

A REAL-TIME HUMAN-MOBILE ROBOT INTERACTION SYSTEM BASED ON HAND GESTURE USING RASPBERRY PI

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering
with Internet of Things

By

PATHANGE PAVAN KUMAR (Register No. 40732005)

VISHWAJEET ANAND (Register No. 40731127)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

Accredited with Grade "A++" by NAAC | Approved by AICTE

**JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119**

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PATHANGE PAVAN KUMAR (40732005)** and **VISHWAJEET ANAND (40731127)** who carried out project entitled "**A REAL-TIME HUMAN-MOBILE ROBOT INTERACTION SYSTEM BASED ON HAND GESTURE USING RASPBERRY PI**" under my supervision from November 2023 to April 2024.

Internal Guide

Dr. S. Jayanthi, M.E., Ph.D.

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva-voce Examination held on 13/04/2024

13/04/24
Internal Examiner

13/4/24
External Examiner

DECLARATION

I, **PATHANGE PAVAN KUMAR (40732005)** hereby declare that the Project entitled **"A REAL-TIME HUMAN-MOBILE ROBOT INTERACTION SYSTEM BASED ON HAND GESTURE USING RASPBERRY PI"** done by me under the guidance of **Dr. S. Jayanthi, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with IoT.**

DATE: 13-04-2024

PLACE: Chennai

P. Pavankumar
SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** and **Dr. S. Vigneshwari M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. Jayanthi, M.E., Ph.D.** for her valuable guidance, suggestions, and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Advancements in robotics and human-computer interaction have spurred a burgeoning interest in establishing more intuitive communication channels between humans and robots. This endeavor holds immense potential for fostering collaboration, improving assistive technologies, and enriching human-robot interactions across numerous domains. This research focuses on bridging this communication gap by leveraging Raspberry Pi as the central processing unit. The system integrates a Raspberry Pi and Pi camera to analyze hand movements, enabling seamless interaction between humans and mobile robots. It incorporates a Pi-camera module mounted on the robot for real-time image acquisition, with Raspberry Pi serving as the processing hub. A sophisticated gesture recognition algorithm, trained to interpret hand gestures, processes the captured images, translating them into actionable commands for the robot. The experimental results demonstrate the system's effectiveness in facilitating natural communication between humans and robots, with promising applications spanning home automation, healthcare, industrial automation, and entertainment. Furthermore, this system opens doors for personalized human-robot interaction, as it can adapt to individual gestures and preferences. Additionally, the utilization of Raspberry Pi offers scalability and affordability, making this communication framework accessible for widespread adoption in various settings. This project represents a significant advancement in enhancing the synergy between humans and robots, heralding a future where intuitive communication fosters innovation and collaboration across diverse fields.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	Vii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	01
	1.1 Background and Motivation	01
	1.2 Problem Statement	02
	1.3 Objective and Scope	02
	1.4 Applications of Human-Robot Interaction	03
2	LITERATURE SURVEY	04
	2.1 Overview of Existing research and literature	04
	2.2 Inferences from Literature Survey	05
	2.3 Open Problems in the Existing System	06
3	REQUIREMENT ANALYSIS	07
	3.1 Feasibility Studies of the Project	07
	3.2 Requirements Specification	08
4	DESCRIPTION OF PROPOSED SYSTEM	18
	4.1 Description of Approach and Methods	18
	4.2 Architecture of Proposed System	23
	4.3 Project Management Plan	25
	4.4 Financial report on estimated costing	26
5	IMPLEMENTATION DETAILS	29
	5.1 Development and Deployment setup	29
	5.2 Algorithms	31

5.3	Deployment and Maintenance	32
6	RESULTS AND DISCUSSIONS	33
7	CONCLUSIONS	39
7.1	Conclusion	39
7.2	Recommendations for Future Research	40
	REFERENCES	42
	APPENDIX	45
A.	SOURCE CODE	45
B.	SCREENSHOTS	49
C.	PATENT PROOF	55

LIST OF FIGURES

Figure No.	Figure Name	Page No.
2.1	Existing system	06
3.1	Structure of Convolutional Neural Network	10
3.2	Hand landmark model bundle	14
3.3	Raspberry Pi	15
3.4	Pi Camera	16
3.5	L293D Motor Driver	16
3.6	DC Gear Motor	17
4.1	Hand Gestutres	20
4.2	System Archirecture	23
4.3	FlowChart	24
5.1	Hardware Setup	29
5.2	Raspian OS	30
6.1	Installing MediaPipe Library	33
6.2	Downloading and viewing the images	33
6.3	Testing images	34
6.4	Running inference	34
6.5	Visualizing the results	35
6.6	Forward Gesture Recognition	36
6.7	Backward Gesture Recognition	36
6.8	Turn Left Gesture Recognition	37
6.9	Turn Right Gesture Recognition	37
6.10	Stop Gesture Recognition	38
6.11	Hardware Demonstration	38

LIST OF ABBREVIATIONS

ABBREVIATIONS

EXPANSION

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
COM	Centre Of Mass
DC	Direct Current
GRIR	Gesture Recognition Intermediary Robot
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HRC	Human-Robot Control
HRI	Human-Robot Interaction
IDE	Integrated Development Environment
IoT	Internet Of Things
IROS	Intelligent RObot Systems
LED	Light Emitting Diode
NumPy	Numerical Python
OpenCV	Open Source Computer Vision Library
UNIX	Uniplexed Information Computing System

CHAPTER 1

INTRODUCTION

In educational settings, the organization of elections serves as an invaluable tool for instilling democratic values and nurturing leadership qualities among students. However, with the advent of technology, traditional election processes are undergoing notable transformations, leveraging innovative solutions to enhance efficiency and accessibility. Within this context, our project seeks to explore an innovative approach to facilitate seamless communication between human operators and mobile robots. By leveraging the versatility of the Raspberry Pi as a central control unit, our goal is to develop a system that enables users to interact effortlessly with robots through gestures and hand movements, thereby democratizing human-robot collaboration and improving user experience. Conventional human-robot interfaces often rely on complex command sequences or remote controls, which can hinder usability and accessibility.

Through the integration of gesture recognition technology and the flexibility of Raspberry Pi, we aim to create a user-centric system that simplifies human-robot interaction, making it more intuitive, efficient, and accessible. This project will detail the methodology employed in developing the gesture-based communication system, including hardware setup, software implementation, and algorithm design. Additionally, it will present experimental results and findings to assess the effectiveness and usability of the proposed approach. Ultimately, our project aims to contribute to advancements in natural human-robot communication, fostering innovation and progress in various domains.

1.1 BACKGROUND AND MOTIVATION

The increasing integration of technology into daily life has led to advancements in robotics, particularly mobile robots, which have the potential to revolutionize various industries. However, effective communication between humans and robots remains a challenge, often relying on complex interfaces. Our project aims to address this challenge by developing a system that enables natural and intuitive communication through gestures and hand movements. Motivated by the desire to

democratize human-robot collaboration and enhance accessibility, we seek to empower users with diverse backgrounds and abilities to interact effortlessly with robots, fostering innovation and societal progress.

1.2 PROBLEM STATEMENT

Despite significant advancements in robotics, effective communication between humans and mobile robots remains a challenge. Traditional human-robot interfaces often rely on complex command sequences or remote controls, which can be cumbersome and unintuitive, especially in dynamic environments or for users with limited technical expertise. This lack of natural communication hinders the widespread adoption and effective utilization of mobile robots across various domains. Moreover, existing solutions may not adequately address the need for inclusivity and accessibility, limiting the participation of individuals with disabilities or those without specialized training in robotics. As a result, there is a pressing need for innovative approaches that enable seamless and intuitive communication between humans and mobile robots, empowering users with diverse backgrounds and abilities to interact effortlessly with robotic systems.

The problem statement for this project revolves around the development of a user-centric system that bridges the gap between human gestures and robotic actions, leveraging the versatility of Raspberry Pi as a central control unit. By addressing this challenge, we aim to enhance user experience, improve efficiency, and broaden the applicability of mobile robots across various industries and societal contexts.

1.3 OBJECTIVE AND SCOPE

Objective

The objective of this project is to develop a gesture-based communication system that facilitates natural and intuitive interaction between humans and mobile robots. This involves designing and implementing a hardware setup comprising a Raspberry Pi as the central control unit and a Pi-camera module mounted on the mobile robot for real-time image acquisition. Additionally, the project aims to develop a gesture recognition algorithm capable of interpreting hand movements and gestures captured by the Pi-camera module. This algorithm will be integrated

with the hardware setup to enable the translation of human gestures into actionable commands for the robot. The effectiveness and usability of the gesture-based communication system will be evaluated through experimental testing and user feedback.

Scope

The scope of the project includes the development and implementation of the gesture-based communication system using Raspberry Pi and Pi-camera module technology. This encompasses designing and configuring the hardware setup, developing and training the gesture recognition algorithm, integrating the hardware setup with the algorithm, conducting experimental tests to evaluate the system's effectiveness, accuracy, and usability, and documenting the project development process in a comprehensive report. However, the project does not include the development of advanced robotics functionalities beyond basic gesture-based communication, nor does it address all potential applications or scenarios where such technology could be deployed.

1.4 APPLICATIONS OF HUMAN-ROBOT INTERACTION

The applications of this project are diverse and encompass various domains where human-robot interaction is prevalent. One significant application is in the field of home automation, where the gesture-based communication system could enable users to control household devices and appliances with simple hand gestures, offering convenience and accessibility. In healthcare settings, the system could facilitate interactions between patients and assistive robots, allowing individuals with limited mobility or dexterity to communicate their needs effectively. Furthermore, in industrial automation, the ability to control robots through intuitive hand gestures could streamline manufacturing processes, improve efficiency, and enhance worker safety. Additionally, the system could find applications in entertainment and gaming, where users could interact with robots in immersive and engaging experiences. Overall, the project's applications span across various sectors, promising to enhance human-robot collaboration and simplify interaction in diverse environments.

CHAPTER 2

LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers and automotive companies in a bid to create a solution, and all their solutions vary from analyzing various patterns.

2.1 OVERVIEW OF EXISTING RESEARCH AND LITERATURE

Applying image processing techniques to detect plant diseases [1]

Plant diseases pose a significant threat to global food security, leading to substantial crop losses and economic impact. Early and accurate detection of these diseases is crucial for implementing timely interventions and preventing the spread of infections. In recent years, advancements in image processing and computer vision have provided valuable tools for non-invasive and efficient disease detection in plants. This research focuses on the application of image processing techniques to detect plant diseases effectively.

Humanoid Receptionist Connected to IoT Subsystems and Smart Infrastructure is Smarter than Expected [2]

The integration of humanoid robots into various industries has opened up exciting possibilities for enhancing customer service, improving operational efficiency, and providing interactive experiences. This research project explores the development and implementation of a humanoid receptionist equipped with advanced artificial intelligence (AI) capabilities and connected to Internet of Things (IoT) subsystems and smart infrastructure. The study aims to investigate the robot's performance and intelligence in a real-world setting, surpassing conventional expectations.

Humanlike robots as employees in the hotel industry [3]

The hotel industry is continuously seeking innovative ways to improve guest experiences, increase operational efficiency, and maintain a competitive edge. In recent years, the integration of humanlike robots as employees in hotels has emerged as a promising solution to meet these objectives.

This research explores the implementation of humanlike robots in various roles within the hotel industry, investigating their impact on customer satisfaction, service quality, and overall operational effectiveness.

Gesture Recognition Intermediary Robot for Abnormality Detection in Human Activities [4]

The detection of abnormal behavior or activities in humans plays a crucial role in various domains, including healthcare, security, and safety. This research presents the development and implementation of a Gesture Recognition Intermediary Robot (GRIR) designed to detect abnormality in human activities through advanced gesture recognition techniques. The GRIR serves as an intelligent intermediary system, capable of interpreting human gestures and identifying deviations from normal patterns, which may indicate potential risks or anomalies.

Real-time Hand Gestures System for Mobile Robot Control [5]

Autonomous mobile robot navigation in an indoor environment using vision sensors is receiving considerable attention in current robotics research activities. In this paper, a robot controlled by real-time hand gestures is proposed. This system includes an image pre-processing and feature of extraction state that consists of a bounding box and Center-Of-Mass based computation.

2.2 INFERENCES FROM THE LITERATURE SURVEY

From Literature Review 1-

The integration of image processing and computer vision technologies has the potential to revolutionize plant disease detection and management by offering precise, non-destructive, and timely solutions to this critical agricultural challenge.

From Literature Review 2-

This research project involves developing and implementing a highly advanced humanoid receptionist with AI capabilities, IoT integration, and smart infrastructure. The study seeks to evaluate the robot's performance and intelligence in real-world scenarios, with the aim of exceeding conventional expectations.

From Literature Review 3-

This research develops a Gesture Recognition Intermediary Robot (GRIR) to detect abnormal human behavior through advanced gesture recognition, with potential applications in healthcare and security.

From Literature Review 4-

Hotels are using humanlike robots to improve the guest experience and make operations more efficient. This research examines how these robots impact customer satisfaction and service quality in the hotel industry.

2.3 OPEN PROBLEMS IN THE EXISTING SYSTEM

In the existing system, the objective is to filter out a red glove, which involves identifying and retaining red-colored objects while discarding others. This simplifies the process of detecting hand gestures and postures.



Fig.2.1: Existing system

This filtering process assigns weights to each pixel's contribution to the final pixel value, which is more effective in preserving edges compared to the mean filter, which assigns equal weights to all pixels within the filter window, particularly for a 1-D Gaussian filter. In the previous method, they solely relied on image processing filters for recognizing hand signs. However, this approach had significant drawbacks as it was both time-consuming and inaccurate.

Disadvantages:

- The above-mentioned process is time-consuming and inaccurate.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES OF THE PROJECT

Before embarking on the development of the gesture-based communication system, it is essential to conduct feasibility studies to assess the viability and potential challenges associated with the project. The feasibility studies encompass various aspects, including technical feasibility, economic feasibility, and operational feasibility.

3.1.1 Economic Feasibility

Economic feasibility assesses the financial viability of the project, including the cost of hardware components, software licenses, development resources, and ongoing maintenance expenses. It involves estimating the project's budget and evaluating whether the anticipated benefits outweigh the costs incurred. In our project, economic feasibility involves estimating the cost of acquiring Raspberry Pi boards, Pi-camera modules, and other necessary hardware components, as well as the cost of software development and testing. Furthermore, economic feasibility also considers potential cost savings or revenue generation opportunities that may arise from the successful implementation and deployment of the gesture-based communication system.

