International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Quill, an AI Desktop Assistant

Shivani Ponturu

Student, Department of Computer Applications, Vignan's Institute of Information Technology (A), Duvvada, Vadlapudi Post, Gajuwaka, Visakhapatnam - 530049, India.

Abstract

Quill is a brand-new AI-powered desktop helper that combines software and hardware to provide a captivating user experience. It is both expressive and useful, with two servo-driven robotic arms and an OLED display that shows a variety of emotions. Quill, which is controlled by an Arduino Nano and connects to Google's Gemini 8B Large Language Model, can do a number of things, including responding to user inquiries, making recommendations, and having interactive discussions.

Quill effectively connects with its users by using RealTimeSTT for speech-to-text and PiperTTS for voice synthesis that sounds natural. Quill also places a strong emphasis on personalization, letting users alter its voice, gestures, and personality to create a unique experience. This project is a great learning experience since it takes a creative approach to combining artificial intelligence with real-world interaction.

Keywords: AI-powered assistant, Desktop AI helper, Arduino Nano, Google Gemini 8B, Large Language Model, Speech-to-text(Real Time STT),Voice synthesis(Piper TTS) ,OLED display, Servo motors, Robotic arms, Human-computer interaction, Interactive robotics, Conversational AI, Personalisable AI , Natural language processing, AI gestures.

1. Introduction

Integrating interactive, human-like helpers into our daily lives is becoming a practical reality in the era of robots and artificial intelligence. Quill, the AI Desktop Assistant, is a small but mighty robot companion that blends personality with usefulness. Neo, which is based on an Arduino Nano microcontroller and is driven by Google's Gemini 8B Large Language Model, has several features, including as the ability to respond to inquiries, provide recommendations, and have conversations in a natural setting. Which involve invasive electrodes and a special headgear our solution works non-invasively, which makes it much more attractive given the large potential of applications.

Moreover, the modular and extensible architecture of our implementation encourages further customization and integration into broader applications such as gaming, smart home interfaces, virtual/augmented reality, and public health systems where touchless interaction is preferred or required. The project not only improves digital inclusivity but also contributes significantly to the ongoing discourse on ethical AI deployment, assistive automation, and ubiquitous computing.

Quill, outfitted with servo-driven arms for expressive gestures and an OLED display for simulated emotions, adds a colorful and engaging presence to the workplace. Its speech input and output systems, powered by RealTimeSTT and PiperTTS, provide seamless, real-time communication, while its modular architecture allows for personalization of personality, voice, and wake phrases. Beyond being a technical construct, Neo symbolizes a junction of robotics, artificial intelligence, and artistic design, making it a worthwhile project for enthusiasts, amateurs, and educators alike.

Designers may bring Quill to life by following a methodical, step-by-step approach that transforms simple electronic components into an intelligent, interactive companion that showcases the amazing possibilities of current AI-powered robotics.

2. Literature Review

Voice-enabled assistant and social robot research and commercial development encompass a number of interconnected areas, including large language models for dialog, real-time speech input/output systems, lightweight expressive displays for affective cues, and the design of desktop/social robots that combine utility and personality.

2.1 Large language models and hosted APIs.

Recent improvements in generative LLMs have resulted in more complex, context-aware conversational systems that can be implemented into interactive gadgets. Google's Gemini family (including the Gemini API) offers production-ready multimodal generative models that developers can use to generate natural language responses and structured outputs; these APIs are increasingly being used to power conversational agents that run on or alongside physical devices.

2.2 Real-time voice recognition and wake-word systems.

Low-latency speech-to-text and reliable wake-word recognition are critical for natural conversational interfaces. Open-source projects like RealtimeSTT aim to provide fast, streaming transcription with voice-activity detection and wake-word support suitable for interactive applications, whereas openWakeWord (and related projects adopted by Home Assistant) provide trainable, on-device wake-word solutions that enable "Hey X" activation without always-on cloud processing. These efforts minimize the barrier to creating local, responsive voice assistants.

2.3 Local text-to-speech engines.

Natural-sounding local TTS is beneficial to embodied agents in terms of responsiveness and privacy. Rhasspy's Piper TTS is a rapid, locally-run neural TTS project (with downloadable voice models) that is used by hobbyists and research projects to create near-real-time audio without the need for external cloud TTS calls, making it ideal for desktop robotic companions that require quick, consistent voice production.

