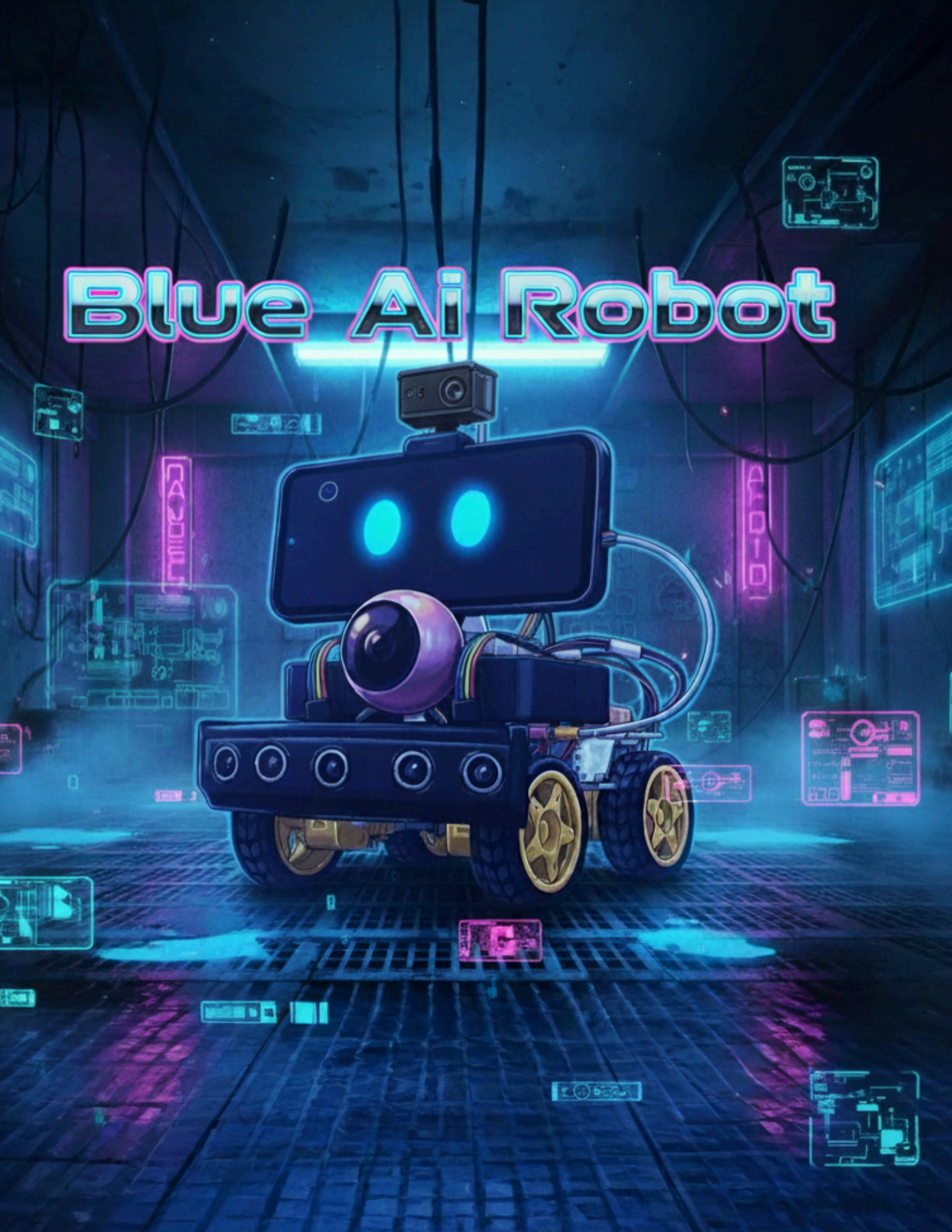

Blue Ai Robot

DEPARTMENT OF SOFTWARE ENGINEERING
Faculty of Computing & Information Technology (FCIT)
University of the Punjab, Lahore – PAKISTAN

Robot Name

BLUE

Introduction

Artificial Intelligence (AI) and robotics are transforming the way humans interact with technology. AI enables machines to think, learn, and make decisions, while robotics provides the physical capability to act in the real world.

