CHAPTER - 1

INTRODUCTION

Robots are rapidly becoming an integral part of modern society, revolutionizing industries and improving the quality of human life. From manufacturing plants and healthcare facilities to educational institutions and smart homes, the presence of robotic systems is expanding at an unprecedented pace. The driving force behind this growth is the increasing demand for machines that can perform human-like tasks while interacting seamlessly and naturally with users. As artificial intelligence (AI) and embedded systems advance, the expectation has shifted from robots that only follow fixed commands to robots capable of understanding human language, gestures, and behavior.

In response to this need, the AI Enhanced Multi-Modal Robot is a cutting-edge technological solution that transcends traditional limitations. Unlike conventional robots that rely solely on button-based inputs or remote controls, this intelligent robotic system can receive and interpret commands through three primary channels—voice, text, and hand gestures. It utilizes a blend of technologies including natural language processing (NLP), computer vision (using OpenCV), and embedded control systems (Arduino with Standard Firmata) to offer a highly interactive and responsive interface.

This robot is designed to enable natural human-robot interaction in diverse scenarios. A user can speak to it, type a message, or use hand movements to issue commands. The robot can see through its camera, listen through its microphone, and speak back using a speaker. It responds to real-time voice commands using speech recognition and replies using text-to-speech (TTS) synthesis. With the integration of gesture recognition modules using Media Pipe and OpenCV, it can identify hand shapes and positions to trigger specific actions. All these functionalities are built upon a robust Python-based software platform that interfaces with the Arduino hardware to control motors, sensors, and actuators effectively.

1.1 Problem Statement:

In today’s rapidly advancing technological world, traditional robots still depend on single-mode inputs like buttons or remotes, limiting their interaction flexibility and accessibility. This creates challenges for users with speech, mobility, or situational restrictions. The need arises for a robot that can understand natural human communication through multiple modalities. The AI-Enhanced Multi-Modal Robot aims to bridge this gap by supporting speech, text, and gesture-based inputs using technologies like OpenCV, NLP, and embedded systems. The goal is to create a more intelligent, adaptable, and inclusive robot that can function in dynamic real-world environments across healthcare, education, and smart living.

Key Features:

- Multi-modal Interaction: Accepts voice, text, and gesture commands.
- Uses speech-to-text and text-to-speech for verbal communication.
- Gesture Recognition: Speech Processing
- Employs OpenCV and Media Pipe for hand gesture control.
- Hardware Integration: Uses Arduino with Python (via Firmata) to manage sensors, motors, and cameras.
- User Accessibility: Designed to be inclusive in noise-sensitive or speech-impaired environments.
- Real-time Feedback: Provides quick and adaptive responses based on user input.

CHAPTER - 2

LITERATURE SURVEY

The concept of Multi-modal interaction in robotics involves the integration of multiple input methods like text, voice, vision and gestures, to make the interaction smoother and user friendly. The importance of this interaction model is improving experience and increasing adaptability of robots in practical, everyday situations. Multimodal interfaces describe systems that seek to leverage natural human capabilities to communicate via voice and gestures. Interacting with multi modal refers to a systems ability to communicate and respond using multi input sources such as voice, gestures, and vision. In Embedded systems and robotics, this approach enhances human-robot interaction by making natural communication. Where traditional robots that works on single-mode of inputs like remote controls, multi modal robots can respond to many different types of human inputs, improving usability and efficiency.

S.No	Author	Title	Year	Observation
1	Phan, Dang Khoa et al.	Innovative Multi-Modal Control for Surveillance Spider Robot	2024	Combines voice and hand gesture recognition for controlling a surveillance robot.
2	Pawar, Suvarna et al.	AI-Based Autonomous Voice-Enabled Robot with Real-Time Object Detection and Collision Avoidance Using Arduino	2023	Uses Arduino for voice-controlled navigation, object detection, and obstacle avoidance.
3	Naeem, Bisma & Yousuf, Nadeem	An AI based voice controlled humanoid robot	2023	Implements voice-controlled AI system in a humanoid robot.
4	Mohd, Tauheed Khan & Javaid, Ahmad Y.	Integrated Control of Robotic Arm through EMG and Speech	2024	Multimodal control using speech and EMG signals via data fusion.
5	Rouillard, José & Vannobel, Jean-Marc	Multimodal interaction for cobot using MQTT	2023	Enables communication for multimodal robotics using MQTT protocol.
6	Rajesh, Y. et al.	AI Enhanced Arduino Based Customized Smart Glasses for Blind People Integrated with Speech Synthesis	2024	Arduino-based system integrating AI and speech synthesis for assistive use.
7	Ahmmed, Faysal et al.	Arduino-Controlled Multi-Function Robot with Bluetooth and nRF24L01+ Communication	2024	Designs a robot controlled by Arduino with Bluetooth and wireless modules.

8	Canh, Thanh Nguyen et al.	Development of a Human-Robot Interaction Platform for Dual-Arm Robots Based on ROS and Multimodal AI	2024	ROS-based system supporting multimodal AI interaction in dual-arm robots.
9	Naeem, Bisma et al.	Voice controlled humanoid robot	2024	Focuses on voice interface for humanoid robot applications.
10	Pacelli, Corrado et al.	Multi-modal communication interactions between socially assistive robot and people with neurodevelopmental disorders	2020	Explores use of multimodal interfaces for social and assistive robotics.
11	Ikram, Sunnia et al.	A Transformer-Based Multimodal Object Detection System for Real-World Applications	2025	Applies transformer architecture to multimodal object detection in real-world environments.

Table: 2.1.1 Reference table

2.1 Existing Systems:

Over the years, robotics has evolved to assist humans in industrial automation, education, healthcare, and smart environments. However, most existing robotic systems operate on single-mode inputs, such as physical buttons, remote controls, or mobile apps. These systems lack adaptability in dynamic or constrained settings where users may not have access to physical controllers.

For example, traditional assistive robots used in healthcare or domestic environments rely heavily on tactile or remote control inputs. Although some modern robots incorporate basic voice recognition, they often struggle with accent variation, background noise, or complex speech patterns. Similarly, gesture-controlled systems lack precision and require users to perform exaggerated or predefined movements under ideal lighting and background conditions.

Moreover, many robots operate in a closed-loop fashion with limited feedback or learning capability. They do not adapt to user behavior or switch seamlessly between communication modes (e.g., voice to gesture). Integration between the robot’s hardware and software often lacks real-time responsiveness, leading to delays or misinterpretation of commands. These shortcomings limit usability, especially for individuals with disabilities or in noisy environments where verbal interaction is difficult.

2.2 Proposed System:

To overcome the limitations of existing systems, we propose the AI-Enhanced Multi-Modal Robot, which supports speech, text, and gesture-based inputs for more natural and intelligent human-robot interaction. The system integrates Natural Language Processing (NLP) for speech-to-text and text-to-speech communication, OpenCV with Media Pipe for gesture and visual recognition, and Standard Firmata via Python and Arduino for real-time hardware control.