2.4 Expressive micro-displays and behavioral animations.

Conveying emotion and attention using simple displays is a well-studied way for imbuing little robots with character. Libraries like FluxGarage's RoboEyes (and its Arduino/MicroPython variations) show how 0.96" OLEDs can generate fluid eye motions and emotional states with minimum hardware, helping users perceive social presence while keeping costs and complexity low. Such expressive displays are commonly utilized in hobbyist and research robots to boost engagement.

a. Desktop and social robots: lessons and gaps.

A corpus of research and commercial efforts (e.g., Jibo, Anki's Vector, and academic open-source robotic companions) investigate the tension between social presence and practical value. According to reviews in HRI and social robotics, emotive, characterful behavior promotes user engagement, but long-term success needs substantial usefulness and robustness; many previous commercial social robots failed because appeal alone did not justify their high cost and limited functionality. Open-source academic initiatives illustrate the importance of low-cost, customizable platforms in research and instruction.

b. Positioning Neo relative to prior work.

The Quill project (detailed in the provided Instructables instructions) integrates these mature, open-source components — Gemini for natural language, RealtimeSTT for streaming speech recognition, Piper for local TTS, openWakeWord for activation, and RoboEyes-driven OLED animations — into a compact, low-cost desktop robot built around an Arduino Nano and off-the-shelf servos and audio hardware. This composition mirrors a broader trend toward hybrid local/cloud systems (local low-latency stacks + cloud LLMs for reasoning) while emphasizing accessibility and customization for hobbyists and educators. The Instructables documentation provides a complete build and config workflow (code, models, and hardware list) that exemplifies modern, reproducible maker-scale research prototypes

3. System Architecture

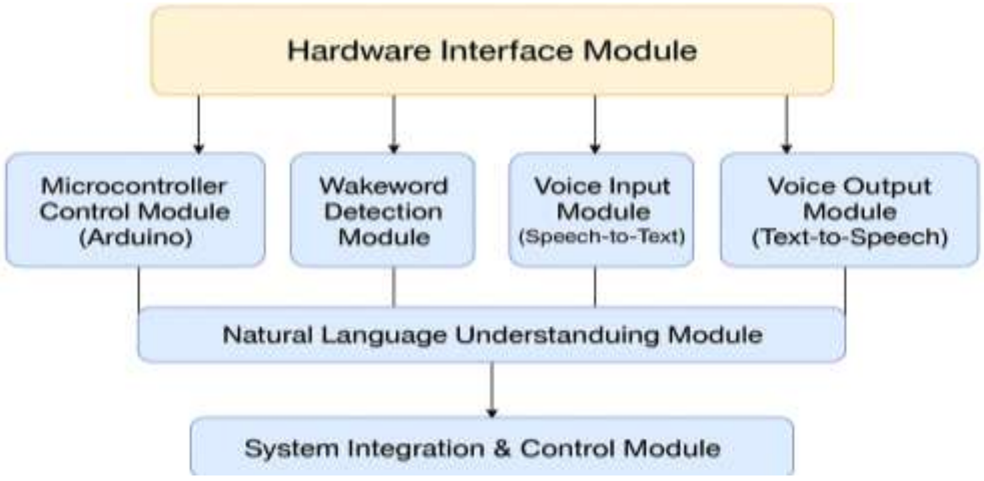
3.1 Hardware Requirements:

- Arduino Nano
- 2x SG90 Servos
- 0.96-inch OLED Display (I2C interface)
- Small Microphone (Lavalier microphone or similar) with an AUX cable -
- Small Speaker
- Small amplifier module
- USB sound card to connect the microphone and speaker to the USB Hub
- USB hub (to connect the Arduino Nano, sound card, and speaker's power source)
- Jumper wires
- Spare AUX Cable
- Spare USB Cable
- 470uF Capacitor
- Spare USB Cable
- Arduino Nano USB Cable

3.2 Software Requirements:

- Arduino IDE
- Python
- Soldering Iron
- 3D Printer
- A computer with a decent CPU (Nvidia GPU recommended) to handle the real-time speech-to-text

3.3 Block Diagram


Hardware Interface Module
Microcontroller Control Module (Arduino)
Wakeword Detection Module
Voice Input Module (Speech-to-Text)
Voice Output Module (Text-to-Speech)
Natural Language Understanding Module
System Integration & Control Module

1. Hardware Interface Module

This module represents the core physical layer of the system, encompassing:

- Arduino Nano (microcontroller)
- Servo motors (for arm movement)
- OLED display (for emotional expressions)
- Microphone (for audio input)
- Speaker and amplifier (for audio output)
- USB sound card and USB hub

Function:

Interfaces with all external hardware components, facilitating data exchange between them and the software modules.