AI-powered robots are increasingly important in industries, healthcare, education, and daily life, as they can perform tasks efficiently, assist humans, and solve complex problems.

Aim of the Project

To design and develop an AI-powered mobile robot capable of understanding voice commands, performing autonomous movements, and interacting intelligently with its environment using sensors and embedded systems.

Problem Statement

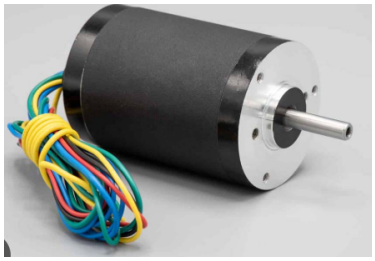
Many tasks in daily life require multitasking and assistance, such as interacting with humans, moving objects, and responding to commands. This project focuses on developing an AI robot that can talk, move, and make basic decisions to assist in such activities.

Objectives

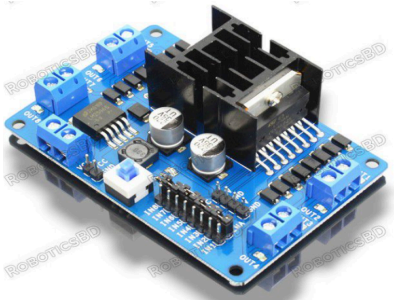
1. Design a robot that can move and navigate its environment.
2. Enable the robot to interact through voice communication.
3. Implement basic AI decision-making for task handling.
4. Integrate sensors and actuators for real-world interaction.

Components

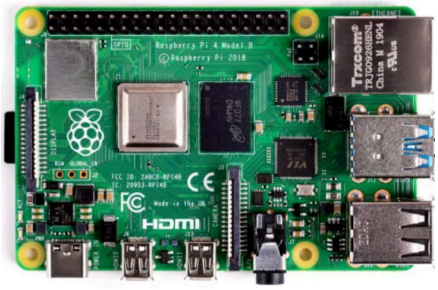
- Omnidirectional Microphone
- USB speakers
- Power bank (5V)
- DC motors + tires



