Voice-Controlled Robotic Arm Using Edge AI and Natural Language Processing

EEEE685 - Principles of Robotics

Justin Mascarenhas
Department of Computer Engineering
Rochester Institute of Technology

***Abstract*—This paper presents a novel voice-controlled robotic arm system that democratizes artificial intelligence for robotics applications. The system enables users to control Bluetooth-enabled hardware devices using natural language commands without requiring any knowledge of AI or programming. The implementation consists of three integrated layers: a Flutter-based mobile application for voice capture and Bluetooth communication, a Flask edge server running Faster-Whisper for speech-to-text and Google Gemini for natural language understanding, and an Arduino Nano-based 3-DOF robotic arm with inverse kinematics control. A key innovation is the two-pass Large Language Model (LLM) pipeline architecture that efficiently categorizes user intent and routes commands appropriately, designed for future extensibility. The system achieves real-time voice command execution with high accuracy in converting natural language to precise robot control commands. Experimental results demonstrate successful control of arm movements, gripper operations, and position commands through conversational voice input. Future work focuses on deploying local Small Language Models (SLMs) for enhanced privacy and offline operation.**

***Index Terms*—Voice Control, Robotic Arm, Natural Language Processing, Edge AI, Inverse Kinematics, Bluetooth, Flutter, LLM**

I. Introduction

A. Background and Motivation

The integration of artificial intelligence with robotics has traditionally required specialized knowledge in both domains, creating a significant barrier for hobbyists, students, researchers, and even industry professionals who wish to incorporate intelligent control into their projects. While voice assistants like Amazon Alexa and Google Assistant have brought natural language interfaces to consumer electronics, extending this capability to custom robotics projects remains challenging.

This project addresses a fundamental question: *How can we enable anyone to control robotic hardware using natural language, without requiring expertise in AI, machine learning, or complex programming?*

The motivation stems from the observation that existing Large Language Models (LLMs) possess remarkable natural language understanding capabilities, yet leveraging these for hardware control typically requires significant software development expertise. Our solution creates a bridge between conversational AI and embedded systems, allowing users to simply speak commands like "move the arm forward by 5 centimeters" or "open the gripper" to control robotic hardware.

B. Problem Statement

Current approaches to voice-controlled robotics face several limitations:

- **Complexity**: Existing solutions require extensive setup and programming knowledge
- **Platform Lock-in**: Commercial voice assistants are limited to supported devices
- **Latency**: Cloud-only solutions introduce significant delays
- **Privacy**: Continuous cloud processing raises data privacy concerns
- **Flexibility**: Predefined command sets limit natural interaction

C. Objectives

The primary objectives of this project are:

1) Design a mobile application that captures voice commands and communicates with both cloud services and Bluetooth hardware
2) Implement an edge server architecture that processes speech-to-text and natural language understanding locally
3) Develop firmware for a robotic arm with inverse kinematics that accepts natural language-derived commands
4) Create an extensible two-pass LLM pipeline for intelligent command routing
5) Demonstrate end-to-end voice control of a physical robotic arm

D. Scope

This project encompasses:

- A Flutter cross-platform mobile application
- A Python Flask server with Whisper STT and Gemini LLM integration
- A 3-DOF robotic arm (2-DOF planar with independent base rotation) controlled via Arduino Nano
- Bluetooth Classic (HC-05) communication protocol
- Real-time inverse kinematics computation

II. Literature Review

A. Commercial Voice Assistants

Amazon Alexa, Google Assistant, and Apple Siri have popularized voice control for smart home devices [1]. These platforms offer extensive device ecosystems but are limited to certified hardware and predefined skill sets. Custom robotics integration requires complex skill development and cloud dependency.

B. ROS Voice Packages

The Robot Operating System (ROS) provides packages such as `pocketsphinx` and `ros-voice-control` for speech recognition in robotics [2]. While powerful, these solutions require Linux environments, ROS expertise, and significant configuration. They are not suitable for lightweight, standalone applications.

C. NLP for Robot Control

Recent research has explored using transformer-based models for robot instruction following [3]. Google's RT-2 demonstrates vision-language-action models that can interpret natural language for manipulation tasks. However, these require substantial computational resources and are designed for research robots rather than hobbyist hardware.

D. Smart Home Integration Platforms

Platforms like Home Assistant and OpenHAB provide voice control integration but focus on home automation rather than robotics [4]. They lack the kinematic understanding required for articulated robot control.