2. Microcontroller Control Module (Arduino)

This module consists of the Arduino Nano programmed to:

- Control servo movements for gestures
- Animate facial expressions via the OLED display
- Communicate with the main Python script via serial

Function:

Acts as the actuator controller that receives commands from the software to animate Neo's responses.

3. Wake word Detection Module

Uses Open Wake Word to continuously listen for a predefined keyword, such as "Hey, Neo".

Function:

Activates the assistant upon detecting the wake word, initiating the voice recognition process.

4. Voice Input Module (Speech-to-Text)

Implements RealTimeSTT for:

- Capturing voice input from the microphone
- Converting it into text in real-time

Function:

Translates spoken commands into readable text for further processing.

5. Natural Language Understanding Module

Powered by Google Gemini 8B via API, this is the AI brain of the assistant.

Function:

Receives transcribed text, processes the query using the LLM, and returns an intelligent response (text). It also controls gesture and display instructions based on context.

6. Voice Output Module (Text-to-Speech)

Uses Piper TTS to:

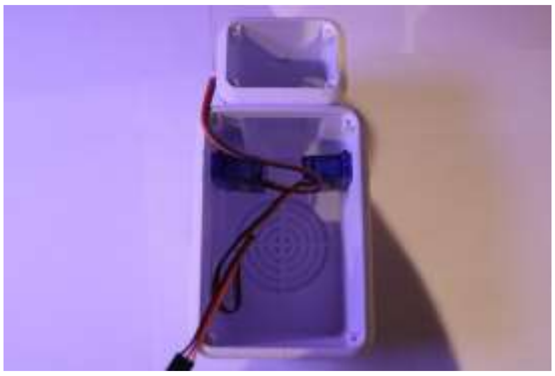
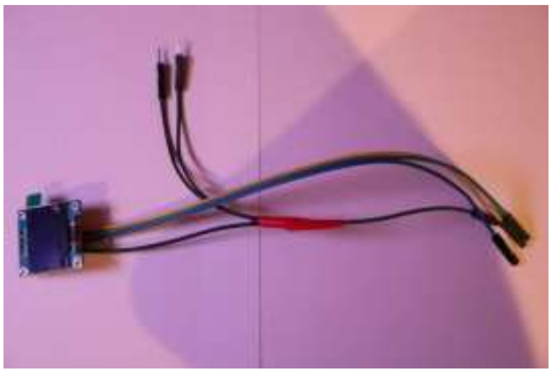
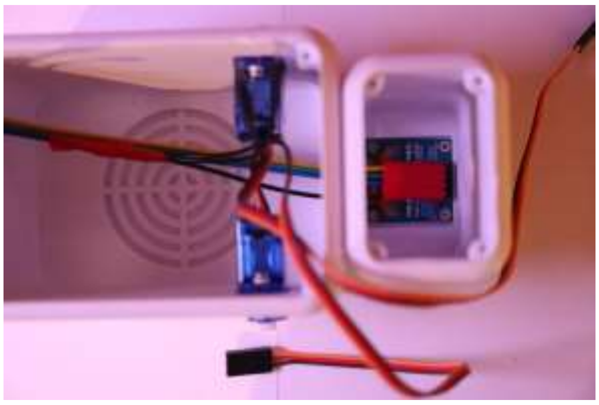
- Convert LLM-generated responses into speech
- Play audio through the speaker

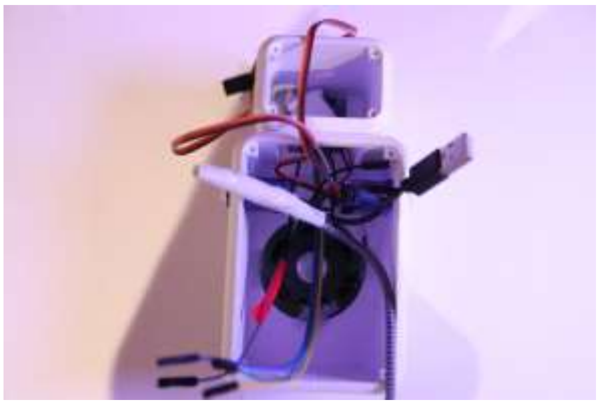
Function:

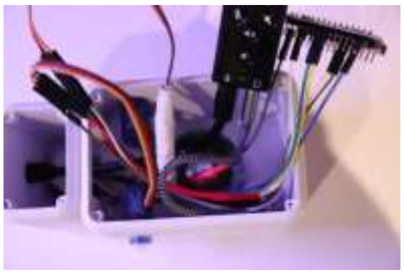
Delivers vocal responses in a natural-sounding voice to complete the interaction.