- L298N Motor Drive



- Raspberry Pi 4



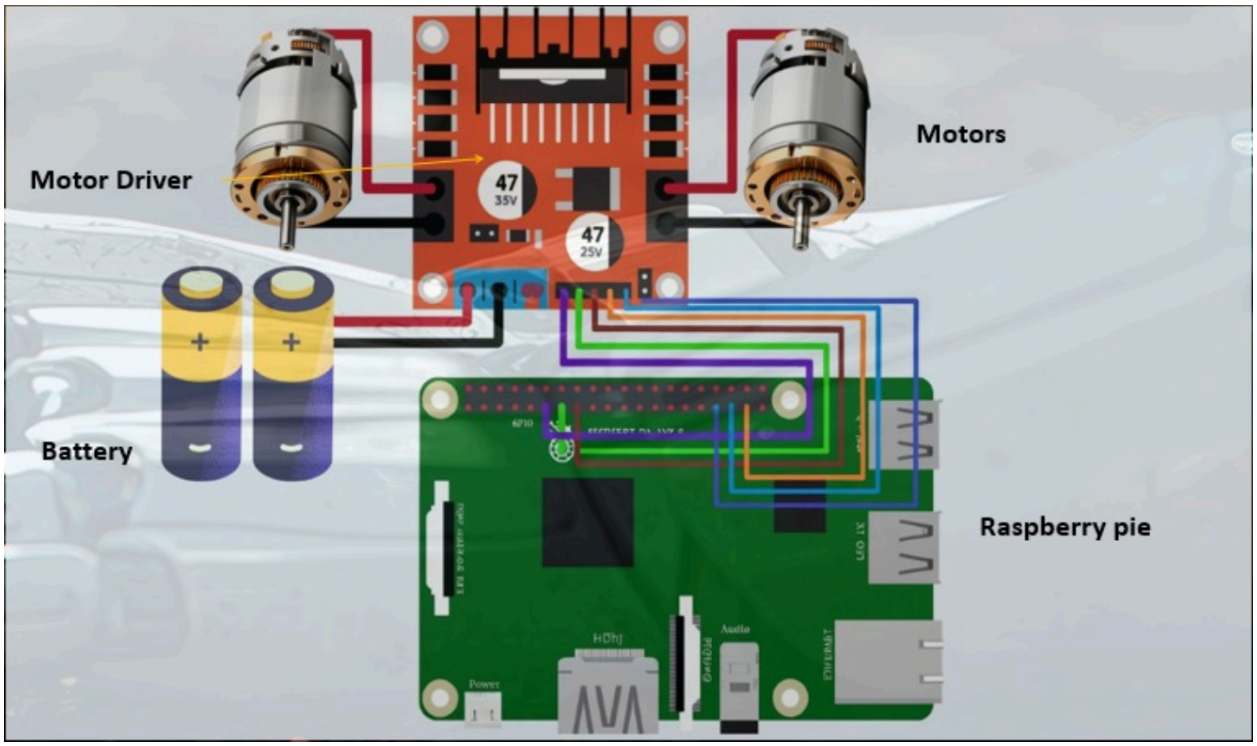
- 18650 batteries

- Ultrasonic sensor

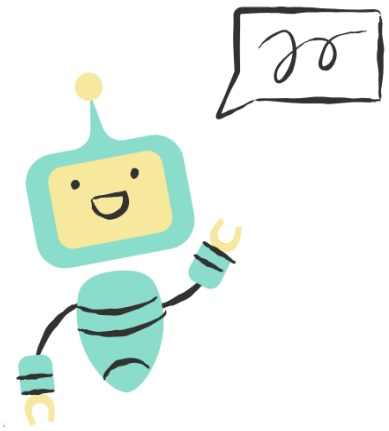
AI Robot Physics Principles

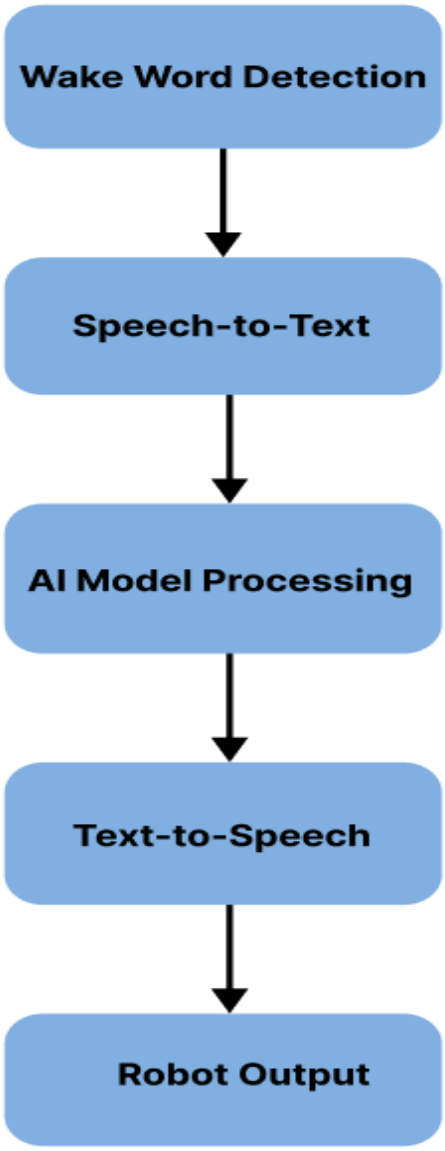
Physics Principle	Where It Is Used	Key Formula	Explanation of Symbols
Newton's Laws of Motion	Movement, acceleration, starting & stopping	$F = m \cdot a$	F = Force (N), m = Mass (kg), a = Acceleration (m/s²)
Torque & Rotational Motion	Motor rotation, wheel movement	$\tau = F \cdot r$	τ = Torque (N·m), F = Force applied (N), r = Distance from rotation axis (m)
Friction	Wheel grip, avoiding slip	$F\_\text{friction} = \mu \cdot N$	F_friction = Frictional force (N), μ = Coefficient of friction (unitless), N = Normal force (N)
Ohm's Law & Electricity	Powering motors, circuitry, sensors, Pi	$V = I \cdot R$	V = Voltage (V), I = Current (A), R = Resistance (Ω)
Electromagnetism (Motor Working)	DC motor rotation	$F = B \cdot I \cdot L$	F = Magnetic force (N), B = Magnetic field strength (T), I = Current (A), L = Length of conductor in field (m)
Ultrasonic Wave Reflection	Obstacle detection and distance measurement	$d = v \cdot t / 2$	d = Distance to obstacle (m), v = Speed of sound (m/s), t = Time taken for echo (s)
Energy Conversion	Battery → mechanical motion & electronics	$P = V \cdot I$	P = Power (W), V = Voltage (V), I = Current (A)

