM pin for Motor B

// Define PWM channels const int pwmChannelA = 0; const int pwmChannelB = 1;

const int pwmFrequency = 1000; // 1 kHz PWM frequency const int pwmResolution = 8; // 8-bit resolution

// Duration for a 90-degree turn (adjust this value based on your rover's configuration) const int turnDuration = 1000; // Duration in milliseconds

void setup() { Serial.begin(115200);

// Initialize motor control pins pinMode(motorA\_in1, OUTPUT); pinMode(motorA\_in2, OUTPUT); pinMode(motorA\_ena, OUTPUT);

pinMode(motorB\_in3, OUTPUT); pinMode(motorB\_in4, OUTPUT); pinMode(motorB\_enb, OUTPUT);

// Configure PWM channels

ledcSetup(pwmChannelA, pwmFrequency, pwmResolution); ledcAttachPin(motorA\_ena, pwmChannelA);

ledcSetup(pwmChannelB, pwmFrequency, pwmResolution); ledcAttachPin(motorB\_enb, pwmChannelB);

// Stop motors initially stopMotors();

// Connect to Wi-Fi

Serial.print("Connecting to "); Serial.println(ssid); WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) { delay(500);

Serial.print(".");

}

Serial.println(""); Serial.println("WiFi connected"); Serial.println("IP address: "); Serial.println(WiFi.localIP());

// Connect to server Serial.print("Connecting to "); Serial.print(host); Serial.print(':'); Serial.println(port);

if (!client.connect(host, port)) { Serial.println("Connection failed."); return;

}

Serial.println("Connected to server");

}

void stopMotors() { digitalWrite(motorA\_in1, LOW); digitalWrite(motorA\_in2, LOW); ledcWrite(pwmChannelA, 0);

digitalWrite(motorB\_in3, LOW); digitalWrite(motorB\_in4, LOW); ledcWrite(pwmChannelB, 0);

}

void handleCommand(const String& command) { Serial.print("Handling command: "); Serial.println(command);

if (command == "forward") { Serial.println("Moving forward");

digitalWrite(motorA\_in1, LOW); digitalWrite(motorA\_in2, HIGH); ledcWrite(pwmChannelA, 255); // Full speed for Motor A

digitalWrite(motorB\_in3, HIGH); digitalWrite(motorB\_in4, LOW); ledcWrite(pwmChannelB, 255); // Full speed for Motor B

delay(turnDuration); // Turn for the specified duration stopMotors(); // Stop after turning

} else if (command == "backward") {

Serial.println("Moving backward"); digitalWrite(motorA\_in1, HIGH); digitalWrite(motorA\_in2, LOW); ledcWrite(pwmChannelA, 255); // Full speed for Motor A

digitalWrite(motorB\_in3, LOW); digitalWrite(motorB\_in4, HIGH); ledcWrite(pwmChannelB, 255); // Full speed for Motor B

delay(turnDuration); // Turn for the specified duration stopMotors(); // Stop after turning

} else if (command == "left") {

Serial.println("Turning left"); digitalWrite(motorA\_in1, LOW); digitalWrite(motorA\_in2, HIGH);

ledcWrite(pwmChannelA, 255); // Full speed for Motor A

digitalWrite(motorB\_in3, LOW); digitalWrite(motorB\_in4, HIGH); ledcWrite(pwmChannelB, 255); // Full speed for Motor B delay(turnDuration); // Turn for the specified duration stopMotors(); // Stop after turning

} else if (command == "three") { Serial.println("Turning right"); digitalWrite(motorA\_in1, HIGH); digitalWrite(motorA\_in2, LOW);

ledcWrite(pwmChannelA, 255); // Full speed for Motor A

digitalWrite(motorB\_in3, HIGH);

digitalWrite(motorB\_in4, LOW); ledcWrite(pwmChannelB, 255); // Full speed for Motor B

delay(turnDuration); // Turn for the specified duration stopMotors(); // Stop after turning

} else if (command == "stop") {

Serial.println("Stopping"); stopMotors();

} else {

Serial.println("Unknown command");