This proposed system focuses on multi-modal communication, allowing users to interact in whichever way suits them best, such as speaking, typing commands, or using hand gestures. It significantly enhances user accessibility, especially in environments where voice is not feasible or for users with speech impairments. It also enables real-time adaptability, by processing different input streams simultaneously and giving immediate, relevant feedback.

Additionally, this robot is modular and scalable, allowing future upgrades like facial expression recognition, autonomous navigation, and IoT integration. The design emphasizes seamless hardware-software coordination, optimizing both performance and power consumption. This makes the system reliable, accessible, and ready for applications in smart homes, education, healthcare, and beyond.

CHAPTER - 3

SYSTEM DESIGN

Design guarantees that every component of the robot operates smoothly and effectively, much like draughting a comprehensive layout before building a house. Without a carefully considered design, the robot may move erratically, react too slowly, or not comprehend directions. Given that this robot may interact in a variety of ways, including speech, text, and motions, the design must carefully arrange the processing of these inputs to maintain system stability and responsiveness. For instance, the robot should priorities the proper input without being confused if someone speaks while gesturing. A robust design guarantees that the robot will perform consistently in real-world scenarios, minimizes delays, and prevents errors.

Additionally, a well-thought-out design facilitates the smooth operation of several technologies, including natural language processing, gesture detection, and speech recognition. Imagine a situation in which a user stops the robot by simultaneously waving their hand and giving a vocal command. It is up to the design to decide which input should be handled first or if both should be done at once. The robot seems more intelligent and perceptive as a result of this coordination, which also avoids conflicting activities. Furthermore, a modular design makes updates simple; if a better speech recognition algorithm is created later, it may be incorporated without interfering with the system’s functionality.

Another important factor in design is accessibility. Not every user interacts with technology in the same manner; some may be in noisy situations where voice instructions are ineffective, while others may have speech problems. By providing alternate input modalities like touch screens, gesture controls, or even eye-tracking, a well-designed robot accounts for these variations.

Lastly, a well-designed interface makes interactions effortless and natural, which improves the user experience. Users won’t trust the robot if it reacts too slowly, misunderstands instructions, or acts erratically. The design guarantees that the robot feels accurate, responsive, and user-friendly by meticulously organizing the processing of inputs, the communication between components, and the handling of failures. A well-designed robot can help people efficiently, adjust to various demands, and function dependably over time in a variety of situations, including homes, workplaces, and medical facilities.

UML DIAGRAMS

3.1.1 USECASE DIAGRAM

The AI-enhanced Multi-modal Robot's use case diagram shows how users engage with the system using different input methods and get results in return. The user, who could be a regular person, a student, an old person, or someone with speech problems, is the main actor in this system. Voice, text, and hand motions are the three primary ways that the user can interact with the robot. With the use of particular AI and embedded technologies, the system is intended to identify and react to each mode, which represents a unique interaction path.

The robot uses speech recognition algorithms to record and process spoken input when the user offers a voice command. After it has been comprehended, it takes the necessary action and can respond orally via text-to-speech. In a similar manner, the robot can be instructed by text input via a direct serial connection, computer interface, or mobile device. When speaking is impractical or impossible, the system interprets this content and responds appropriately, making it accessible. The robot uses a camera and computer vision (OpenCV) to understand hand gestures in the third interaction mode, known as gesture control. For people with speech impairments or in settings where silence is necessary, this capability is especially helpful.

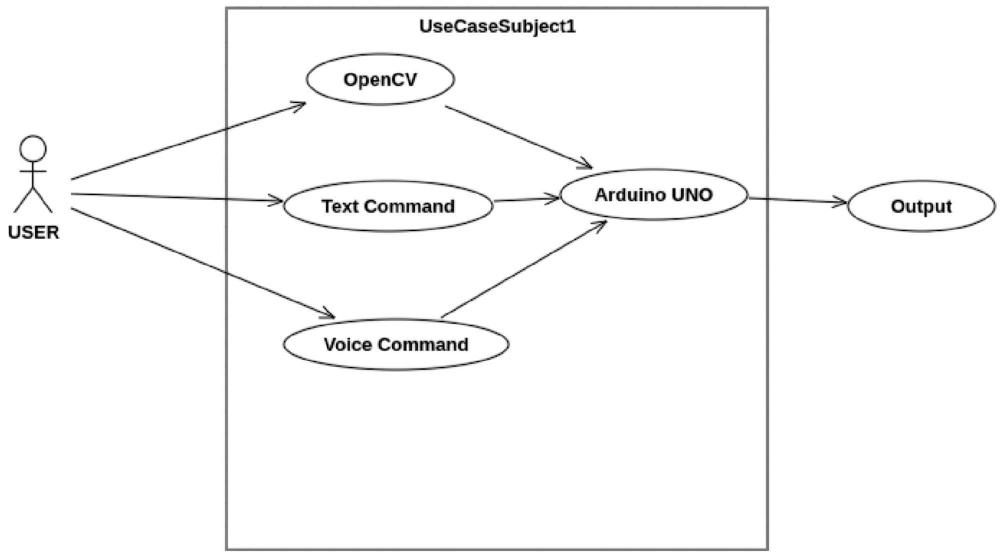


Fig3.2.1: Use case Diagram

3.1.2 CLASS DIAGRAM

The AI-enhanced Multi-modal Robot's class diagram serves as a model for the system's design by capturing the essential elements and how they interact. The Robot Controller class is essential to the architecture because it coordinates the actions of software modules like AI models and communication interfaces with hardware elements like sensors, motor drivers, and the camera. Classes like Voice Command, Text Command, and Gesture Command, which are each in charge of managing particular kinds of user inputs, reflect the many input modalities that the system supports. These input classes communicate with specific processing modules: Text commands are parsed and handled properly, Gesture Recognizer uses computer vision to understand hand gestures, and Speech Recognizer uses natural language processing to analyze voice commands.