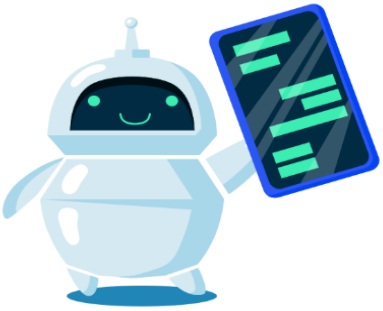
Overall Circuit Architecture



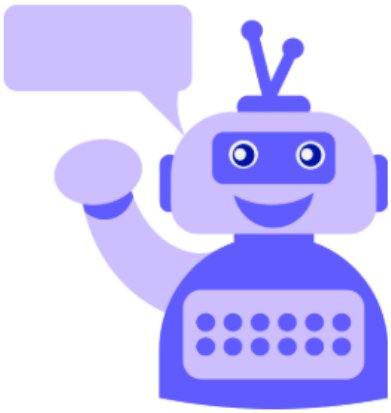
System WorkFlow







Hardware



1. **First, we gathered all the required components.**
   We selected the Raspberry Pi, DC motors, motor driver, microphone, mobile (cell) holder, 18650 cells, power bank, jumper wires, chassis, and sensors. Each component was chosen based on compatibility and the robot's functional requirements.

2. **Next, we mounted and arranged the hardware.**
   We attached the motors to the robot chassis, fixed the Raspberry Pi in place, and installed the mobile holder for stable camera/phone support. The microphone was positioned in a spot where it could clearly capture voice commands.

3. **Then, we completed all wiring connections.**
   Using jumper wires, we connected the motors to the motor driver, the motor driver to the Raspberry Pi, and the sensors to their respective GPIO pins. The 18650 cell setup and power bank were arranged to supply stable power to different modules.

4. **After that, we set up the power system.**
   The power bank was used to run the Raspberry Pi, while the 18650 cells supported additional components like motors. We ensured safe connections and tested the voltage supply to avoid overloads or drops.

5. **Next, we configured the software environment.**
   We installed necessary libraries for speech recognition, text-to-speech, AI communication, motor control, and sensor handling. This included enabling GPIO control and preparing the Pi for microphone input.

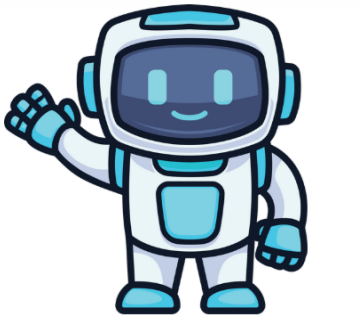
6. **Then, we integrated the voice system.**
   We connected the microphone for Speech-to-Text (STT) input and set up Text-to-Speech (TTS) for audio output. The system was tested to ensure the robot could clearly hear commands and respond back naturally.

7. **After that, we developed AI logic.**
   We programmed the robot to interpret voice instructions, process them through AI, and generate meaningful responses. This created the core “smart assistant” functionality of the robot.

8. **Next, we linked movement with AI commands.**
   Motor control functions were connected with the AI output so the robot could move forward, reverse, turn, or stop based on interpreted instructions. Obstacle sensors were also integrated for safety.