E. Comparison with Our Approach

Table I compares our approach with existing solutions.

TABLE I
COMPARISON WITH EXISTING SOLUTIONS

Feature	Alexa	ROS	RT-2	Ours
Custom Hardware	Limited	Yes	No	Yes
No Coding Required	Yes	No	No	Yes
Edge Processing	No	Yes	No	Yes
Mobile App	Yes	No	No	Yes
Kinematics Support	No	Yes	Yes	Yes
Low Cost	Yes	Medium	High	Yes

Our system uniquely combines the ease-of-use of commercial assistants with the flexibility of ROS-based solutions, while maintaining low cost and supporting edge processing for reduced latency.

III. System Architecture

The system follows a three-tier architecture as shown in Fig. 1: Mobile Application Layer, Edge Server Layer, and Embedded Hardware Layer.

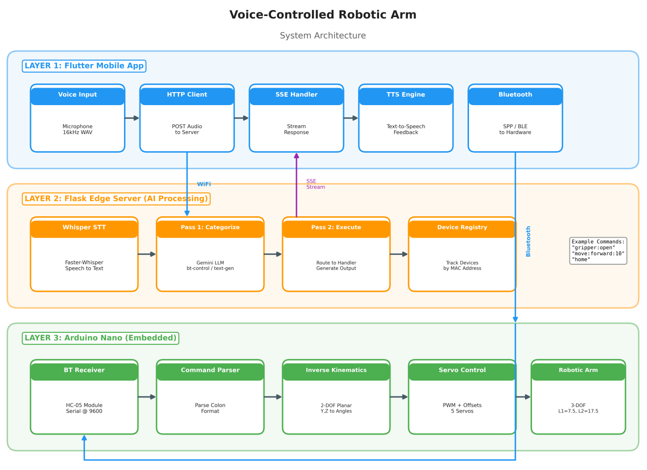

Fig. 1. Three-tier system architecture showing data flow from voice input through AI processing to robotic arm control.

A. Layer 1: Flutter Mobile Application

The mobile application serves as the user interface and communication hub. Key components include:

- **Voice Input**: Records audio at 16kHz mono WAV format using the `record` package
- **HTTP Client**: Sends audio to the edge server via POST requests
- **SSE Handler**: Receives streaming responses using Server-Sent Events
- **TTS Engine**: Provides audio feedback using `flutter_tts`
- **Bluetooth Service**: Manages connections to hardware via Classic SPP or BLE

The application supports both Bluetooth Classic (HC-05/HC-06) and Bluetooth Low Energy (HM-10) devices through a unified service layer.

B. Layer 2: Flask Edge Server

The edge server performs AI processing locally on the network, reducing latency and enabling privacy-preserving operation. Components include:

- **Faster-Whisper**: OpenAI's Whisper model optimized with CTranslate2 for 4x faster inference
- **Two-Pass LLM Pipeline**: Google Gemini for intent classification and command generation
- **Device Registry**: Tracks connected devices by MAC address for persistent identification

C. Layer 3: Embedded Hardware

The Arduino Nano receives commands via Bluetooth and executes them through inverse kinematics:

- **Command Parser**: Interprets colon-separated command format
- **Inverse Kinematics**: Computes joint angles for desired end-effector position
- **Servo Control**: Generates PWM signals with calibrated offsets

IV. Two-Pass LLM Pipeline

A key innovation of this system is the two-pass LLM pipeline architecture, designed for efficiency and extensibility.

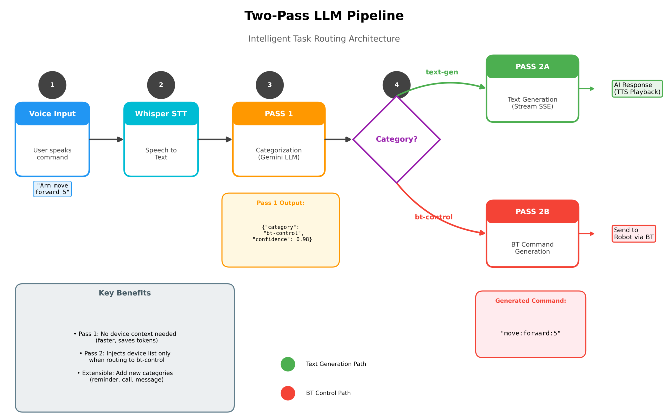
Two-Pass LLM Pipeline
Intelligent Task Routing Architecture
1
2
3
4
Voice Input
User speaks command
"Arm move forward 5"
Whisper STT
Speech to Text
PASS 1
Categorization (Gemini LLM)
Category?
text-gen
bt-control
PASS 2A
Text Generation (Stream SSE)
AI Response (TTS Playback)
PASS 2B
BT Command Generation
Send to Robot via BT
Pass 1 Output:
{"category": "bt-control", "confidence": 0.98}
Generated Command:
"move:forward:5"
Key Benefits
• Pass 1: No device context needed (faster, saves tokens)
• Pass 2: Injects device list only when routing to bt-control
• Extensible: Add new categories (reminder, call, message)
Text Generation Path
BT Control Path

