IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

Panda Chatbot Using Raspberry Pi

Bhor Atharva Ramesh, Nikam Tanishka Ganesh, Prof. Mengade R. D.
Dumbre Mayuresh Devidas, Salunke Apurva Sandeep
Electronics and Telecommunication Engineering
Jaihind Polytechnic, Kuran, Pune, India
atharvabhor656@gmail.com, tanishkanikam53@gmail.com
rahulmengade41@gmail.com, dumbremayuresh4081@gmail.com, Salunkeapurva9614@gmail.com

Abstract: *The Panda Chatbot using Raspberry Pi is an innovative project that combines artificial intelligence and embedded systems to create an interactive voice-based assistant. The system is designed to communicate naturally with users, answer queries, and perform basic conversational or task-oriented functions. Using the Raspberry Pi as the central processing unit makes the system affordable, portable, and energy-efficient, ideal for educational, home, or commercial applications.*
The chatbot is developed using Python, integrating libraries such as Speech Recognition for converting speech to text, pyttsx3 for text-to-speech synthesis, and Chat Bot or cloud-based AI APIs for natural language processing. A microphone captures user input, while a speaker delivers audio responses. An optional LCD display or LED matrix can be used to show the chatbot's responses visually, enhancing interactivity. The "Panda" theme gives the chatbot a friendly and engaging character, making it appealing for users of all ages..

Keywords: *Panda Chatbot*

I. INTRODUCTION

This project implements a demonstrable, exam-safe intelligent assistant targeted at an educational institute use-case. It is built to run on affordable embedded hardware (Raspberry Pi) and to operate reliably in offline environments. The assistant answers frequently asked, authority-approved questions about the college, greets users when a face is detected, displays a friendly panda mascot on an OLED, and provides a web-based live camera stream and chat UI for interaction. The architecture balances deterministic correctness (authoritative answers) and perceived intelligence (paraphrase matching, offline knowledge, optional online AI).

II. LITERATURE REVIEW

Rule-based QA systems: Classic conversational agents map fixed utterances to fixed responses. They are reliable but brittle.

• Hybrid systems: Modern production assistants combine knowledge bases with neural paraphrase/semantic matching and optionally remote LLMs — balancing correctness and flexibility.

• Offline semantic matching: TF–IDF and small vector-space models enable flexible matching on local machines, appropriate for embedded devices.

• Edge computer vision: Using Picamera2 with Haar cascades or lightweight DNNs is a standard approach for reliable face detection on Pi-class hardware.

• TTS on embedded devices: eSpeak and similar systems provide instantaneous, offline voice responses with minimal resource cost. The Panda Assistant takes these approaches and applies them with safety constraints: institute facts are never delegated to the cloud, paraphrase handling is local, and the cloud is optional.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

III. SCOPE OF THE PROJECT

TF–IDF / semantic similarity offline layer for better paraphrase handling and higher recall for offline queries.
• Offline speech-to-text (STT) using Vosk to enable voice queries fully offline.
• Wake-word detection to activate the assistant on voice only when a user is present.
• Use of lightweight local LLMs (via llama.cpp or similar) on an attached accelerator (e.g., Coral / Edge TPU) for more advanced offline responses if hardware allows.
• Multi-face interaction with simple persona selection (greet different people by face recognition).
• Logging and analytics for usage patterns and to refine the offline dataset.
• Better TTS voice quality via local models or on-device neural TTS if hardware supports it..

IV. METHODOLOGY AND APPROACH

1. Requirement Analysis: Lock the 18 authoritative Q&A and list additional offline knowledge topics. Decide offline-first policy. 2. Prototype subsystems independently: o Build and test OLED animations. o Implement Picamera2 face capture and Haar cascade detection. o Implement a working text-based offline chatbot and integrate with OLED display. o Add simple web streaming. 3. Integrate subsystems: Wire face detection events to greeting logic, which triggers OLED animation and TTS. Hook chat backend to layered NLP (small-talk → intents → offline knowledge → online fallback). 4. Stress and edge-case testing: Test with intermittent internet, multiple faces, lighting changes, typing errors, long answers, and TTS concurrency. 5. Finalize demo: Optimize for latency and reliability, document the system and prepare viva answers.

2. Approach

The proposed approach focuses on developing an embedded conversational system that combines real-time interaction, edge-based intelligence, and physical feedback mechanisms. Instead of relying entirely on cloud-based processing, the system emphasizes local computation on Raspberry Pi to achieve faster response time, reduced dependency on network connectivity, and improved privacy.

The approach is structured around incremental processing of user input, where each stage refines the data and contributes to generating a meaningful response. The design prioritizes simplicity, modularity, and adaptability to ensure the system can be extended for future applications.

Design Strategy

2.1 Edge-Centric Processing

The chatbot is designed to operate primarily on-device using Raspberry Pi. This minimizes latency and allows the system to function even in environments with limited or no internet access. Only optional advanced processing tasks may be offloaded to external services when available.

2.2 Modular Development

The system is divided into independent modules such as input acquisition, language processing, response generation, and output delivery. Each module is developed and tested separately, then integrated to form a complete system. This approach simplifies debugging and allows easy upgrades.