9. **Finally, we performed testing and optimization.**
   We tested the robot's movement, voice recognition, response accuracy, and power stability. Any wiring issues or code errors were fixed. After multiple iterations, the robot behaved smoothly and reliably.

Core Functional Capabilities of the AI-Powered Robotic System



1. Voice Command Recognition
2. Natural Speech Response System
3. Autonomous Mobility Control
4. Real-Time Obstacle Detection & Avoidance
5. Decision-Making Engine
6. Wireless Connectivity & Remote Access

Software

Language :python
Tool :Visual Studio Code

Code Part

```
import speech_recognition as sr
import os
import json
import time
import subprocess
import requests
import base64
from gpiozero import Robot, DistanceSensor

# ==========================================
# ⚙️  CONFIGURATION
# ==========================================
LAPTOP_IP = "192.168.1.20"  # <--- CHECK THIS IP
PORT = "11434"
WAKE_WORD = "hello robot"
SAFE_DISTANCE = 0.3  # 30 cm

# API URL
BASE_URL = f"http://{LAPTOP_IP}:{PORT}/api/chat"

# ==========================================
# 🛠️  SMART HARDWARE SETUP
# ==========================================
SIMULATION_MODE = False

try:
    # 1. Motors (Left: 17,27 | Right: 22,23)
    my_robot = Robot(left=(17, 27), right=(22, 23))

    # 2. Sensor (Echo: 24 | Trig: 25)
```

```
    sensor = DistanceSensor(echo=24, trigger=25, max_distance=2.0)
    print("✅ HARDWARE CONNECTED: Motors & Sensors Ready.")

except Exception as e:
    # If GPIO fails (no wires connected), switch to Simulation
    print(f"⚠ HARDWARE NOT FOUND: {e}")
    print("🔄 SWITCHING TO SIMULATION MODE (Printing text only)")
    SIMULATION_MODE = True

    class DummyRobot:
        def forward(self): print("   🚗 [SIM] Motors FORWARD")
        def backward(self): print("   🚗 [SIM] Motors BACKWARD")
        def left(self): print("   🔄 [SIM] Motors LEFT")
        def right(self): print("   🔄 [SIM] Motors RIGHT")
        def stop(self): pass
        def close(self): pass

    class DummySensor:
        @property
        def distance(self): return 0.5 # Always safe

    my_robot = DummyRobot()
    sensor = DummySensor()

# ==========================================
# 🧠 BRAIN & UTILS
# ==========================================
def speak(text):
    print(f"🤖 Robot: {text}")
    clean_text = text.replace("'", "").replace('"', "")
    os.system(f'espeak -s 140 "{clean_text}" &')

def capture_image(filename="view.jpg"):
    # 📸 OPTIMIZED CAMERA SETTINGS
    cmd = (
        f"rpicam-still -o {filename} -t 1000 "
        "--width 640 --height 480 -q 50 -n --ev 2.0 --metering spot"
```

```
    )
    subprocess.run(cmd, shell=True)
    return filename

def send_to_brain(model, prompt, image_path=None):
    print(f"📡 Sending to {model}...")

    # KEEP ALIVE: 60m prevents the AI from going to sleep
    payload = {
        "model": model, "stream": False, "keep_alive": "60m",
        "messages": [{"role": "user", "content": prompt}]
    }

    if image_path:
        try:
            with open(image_path, "rb") as img:
                payload["messages"][0]["images"] =
[base64.b64encode(img.read()).decode('utf-8')]
        except: return "ERROR"

    try:
        res = requests.post(BASE_URL, json=payload, timeout=120)
        return res.json()['message']['content'] if res.status_code == 200 else "ERROR"
    except: return "TIMEOUT"

def decide_action(text):
    sys_prompt = """
    You are a robot. Output JSON ONLY.
    Valid Types: "CHAT", "MOVE", "SEARCH", "EXPLORE".
    Examples:
    - "Move forward" -> {"type": "MOVE", "payload": [{"dir": "FORWARD", "time":
1.0}]}
    - "Find the cup" -> {"type": "SEARCH", "payload": "cup"}
    - "Explore the room" -> {"type": "EXPLORE", "payload": null}
    """