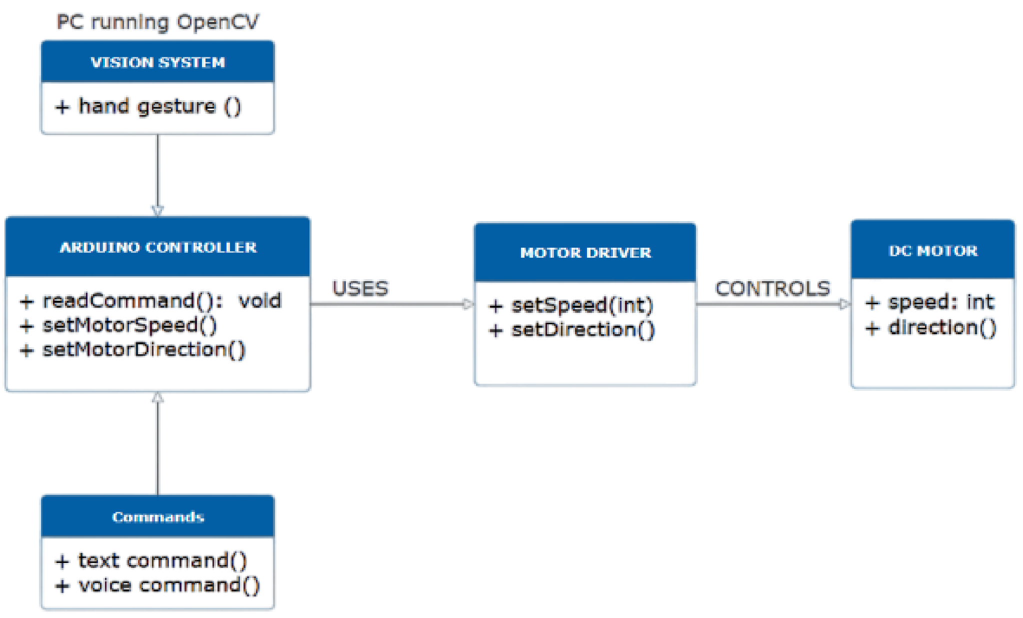


Fig3.2.2: class diagram

3.1.3 SEQUENCE DIAGRAM

The AI-enhanced Multi-modal Robot's sequence diagram shows how the user and system components interact dynamically during multi-modal communication. It starts when the user initiates an input, which could be a hand gesture, written command, or voice command. The associated processing module—Speech Recognizer for voice, Text Processor for text, or Gesture Recognizer for hand movements—receives this input first. It is then in charge of employing AI and embedded system techniques to analyze the particular type of data. Following recognition and comprehension of the input, these modules transmit the interpreted command to the Robot Controller, which serves as the main coordinator for the hardware and software components of the robot.

After that, the Robot Controller executes the command by using the Motor Driver and Sensor classes to activate the appropriate hardware, such as motors or sensors, or by using the Camera module to do visual recognition tasks like object or face detection. To ensure natural and real-time engagement, the controller uses additional communication channels to show output or calls the Text To Speech module to produce a voice response if the interaction calls for feedback. In order to ensure dependability and safety during this sequence, the system continuously analyses power usage and error states, implementing fallback methods in case of unforeseen circumstances.

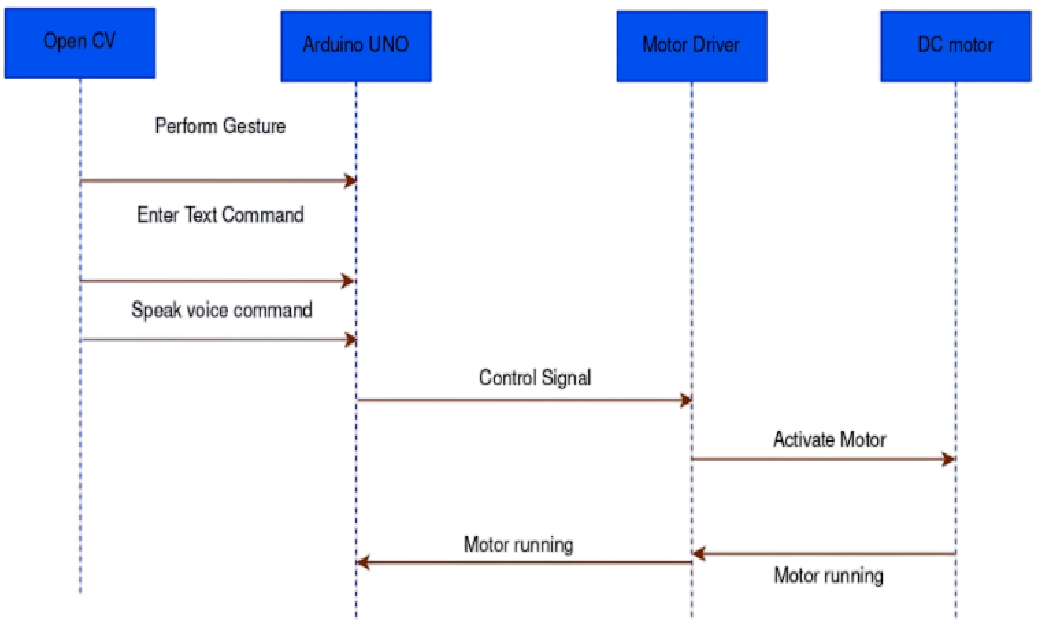


Fig3.2.3: Sequence Diagram

3.1.4 ACTIVITY DIAGRAM

The AI-enhanced Multi-modal Robot’s activity diagram shows the sequential method for interpreting user input and managing the robot’s reactions. The robot waits for an input at the start of the procedure, which could be a hand gesture, text input, or voice command. After detecting an input, the system divides into different activity paths based on the type of input: hand gestures are recorded by the camera and interpreted using computer vision techniques, text inputs are parsed and examined, and voice commands are processed using speech recognition and natural language understanding.

Following processing, the interpreted command is transmitted to the central controller of the robot, which assesses it and decides on the necessary hardware actions.