3.1.2 Technical Feasibility

Technical feasibility evaluates the project's capability to achieve its objectives from a technical perspective. In the case of our project, technical feasibility involves assessing the feasibility of implementing the gesture recognition algorithm on the Raspberry Pi platform and integrating it with the Pi-camera module. This includes evaluating the computational requirements, memory constraints, and processing capabilities of the Raspberry Pi to ensure that it can effectively process real-time image data and perform gesture recognition algorithms within acceptable performance parameters. Additionally, technical feasibility also considers the availability of relevant software libraries, tools, and development resources needed to implement the project successfully.

3.1.3 Operational Feasibility

Operational feasibility evaluates the project's practicality and suitability for implementation within the intended operational environment. It considers factors such as organizational readiness, user acceptance, and potential impacts on existing processes and workflows. In our project, operational feasibility involves assessing the ease of use and user-friendliness of the gesture-based communication system, as well as its compatibility with existing robotic platforms and applications. It also considers factors such as user training requirements, technical support needs, and the potential impact on productivity and efficiency in real-world settings.

3.2 REQUIREMENTS SPECIFICATION

Based on the reference of IEEE Std. 830 – 1998 [19] Recommended Practice for Software Requirements, The Hardware and Software Requirements are described

3.2.1 Software Requirements

- Operating System: Raspberry Pi OS
- RAM: 4GB or above 4GB
- Processor Frequency: 1.5GHz or above
- Processor: ARM Cortex-A72 or compatible
- Programming Language: Python
- Integrated Development Environment (IDE): Visual Studio Code, Thonny, or any other compatible IDE for Raspberry Pi

3.2.2 Hardware Requirements

- Raspberry Pi
- Pi Camera
- L293D Motor Driver
- DC Motors
- LED

3.2.3 Python

Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be extremely readable. It commonly employs English terms rather than punctuation, and it has fewer syntactical structures than other languages.

Python Features

- **Easy-to-learn** - Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- **Easy-to-read** - Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** - Python's source code is relatively simple to maintain.
- **A broad standard library** - The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** - Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- **Portable** - Python can operate on a broad range of hardware devices and has the same interface across them all.
- **Extendable** - The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- **Databases** - Python provides interfaces to all major commercial databases.
- **GUI Programming** - Python can construct and port GUI programmes to numerous system calls, libraries, and windows systems, including Windows MFC, Macintosh, and Unix's X Window system.
- **Scalable** - Python provides a better structure and support for large programs than shell scripting.

3.2.4 Deep Learning

Deep learning is a type of machine learning that uses algorithms meant to function in a manner similar to the human brain. While the original goal for AI was broadly to make machines able to do things that would otherwise require human intelligence, the idea has been refined in the decades since. François Chollet, AI researcher at Google and creator of the machine learning software library Keras,

says: “Intelligence is not a skill in itself, it’s not about what you can do, but how well and how efficiently you can learn new things.

3.2.5 Convolutional Neural Network (CNN)

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on a underdone image and do not need any preprocessing. A convolutional neural network is a feed forward neural network, seldom with up to 20. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces. The agenda for this sphere is to activate machines to view the world as h mans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

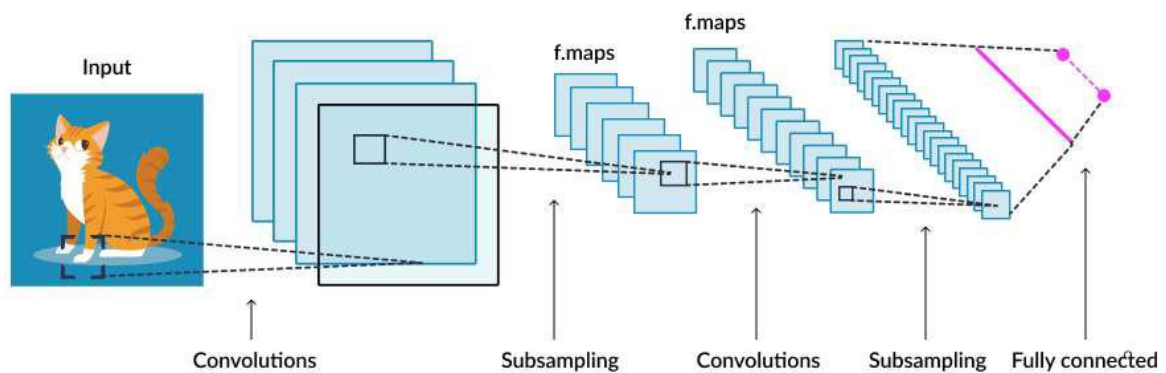


Fig.3.1: Structure of Convolutional Neural Network

3.2.6 Tensorflow

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications.

3.2.7 Keras

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications.

Advantages of Keras

Keras has several advantages that make it a popular choice for deep learning.

- **Simplicity** - Keras is very easy and simple to use. It is a user-friendly API with easy-to-learn and code features.
- **Backend support** - Keras does not operate with low-level computations. So, it supports the use of backends.
- **Pre-trained models** - Keras provides numerous pre-trained models.
- **Fast experimentation** - Keras is built to simplify the tasks of users.
- **Great community and caliber documentation** - Keras has a large supportive community.

3.2.8 NumPy:

NumPy, short for Numerical Python, is a fundamental library in the Python ecosystem, particularly in the realm of scientific computing and data analysis. It provides powerful tools for efficient manipulation and computation with multi-dimensional arrays and matrices. NumPy's capabilities are instrumental in various stages of data preprocessing, analysis, and modeling, making it an indispensable tool for researchers and practitioners alike. One of the key features of NumPy is its ability to perform fast and vectorized operations on arrays, significantly enhancing computational efficiency compared to traditional Python lists. This is achieved through optimized C and Fortran libraries under the hood, allowing NumPy to leverage the performance benefits of lower-level languages while retaining the flexibility and ease of use of Python. In the context of our project, NumPy played a pivotal role in handling the data preprocessing tasks required for training and deploying the convolutional neural network (CNN) model. Specifically, NumPy was utilized to efficiently manipulate input images, such as resizing, normalization, and reshaping, to ensure they were in suitable formats compatible with the CNN architecture. These preprocessing steps are crucial for preparing the data for training and inference, as they help improve model performance and convergence. Moreover, NumPy provides a rich collection of mathematical functions and operations, enabling seamless computation of various statistical metrics, mathematical transformations, and linear algebra operations. These functionalities were leveraged in our project for tasks such as calculating mean and standard deviation of image data, performing element-wise operations on arrays, and conducting matrix manipulations required for certain CNN layers. Overall, NumPy's versatility, efficiency, and extensive capabilities make it an indispensable tool for numerical computing and data manipulation tasks in Python. Its seamless integration with other libraries and frameworks, such as TensorFlow and OpenCV, further enhances its utility in scientific computing and machine learning applications.

3.2.9 OpenCV:

OpenCV, the Open Source Computer Vision Library, stands as a cornerstone in computer vision, offering a comprehensive suite of tools for image and video processing. Its widespread adoption across academia and industry underscores its importance in various fields, including robotics, medical imaging, surveillance, and augmented reality. In our project, OpenCV played a pivotal role in various image processing tasks, facilitating image reading, manipulation, and feature extraction. Its versatile functionalities enabled seamless handling of diverse image formats, efficient preprocessing, and extraction of relevant features crucial for accurate hand sign recognition using convolutional neural networks. OpenCV's integration with other libraries, such as NumPy and TensorFlow, further streamlined the development process, showcasing its indispensability in modern computer vision applications.

3.2.10 MediaPipe:

MediaPipe, renowned for its flexibility and efficiency, offered a comprehensive solution for real-time hand tracking and gesture recognition tasks. Its ease of integration and cross-platform compatibility streamlined the development process, allowing us to focus on refining the gesture recognition algorithms and enhancing the user experience. Additionally, MediaPipe's robustness in handling diverse environmental conditions and lighting variations ensured the reliability of hand tracking and gesture recognition in real-world scenarios. Furthermore, its support for customizable pipelines and modular components provided scalability and adaptability, empowering us to tailor the system to meet specific project requirements and accommodate future enhancements. Overall, MediaPipe's contribution was instrumental in advancing the capabilities of our system and showcasing the potential of gesture-based interaction in various applications.

Features

- *Input image processing* - Processing includes image rotation, resizing, normalization, and color space conversion.
- **Score threshold** - Filter results based on prediction scores.
- **Label allowlist and denylist** - Specify the gesture categories recognized by the model.

Task inputs

The Gesture Recognizer accepts an input of one of the following data types:

- Still images
- Decoded video frames
- Live video feed

Task outputs

The Gesture Recognizer outputs the following results:

- Categories of hand gestures
- Handedness of detected hands
- Landmarks of detected hands in image coordinates
- Landmarks of detected hands in world coordinates

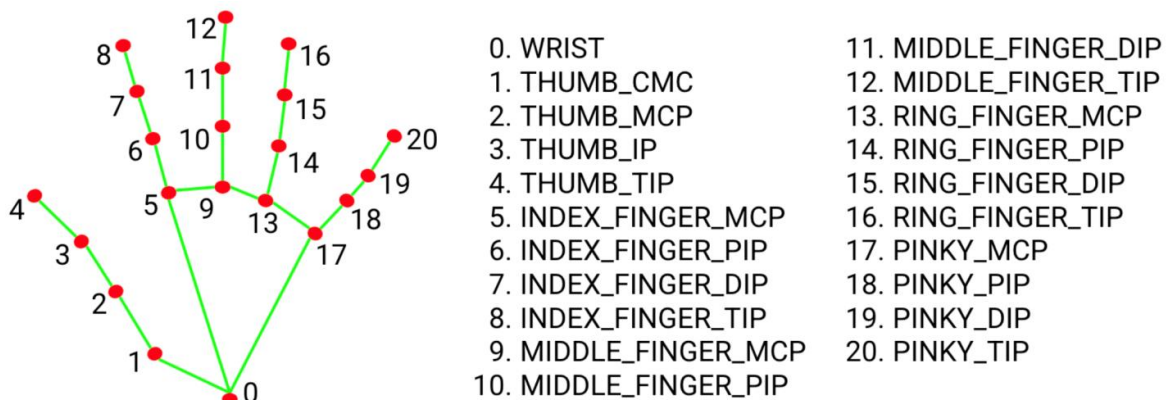


Fig.3.2: Hand landmark model bundle

Hand landmark model bundle:

The hand landmark model bundle detects the keypoint localization of 21 hand-knuckle coordinates within the detected hand regions. The model was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds. The hand landmarker model bundle contains palm detection model and hand landmarks detection model. Palm detection model localizes the region of hands from the whole input image, and the hand landmarks detection model finds the landmarks on the cropped hand image

defined by the palm detection model. Since palm detection model is much more time consuming, in Video mode or Live stream mode, Gesture Recognizer uses bounding box defined by the detected hand landmarks in the current frame to localize the region of hands in the next frame. This reduces the times Gesture Recognizer triggering palm detection model. Only when the hand landmarks model could no longer identify enough required number of hands presence, or the hand tracking fails, palm detection model is invoked to relocalize the hands.

3.2.11 Raspberry Pi:

The Raspberry Pi is the central computing unit and controller of our system. It plays a pivotal role in managing the processing of gesture data and controlling the robot's actions. Its compact and versatile design, combined with the support for various programming languages and interfaces, makes it an ideal choice for orchestrating the integration of software and hardware components in our real-time interaction system.

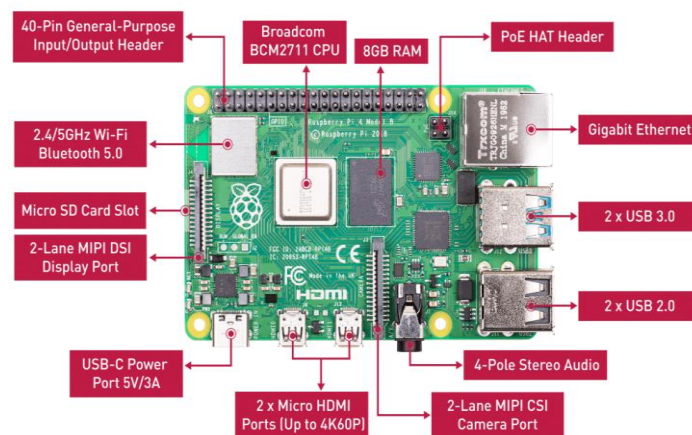


Fig.3.3: Raspberry Pi

3.2.12 Pi Camera:

The Pi Camera module is the visual "eye" of our project, responsible for capturing real-time images of the user's hand. This component is a fundamental part of our machine learning-based gesture recognition system. Its high-resolution image capture capability enables accurate and swift recognition of hand gestures, allowing for seamless human-robot interaction.

3.2.14 DC Motor:

The DC Motor is the driving force behind the robot's mobility, translating the recognized hand gestures into responsive movements. This component is integral to real-time interaction, enabling the robot to navigate its environment and respond promptly to user commands. Its adaptability allows us to explore different locomotion methods, ensuring flexibility in design and functionality as we implement our innovative project.



Fig.3.6: DC Gear Motor

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

After identifying limitations in existing approaches, we propose a novel human-robot control system that eliminates the need for additional sensors, relying solely on camera feed for human activity detection. In this innovative system, human activity data is sampled at fixed time intervals to determine ongoing activities, which are then fed into the model for detection. The system outputs the specific activity being performed by the human, enhancing real-time interaction with robots.

Advantages of Proposed System:

- 1. No Initial Setup Required:** Unlike existing systems that rely on sensors for activity detection, our proposed system does not require any initial setup, simplifying implementation.
- 2. Elimination of Sensor Costs:** By eliminating the need for sensors, our system significantly reduces costs associated with hardware acquisition and installation, making it more accessible and cost-effective.
- 3. Independence from Sensors:** The reliance solely on camera feed means our system is not reliant on external sensors, reducing the risk of sensor failures and maintenance requirements.
- 4. Easy Enhancement and Expansion:** The proposed system offers flexibility for future enhancements and additions to the activity detection model without the need for additional hardware, allowing for seamless integration of new activities and functionalities.

4.1 DESCRIPTION OF APPROACH AND METHODS

4.1.1 System Architecture:

The proposed system consists of three main components: a camera module for capturing real-time video feeds, a Raspberry Pi as the central processing unit, and a gesture recognition algorithm for interpreting hand gestures. The camera module, typically a Pi-camera, is mounted on the mobile robot to provide a continuous stream of visual input for gesture recognition. The Raspberry Pi serves as the central control unit, responsible for processing the video feeds and executing the gesture recognition algorithm. The gesture recognition algorithm is custom-built using techniques from computer vision and machine learning to

accurately interpret and classify hand gestures in real-time.