    full_prompt = f"{sys_prompt}\nUser Command: {text}"
    res = send_to_brain("llama3.2:3b", full_prompt)
```

```
    try:
        start = res.find('{')
        end = res.rfind('}') + 1
        data = json.loads(res[start:end])
        return data.get('type', 'CHAT'), data.get('payload', 'I am confused.')
    except: return "CHAT", res

# ==========================================
# 🛡️ MOVEMENT, SEARCH & EXPLORE
# ==========================================
def execute_move(moves):
    speak("Moving.")
    for move in moves:
        d = move['dir'].upper()
        t = move.get('time', 0.5)

        # Safety Check
        if d == "FORWARD" and sensor.distance < SAFE_DISTANCE:
            speak("Obstacle detected. Stopping.")
            break

        if d == "FORWARD": my_robot.forward()
        elif d == "BACK": my_robot.backward()
        elif d == "LEFT": my_robot.left()
        elif d == "RIGHT": my_robot.right()

        time.sleep(t)
        my_robot.stop()

def run_search_mode(object_name):
    speak(f"Looking for {object_name}.")
    for _ in range(4): # Scan 360 degrees
        capture_image("view.jpg")

        loc = send_to_brain("moondream", f"Where is the {object_name}? Reply exactly:
LEFT, RIGHT, CENTER, or NO.", "view.jpg").upper()
        print(f"👀 Saw: {loc}")
```

```
        if "LEFT" in loc:
            speak("Found Left."); my_robot.left(); time.sleep(0.4); my_robot.stop();
execute_move([{"dir": "FORWARD", "time": 1.0}]); return
        elif "RIGHT" in loc:
            speak("Found Right."); my_robot.right(); time.sleep(0.4); my_robot.stop();
execute_move([{"dir": "FORWARD", "time": 1.0}]); return
        elif "CENTER" in loc:
            speak("Found Ahead."); execute_move([{"dir": "FORWARD", "time": 1.0}]);
return

        speak("Scanning...")
        my_robot.right(); time.sleep(0.6); my_robot.stop()

    speak(f"I could not find the {object_name}.")

def run_explore_mode():
    speak("Exploring the area.")
    observations = []

    for angle in ["left", "center", "right"]:
        capture_image("view.jpg")
        print(f"👀 Analyzing {angle} view...")
        desc = send_to_brain("moondream", "Describe this scene in one very short
sentence.", "view.jpg")
        observations.append(desc.replace("\n", " "))

        if angle != "right":
            my_robot.right()
            time.sleep(0.5)
            my_robot.stop()

    full_summary = ". ".join(observations)
    print(f"📝 Summary: {full_summary}")
    final_speech = send_to_brain("llama3.2:3b", f"Summarize these robot observations
into 2 sentences: {full_summary}")
    speak(final_speech)
```

```
# ==========================================
# 🎤 MAIN LOOP
# ==========================================
def main():
    r = sr.Recognizer()

    # ⚠ AMPLIFICATION CONFIG
    # We set this to None so it uses the 'default' device defined in .asoundrc
    mic_index = None

    speak("System Booting.")
    print("🔥 Warming up Brain...")
    send_to_brain("llama3.2:3b", "ping")
    speak("System Online.")

    print(f"\n🎤 Listening on Default Device (Amplified)...")

    with sr.Microphone(device_index=mic_index) as source:
        r.adjust_for_ambient_noise(source, duration=1)
        r.energy_threshold = 3000

        while True:
            print("\n🔴 Listening...")
            try:
                # 1. Listen for Wake Word
                audio = r.listen(source, timeout=5, phrase_time_limit=5)
                text = r.recognize_google(audio).lower()
                print(f" 👂 Heard: {text}")

                if WAKE_WORD in text:
                    speak("Yes?")

                    # 2. Listen for Command
                    audio = r.listen(source, timeout=8)
                    cmd = r.recognize_google(audio).lower()
                    print(f"🗣 Command: {cmd}")
```