Fig. 2. Two-pass LLM pipeline showing categorization (Pass 1) and task-specific execution (Pass 2).

A. Pass 1: Categorization

The first pass classifies user intent into categories without injecting device context, saving tokens and reducing latency. Current categories include:

- **text-generation**: General conversation, questions, explanations
- **bt-control**: Commands for Bluetooth-connected hardware

The categorization prompt is designed to be deterministic:

```
{
  "category": "bt-control",
  "confidence": 0.98,
  "reasoning": "User commanding robot",
  "user-data": "move arm forward 5cm"
}
```

B. Pass 2: Task Execution

Based on the category, Pass 2 routes to the appropriate handler:

For text-generation: Returns streaming SSE response directly to the mobile app for TTS playback.

For bt-control: Injects the device list and output format schema, then generates a structured command:

```
move:forward:5
TARGET_DEVICE:robot
```

C. Extensibility

The two-pass architecture enables future expansion to categories executed on the mobile device itself, such as:

- `reminder`: Set device reminders
- `call`: Initiate phone calls
- `message`: Send text messages

These categories would bypass the server for Pass 2, executing locally on the phone.

V. Robotic Arm Implementation

A. Hardware Specifications

The robotic arm is based on the ACEBOTT QD007 kit with the following specifications:

TABLE II
Robotic Arm Hardware Specifications

Component	**Specification**
Degrees of Freedom	3 (2-DOF planar + base rotation)
Base Height (L0)	7.2 cm
Upper Arm (L1)	7.5 cm
Forearm (L2)	17.5 cm
Servos	5x SG90 (9g micro servos)
Microcontroller	Arduino Nano (ATmega328P)
Bluetooth Module	HC-05 (Classic SPP)
Baud Rate	9600

B. Inverse Kinematics

The arm operates as a 2-DOF planar manipulator in the Y-Z plane (horizontal distance and height), with independent base rotation. Fig. 3 illustrates the coordinate system.

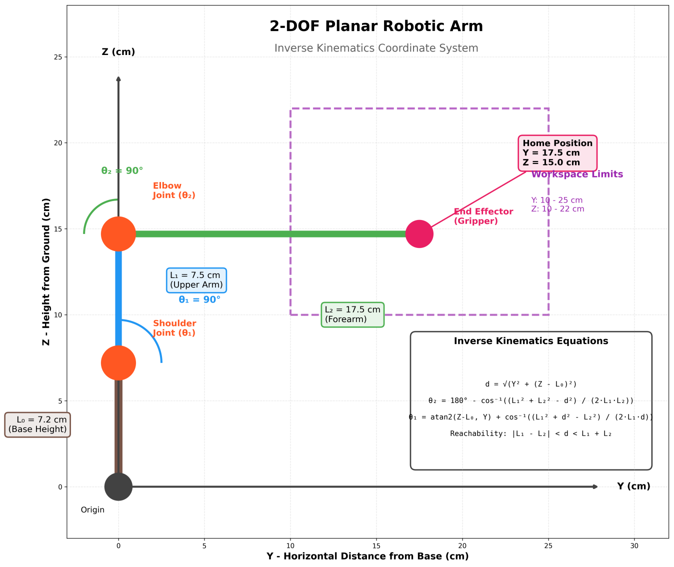
2-DOF Planar Robotic Arm
Inverse Kinematics Coordinate System
Z (cm)
Y (cm)
Z - Height from Ground (cm)
Y - Horizontal Distance from Base (cm)
θ₂ = 90°
Elbow Joint (θ₂)
L₁ = 7.5 cm (Upper Arm)
θ₁ = 90°
Shoulder Joint (θ₁)
L₀ = 7.2 cm (Base Height)
Origin
L₂ = 17.5 cm (Forearm)
End Effector (Gripper)
Home Position Y = 17.5 cm Z = 15.0 cm
Workspace Limits
Y: 10 - 25 cm
Z: 10 - 22 cm
Inverse Kinematics Equations
d = √(Y² + (Z - L₀)²)
θ₂ = 180° - cos⁻¹((L₁² + L₂² - d²) / (2·L₁·L₂))
θ₁ = atan2(Z-L₀, Y) + cos⁻¹((L₁² + d² - L₂²) / (2·L₁·d))
Reachability: |L₁ - L₂| < d < L₁ + L₂