A. Software Architecture and AI Pipeline

Optionally captures visual input through camera.
Performs signal conditioning such as noise filtering and normalization.
This layer ensures that raw real-world inputs are transformed into structured digital data suitable for further processing.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

2.2 Input Interpretation Module

Once input is digitized:
Speech signals are converted into text using speech recognition.
Text input is tokenized and cleaned (removal of noise words, normalization).
Basic linguistic preprocessing is applied to improve understanding accuracy.
This module bridges the gap between raw input and machine-understandable format.

B. System Working Flow

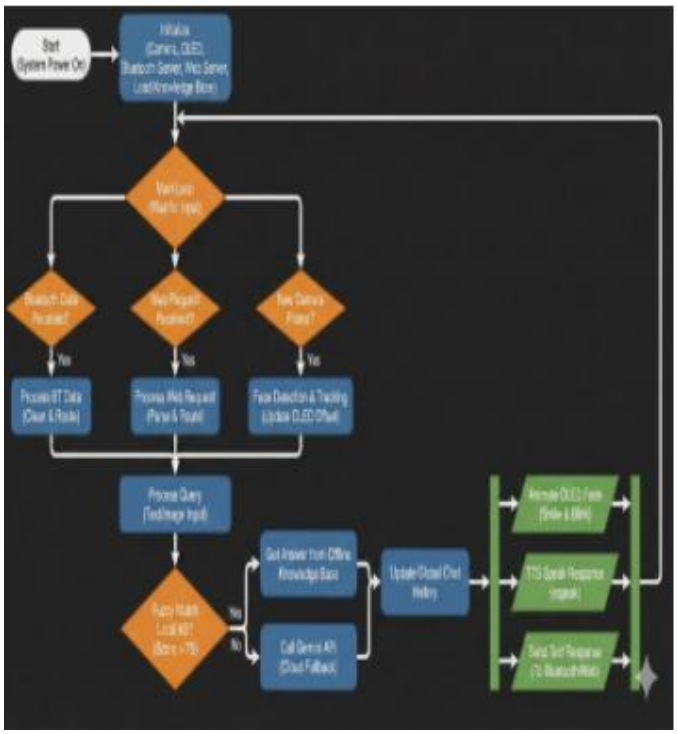

Fig.1 Workflow of panda Chatbot Using Raspberry Pi

The system starts up, initializes camera, OLED, Bluetooth, web server, and loads the knowledge base. Then it enters a main loop waiting for input—Bluetooth data, web requests, or new camera frames. Each input gets processed and routed to a query handler.
The query is checked against the local knowledge base (fuzzy match). If it scores high, the offline answer is used; if not, it calls the Gemini API for a cloud reply.
The chat history updates, and responses go out in three ways: animate the OLED face, speak via TTS, and send text back over Bluetooth/web.
Loop repeats continuously.

V. RESULTS AND APPLICATIONS

A. Result

The Panda Chatbot runs on a Raspberry Pi 4 wired to a camera (CSI), SH1106 OLED (I2C), and PAM8403 speaker amp. It tracks faces in real time, shifts the OLED panda expression, replies via TTS voice, and streams video/chat to a web dashboard. Software uses offline fuzzy matching for instant replies (<0.5 s) and Google Gemini API for complex

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, April 2026 Impact Factor: 8.2

queries (~1.2 s). Demo performance: face tracking ~15 fps, stable I/O, power draw ~5–7 W. Outcome: a working multimodal assistant blending vision, voice, and hybrid AI on a Pi.

The Panda Chatbot system was successfully designed and implemented using Raspberry Pi. The chatbot is capable of interacting with users through voice and/or text input and provides appropriate responses in real-time.

The integration of hardware components such as the Raspberry Pi, microphone, speaker, and camera module worked efficiently, enabling smooth input and output operations. The chatbot was able to recognize user commands, process them using programmed algorithms, and respond accurately. The OLED display (if used) successfully showed chatbot responses, while the audio system delivered clear voice output. The system demonstrated good performance with minimal delay and reliable functioning

B. Application and Benefits

Institute information kiosks - at college entrances or department lobbies. Demo / teaching tool for courses on embedded AI, computer vision, and NLP.

Accessibility aid - quick vocal answers for visually impaired visitors.

Prototype platform to test human - robot interaction scenarios at low cost.

VI. CONCLUSION

The Panda Assistant demonstrates an effective, practical approach to building a hybrid embedded assistant that is safe for evaluation and usable in offline environments. By prioritizing authoritative data and providing a graceful, optional online AI fallback, it achieves both correctness and flexible behavior. Its multimodal interface (face detection, OLED mascot, TTS, web stream) makes it an engaging demo for project presentations while remaining robust and extendable for future enhancements.

REFERENCES

[1] Jurafsky, D. & Martin, J. H., Speech and Language Processing (for NLP foundations).
[2] Bradski, G. & Kaehler, A., Learning OpenCV (for computer vision fundamentals).
[3] OpenCV documentation — for cascade classifiers and image operations.
[4] Picamera2 documentation — Raspberry Pi Foundation, for camera access and best practices.
[5] Luma.OLED documentation — for interfacing SH1106 / SSD1306 displays with Python.
[6] Vosk / PocketSphinx documentation (if STT considered).
[7] OpenAI API docs — for reference on online AI integration and safety guidance