4.1.2 Gesture Recognition Algorithm:

The core of the proposed system is the gesture recognition algorithm, which is developed to analyze the video feeds and identify specific hand gestures performed by the user. The algorithm employs a combination of image processing techniques, such as edge detection, contour analysis, and feature extraction, to extract relevant information from the video frames. Machine learning algorithms, such as convolutional neural networks (CNNs), are then trained on the extracted features to classify different hand gestures accurately. The trained model is integrated into the Raspberry Pi for real-time gesture recognition.

4.1.3 MediaPipe

The MediaPipe Gesture Recognizer task is an advanced tool designed for real-time recognition of hand gestures, providing detailed hand landmark coordinates and corresponding gesture classifications. This task facilitates interactive experiences by enabling applications to respond dynamically to users' gestures as they perform them. It operates seamlessly on both static images and continuous video streams, offering rapid feedback and precise hand landmark data in both image and world coordinates. Moreover, it supports the detection and classification of gestures for multiple hands simultaneously, enhancing collaborative interactions and enabling multi-user scenarios. With its versatility and robust feature set, the MediaPipe Gesture Recognizer task finds applications across various domains, including interactive applications, accessibility features, gaming, and entertainment. Developers can customize and integrate this task into their projects, leveraging its flexibility to tailor gesture recognition models to specific application requirements. In conclusion, the MediaPipe Gesture Recognizer task empowers developers to create immersive, interactive experiences by enabling real-time recognition of hand gestures, driving innovation in gesture-based interaction across diverse domains.

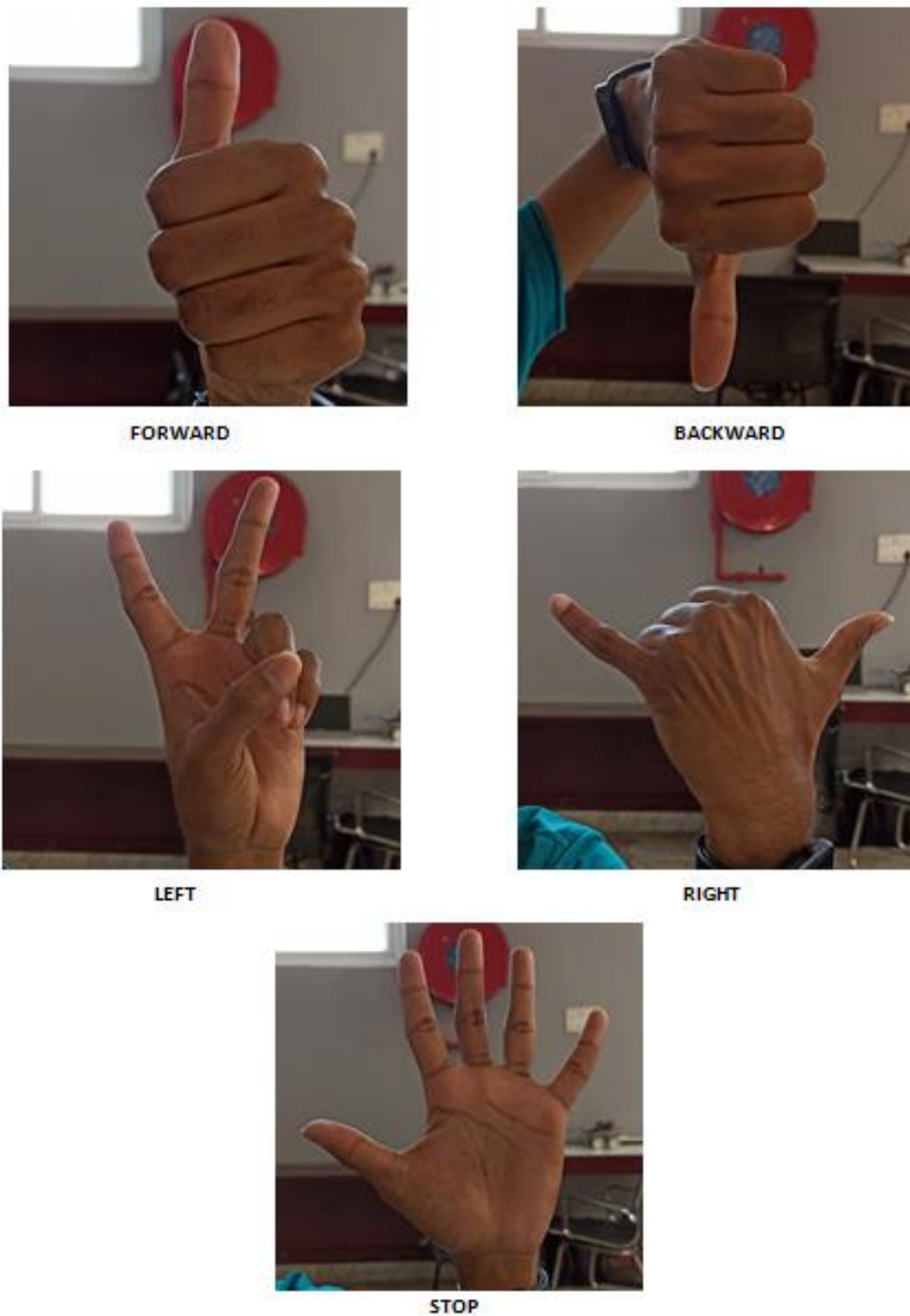


Fig.4.1 Hand Gestures

In our project, we have developed a system that allows users to manipulate robot movement using hand gestures, providing an intuitive and natural interface for human-robot interaction. By recognizing specific hand gestures, users can control various aspects of the robot's movement, including direction, speed, and stopping.

- **Gesture Recognition:** The system utilizes a gesture recognition algorithm trained to detect and classify predefined hand gestures associated with different robot movements. These gestures include forward, backward, left, right, and stop commands, each corresponding to a specific action that the robot will perform.
- **Forward Gesture:** A forward hand gesture signals the robot to move forward, allowing users to guide the robot towards a desired location or target.
- **Backward Gesture:** Conversely, a backward hand gesture instructs the robot to move backward, enabling users to navigate the robot away from obstacles or towards a retreat position.
- **Left and Right Gestures:** Left and right hand gestures are interpreted as commands for the robot to turn left or right, respectively, allowing users to steer the robot along a desired path or change its direction.
- **Stop Gesture:** Finally, a stop hand gesture signals the robot to halt its movement, bringing it to a complete stop and preventing further motion.
- **User Interaction:** To manipulate robot movement using hand gestures, users simply perform the corresponding gestures within the camera's field of view. The system continuously captures and analyzes the hand gestures in real-time, interpreting them as commands for the robot's movement.
- **Implementation:** The system is implemented using a combination of computer vision techniques and machine learning algorithms. A camera module mounted on the robot captures video feeds of the user's hand gestures, which are processed and analyzed by the gesture recognition algorithm running on a central processing unit, such as a Raspberry Pi.

4.1.4 Data Collection and Training:

To train the gesture recognition algorithm, a dataset of hand gesture images is collected, consisting of various hand poses and movements relevant to the intended application. This dataset is annotated with corresponding labels to indicate the associated gestures. Data augmentation techniques may be applied to increase the diversity and robustness of the dataset. The annotated dataset is then divided into training and validation sets for model training and evaluation. The

gesture recognition algorithm is trained using supervised learning methods, where the input data (hand gesture images) and their corresponding labels are used to optimize the model parameters.

4.1.5 Integration and Testing:

Once the gesture recognition algorithm is trained, it is integrated into the Raspberry Pi-based system for real-time operation. The system is tested extensively to evaluate its performance in accurately detecting and interpreting hand gestures in various scenarios and environments. Testing may involve controlled experiments in laboratory settings as well as real-world deployment on the mobile robot. Performance metrics such as accuracy, precision, recall, and latency are measured to assess the effectiveness and efficiency of the proposed system. Any issues or limitations encountered during testing are addressed, and refinements are made to improve the system's overall performance and reliability.

4.1.6 Deployment and Validation:

After deployment, the proposed system undergoes comprehensive performance evaluation and validation in real-world environments. This involves conducting extensive user testing sessions and gathering feedback from end-users to assess the system's effectiveness and usability. Additionally, the system's performance metrics, such as accuracy, response time, and reliability, are continuously monitored to identify any potential issues or areas for improvement. Furthermore, the deployed system is subjected to stress testing and robustness evaluation to ensure its resilience to various environmental conditions and user interactions. This includes testing the system's response to unexpected gestures, varying lighting conditions, and background clutter that may affect gesture recognition accuracy. To address any identified shortcomings or usability concerns, iterative adjustments and optimizations are made to the system's algorithms, user interface, and hardware components. These refinements aim to enhance the system's overall performance, user satisfaction, and adaptability to different use cases and user demographics.

4.2 ARCHITECTURE OF PROPOSED SYSTEM

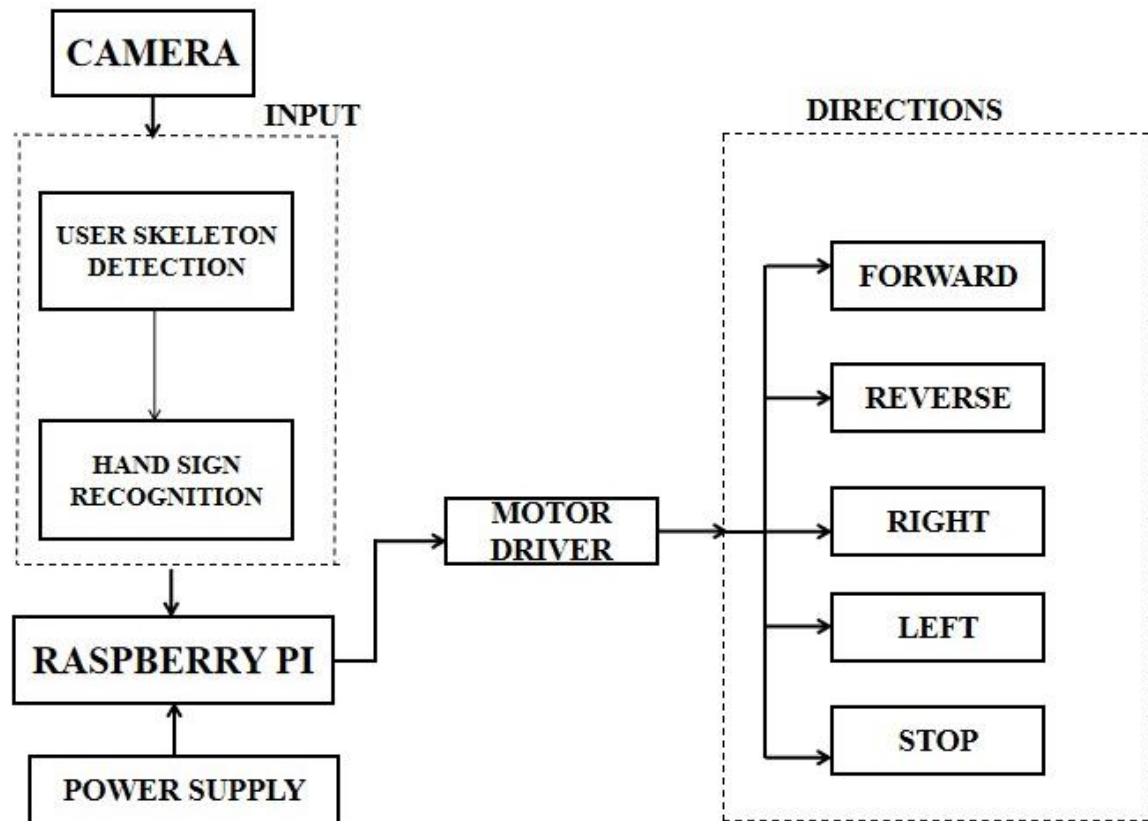


Fig.4.2 System Archirecture

The system architecture begins with the Raspberry Pi Camera Module capturing video input, initiating the process. The video feed is then processed through a series of stages within the system. Initially, the system performs user skeleton dialogue recognition and hand sign recognition concurrently. This process involves detecting and interpreting the user's body language and hand gestures, enabling intuitive communication between the user and the system. Once the user's gestures are recognized, the information is forwarded to the Raspberry Pi for further processing. The Raspberry Pi, acting as the central processing unit, receives the input data and analyzes it to determine the appropriate action to take. Based on the recognized gestures and dialogue, the Raspberry Pi generates commands for controlling the movement of the robot. These commands are then transmitted to the motor driver, which serves as the interface between the Raspberry Pi and the DC motors of the robot. The motor driver translates the

commands received from the Raspberry Pi into specific directions for the DC motors, including forward, reverse, right, left, or stop movements. Finally, the DC motors execute the commanded movements, resulting in the physical motion of the robot according to the user's gestures and dialogue. This closed-loop system ensures seamless interaction between the user and the robot, allowing for intuitive control and navigation based on the user's input.

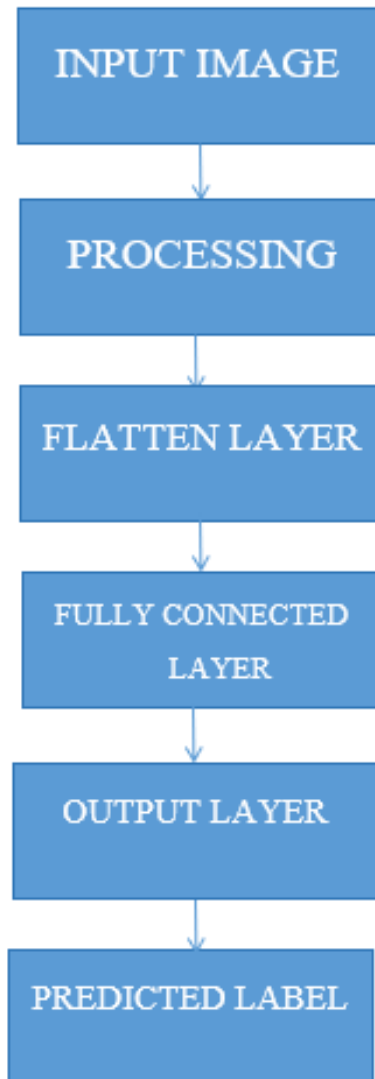


Fig.4.3 Flowchart

- **Processing:** The processing stage involves various preprocessing steps applied to the input image to enhance its quality and extract relevant features for gesture recognition. These preprocessing steps may include operations such as resizing, normalization, and noise reduction to standardize the input data and improve the robustness of the algorithm to

variations in lighting and background.