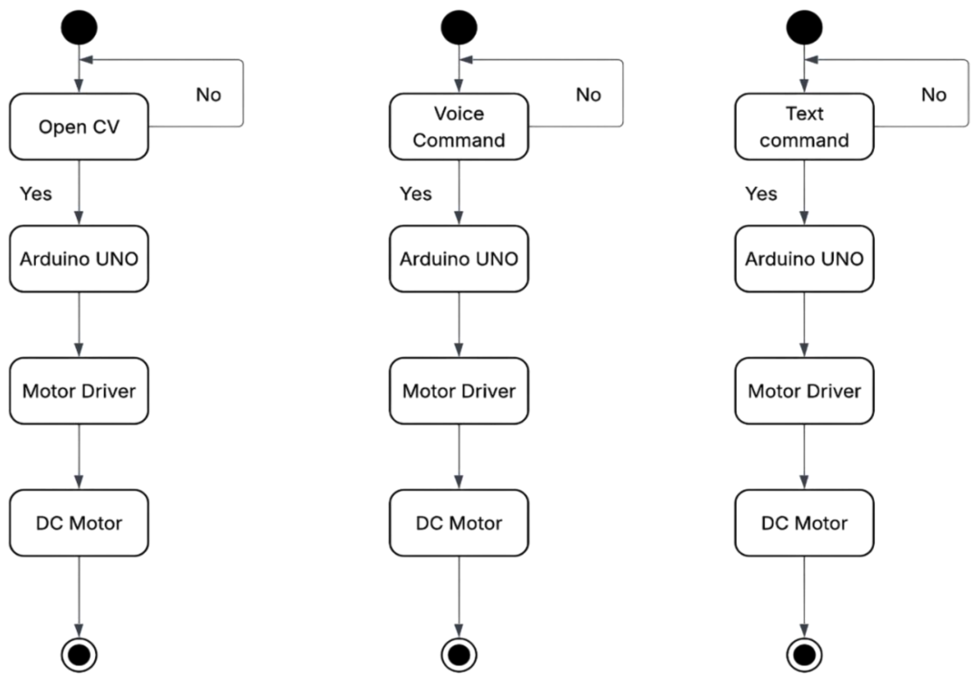


Fig3.2.4: Activity diagram

3.1.5 SYSTEM ARCHITECTURE

The system architecture of the AI-Enhanced Multi-Modal Robotic System defines the structured interaction between various hardware and software components to enable intelligent, real-time human-robot communication. This architecture is designed to support multi-modal inputs—such as voice, text, and hand gestures—processed through natural language processing, computer vision, and embedded control systems. By organizing the system into clearly defined layers, the architecture ensures modularity, scalability, and efficient task execution. Each layer plays a distinct role, ranging from user interaction to hardware-level actuation, enabling the robot to function as an adaptable, responsive, and human-friendly assistant in real-world applications like healthcare, smart homes, and education.

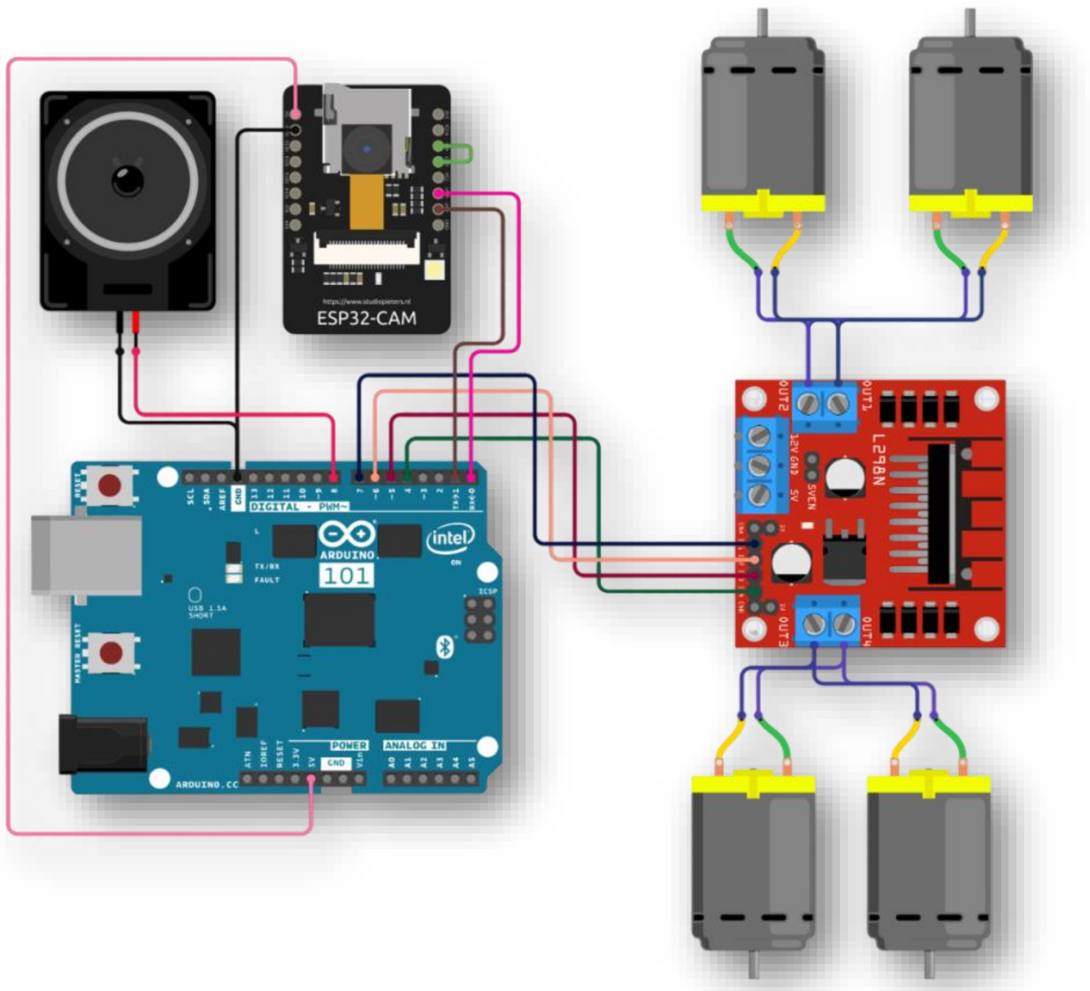


Fig3.2.5: System architecture of Ai Enhanced Multi Model Robot

ARDUINO UNO:

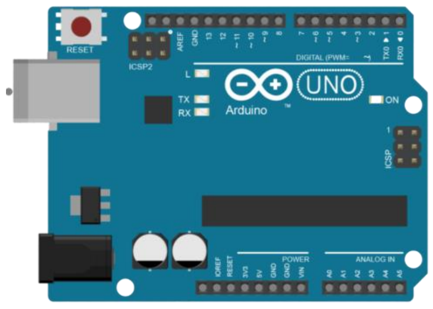


Fig3.2.6: Arduino UNO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.

Arduino boards are powered via USB or external power sources and are known for their flexibility and simplicity, making them ideal for beginners, students, and professionals. Due to its compatibility with a wide range of sensors and modules, Arduino is commonly used in robotics, automation, IoT, and embedded systems development.

L298N MOTOR DRIVER

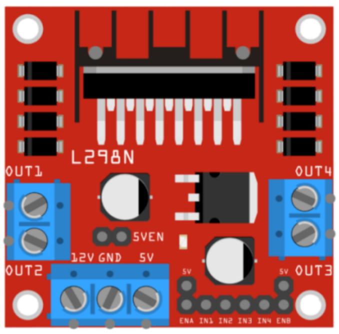


Fig 3.2.7: L298N Motor Driver

The L298N motor driver module is a key component in robotics and embedded systems, enabling control of the direction and speed of motors. It is based on the L298N Motor Driver**,** a dual full-bridge driver that can control two independent DC motors or a single stepper motor. The module is widely used due to its low cost, reliability, and ease of integration with microcontrollers like Arduino, Raspberry Pi, and NodeMCU.

Controlling motors directly using microcontrollers is not feasible because motors often require higher voltage and current than microcontrollers can provide. The L298N acts as an interface between the microcontroller and motors, allowing precise control over motor functions using simple digital signals.