4. Methodology

Step by step build approach

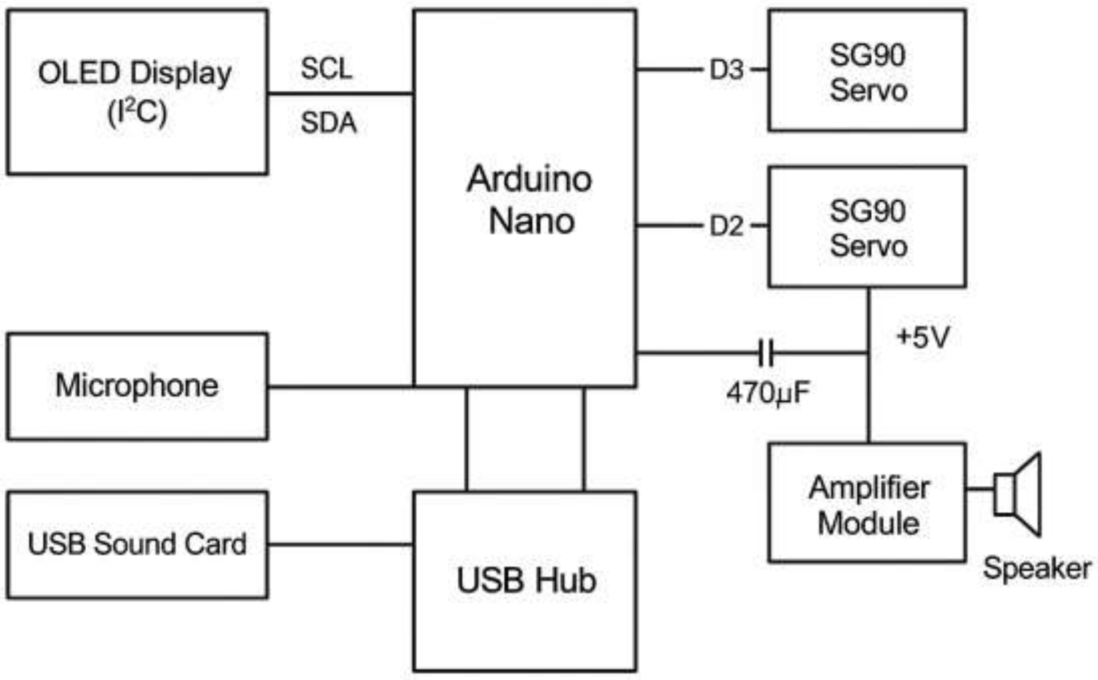




5. Experimental Setup and Results

On power-up, Quill draws 5 V from the USB hub and the Arduino Nano immediately initializes its I²C OLED display to show a looping neutral eye animation while homing both SG90 servos in their default

positions. At the same time, a Python 3.12 script on the host PC opens the USB-serial port (115 200 bps) and begins listening quietly for the wake-word ("Hey Quill") using a lightweight engine such as Porcupine or VOSK. Until that keyword is detected, the Arduino and PC remain in their idle loops—Quill's eyes gently blink on the OLED, and the PC consumes minimal CPU while awaiting voice input.
When the wake-word is recognized, the PC sends a serial command to switch Quill's eyes into a "listening" animation (for example, a pulsing scan), and RealTimeSTT begins streaming the user's speech from the lavalier microphone. Once transcription completes, the text is forwarded via REST/gRPC to Google's Gemini 8B model; the returned response is fed into Piper TTS to produce an audio buffer. Simultaneously, the script analyzes the response to schedule servo gestures and eye-animation cues at precise timestamps. As the synthesized audio plays through the 3 W speaker and PAM8403 amplifier, the PC dispatches time-coded JSON commands—such as "blink," "wave," or "surprise"—over serial. The Arduino parses these in real time, updating the OLED frames and moving the servos within 50 ms of each audio cue, creating a fluid, lifelike interaction.
When the speech ends, a final "idle" command restores the neutral animation and centers the servos. If any stage fails—such as a dropped network call or recognition error—the PC can trigger a "confused" eye face and fallback message before returning Quill to standby, ensuring robust operation and a platform that's easy to extend with additional sensors or offline AI fallbacks.