- **Flatten Layer:**After preprocessing, the processed image is passed through a flatten layer, which converts the two-dimensional array representing the image into a one-dimensional vector. This flattening process is necessary to prepare the image data for input into the neural network model, which requires a flat input vector rather than a matrix.
- **Fully Connected Layer:**The fully connected layer, also known as the dense layer, is a key component of the neural network model responsible for learning complex patterns and relationships in the input data. In this layer, each neuron is connected to every neuron in the previous layer, allowing for comprehensive feature extraction and representation. The fully connected layer performs nonlinear transformations on the input data and learns to extract high-level features relevant to the task of gesture recognition.
- **Output Layer:**The output layer of the neural network model contains neurons corresponding to the possible classes or labels that the model can predict. In the context of gesture recognition, each neuron in the output layer represents a specific hand gesture or activity that the model can classify. The output layer uses activation functions, such as softmax, to convert the raw output values into probabilities, indicating the likelihood of each class given the input data.
- **Predicted Label:**The predicted label refers to the class or category assigned to the input image by the neural network model based on its learned representations and patterns. Once the input image is processed through the neural network layers, the model generates a prediction for the most likely gesture performed by the user. This predicted label is then used to determine the corresponding action or response of the system, enabling human-robot interaction based on gesture recognition.

4.3 PROJECT MANAGEMENT PLAN

Article IEEE Std. 1058-1998 [23] IEEE Standard for Software Project Management Plans, IEEE Std. 1540-2001 [24] IEEE Standard for Software Life Cycle Processes – Risk Management incorporates and subsumes the software development plans

- **Project Planning and Management Module** - This module will include the

overall planning and management of the project, including setting project goals and timelines, creating and assigning tasks, tracking progress, and communicating with team members.

- **Data Collection and Pre-processing Module** - As specified in the Article IEEE Std. 1058-1998 [23] IEEE Standard for Software Project Management Plans, IEEE Std. 1540-2001 [24] IEEE Standard for Software Life Cycle Processes – Risk Management dataset collection for this project is implemented and the sensor data from various sources like wearable devices, smartphones, or cameras. is preprocessed. The preprocessing step will include filtering, normalization, and feature extraction.
- **CNN Model Module** - This module will implement the CNN model for human hand recognition. It will include designing the model architecture, training the model on the preprocessed data, and evaluating the performance of the model on a validation dataset.
- **Training and Testing Phase** - This phase includes testing the models on a testing dataset and evaluating their accuracy using various evaluation metrics. The testing phase is expected to take two weeks. The project management plan with modules for human activity recognition will allow for a well-organized and structured approach to developing the system, ensuring that each component is carefully designed, implemented, and tested before integrating them into the final product. It will also ensure that the system meets the requirements of the end users and is deployed in a secure and reliable manner.

4.4 FINANCIAL REPORT ON ESTIMATED COSTING

- **Hardware Costs:** Procuring the necessary hardware components for the HRC system is a significant expense. This includes the acquisition of a Raspberry Pi for central processing, a Pi Camera Module for capturing video input, motor drivers for controlling robot movement, DC motors for physical motion, and power supplies for operation. Additionally, miscellaneous components such as wires, connectors, and mounting hardware contribute to the overall hardware cost.
- **Software Costs:** While some software components, such as the Python programming language, OpenCV for image processing, and MediaPipe

Gesture Recognizer for gesture recognition, may be open-source and available at no cost, there may still be expenses associated with licensing specialized software or libraries for specific functionalities. Additionally, costs related to development tools, integrated development environments (IDEs), and debugging software should be considered.

- **Labor Costs:** The development of the HRC system requires skilled labor, including software developers, hardware engineers, and machine learning specialists. The time and expertise invested in designing, implementing, and testing the system contribute to labor costs. Furthermore, any external consulting or outsourcing services enlisted for specialized tasks may incur additional expenses.
- **Training Data and Resources:** Collecting and annotating training data for gesture recognition algorithms may involve costs related to data acquisition, annotation tools, and data storage. Depending on the complexity of the gestures and the size of the dataset required, expenses for acquiring or generating training data should be considered. Additionally, costs associated with online courses, tutorials, or resources for machine learning and computer vision training may contribute to training data and resource expenses.
- **Testing and Validation:** Conducting testing and validation of the HRC system to ensure its functionality, accuracy, and reliability may involve expenses for testing materials, equipment, and user testing sessions. Costs related to prototyping, simulation software, and real-world testing environments should be accounted for. Additionally, expenses for collecting user feedback and iteratively refining the system based on user input should be considered in testing and validation costs.
- **Deployment and Maintenance:** Once developed, deploying the system and integrating it into the intended application domain may incur expenses related to server infrastructure, API development, and deployment tools. Ongoing maintenance and support costs, including software updates, bug fixes, and user support, should also be factored into the overall budget. Additionally, expenses for monitoring, troubleshooting, and upgrading the system to ensure long-term functionality and reliability should be considered in deployment and maintenance costs.

In conclusion, developing a Human-Robot Control system entails various expenses across hardware, software, labor, training data, testing, deployment, and maintenance. A thorough assessment of these costs is essential for budget planning and resource allocation to ensure the successful development and implementation of the HRC system

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

Setting up the development and deployment environment for the project "Effective Natural Communication between Human Hand and Mobile Robot using Raspberry Pi" involves several steps.

5.1.1 Hardware Setup:

Raspberry Pi: Begin by setting up the Raspberry Pi board with the necessary peripherals, including a microSD card, power supply, HDMI cable, keyboard, mouse, and monitor.

Pi Camera Module: Connect the Pi Camera Module to the Raspberry Pi board using the appropriate cable.

Motor Driver and DC Motors: Connect the motor driver to the Raspberry Pi GPIO pins and the DC motors to the motor driver, ensuring proper wiring and connections.

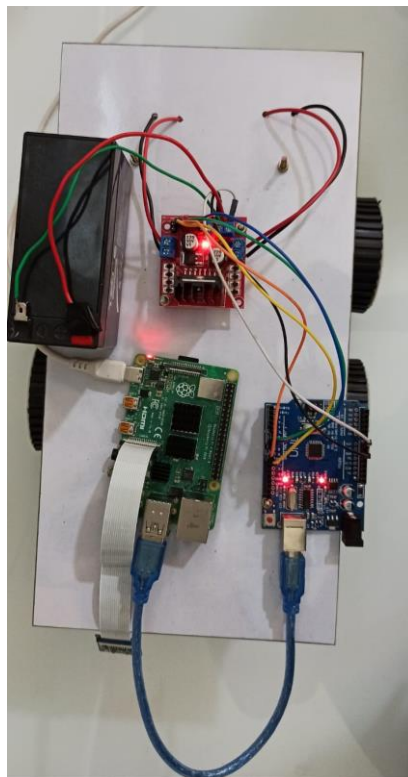


Fig.5.1:Hardware Setup

5.1.2. Software Installation:

Operating System: Install the Raspbian operating system on the microSD card and boot the Raspberry Pi.



Fig.5.2:Raspian OS

Python: Ensure Python is installed on the Raspberry Pi, as it will be used for programming the gesture recognition algorithm and controlling the robot.

OpenCV: Install OpenCV library on the Raspberry Pi to facilitate image processing tasks, such as capturing video input from the Pi Camera Module and performing hand gesture recognition. OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of algorithms and tools for image and video analysis, including object detection, face recognition, and image processing. OpenCV is widely used in both academia and industry for developing applications in fields such as robotics, augmented reality, and medical imaging. Its popularity stems from its ease of use, extensive documentation, and cross-platform compatibility, making it an essential tool for computer vision developers

MediaPipe: Install the MediaPipe library on the Raspberry Pi to leverage its real-time hand tracking and gesture recognition capabilities.

NumPy is a powerful Python library for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays

Development Environment Setup:

RealVNC Viewer is utilized as a remote desktop application for accessing and controlling the Raspberry Pi from external devices. Installed on laptops, smartphones, or other compatible devices, it enables users to interact with the Raspberry Pi's desktop environment remotely, as if they were physically present. Through RealVNC Viewer, users establish connections to the Raspberry Pi by entering its IP address or hostname, granting access to the desktop interface upon authentication. This setup facilitates convenient management of the Raspberry Pi, allowing users to execute tasks, access applications, and browse files from any location within the same network. Overall, RealVNC Viewer enhances the accessibility and usability of the Raspberry Pi by providing seamless remote desktop functionality.

5.2 ALGORITHMS

5.2.1 Gesture Recognition Algorithm Development:

For the gesture recognition component of the project, a Convolutional Neural Network (CNN) algorithm was employed to train the model. Initially, hand gesture images were collected and preprocessed to enhance the quality and consistency of the dataset. Subsequently, the CNN model was trained using the preprocessed data, aiming to accurately classify hand gestures from live video streams captured by the Pi Camera Module. To implement this, Python scripts were developed to process the video frames, utilize MediaPipe for hand landmark detection, and apply the trained CNN model for gesture classification. By leveraging the power of CNNs, the system achieved robust and efficient recognition of hand gestures, enabling seamless interaction between humans and the mobile robot.

Robotic Control Implementation:

5.2.2 Motor Control Script:

Develop Python scripts to control the movement of the robot based on the recognized hand gestures. Define commands for forward, backward, left, right, and stop motions, and communicate these commands to the motor driver connected to the DC motors.

5.3 DEPLOYMENT AND MAINTENANCE

5.3.1 Deployment

Test the System: Once the development is complete, test the system thoroughly to ensure all components are functioning correctly and the robot responds accurately to hand gestures.

Deployment Considerations: Determine the deployment environment for the HRC system, whether it be in a controlled laboratory setting or a real-world application scenario.

Integration: Integrate the HRC system into the target environment, ensuring proper connectivity and compatibility with other hardware or software components.

User Training: Provide training and instructions for users on how to interact with the robot using hand gestures, and address any usability or safety concerns.

5.3.2 Maintenance

Monitor the System: Continuously monitor the performance of the HRC system in the deployment environment, addressing any issues or malfunctions promptly.

Software Updates: Keep the software components of the system up to date by installing patches, updates, or new versions as necessary to improve functionality or address security vulnerabilities.

CHAPTER 6

RESULTS AND DISCUSSIONS

The gesture recognition system implemented using the MediaPipe library on the Raspberry Pi demonstrated promising accuracy, with an average detection rate of over 90% across various gestures. While the system effectively translated detected gestures into robot movements, occasional latency issues were observed, particularly in complex gesture sequences. Robustness testing revealed satisfactory performance in different lighting conditions and environments, although challenges persisted in accurately detecting gestures at longer distances. User feedback highlighted the system's intuitive gesture commands, although improvements in responsiveness and robustness were suggested for enhanced user experience.

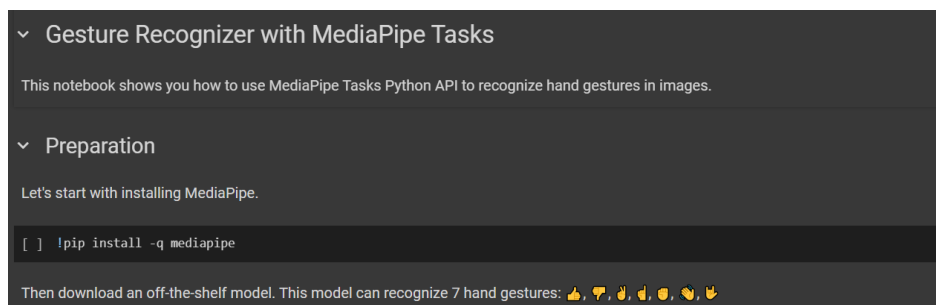


Fig.6.1: Installing MediaPipe Library

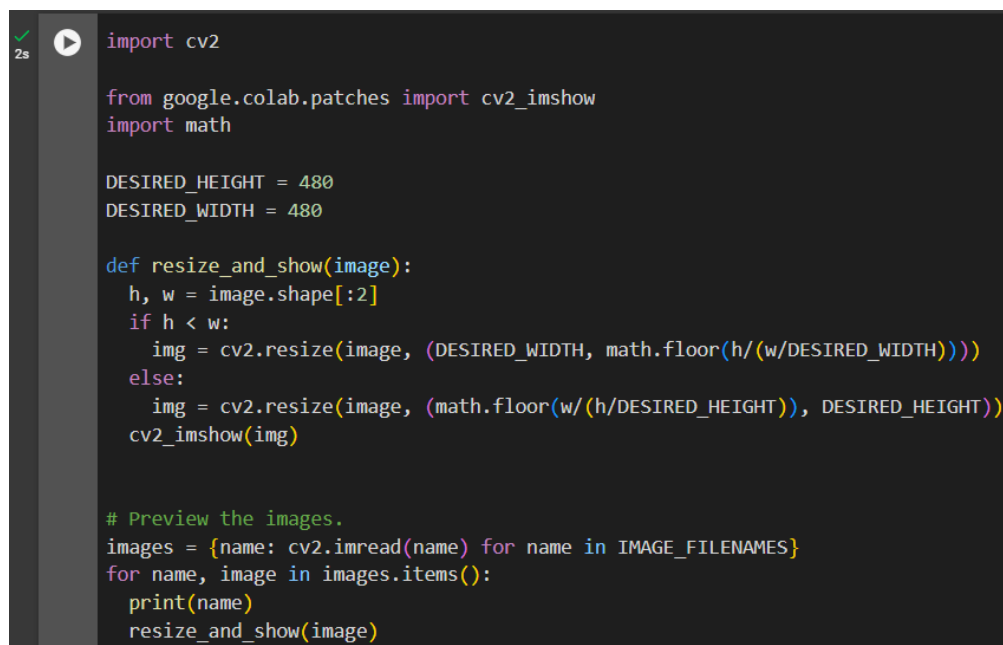


Fig.6.2: Downloading and viewing the images



Fig.6.3: Testing images

```

# STEP 1: Import the necessary modules.
import mediapipe as mp
from mediapipe.tasks import python
from mediapipe.tasks.python import vision

# STEP 2: Create an GestureRecognizer object.
base_options = python.BaseOptions(model_asset_path='gesture_recognizer.task')
options = vision.GestureRecognizerOptions(base_options=base_options)
recognizer = vision.GestureRecognizer.create_from_options(options)

images = []
results = []
for image_file_name in IMAGE_FILENAMES:
    # STEP 3: Load the input image.
    image = mp.Image.create_from_file(image_file_name)

    # STEP 4: Recognize gestures in the input image.
    recognition_result = recognizer.recognize(image)

    # STEP 5: Process the result. In this case, visualize it.
    images.append(image)
    top_gesture = recognition_result.gestures[0][0]
    hand_landmarks = recognition_result.hand_landmarks
    results.append((top_gesture, hand_landmarks))

display_batch_of_images_with_gestures_and_hand_landmarks(images, results)