Technical Specifications:

Operating Voltage: 5V to 35V (motor voltage)

Logic Voltage: 5V (for input signals)

Current: Up to 2A per channel

Number of Channels: 2 (can drive 2 DC motors or 1 stepper motor)

Enable Pins: Allow speed control via PWM (Pulse Width Modulation)

Pin Description:

Pin Name	Description
IN1,IN2	Control motor A direction
IN3,IN4	Control motor B direction
ENA, ENB	Enable motor A and B respectively (used for speed control)

OUT1, OUT2	Motor A connection
OUT3,OUT4	Motor B connection
12v	Motor power supply
GND	Ground Pin

Fig.3.2.1.1: Motor Driver Pin Description

WORKING PRINCIPLE

The L298N motor driver works based on the **H-bridge circuit**, which allows current to flow in either direction through the motor, enabling forward and reverse motion. Each motor has two input pins (IN1 & IN2 for Motor A, IN3 & IN4 for Motor B). By changing the logic levels (HIGH or LOW) on these pins, the motor can be made to rotate forward, reverse, or stop.

IN1	**IN2**	**Motor A Action**
1	0	Forward
0	1	Reverse
0	0	Stop

Fig.3.2.1.2: Working Principle of Motor driver

ESP32 CAM:

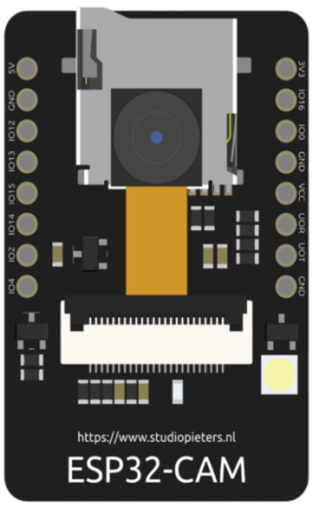


Fig3.2.8: ESP32 CAM

The ESP32-CAM is a compact, low-cost development board based on the ESP32-S microcontroller with an integrated camera and Wi-Fi/Bluetooth capabilities. It is designed primarily for image and video capture applications and is popular in IoT, surveillance, home automation, AI-based vision, and robotics projects**.** The board combines the powerful dual-core 32-bit processor of the ESP32 with an OV2640 camera module, making it capable of handling computer vision tasks, streaming, and wireless communication without the need for external processors.

Despite its small form factor, the ESP32-CAM offers a range of features including micro SD card support, GPIOs for connecting sensors/actuators, and deep-sleep modes for power efficiency. Its ability to capture and stream images wirelessly makes it one of the most powerful tools for smart vision systems at an affordable price.

The ESP32-CAM is a feature-rich, affordable module that combines processing power, camera functionality, and wireless communication in a compact package. It is a go-to choice for vision-based IoT applications such as surveillance, face recognition, smart automation, and portable imaging. While it has some limitations like limited GPIO access and the need for an FTDI adapter, its performance and flexibility far outweigh these concerns, making it a favorite among hobbyists, students, and professionals working in computer vision and AI-based projects.

PIN DISCRIPTION

Pin	**Function**	**Description**
3V	Power Input	3.3V supply (usually powered via 5V pin)
5V	Power Input	Recommended input (5V regulated)
GND	Ground	Common ground
U0R	UART RX	Serial communication (to FTDI TX)
U0T	UART TX	Serial communication (to FTDI RX)
GPIOs	Digital I/O	General purpose I/O, shared with camera

Fig.3.2.1.3: ESP32 CAM Pin Description

FUNCTIONAL REQUIREMENTS

HARDWARE:

1. **Microcontroller (Arduino Uno)**

- **Function**: Acts as the central control unit for interpreting signals from sensors and executing commands by actuating motors or responding via communication modules.
- **Requirement**: Must support serial communication with the computer (Standard Firmata), be able to read analog/digital inputs, and control outputs (motors, actuators).

2. **Motor Driver (L298N or L293D)**

- **Function**: Controls the robot's wheel movement based on interpreted input commands.
- **Requirement**: Must be able to drive DC motors with sufficient current and voltage.

3. **Camera Module (USB/Webcam)**

- **Function**: Captures visual input for gesture recognition, face/object detection.
- **Requirement**: Should offer real-time video capture support with minimum 30 FPS for accurate gesture tracking using OpenCV.

SOFTWARE:

1. **Python (Main Program Logic)**

- **Function**: Acts as the brain of the robot for processing voice, text, and visual commands.
- **Requirement**: Must support serial communication (via `pyFirmata`), OpenCV for image processing, and speech/text libraries for voice interactions.

2. **Standard Firmata Protocol (Arduino)**

- **Function**: Allows Arduino to be controlled directly from the Python program.
- **Requirement**: Arduino must be flashed with Standard Firmata firmware to accept Python-based instructions.

3. **OpenCV Library**

- **Function**: Performs computer vision tasks like gesture, face, and object recognition.
- **Requirement**: Must detect and classify hand gestures in real time from webcam feed.

4. **Speech Recognition Library (e.g.,** speech_recognition**)**

- **Function**: Converts spoken input to text commands for the robot.
- **Requirement**: Must work with microphone input and support offline/online recognition engines.

5. **Text-to-Speech Engine (e.g.,** pyttsx3 **or** gTTS**)**

- **Function**: Converts robot's response from text to spoken output.
- **Requirement**: Should provide audible, clear responses to user queries.

6. **Serial Communication (**pyFirmata**)**

- **Function**: Facilitates Python-to-Arduino communication to control motors and read sensors.
- **Requirement**: Must establish real-time, bi-directional communication between Python and Arduino over USB.

CHAPTER – 4

MODULE DESCRIPTION

4.1.1 Open CV-Based Gesture Control Module:

The OpenCV-based gesture control module is one of the most interactive components in the AI Enhanced Multi-Model Robot. This module uses a webcam and computer vision technologies to interpret hand gestures and control the robot accordingly. Built using Python, it integrates OpenCV and MediaPipe libraries to capture and process real-time video feed. MediaPipe, developed by Google, provides an advanced hand-tracking solution that can detect 21 landmarks on each hand, including fingertips, knuckles, and wrist. With the help of these landmarks, the system identifies the position and orientation of each finger.

The logic of this module works by comparing the y-coordinates of finger tips with their corresponding lower joints. For example, if the tip of the index finger is higher than the PIP joint (proximal interphalangeal), the system identifies it as being raised. Based on this detection, specific gestures are mapped to robot movements. A simple example is: when only the index finger is up, the robot moves forward; when the middle finger is raised, it moves backward; and when the ring finger is up, it stops. These control signals are sent to the microcontroller ArduinoUno through serial communication.

This system removes the need for physical switches or remotes and makes the robot contactless, user-friendly, and accessible. It is especially useful in educational, research, and assistive technology projects. Gesture control through OpenCV not only makes robot navigation intuitive but also introduces students to real-world computer vision applications in robotics.