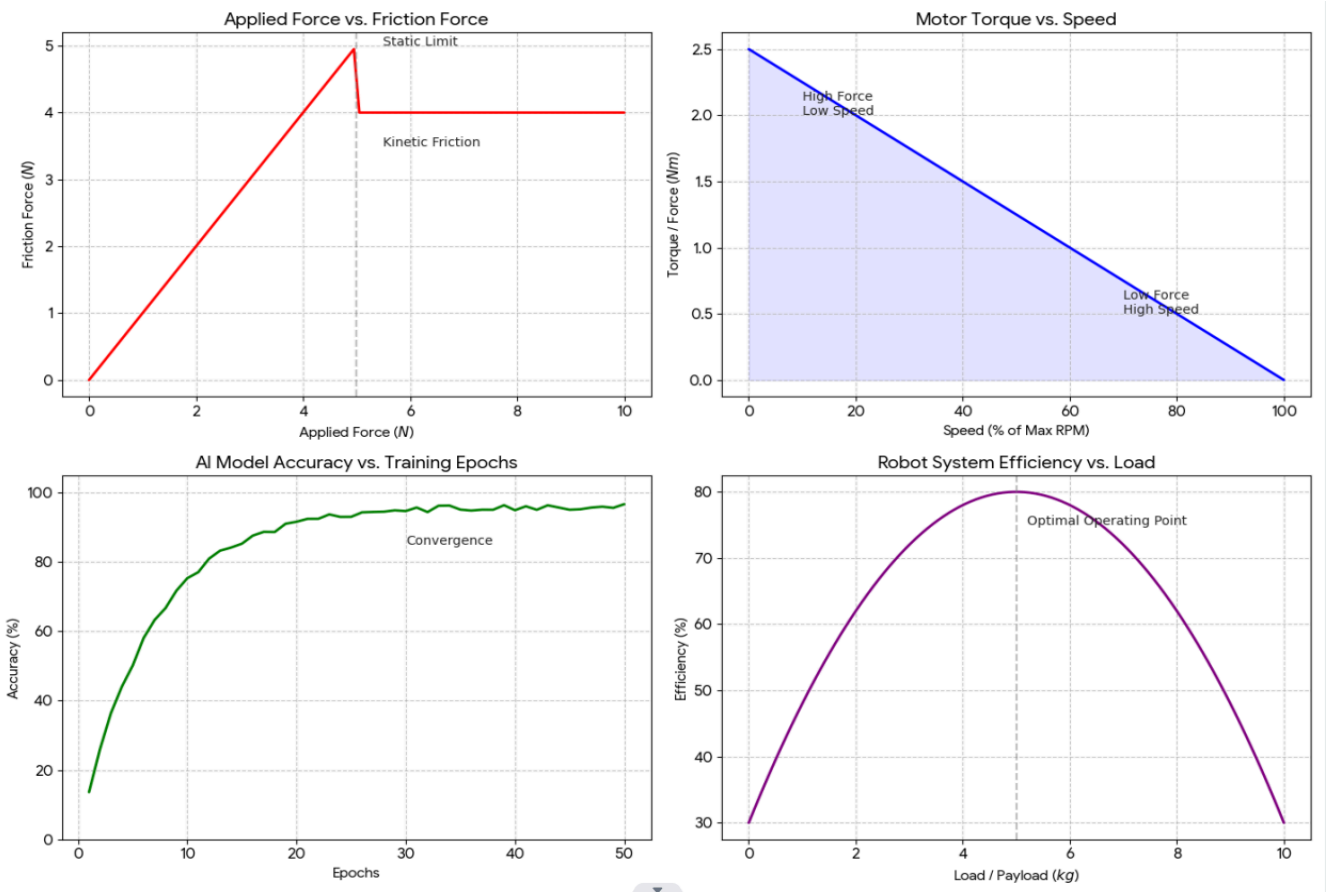
```
                # 3. Decide & Act
                action, payload = decide_action(cmd)
                print(f"⚡ Action: {action}")

                if action == "CHAT": speak(payload)
                elif action == "SEARCH": run_search_mode(payload)
                elif action == "EXPLORE": run_explore_mode()
                elif action == "MOVE": execute_move(payload)

            except sr.WaitTimeoutError: pass
            except sr.UnknownValueError: pass
            except Exception as e: print(f"Error: {e}")

if _name_ == "_main_":
    try: main()
    except KeyboardInterrupt:
        if not SIMULATION_MODE:
            my_robot.close()
            sensor.close()
        print("\nShutdown.")
```

Graphs



1. The Physics of Movement (Friction & Force)
   Top-Left Graph: shows how our robot starts moving.
   Static Friction: As the motors apply force, friction pushes back equally. The robot doesn't move yet.
   Kinetic Friction: Once moving, friction drops slightly and stays constant. This is why it takes more power to start the robot than to keep it moving.