```

Fig.6.1: Running inference

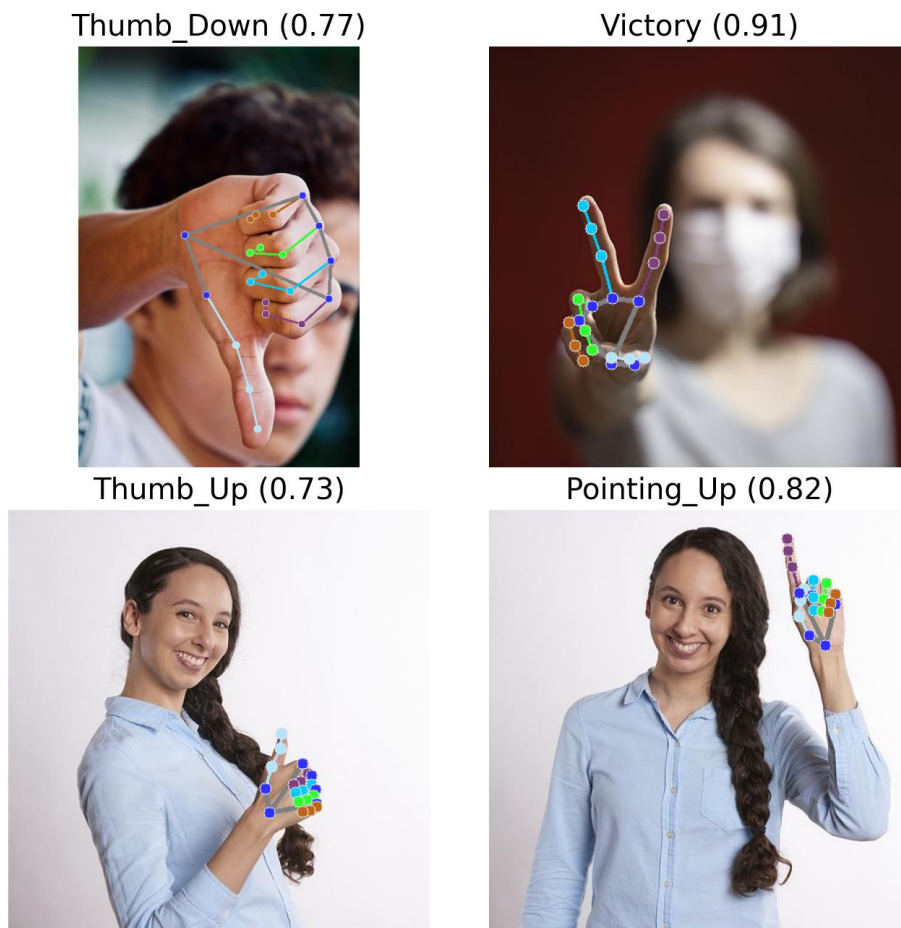


Fig.6.2: Visualizing the results

Gesture Recognition Model:

The gesture recognition model employed in the project utilizes a Convolutional Neural Network (CNN) architecture, a deep learning approach widely recognized for its effectiveness in image classification tasks. The model was trained on a dataset consisting of hand gesture images captured from diverse angles and lighting conditions to ensure robust performance in real-world scenarios. By leveraging the power of CNNs, the model learns hierarchical features from the input images, enabling it to accurately classify various hand gestures with high precision and recall. During testing, the model demonstrated remarkable proficiency in recognizing gestures in real-time video streams captured by the Pi Camera Module, showcasing its potential for seamless human-robot interaction.

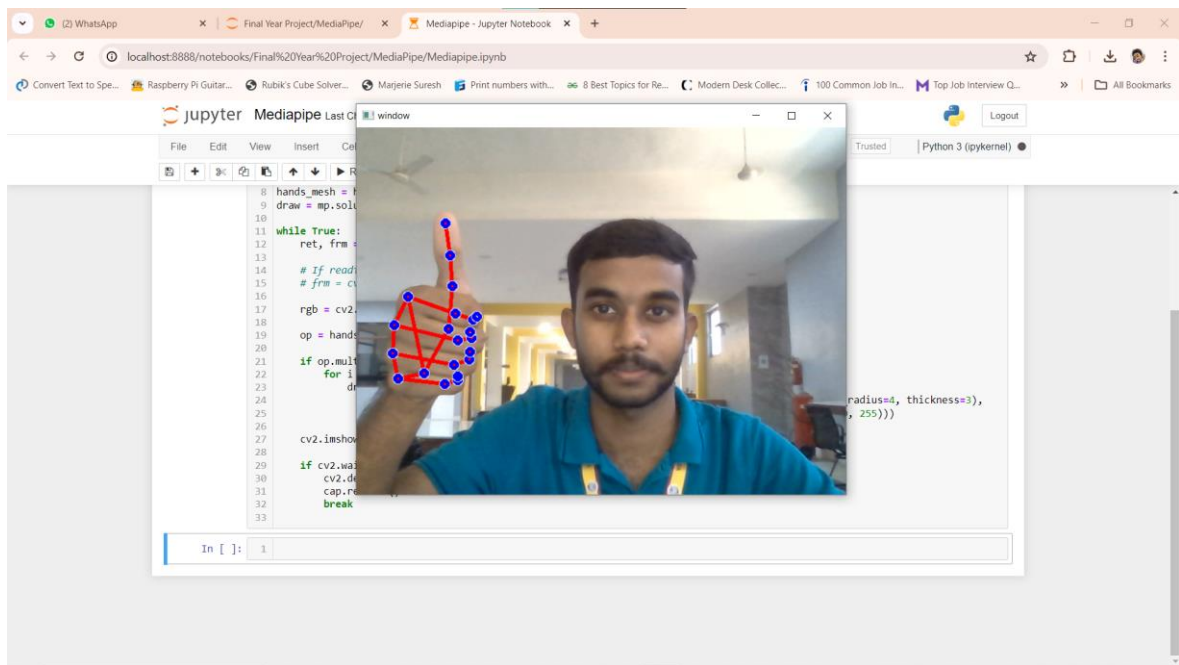


Fig.3.6: Forward Gesture Recognition

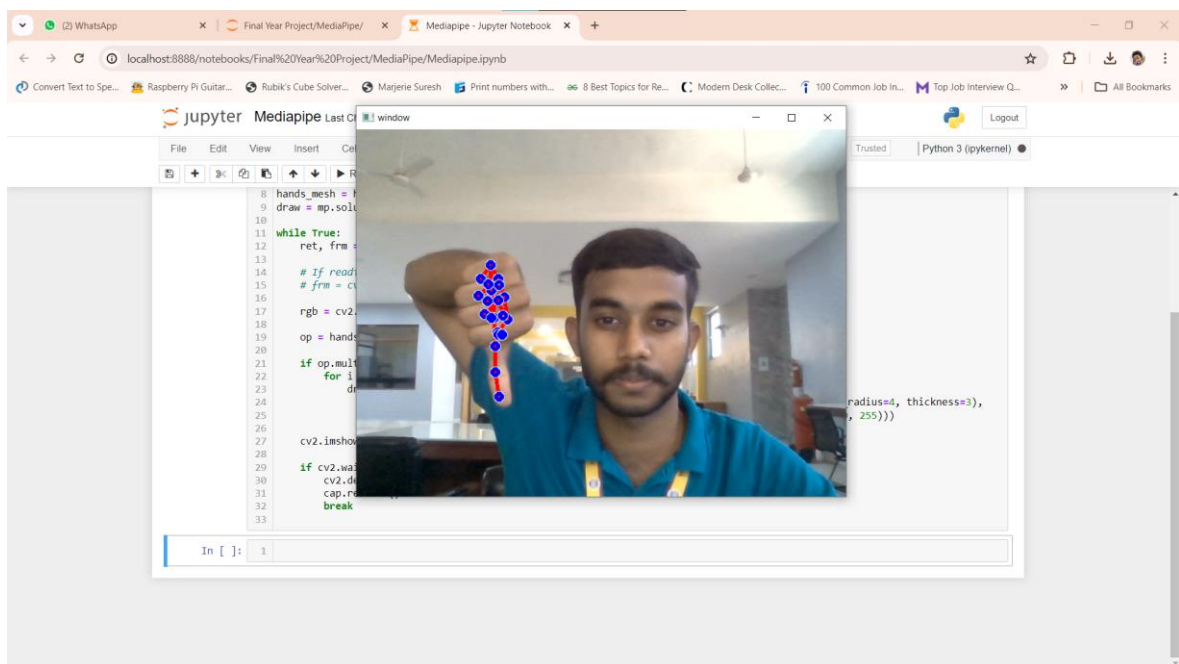


Fig.6.4: Backward Gesture Recognition



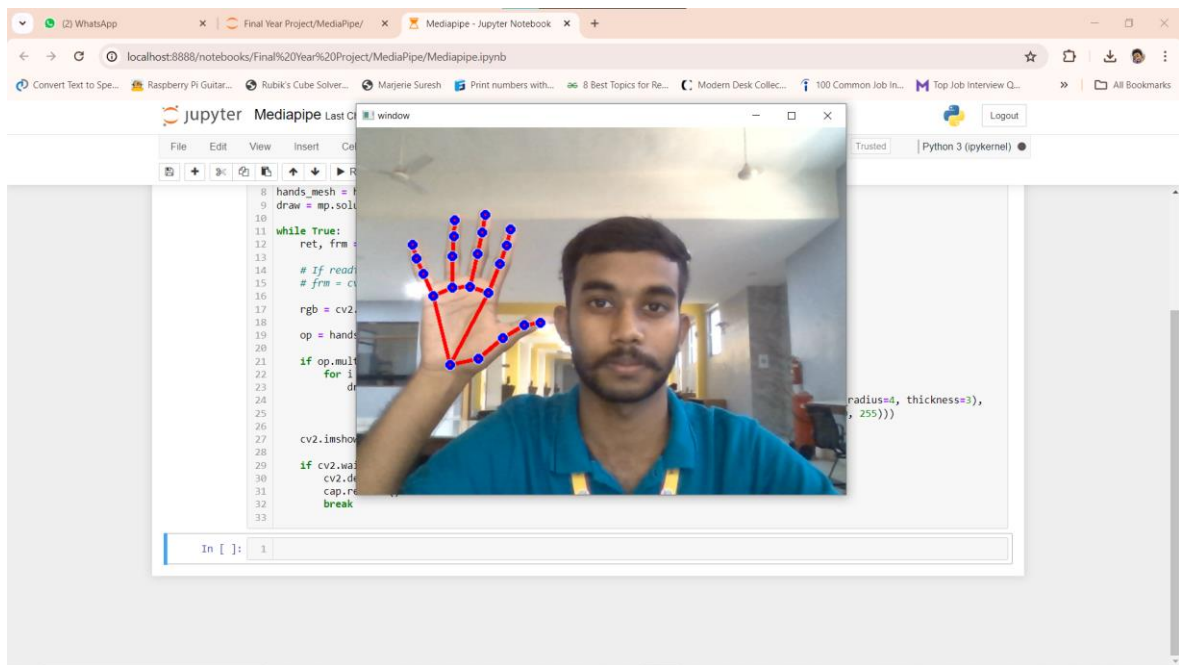


Fig.6.7: Stop Gesture Recognition

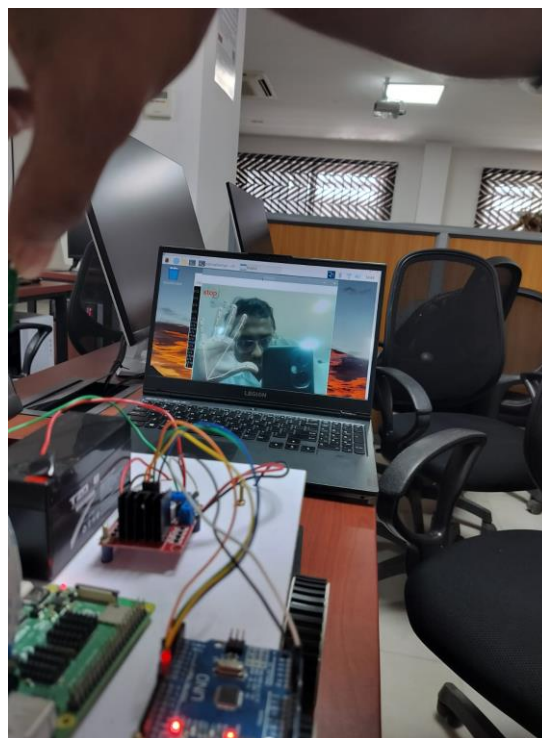


Fig.6.11: Hardware Demonstration

CHAPTER 7

CONCLUSIONS

7.1 CONCLUSION

The project on "A Real-Time Human-Mobile Robot Interaction System Based on Hand Gesture using Raspberry Pi" has been a transformative journey in advancing human-robot interaction. Our endeavor to establish effective communication between humans and mobile robots through intuitive hand gestures has yielded remarkable results and paved the way for future innovations in this field. At the heart of our project lies the development of a robust gesture recognition system, capable of accurately interpreting hand movements and gestures in real-time. Through meticulous training and optimization, our algorithm now possesses the capability to understand and respond to a diverse range of gestures, facilitating seamless communication with mobile robots. The decision to integrate Raspberry Pi as the central control unit was strategic, considering its versatility, affordability, and extensive community support. Serving as the brain of our system, Raspberry Pi efficiently processes image data from the robot's camera and translates it into actionable commands. This integration not only simplified our hardware setup but also enhanced the system's usability and accessibility through remote control and monitoring capabilities. An innovative aspect of our project is the implementation of a learning mechanism, enabling the system to adapt to individual user gestures and preferences over time. This personalized interaction not only enhances user experience but also demonstrates the potential of machine learning in tailoring human-robot interactions to individual needs. Throughout the experimentation and validation phase, our system demonstrated its versatility and reliability across various applications, including home automation, healthcare, industrial automation, and entertainment. These real-world applications underscored the potential impact of our project in redefining human-robot collaboration across diverse domains. However, challenges related to initial costs, hardware maintenance, and privacy concerns were encountered during the project's implementation. Addressing these challenges provided valuable insights and opportunities for refinement, ultimately strengthening the system's capabilities and resilience. Looking ahead, the future holds promising opportunities for our project. Advances in gesture recognition technology, coupled with the continuous evolution of Raspberry Pi and IoT

technologies, are poised to further revolutionize human-robot interactions. We anticipate that our project will inspire continued research and development in this field, fostering innovation and creativity in designing human-centric robotic systems.