}

}

void loop() {

// Check if connected

if (client.connected()) {

// Receive data from the server if (client.available()) {

String received\_message = client.readStringUntil('\n'); received\_message.trim(); // Trim the received message Serial.print("Received: "); Serial.println(received\_message);

// Handle the received command handleCommand(received\_message);

}

} else {

Serial.println("Server closed the connection."); client.stop();

// Optionally, you can attempt to reconnect here delay(5000); // Wait before retrying

if (client.connect(host, port)) { Serial.println("Reconnected to server");

} else {

Serial.println("Reconnection failed.");

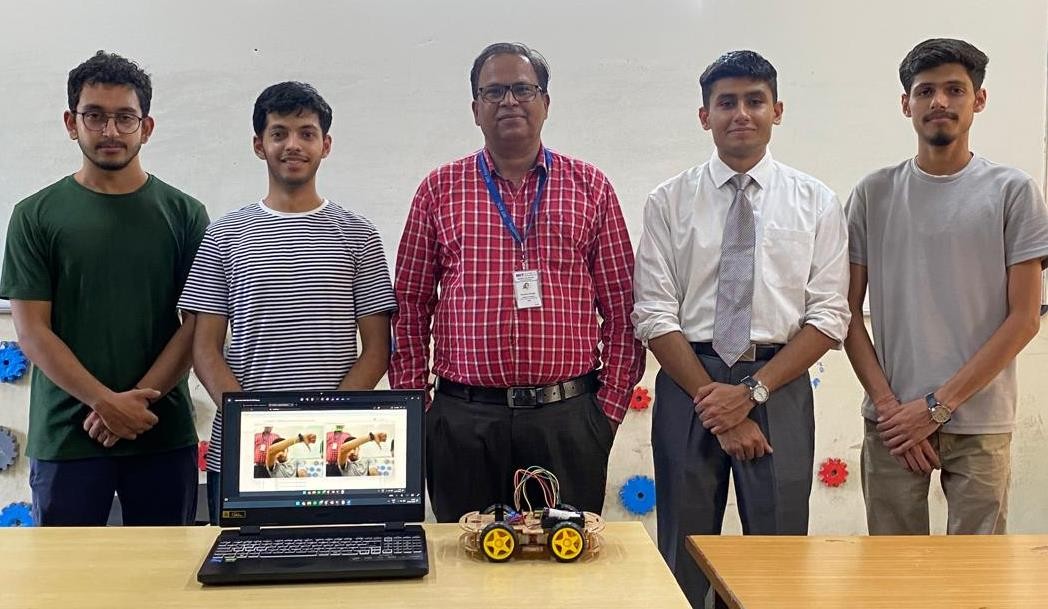
}

}

delay(100); // Adjust the delay as needed

}

## Appendix -2



***Prof. Isaq Shikalgar (Project guide); middle, next to him (right); Tuhinansh Sharma, Mayank Soni; right most, Siddhant Mishra; left most, next to him(right); Pranav Khatavkar, with their project Hand gesture-controlled rover, May 28, 2024.***

### Link for project video:

* [https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5\_RWV60dYN-](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing) [PZbFm1MUMzCiXfkP74qw/edit?usp=sharing](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing)

### Link for digital portfolio:

Team members:

1. Mayank Soni

PRN: 202201090031

* + [https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5\_RWV60dYN-](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing) [PZbFm1MUMzCiXfkP74qw/edit?usp=sharing](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing)

1. Pranav Khatavkar PRN: 202201090028
   * [https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5\_RWV60dYN-](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing) [PZbFm1MUMzCiXfkP74qw/edit?usp=sharing](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing)
2. Tuhinansh Sharma PRN: 202201060010
   * [https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5\_RWV60dYN-](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing) [PZbFm1MUMzCiXfkP74qw/edit?usp=sharing](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing)
3. Siddhant Mishra PRN: 202201040036
   * [https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5\_RWV60dYN-](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing) [PZbFm1MUMzCiXfkP74qw/edit?usp=sharing](https://docs.google.com/document/d/1Jt-ZQ1ajl2rQ5_RWV60dYN-PZbFm1MUMzCiXfkP74qw/edit?usp=sharing)