2. Motor Performance (Force/Torque)

Top-Right Graph: Motors provide maximum pushing force (Torque) at zero speed (starting) and minimum force at top speed.

Trade-off: If our robot is heavy, it will move slower because it operates on the left side of this graph.

3. AI Performance (Accuracy)
   Bottom-Left Graph: Accuracy increases rapidly at first and then plateaus.

4. Energy Management (Efficiency)
   Bottom-Right Graph: Design chassis so the typical weight places the motors in their "Optimal Operating Point" (the peak of the curve) to save battery life.

Risk Assessment and Precautionary Actions

Category	Possible Error During Development	Precaution
Power & Wiring	Incorrect wiring causing short-circuits	Double-check connections; follow wiring diagrams; use color-coded wires
Motors	Overloading the motor causing heating or failure	Ensure correct voltage/current; add motor driver and heat protection

Speech/AI Module	Misinterpretation of voice commands	Calibrate microphone; use noise reduction; allow retry option
Bluetooth/Wi-Fi Connectivity	Connection dropouts	Keep within range; avoid obstacles; test connectivity strength
Software & Integration	Module conflicts (speech + motor + sensor together)	Integrate in stages; test each module separately first
Power Supply (Battery)	Battery drains too fast	Use correct Li-ion or Li-Po battery; test load requirements

Cost Analysis

Quantity	Item	Amount
2	Chassis	400
4	Wheel	500
4	Motor T.T	500
3	SR-04	540
1	L298N	350
1	INM441	600
1	5MP camera	650
1	RP 4 4G13	15000
1	3 Cell holder	60
3	18650 Cell	540
6	Jumper Wires	200

Project SDG Alignment

SDG	Why It Aligns with our Project
SDG 4: Quality Education	Our robot supports STEM learning by teaching robotics, AI, programming, and hands-on engineering. This helps promote "inclusive and equitable quality education."
SDG 9: Industry, Innovation, and Infrastructure	The project uses innovative technology (AI, embedded systems, robotics) to build infrastructure for smart automation. This aligns with the goal of "fostering innovation" and "building resilient infrastructure."
SDG 12: Responsible Consumption and Production	By using rechargeable batteries and reusable electronics parts, our robot contributes to more sustainable production and consumption patterns. This supports the goal of "responsible consumption."
SDG 13: Climate Action	Although our robot may not directly reduce emissions, its use of efficient components and potential for energy-efficient design aligns with the broader aim of "taking urgent action to combat climate change."
SDG 17: Partnerships for the Goals	To build and improve the robot, you'll likely collaborate (or could collaborate) across teams / disciplines (software, hardware, AI). That supports "revitalizing global partnership" for sustainable development.

Conclusion

Strengths

The system offers reliable speech interaction, supported by robust

and high-quality hardware components. Additionally, the integration of AI models enables intelligent and natural responses.

Weaknesses

The device's limited processing power restricts the implementation of more advanced AI features. Furthermore, background noise can negatively impact the accuracy of the microphone.

Opportunities

There is significant potential to enhance the system with advanced sensors and cloud-based AI capabilities. Moreover, the technology has a broad application scope in education, personal assistance, and research sectors.

Threats

Rapid technological advancements in the field increase competitive pressures. Additionally, privacy and data security concerns pose risks for voice-based systems

References

Research and Education in Robotics : https://doi.org/10.3390/jsan14040076
Chat gpt :https://chat.openai.com/
Youtube : https://youtu.be/e-nbSGRFP4Q?si=wnXoAmui734rYBJo