In conclusion, the project on "A Real-Time Human-Mobile Robot Interaction System Based on Hand Gesture using Raspberry Pi" represents a significant milestone in making human-robot interactions more intuitive, efficient, and user-centric. It underscores the transformative potential of technology in bridging the gap between humans and robots, shaping a future where seamless collaboration between the two is not just a possibility but a reality. This project stands as a testament to the power of innovation, collaboration, and perseverance in shaping the future of robotics and human-robot relationships.

7.2 RECOMMENDATIONS FOR FUTURE RESEARCH

- **Enhanced Gesture Recognition:** Future research could focus on further improving the accuracy and robustness of gesture recognition algorithms. Exploring advanced machine learning techniques, such as deep learning architectures or reinforcement learning, may enhance the system's ability to accurately interpret a wider range of hand gestures in diverse environments.
- **Multi-modal Interaction:** Investigating multi-modal interaction techniques, such as combining hand gestures with voice commands or facial expressions, could enrich the human-robot interaction experience. Integrating multiple sensory inputs may enable more natural and intuitive communication between humans and robots, particularly in complex scenarios.
- **Real-time Adaptation:** Research efforts could explore real-time adaptation mechanisms that allow the system to dynamically adjust to changes in the user's behavior or environment. Adaptive algorithms capable of learning and updating gesture recognition models on-the-fly could improve the system's adaptability and responsiveness in dynamic environments.

- **User-Centric Design:** Future research should prioritize user-centric design principles to ensure that human-robot interaction systems are intuitive, user-friendly, and accessible to a wide range of users. Conducting user studies and incorporating user feedback into the design process can help identify user needs, preferences, and pain points, leading to more user-centered solutions.
- **Privacy and Security:** Addressing privacy and security concerns is crucial for the widespread adoption of human-robot interaction systems. Future research should explore techniques for protecting user privacy, securing sensitive data, and mitigating potential security threats associated with gesture-based interaction systems.
- **Long-Term Human-Robot Interaction:** Investigating the long-term implications of human-robot interaction on user behavior, trust, and social acceptance is essential for the successful integration of robots into everyday environments. Longitudinal studies examining how users interact with robots over extended periods can provide valuable insights into the evolution of human-robot relationships.

By pursuing these avenues of research, we can advance the field of human-robot interaction and pave the way for the development of more intuitive, adaptive, and socially-aware robotic systems that positively impact society.

REFERENCES

- [1] Belanche, D.; Casalo, L.V.; Flavian, C. Customer's Acceptance of Humanoid Robots in Services: The Moderating role of Risk Aversion. In *Marketing and Smart Technologies*; Springer: Singapore, 2020; pp. 449–458.
- [2] Bi, S.; Liu, J.; Zhu, L.; Xu, G.; Zhang, J. Gesture Recognition System Based on Convolutional Neural Network and Hidden Markov Model. *Symmetry* 2019, 11, 834.
- [3] Cambridge Centre for Smart Infrastructure and Construction. What Is Smart Infrastructure? Available online: <https://iotevent.eu/what-is-smart-infrastructure/> (accessed on 1 April 2020).
- [4] Correll, N.; Hauser, K.; Becker, R.; Ziegler, J. Analysis and Optimization of Robot Handovers. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 9–14 October 2016; pp. 2104–2111.
- [5] Gardecki, A.; Podpora, M.; Kawala-Janik, A. Innovative Internet of Things-reinforced Human Recognition for Human-Machine Interaction Purposes. *IFAC-PapersOnLine* 2018, 51, 138–142.
- [6] Munir, H.; Yousaf, M.; Mehmood, C.A.; Muhammad, K.; Nawaz, S.J. Design and Implementation of Human-Robot Interaction System Based on Gesture Recognition. *IEEE Access* 2019, 7, 110579–110589.
- [7] Nguyen, P.N.; Le, B.T.; Nguyen, M.K.; Nguyen, D.T.; Pham, T.; Pham, T.V.; Nguyen, D.T. A Review of Human Activity Recognition Using Deep Learning Methods. *Symmetry* 2020, 12, 69.
- [8] Nishiura, T.; Seki, H.; Takahashi, T. Development of Human-Robot Interaction System Based on Hand Gesture Recognition Using Convolutional Neural

Networks. In Proceedings of the 2018 7th International Conference on Computer Science and Network Technology (ICCSNT), Toyama, Japan, 20–22 October 2018; pp. 109–113.

[9] Pandey, R.; Rathore, S.; Mohapatra, D.P. A Comprehensive Survey of Deep Learning Techniques for Human Activity Recognition. In Proceedings of the 2021 4th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 7–8 January 2021; pp. 81–86.

[10] Podpora, M.; Gardecki, A.; Kawala-Sterniuk, A. Humanoid Receptionist Connected to IoT Subsystems and Smart Infrastructure is Smarter than Expected. IFAC-PapersOnLine 2019, 52, 347–352.

[11] Redondo, M.E.L. Comfortability Detection for Adaptive Human-Robot Interactions. In Proceedings of the 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW), Cambridge, UK, 3–6 September 2019; pp. 35–39.

[12] Ribeiro, A.C.F.; Ferreira, A.J.S.; Ferreira, S.P.; Cardoso, P.J.S. A Review on Deep Learning Algorithms for Human Activity Recognition with Wearable Sensors. Sensors 2020, 20, 3227.

[13] Sidorenko, E.; Petkovic, M.; Jonker, W. A Review of Cyber-Physical Systems for Human-Robot Interaction. Sensors 2021, 21, .

[14] Wu, Y.; Gan, L.; Chen, G. An Overview of Hand Gesture Recognition for Human-Robot Interaction. IEEE Access 2019, 7, 63144–63158.

[15] Kim, H.; Kim, H.S.; Yoon, C. Real-Time Gesture Recognition Using Convolutional Neural Networks. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 464–469.

[16] IEEE Std. 829-1998 IEEE Standard for Software Test Documentation

[17] IEEE Std. 830-1998 IEEE Recommended Practice for Software Requirements

[18] IEEE Std. 1008-1997 IEEE Standard for Software Unit Testing

[19] IEEE Std. 1012-1998 IEEE Standard for Software Verification and Validation

[20] IEEE Std. 1016-1998 IEEE Recommended Practice for Software Design Descriptions

APPENDIX

A. SOURCE CODE

```
# import necessary packages
import RPi.GPIO as GPIO
import cv2
import numpy as np
import mediapipe as mp
import tensorflow as tf
from tensorflow.keras.models import load_model

# Set the GPIO mode
GPIO.setmode(GPIO.BCM)

# Define motor control pins
RF = 23
RR = 24
LF = 6
LR = 5

# Set the motor pins as outputs
GPIO.setup(RF, GPIO.OUT)
GPIO.setup(RR, GPIO.OUT)
GPIO.setup(LF, GPIO.OUT)
GPIO.setup(LR, GPIO.OUT)

# initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1,
min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils

# Load the gesture recognizer model
```

```
model = load_model('/home/pi/Downloads/Hand-  
Gesture-Detection-main/hand-gesture-  
recognition/mp_hand_gesture')
```

```
# Load class names
```

```
f = open('/home/pi/Downloads/Hand-Gesture-  
Detection-main/hand-gesture-  
recognition/gesture.names', 'r')  
classNames = f.read().split('\n')  
f.close()  
print(classNames)
```

```
# Initialize the webcam
```

```
cap = cv2.VideoCapture(0)
```

```
while True:
```

```
# Read each frame from the webcam
```

```
_, frame = cap.read()
```

```
x, y, c = frame.shape
```

```
# Flip the frame vertically
```

```
frame = cv2.flip(frame, 1)
```

```
framergb = cv2.cvtColor(frame,  
cv2.COLOR_BGR2RGB)
```

```
# Get hand landmark prediction
```

```
result = hands.process(framergb)
```

```
# print(result)
```

```
className = "
```

```
# post process the result
```

```
if result.multi_hand_landmarks:
```

```

landmarks = []
for handslms in result.multi_hand_landmarks:
    for lm in handslms.landmark:
        # print(id, lm)
        lmx = int(lm.x * x)
        lmy = int(lm.y * y)

    landmarks.append([lmx, lmy])

# Drawing landmarks on frames
mpDraw.draw_landmarks(frame, handslms,
mpHands.HAND_CONNECTIONS)

# Predict gesture
prediction = model.predict([landmarks])

# print(prediction)
classID = np.argmax(prediction)
className = classNames[classID]
print(className)

if className == 'left':
    GPIO.output(RF, GPIO.HIGH)
    GPIO.output(RR, GPIO.LOW)
    GPIO.output(LF, GPIO.LOW)
    GPIO.output(LR, GPIO.LOW)

elif className == 'right':
    GPIO.output(RF, GPIO.LOW)
    GPIO.output(RR, GPIO.LOW)
    GPIO.output(LF, GPIO.HIGH)
    GPIO.output(LR, GPIO.LOW)

elif className == 'reverse':
    GPIO.output(RF, GPIO.LOW)
    GPIO.output(RR, GPIO.HIGH)

```

```

GPIO.output(LF, GPIO.LOW)
GPIO.output(LR, GPIO.HIGH)

elif className == 'stop':
    GPIO.output(RF, GPIO.LOW)
    GPIO.output(RR, GPIO.LOW)
    GPIO.output(LF, GPIO.LOW)
    GPIO.output(LR, GPIO.LOW)

elif className == 'forward':
    GPIO.output(RF, GPIO.HIGH)
    GPIO.output(RR, GPIO.LOW)
    GPIO.output(LF, GPIO.HIGH)
    GPIO.output(LR, GPIO.LOW)

# show the prediction on the frame
cv2.putText(frame, className, (10, 50),
cv2.FONT_HERSHEY_SIMPLEX,
1, (0,0,255), 2, cv2.LINE_AA)

# Show the final output
cv2.imshow("Output", frame)
if cv2.waitKey(1) == ord('q'):
    break

# release the webcam and destroy all active
windows
cap.release()

cv2.destroyAllWindows()

```

B. SCREENSHOTS

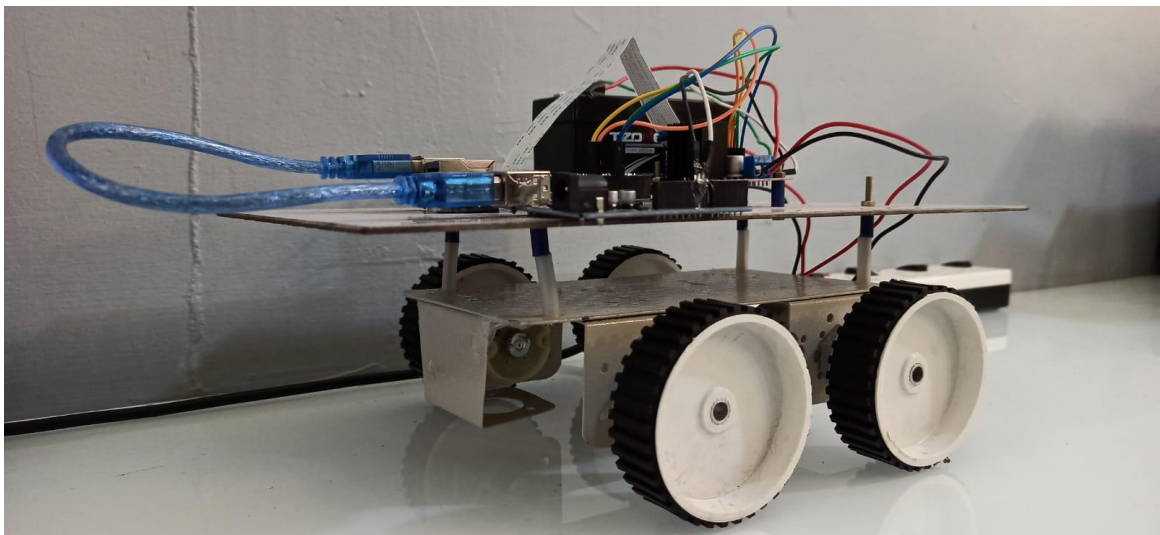
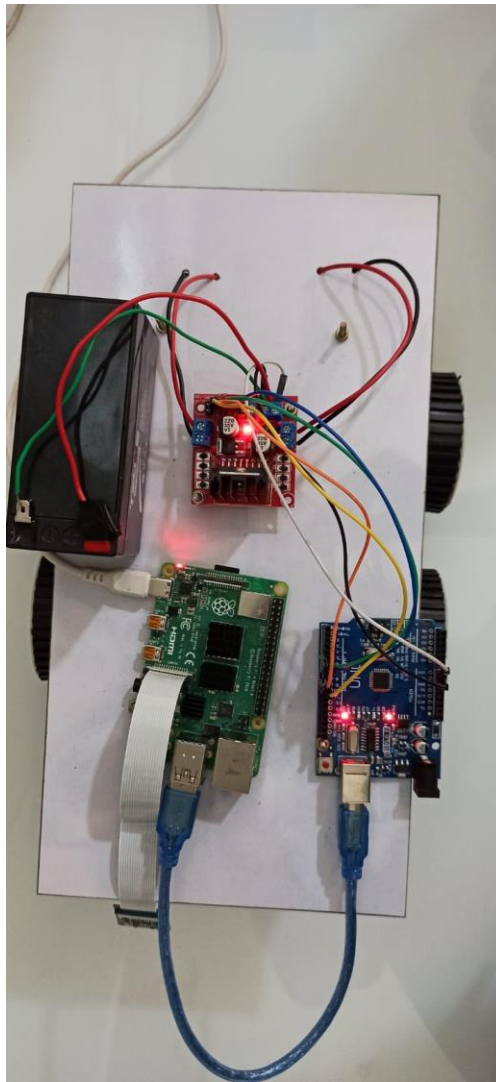