4.1.2 Speech-to-Text (Voice Command Module)

The speech-to-text module in this project enables voice-based control, allowing users to operate the robot using simple spoken commands. This module is developed using Python and the Speech Recognition library, which acts as an interface to multiple speech-to-text engines. Most commonly, the Google Speech Recognition API is used for its high accuracy and reliability. When the user speaks into the microphone, the module captures the audio and converts it into text using cloud-based or offline speech engines.

Once the speech is converted into text, the system checks whether the spoken words match any pre-defined commands such as “move forward”, “stop”, “turn left”, or “go back”. If a match is found, a corresponding signal is sent to the robot controller via serial or wireless communication. This signal is then interpreted by the microcontroller to trigger a physical action, such as moving motors or stopping them.

This voice command feature adds a hands-free dimension to the robot’s operation. It is especially beneficial for users who are visually impaired or unable to use hand gestures. The system can also be customized to recognize commands in different languages or dialects, making it versatile and inclusive. Furthermore, with the use of natural language processing techniques, the module can be expanded to handle more complex and conversational commands in future upgrades.

4.1.3. Text-to-Speech (Voice Feedback Module):

The text-to-speech (TTS) module is responsible for making the robot talk. It provides verbal feedback based on the actions being performed or responses to the user’s commands. Implemented in Python using the `pyttsx3` library, this module operates offline and does not require an internet connection, which makes it reliable and responsive. The TTS engine converts plain text messages into audible speech, which is played through a speaker connected to the robot system.

This feature enhances human-robot interaction by informing the user about the robot’s status. For example, when a voice or gesture command is received, the robot might respond by saying “Moving forward”, “Turning left”, or “Command not recognized”. These voice responses give the user clear confirmation that the robot has understood the command and is acting accordingly. It also helps in debugging or troubleshooting since users can audibly know what the system is processing at any given moment.

The pyttsx3 library supports different voices, speech rates, and volumes, making the voice customizable according to project requirements. The module can also be extended for multilingual speech output or used in educational settings where robots act as voice-based tutors. Overall, the text-to-speech module not only adds personality to the robot but also improves the accessibility and user experience by providing real-time audible feedback.

4.2 SOURCE CODE:

```
import cv2
import mediapipe as mp
import pyfirmata2
import pyttsx3
import queue
import threading
# ========== Speech Engine ==========
engine = pyttsx3.init()
engine.setProperty(“rate”, 150)
engine.setProperty(“volume”, 1)
speech_queue = queue.Queue()
def process_speech():
while True:
text = speech_queue.get()
if text == “EXIT”:
break
engine.say(text)
engine.runAndWait()
speech_thread = threading.Thread(target=process_speech, daemon=True)
speech_thread.start()
# ========== Arduino Setup ==========
port = ‘COM4’
board = pyfirmata2.Arduino(port)
print(“SUCCESS: Arduino connected”)
# Motor A (Left)
in1_pin = 8
in2_pin = 9
ena_pin = 10
# Motor B (Right)
```

```
in3_pin = 11
in4_pin = 12
enb_pin = 5
# Set pin modes
for pin in [in1_pin, in2_pin, in3_pin, in4_pin]:
board.digital[pin].mode = pyfirmata2.OUTPUT
board.digital[ena_pin].mode = pyfirmata2.PWM
board.digital[enb_pin].mode = pyfirmata2.PWM
# Default speed
default_speed = 0.1
board.digital[ena_pin].write(default_speed)
board.digital[enb_pin].write(default_speed)
print(f"Motor speed set to {int(default_speed * 100)}%")
motor_state = "stopped"
def motor_forward():
global motor_state
if motor_state != "forward":
    board.digital[in1_pin].write(1)
    board.digital[in2_pin].write(0)
    board.digital[ena_pin].write(default_speed)
    board.digital[in3_pin].write(1)
    board.digital[in4_pin].write(0)
    board.digital[enb_pin].write(default_speed)

    motor_state = "forward"
    print("Motors running forward")
    speech_queue.put("Motors running forward")
    def motor_backward():
    global motor_state
    if motor_state != "backward":
    board.digital[in1_pin].write(0)
    board.digital[in2_pin].write(1)
    board.digital[ena_pin].write(default_speed
    board.digital[in3_pin].write(0)
    board.digital[in4_pin].write(1)
```

```
        board.digital[enb_pin].write(default_speed)
        motor_state = “backward”
        print(“Motors running backward”)
        speech_queue.put(“Motors running backward”)
        def motor_stop():
        global motor_state
        if motor_state != “stopped”:
         board.digital[in1_pin].write(0)
         board.digital[in2_pin].write(0)
         board.digital[ena_pin].write(0)
         board.digital[in3_pin].write(0)
         board.digital[in4_pin].write(0)
         board.digital[enb_pin].write(0)
         motor_state = “stopped”
         print(“Motors stopped”)
         speech_queue.put(“Motors stopped”)
# ========== Text Command Mode ==========
def text_command_mode():
print(“Text Command Mode Activated!”)
while True:
cmd = input(“Enter ‘f’=forward, ‘b’=backward, ‘s’=stop, ‘e’=exit: “).strip().lower()
if cmd == ‘f’:
motor_forward()
elif cmd == ‘b’:
motor_backward()
elif cmd == ‘s’:
motor_stop()
elif cmd == ‘e’:
motor_stop()
print(“Exiting text mode…”)
break
else:
            print(“Invalid command. Try again.”)
# ========== Gesture Control Mode ==========
def gesture_control_mode():
```

```
print(“Gesture Control Mode Activated!”)
mp_hands = mp.solutions.hands
hands=mp_hands.Hands(min_detection_confidence=0.7,
min_tracking_confidence=0.7)
mp_drawing = mp.solutions.drawing_utils
cap = cv2.VideoCapture(0)
if not cap.isOpened():
print(“Camera not detected”)
return
print(“Index = Forward | Middle = Backward | Ring = Stop | ‘q’ to Quit”)
try:
while True:
ret, frame = cap.read()
if not ret:
break
frame = cv2.flip(frame, 1)
h, w, _ = frame.shape
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results = hands.process(rgb)
fingers_up = [False] * 5
if results.multi_hand_landmarks:
for hand in results.multi_hand_landmarks:
mp_drawing.draw_landmarks(frame, hand, mp_hands.HAND_CONNECTIONS)
tips = [4, 8, 12, 16, 20]
lowers = [3, 7, 11, 15, 19]
for i in range(5):
tip_y = hand.landmark[tips[i]].y * h
lower_y = hand.landmark[lowers[i]].y * h
fingers_up[i] = tip_y < lower_y
if fingers_up[1]:  # Index
motor_forward()
elif fingers_up[2]:  # Middle
motor_backward()
elif fingers_up[3]:  # Ring
motor_stop()
```

```
cv2.putText(frame, “Index=Forward | Middle=Backward | Ring=Stop | ‘q’=Quit”,
(10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2)
cv2.imshow(“Gesture Motor Control”, frame)
if cv2.waitKey(1) & 0xFF == ord(‘q’):
motor_stop()
break
except Exception as e:
        print(f”Error: {e}”)
finally:
cap.release()
cv2.destroyAllWindows()
hands.close()
# ========== Mode Selection ==========
def main():
try:
        print(“Select Control Mode:”)
        print(“1. Text Command Mode”)
        print(“2. Gesture Control Mode”)
choice = input(“Enter choice (1 or 2): “).strip()
if choice == ‘1’:
text_command_mode()
elif choice == ‘2’:
gesture_control_mode()
else:
        print(“Invalid choice. Exiting…”)
finally:
motor_stop()
speech_queue.put(“EXIT”)
speech_thread.join(timeout=3)
board.exit()
        print(“Clean exit completed.”)
if __name__ == “__main__”:
        main()
```