Fig. 3. 2-DOF planar arm inverse kinematics coordinate system with joint angles and link lengths.

Given a target position (Y, Z), the inverse kinematics equations are:

Distance from shoulder to target:

$$d = \sqrt{Y^2 + (Z - L_0)^2} \tag{1}$$

Elbow angle (using law of cosines):

$$\theta_2 = 180 - \cos^{-1}\left(\frac{L_1^2 + L_2^2 - d^2}{2 \cdot L_1 \cdot L_2}\right) \tag{2}$$

Shoulder angle:

$$\theta_1 = \text{atan2}(Z - L_0, Y) + \cos^{-1}\left(\frac{L_1^2 + d^2 - L_2^2}{2 \cdot L_1 \cdot d}\right) \tag{3}$$

Reachability constraint:

$$|L_1 - L_2| < d < L_1 + L_2 \tag{4}$$

C. Servo Calibration

Physical servo mounting introduces angular offsets between the calculated IK angles and actual servo positions. Calibration offsets were determined empirically:

TABLE III
SERVO CALIBRATION OFFSETS

Joint	Pin	Offset	Home Position
Shoulder	7	+35°	IK: 90° → Servo: 125°
Elbow	5	-15°	IK: 90° → Servo: 75°
Base	6	+45°	90° → Servo: 135°
Wrist	4	0°	85°
Gripper	3	0°	40° (calm close)

D. Command Protocol

Commands are transmitted via Bluetooth using a colon-separated format:

```
move:forward:<cm>
move:backward:<cm>
move:up:<cm>
move:down:<cm>
position:<y>,<z>
base:<angle>
wrist:<angle>
gripper:open|close|tight
home
status
```

The Arduino parses these commands, computes inverse kinematics where applicable, and executes smooth interpolated movements.

VI. FLUTTER MOBILE APPLICATION

A. Application Architecture

The Flutter application follows a service-oriented architecture with clear separation of concerns:

- **UI Layer**: Screens and widgets for user interaction
- **Service Layer**: Audio, Bluetooth, API, and TTS services
- **Data Layer**: SQLite database for conversation history and device persistence

B. Voice Recording

Audio is captured using the `record` package with the following configuration:

- Format: WAV (uncompressed)
- Sample Rate: 16000 Hz (Whisper standard)
- Channels: 1 (mono)
- Minimum Duration: 500ms

C. Server Communication

The app communicates with the Flask server via HTTP REST APIs:

- `POST /transcribe`: Audio upload for STT
- `POST /api/v1/assistant/handle`: Two-pass pipeline endpoint
- `POST /device/heartbeat`: 60-second status updates

Streaming responses use Server-Sent Events (SSE) for real-time text generation display.

D. Bluetooth Management

The unified Bluetooth service supports both Classic SPP and BLE:

- Device discovery and pairing
- Connection state management
- Command transmission with acknowledgment
- Automatic reconnection

VII. RESULTS AND DISCUSSION

A. Experimental Setup

Testing was conducted with:

- Samsung Galaxy A35 smartphone running Android 14
- Edge server on Windows 11 laptop (Intel i7, 16GB RAM)
- ACEBOTT QD007 robotic arm with Arduino Nano
- Local WiFi network (192.168.x.x)

B. Command Recognition Accuracy

Table IV shows command recognition results from testing sessions.