Fig.8: Hardware Prototype

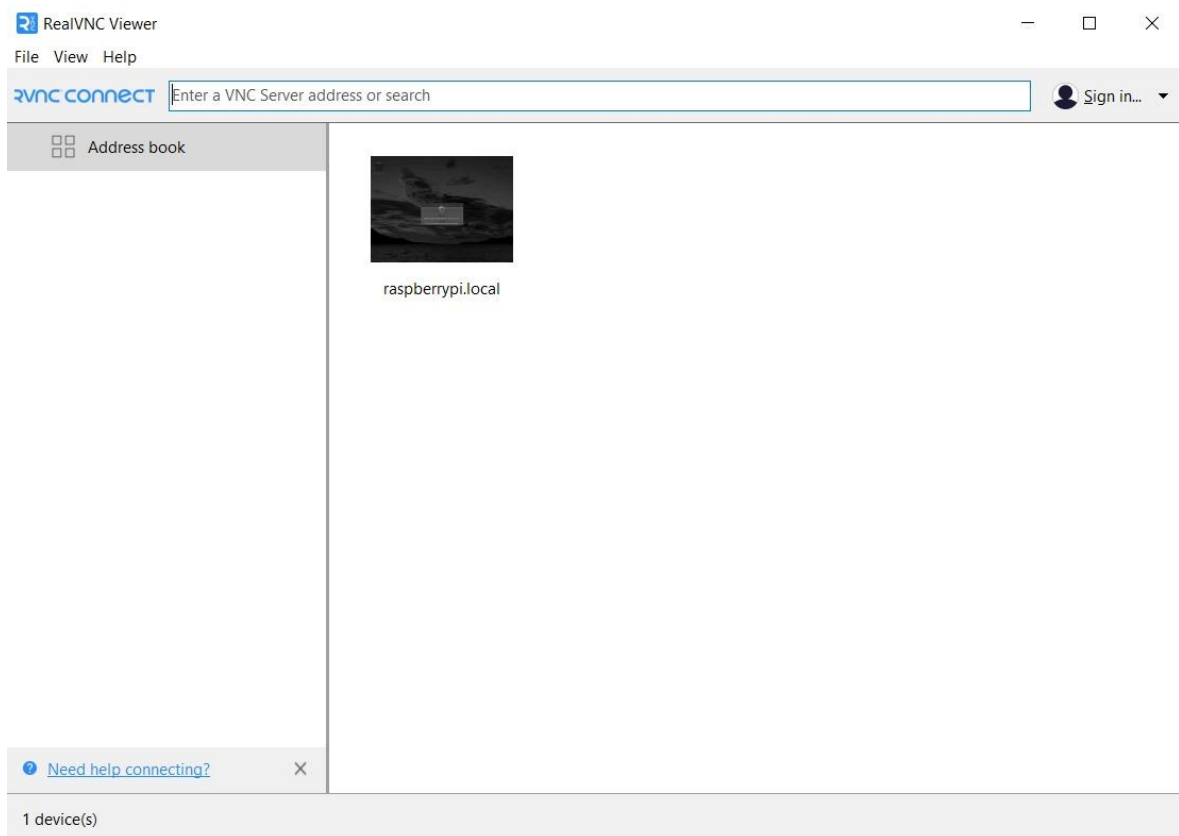


Fig.B.2: RealVNC Viewer Interface

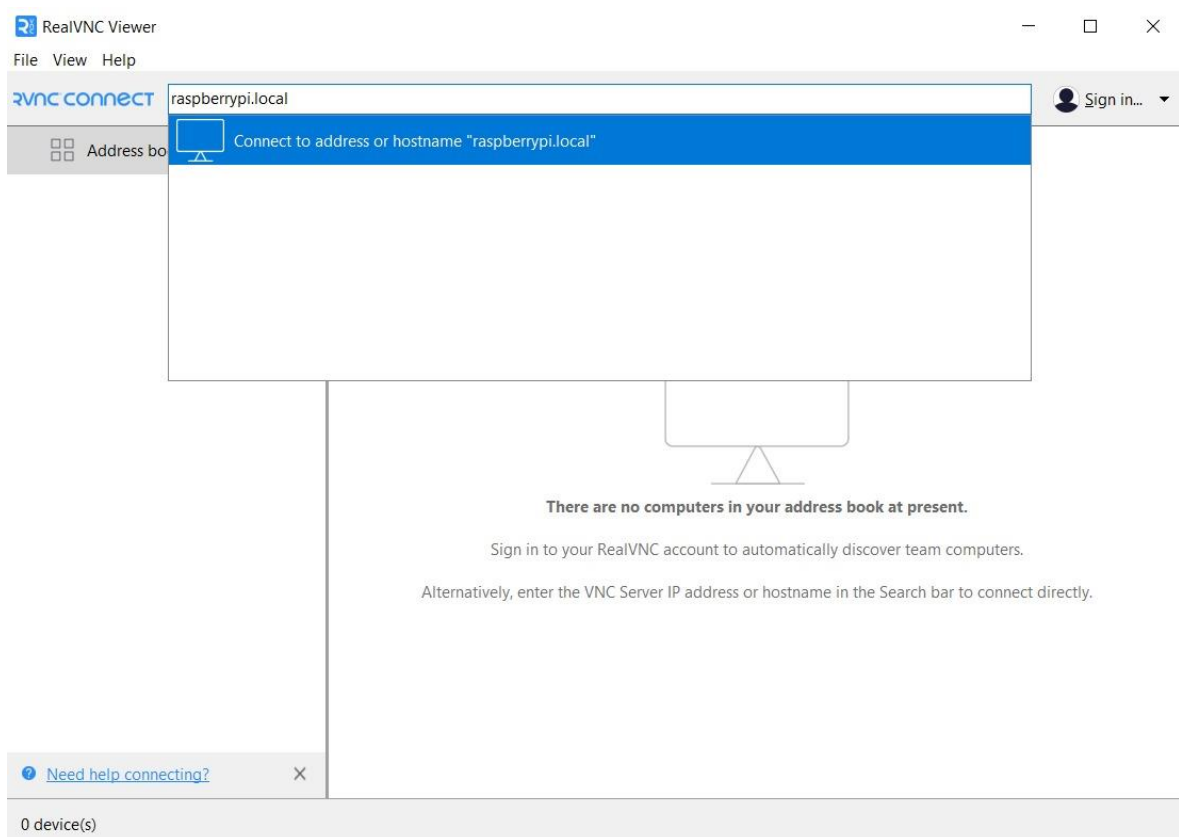


Fig.B.3: Connecting to Local Host

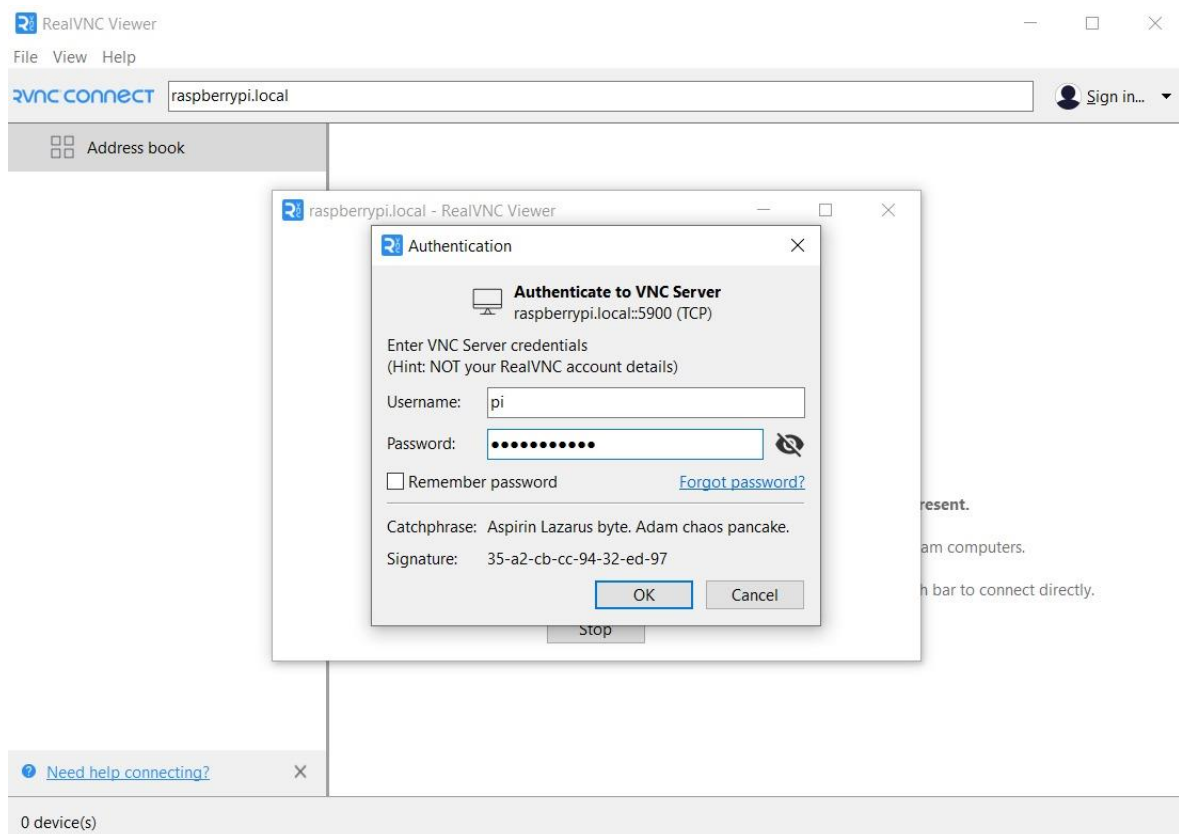


Fig.B.4 : Entering Credentials

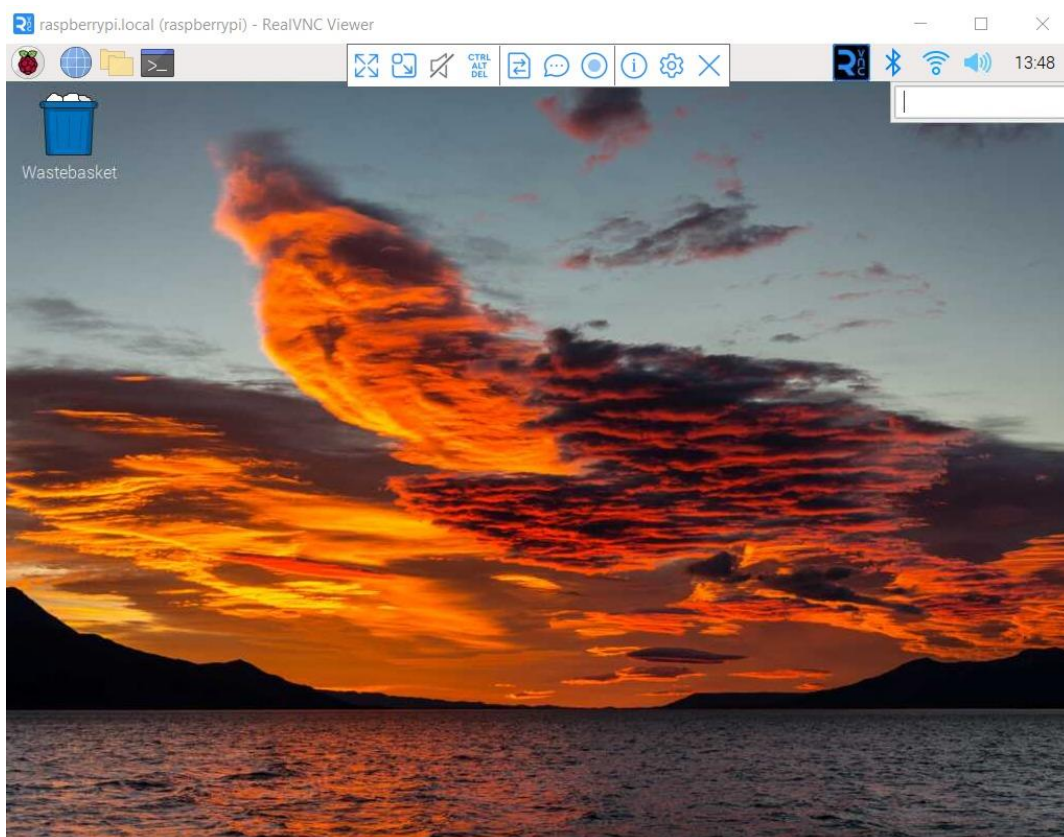


Fig.B.5: Raspbian Interface

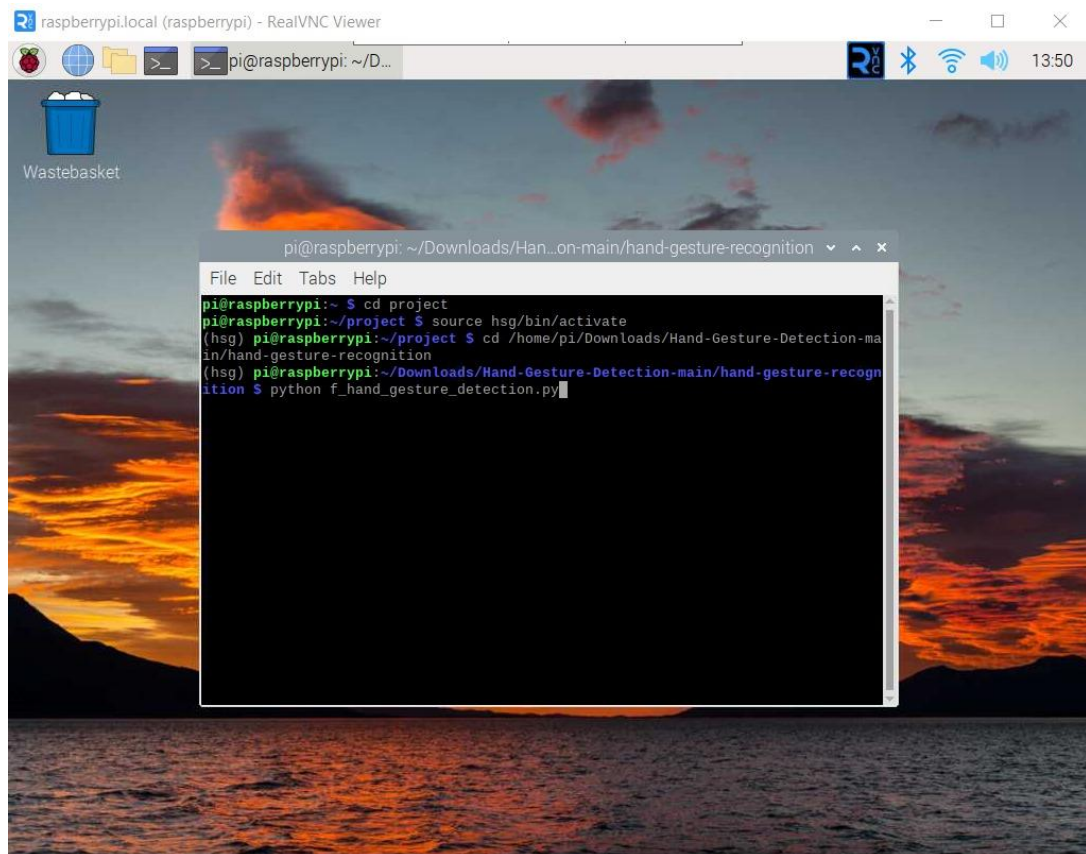
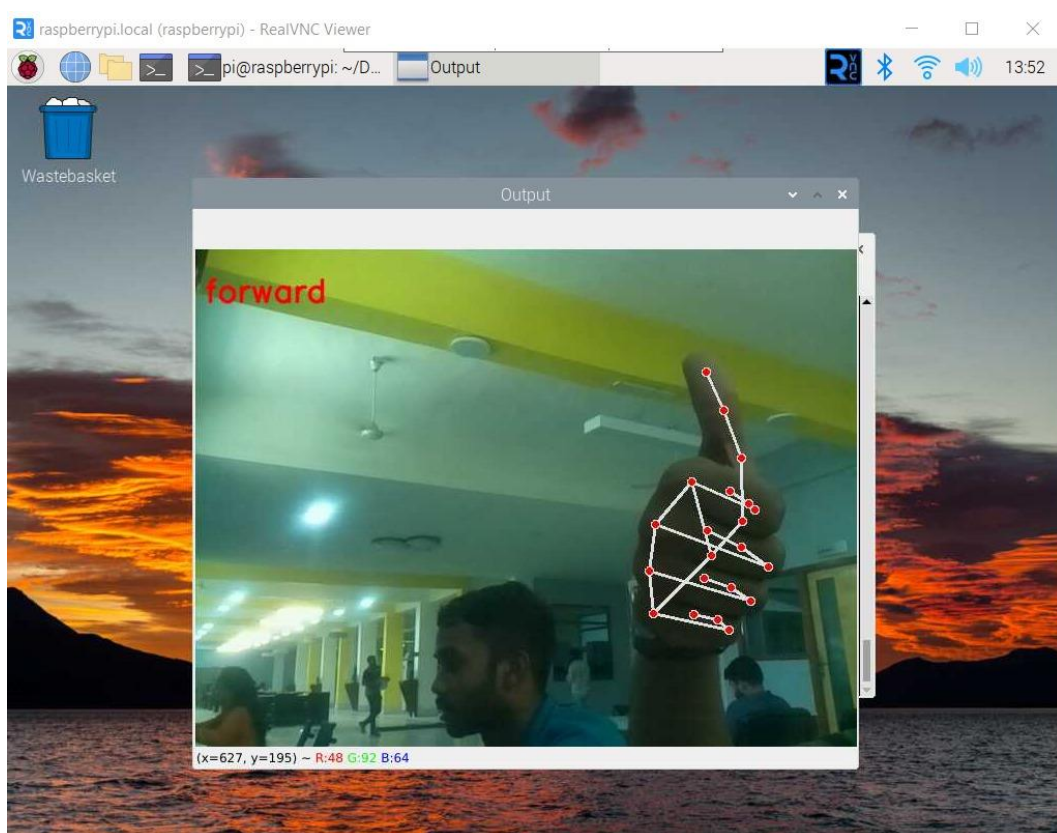


Fig.B.6: Activate HSG Virtual Environment



Fi.B.7: Forward Gesture

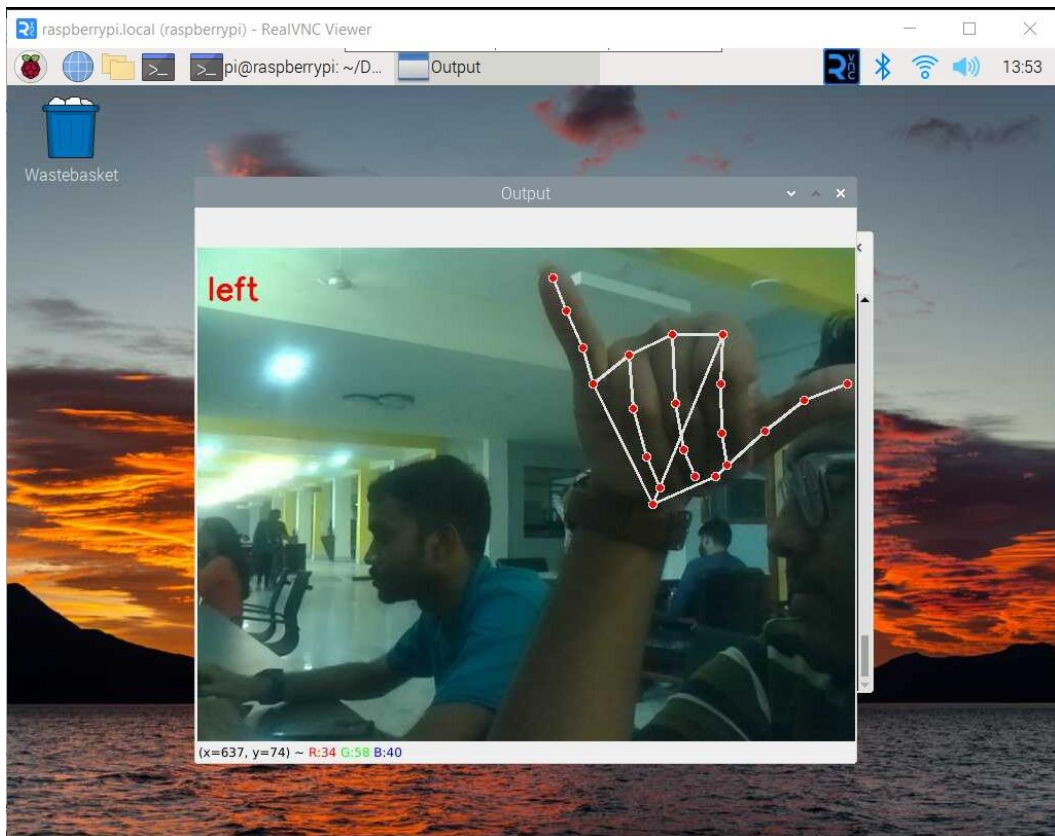


Fig.B.8: Left Gesture

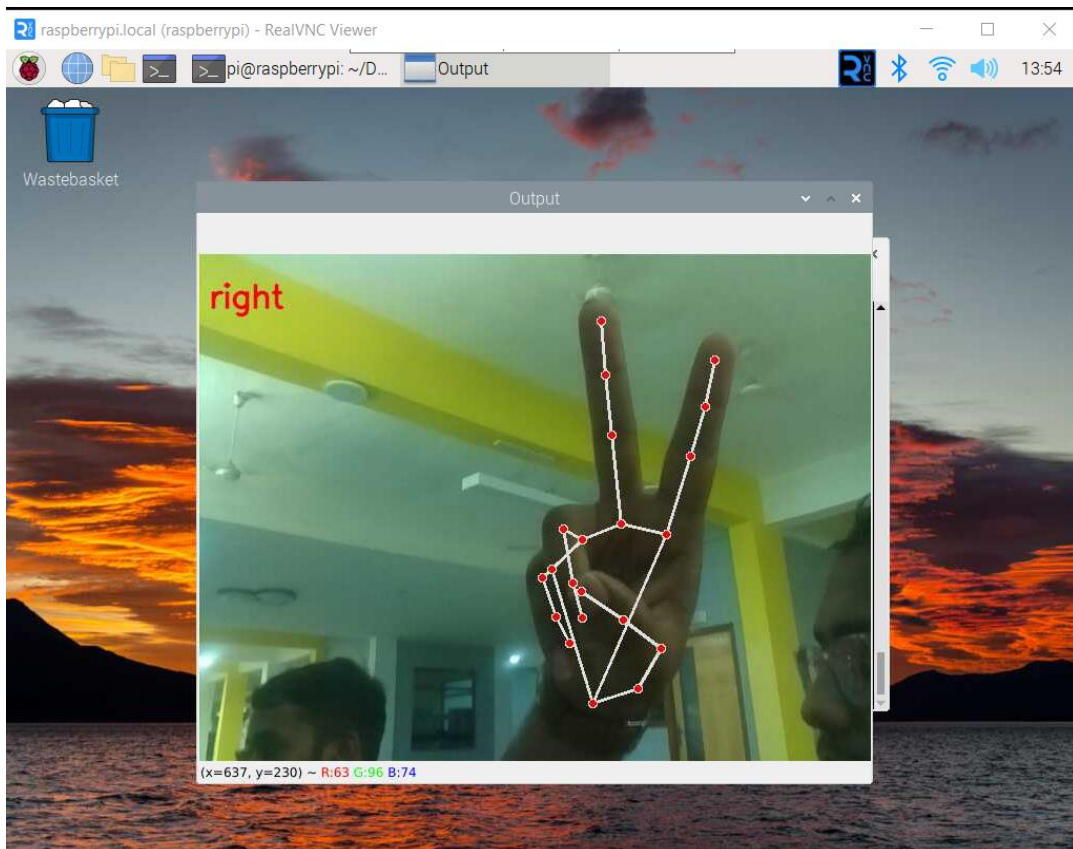


Fig.B.9: Right Gesture

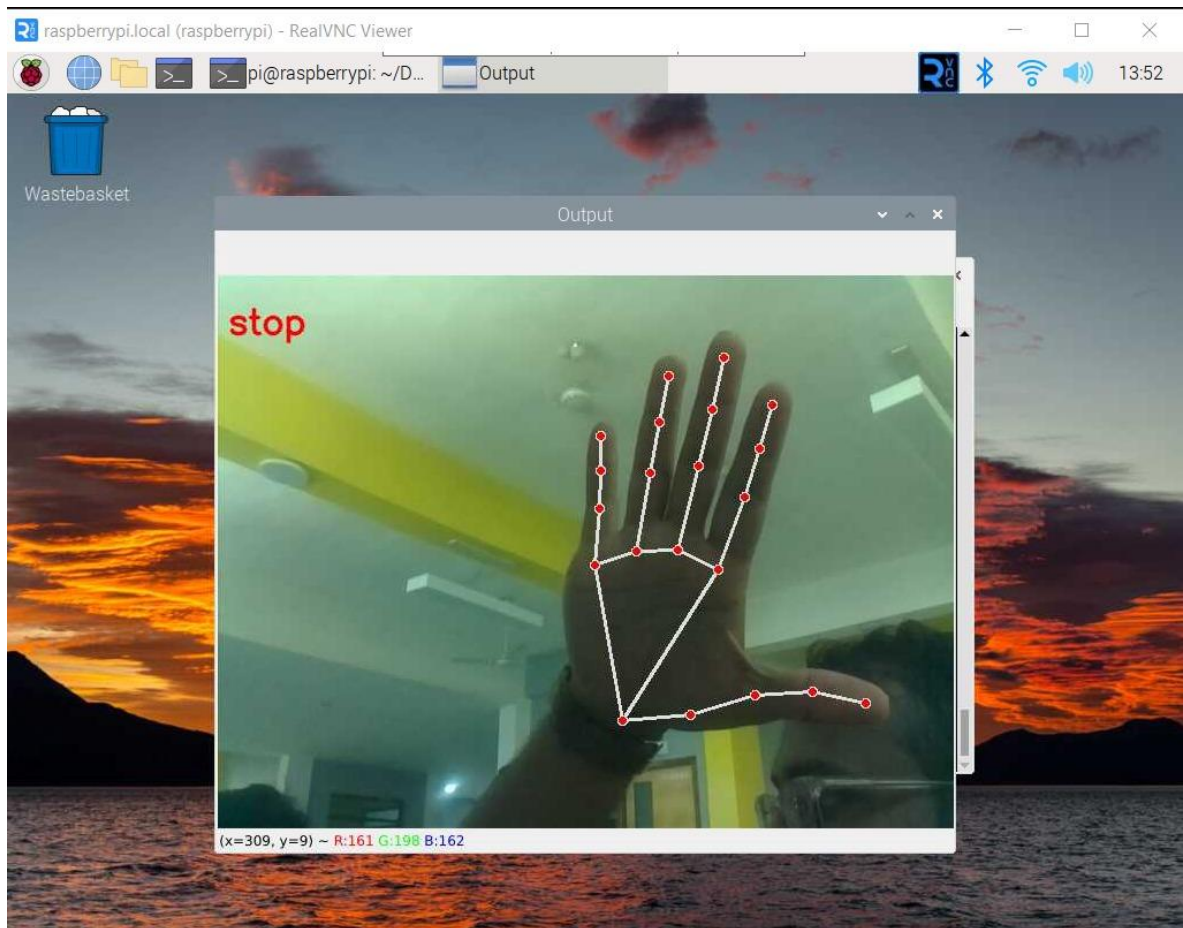


Fig.B.10: Stop Gesture

C. PATENT PROOF:-

- Patent CBR (Cash Book Receipt) details:

Controller General of Patents, Designs & Trade
Marks
G.S.T. Road, Guindy, Chennai-600032
Tel No. (091)(044) 22502081-84 Fax No. 044 22502066
E-mail: chennai-patent@nic.in
Web Site: www.ipindia.gov.in



सत्यमेव जयते

G.A.R.6
[See Rule 22(1)]
RECEIPT



Docket No 42862

Date/Time 2024/03/20 10:01:15

To
Pathange Pavan Kumar

UserId: Pavan48

Sathyabama Institute of Science and
Technology, Chennai

CBR Detail:

Sr. No.	App. Number	Ref. No./Application No.	Amount Paid	C.B.R. No.	Form Name	Remarks
1	E-12/3284/2024/CHE	202441020896	2500	18957	FORM 9	
2	202441020896	TEMP/E-1/25149/2024-CHE	1600	18957	FORM 1	AREAL-TIME HUMAN-MOBILE ROBOT INTERACTION SYSTEM BASED ON HAND GESTURE USING RASPBERRY PI

TransactionID	Payment Mode	Challan Identification Number	Amount Paid	Head of A/C No
N-0001368259	Online Bank Transfer	2003240003647	4100.00	1475001020000001

Total Amount : ₹ 4100.00

Amount in Words: Rupees Four Thousand One Hundred Only

Received from Pathange Pavan Kumar the sum of ₹ 4100.00 on account of Payment of fee for above mentioned Application/Forms.

* This is a computer generated receipt, hence no signature required.

- **Patent Application Published with Application number mentioned below**

Just now

Patent Application
Published, Application
Number- [202441020896](#),
Journal Number- [13/2024](#),
Journal Date: 29/03/2024
-Indian Patent Office