CHAPTER – 5

RESULTS

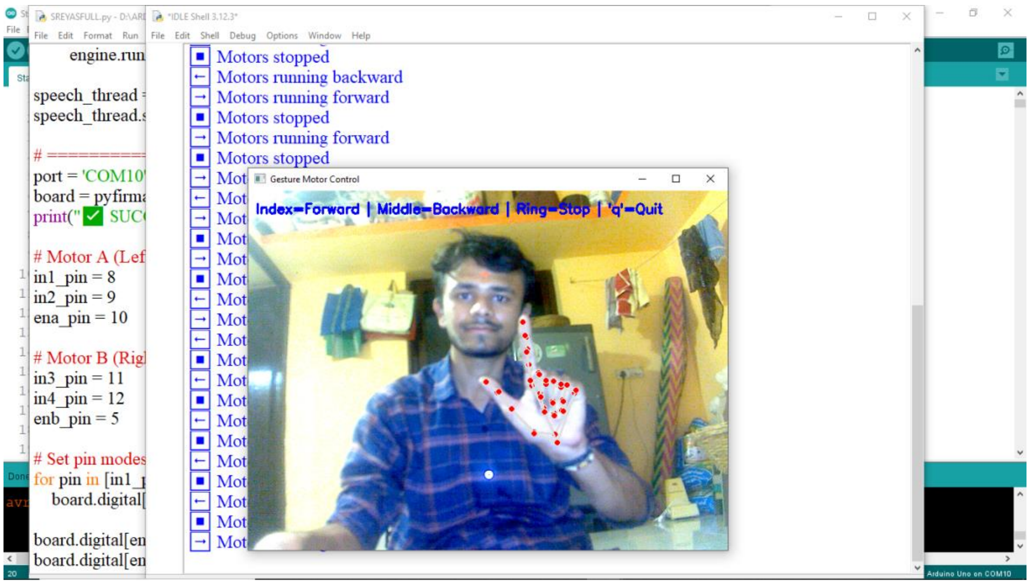
An Open-source software library for computer vision and machine learning is called OpenCV, short for open source computer vision library. It was first created by Intel and is currently maintained by the OpenCV foundation, a development community. This article delves into Open CV, examining its features, uses, and real world examples. OpenCV is a vast open source library for image processing, machine learning, and computer-vision is called OpenCV. These days, it has a significant impact on real-time operation, which is critical for modern systems. It may be used to process photos and videos in order to recognize faces, objects, or even human hand writing.

An Open-source software library for computer vision and machine learning is called OpenCV, short for open source computer vision library. It was first created by Intel and is currently maintained by the OpenCV foundation, a development community. This article delves into Open CV, examining its features, uses, and real world examples. OpenCV is a vast open source library for image processing, machine learning, and computer-vision is called OpenCV. These days, it has a significant impact on real-time operation, which is critical for modern systems. It may be used to process photos and videos in order to recognize faces, objects, or even human hand writing.

Image processing: Real-time hand gesture recognition is possible using the Media Pipe gesture recogniser job, which also displays the landmarks of the hands that were detected. The task allows you to identify particular hand motions made by user and activate the functionalities of the program that go along with those gestures. This assignment uses a machine learning model to process image data can take either a continuous stream or static data. Hand landmarks in picture co ordinates, hand landmarks in world co-ordinates, handedness (left/right hand), and the categories of hand gestures for multiple hands are all output by the job.

Image processing: Real-time hand gesture recognition is possible using the Media Pipe gesture recogniser job, which also displays the landmarks of the hands that were detected. The task allows you to identify particular hand motions made by user and activate the functionalities of the program that go along with those gestures. This

assignment uses a machine learning model to process image data can take either a continuous stream or static data. Hand landmarks in picture co ordinates, hand landmarks in world co-ordinates, handedness (left/right hand), and the categories of hand gestures for multiple hands are all output by the job.



```
    engine.run

speech_thread =
speech_thread.s

# ==========
port = 'COM10'
board = pyfirma
print("✅ SUC

# Motor A (Lef
in1_pin = 8
in2_pin = 9
ena_pin = 10

# Motor B (Rig
in3_pin = 11
in4_pin = 12
enb_pin = 5

# Set pin modes
for pin in [in1_
    board.digital[

board.digital[en
board.digital[en
```

```
■ Motors stopped
← Motors running backward
→ Motors running forward
■ Motors stopped
→ Motors running forward
■ Motors stopped
→ Mot
← Mot
→ Mot
■ Mot
→ Mot
■ Mot
← Mot
→ Mot
← Mot
■ Mot
← Mot
■ Mot
← Mot
■ Mot
← Mot
■ Mot
← Mot
■ Mot
→ Mot
```

Fig 5.1.1 OpenCV with hand gesture

The text to speech uses deep neural to generate computer voices almost identical to human recordings. Neural text to speech considerably lessens listening fatigue when consumers engage with AI systems because of the clear word articulation. Prosody refers to the intonation and stress patterns found in spoken language. Conventional text-to-speech systems divide prosody into distinct acoustic prediction and linguistic analysis processes that are controlled by different models. Voice synthesis may become buzzy and muted as a result. Further details regarding the speech services neural text-to-speech feature and how they get around the drawbacks of conventional text-to-speech systems are provided:

a. Real-time speech synthesis: Convert text to speech using either bespoke or prebuilt neural voices by utilizing the speck SDK or REST API.

b. Asynchronous synthesis of lengthy audio: To asynchronously synthesise text to speech files longer than then minutes (such as lectures or audio books), utilise the batch synthesis API. Unlike synthesis conducted using the speech SDK or Speech to text REST API, responses are not returned in real-time. When the service makes synthesised audio available, it is expected that requests will be sent asynchronously, replies will be polled for, and the audio will be downloaded.

c. Speech to text recognition:

   Real-time and batch transcription of audio streams into text is made possible by Speech to text, sometimes referred to as speech recognition, which is provided by the speech service. It also allows for real-time pronunciation evaluation and provides speakers with fee back on the precision and fluidity of spoken audio when more reference text input is provided.