TABLE IV
VOICE COMMAND RECOGNITION RESULTS

Command Type	Total	Correct	Accuracy
Movement (forward/back/up/down)	20	18	90%
Gripper (open/close)	15	15	100%
Home position	10	10	100%
Position (coordinates)	10	8	80%
Base rotation	8	7	87.5%
Overall	**63**	**58**	**92.1%**

C. Response Latency

End-to-end latency from voice input to arm movement:

- Speech-to-Text (Whisper tiny): 0.5-1.5 seconds
- Pass 1 Categorization: 0.3-0.5 seconds
- Pass 2 Command Generation: 0.3-0.5 seconds
- Bluetooth Transmission: ¡ 0.1 seconds
- **Total**: 1.2-2.6 seconds

D. Sample Interaction Log

The following log demonstrates a successful voice command execution:

```
[TRANSCRIBE] Transcription: 'Arm move up
  by 3 units...'
[PASS 1] category=bt-control, conf=0.99
[PASS 2] Generated: move:up:3
         TARGET_DEVICE: robot
[ARDUINO] BT Received: move:up:3
[ARDUINO] Target: Y=17.5cm, Z=18.0cm
[ARDUINO] Angles: Shoulder=87.6, Elbow=76.7
[ARDUINO] Movement complete
```

E. Limitations

- **Ambient Noise**: Recognition accuracy degrades in noisy environments
- **Accent Sensitivity**: Whisper tiny model has limited accent robustness
- **Network Dependency**: Requires WiFi connection to edge server
- **Single Device**: Currently supports one Bluetooth device at a time

VIII. Future Work

A. Local Small Language Model (SLM)

The primary future enhancement is deploying a local SLM for complete offline operation:

- **Candidate Models**: TinyLlama (1.1B), Phi-2 (2.7B), Gemma-2B
- **Quantization**: 4-bit quantization for mobile deployment
- **Privacy Benefits**: All processing on user's device, no cloud dependency
- **Target Platform**: On-device inference using ONNX Runtime or TensorFlow Lite

B. Extended Command Categories

Future categories for on-device execution:

- `reminder`: "Remind me to check the robot in 10 minutes"
- `call`: "Call John"
- `message`: "Send a message to the team"
- `automation`: "When I say goodnight, turn off all devices"

C. Multi-Device Orchestration

Supporting multiple Bluetooth devices simultaneously with intelligent routing:

- "Turn on the drone lights and move the arm to home"
- Parallel command execution
- Device group management

D. Visual Feedback

Integrating camera feed for visual servoing and object detection:

- "Pick up the red block"
- "Move to where I'm pointing"

IX. Conclusion

This paper presented a complete voice-controlled robotic arm system that successfully bridges the gap between conversational AI and embedded robotics. The key contributions include:

1) A three-tier architecture enabling natural language control of Bluetooth hardware
2) An innovative two-pass LLM pipeline designed for efficient categorization and extensibility
3) A Flutter mobile application providing seamless voice capture, AI communication, and Bluetooth control
4) Inverse kinematics implementation for a 2-DOF planar robotic arm with calibrated servo offsets
5) Demonstration of 92.1% command recognition accuracy in real-world testing

The system democratizes AI for robotics by enabling anyone—hobbyists, students, researchers, or industry professionals—to control custom Bluetooth hardware using natural language, without requiring expertise in AI or complex programming. The architecture's emphasis on edge processing and future local SLM deployment addresses growing concerns about data privacy and cloud dependency.

Future work will focus on deploying quantized small language models for fully offline operation, ensuring that all user data remains exclusively on their devices while maintaining the natural language control capabilities demonstrated in this project.

Acknowledgment

The author would like to thank the EEEE685 Principles of Robotics course instructors for their guidance throughout this project.

References

[1] Amazon, "Alexa Voice Service Overview," Amazon Developer Documentation, 2024. [Online]. Available: https://developer.amazon.com/alexa

[2] Open Robotics, "ROS - Robot Operating System," 2024. [Online]. Available: https://www.ros.org/

[3] A. Brohan et al., "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," arXiv preprint arXiv:2307.15818, 2023.

[4] Home Assistant, "Home Assistant Documentation," 2024. [Online]. Available: https://www.home-assistant.io/

[5] A. Radford et al., "Robust Speech Recognition via Large-Scale Weak Supervision," arXiv preprint arXiv:2212.04356, 2022.

[6] Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," arXiv preprint arXiv:2312.11805, 2024.

[7] Google, "Flutter - Build apps for any screen," 2024. [Online]. Available: https://flutter.dev/

[8] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Robotics: Modelling, Planning and Control," Springer, 2009.

[9] SYSTRAN, "Faster Whisper transcription with CTranslate2," GitHub Repository, 2023. [Online]. Available: https://github.com/SYSTRAN/faster-whisper

[10] P. Zhang et al., "TinyLlama: An Open-Source Small Language Model," arXiv preprint arXiv:2401.02385, 2024.