CHAPTER 6

CONCLUSION

Furthermore, this project has immense potential for application in fields like **education, healthcare, and elder care**. In education, the robot can act as a teaching assistant, interacting with students in multiple languages and through different input modes. In healthcare and elder care, the robot can remind patients about medications, detect falls or distress, and provide immediate alerts to caregivers.

Lastly, future developments may include building a mobile app interface for remote control, integration with virtual assistants like Alexa or Google Assistant, and energy-efficient operation using solar-powered modules or optimized power management.

In conclusion, the AI-Enhanced Multi-Modal Robot is a scalable and versatile platform with a strong foundation for future innovation. By incorporating advanced AI, better hardware, and enhanced connectivity, this system can evolve into a fully autonomous, intelligent assistant capable of transforming the way humans interact with machines in everyday life.

CHAPTER – 7

Future Scope

The development of the AI-Enhanced Multi-Modal Robot opens up a broad range of future possibilities in both technological advancement and real-world applications. With rapid progress in artificial intelligence, embedded systems, and human-computer interaction, the potential for enhancing this system is significant and far-reaching.

One of the most promising directions for future development is the integration of machine learning-based adaptive behavior**.** Currently, the robot executes predefined actions based on gesture, voice, or text input. By incorporating learning algorithms, the robot can personalize its responses based on user behavior over time. This adaptation will allow the robot to understand user preferences, optimize task execution, and make predictive decisions, which can significantly improve the overall user experience.

Another important future enhancement is the deployment of cloud-based processing and storage**.** While the current model operates locally, cloud integration will enable more powerful computation, better storage management, and real-time data synchronization. Cloud services can facilitate collaborative learning between multiple robots, enable remote diagnostics, and allow real-time updates to improve performance and security.

In terms of hardware, expanding the sensor suite to include environmental sensors (gas, temperature, and humidity), GPS modules, and cameras with depth sensing (such as Intel Real Sense or LiDAR) can allow the robot to operate autonomously in complex environments. This would transform the robot from a static assistant to a mobile autonomous unit capable of performing tasks such as navigation, monitoring, and remote surveillance in areas like smart homes, factories, and healthcare institutions.

The future scope also includes extending the communication capabilities of the robot through IoT (Internet of Things) integration. By enabling Wi-Fi, Bluetooth, and GSM-based communication, the robot can interact with other smart devices, share sensor data, and provide alerts or updates to users' smart phones or cloud dashboards. This allows for seamless interoperability in a connected environment.

On the interaction side**,** natural language understanding (NLU) and multi-language support will make the robot more versatile and accessible. Instead of responding to only basic voice commands, it can understand conversational context, emotional tone, and respond more intelligently. Additionally, expanding gesture recognition to full-body gestures or sign language can improve accessibility for people with speech or hearing impairments.

Furthermore, this project has immense potential for application in fields like education, healthcare, and elder care. In education, the robot can act as a teaching assistant, interacting with students in multiple languages and through different input modes. In healthcare and elder care, the robot can remind patients about medications, detect falls or distress, and provide immediate alerts to caregivers.

Future developments may include building a mobile app interface for remote control, integration with virtual assistants like Alexa or Google Assistant, and energy-efficient operation using solar-powered modules or optimized power management.

The AI-Enhanced Multi-Modal Robot is a scalable and versatile platform with a strong foundation for future innovation. By incorporating advanced AI, better hardware, and enhanced connectivity, this system can evolve into a fully autonomous, intelligent assistant capable of transforming the way humans interact with machines in everyday life**.**

REFERENCES

4.3.1.1 Phan, Dang Khoa, Phuong-Nam Tran, Nhat Truong Pham, Tra Huong Thi Le, and Duc Ngoc Minh Dang. "Innovative Multi-Modal Control for Surveillance Spider Robot: An Integration of Voice and Hand Gesture Recognition." In *Proceedings of the 2024 9th International Conference on Intelligent Information Technology*, pp. 141-148. 2024.

4.3.1.2 Pawar, Suvarna, Pravin Futane, Nilesh Uke, Sourav Patil, Riya Shah, Harshi Shah, and Om Jain. "AI-Based Autonomous Voice-Enabled Robot with Real-Time Object Detection and Collision Avoidance Using Arduino." In *AI, IoT, Big Data and Cloud Computing for Industry 4.0*, pp. 199-218. Cham: Springer International Publishing, 2023.

4.3.1.3 Naeem, Bisma, and Nadeem Yousuf. "An AI based voice controlled humanoid robot." (2023).

4.3.1.4 Mohd, Tauheed Khan, and Ahmad Y. Javaid. "Integrated Control of Robotic Arm through EMG and Speech: Decision-Driven Multimodal Data Fusion." *arXiv preprint arXiv:2404.15283* (2024).

4.3.1.5 Rouillard, José, and Jean-Marc Vannobel. "Multimodal interaction for cobot using MQTT." *Multimodal Technologies and Interaction* 7, no. 8 (2023): 78.

4.3.1.6 Rajesh, Y., K. Bhaskar, K. Audi Dinakar, T. Kiruba Devi, and M. Priyanga. "AI Enhanced Arduino Based Customized Smart Glasses for Blind People Integrated with Speech Synthesis." In *International Conference on Cognitive Computing and Cyber Physical Systems*, pp. 317-331. Cham: Springer Nature Switzerland, 2024.

4.3.1.7 Ahmmed, Faysal, Asef Rahman, Amirul Islam, Ajmy Alaly, Samanta Mehnaj, Prottoy Saha, and Tamim Hossain. "Arduino-Controlled Multi-Function Robot with Bluetooth and nRF24L01+ Communication." *International Journal of Robotics & Control Systems* 4, no. 3 (2024).

4.3.1.8 Canh, Thanh Nguyen, Ba Phuong Nguyen, Hong Quan Tran, and Xiem HoangVan. "Development of a Human-Robot Interaction Platform for Dual-Arm Robots Based on ROS and Multimodal Artificial Intelligence." *arXiv preprint arXiv:2411.05342* (2024).

4.3.1.9 Naeem, Bisma, Wasey Kareem, Naureen Naeem, and Roha Naeem. "Voice controlled humanoid robot." *International Journal of Intelligent Robotics and Applications* 8, no. 1 (2024): 61-75.

4.3.1.10 PACELLI, CORRADO, PALLIMULLA HEWA GEEGANAGE, and THARUSHI KINKINI DE SILVA. "Multi-modal communication interactions between socially assistive robot and people with neurodevelopmental disorders." (2020).

4.3.1.11 Ikram, Sunnia, Imran Sarwar, Amna Ikram, and M. Abdullah-AI-Wahud. "A Transformer-Based Multimodal Object Detection System for Real-World Applications." *IEEE Access* (2025).