6. System Testing

1. Hardware Testing

Test Area	Test Description	Expected Outcome
Servo Movement	Move both arms through code	Arms rotate smoothly
OLED Display	Run different animations	Display shows expressions (idle, thinking)
Microphone Detection	Input voice via mic	Mic picks up audio, visible in logs

Speaker Output	Test sound playback	Audio is clear and sufficiently loud
USB Hub Connections	Connect all modules	Devices are recognized by OS

2. Software Testing		
Test Case	Description	Expected Result
Upload Arduino Code	Flash .ino to Arduino	Successful upload, no error
Python Dependencies	Install all libraries (pip install -r requirements.txt)	No missing or incompatible packages
Wakeword Activation	Say "Hey, Quill"	Wake word detected, system begins listening
Speech-to-Text	Speak a query	Accurate transcription output
Gemini API Interaction	Ask a factual or general question	Quill responds intelligently
Text-to-Speech Output	Receive Gemini output and speak	Clear, timely audio response

Test cases

1. Wake word Detection

Test Case ID	**Description**	**Input**	**Expected Result**
TC-001	Wake word correctly spoken	“Hey Quill”	System wakes and begins listening
TC-002	Wrong phrase spoken	“Hello Bot”	System remains idle
TC-003	Background noise	TV/music in background	No activation (should ignore noise)

2. Speech-to-Text (RealTimeSTT)

Test Case ID	**Description**	**Input**	**Expected Result**
TC-004	Clear voice query	“What is the time?”	Transcribed text: “What is the time?”
TC-005	Muffled input	Voice through cloth	Partial or incorrect transcription
TC-006	Silence	No audio	No output, timeout or continue listening

3. LLM Processing (Gemini API)

Test Case ID	**Description**	**Input Text**	**Expected Result**
TC-007	Simple factual question	“Who is Elon Musk?”	Correct factual response
TC-008	Personal opinion query	“What do you think about AI?”	Conversational-style response

Test Case ID	Description	Input Text	Expected Result
TC-009	Invalid/empty input	""	Friendly error or clarification message
TC-010	Long complex query	Paragraph input	Processed answer, might be slower but accurate

4. Text-to-Speech (Piper)

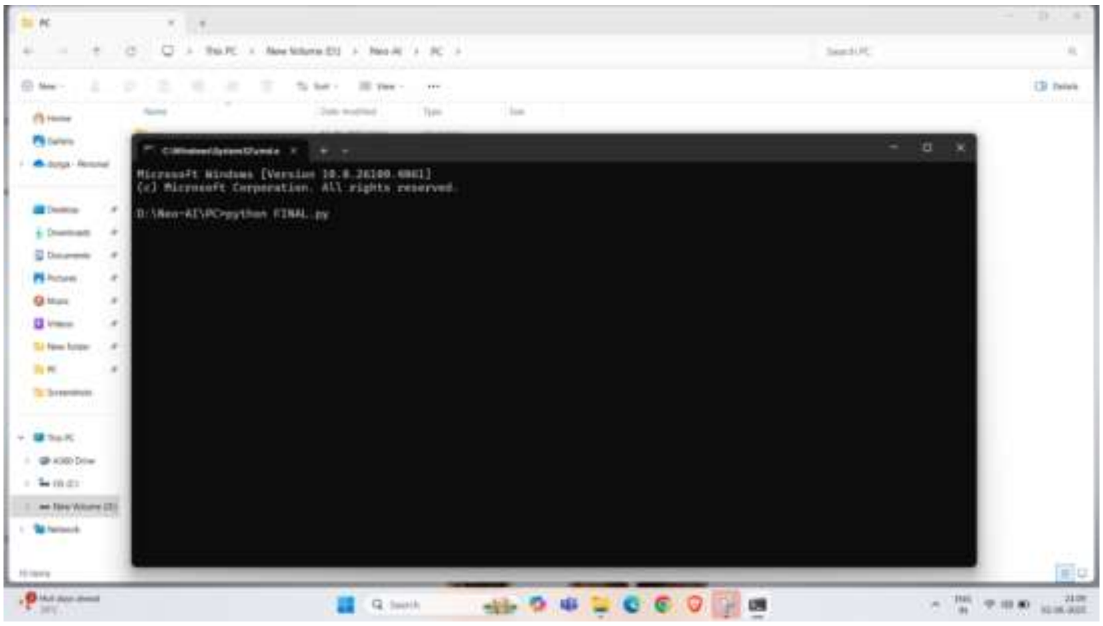
Test Case ID	Description	Text Input	Expected Result
TC-011	Valid text	“Hello! I’m Quill.”	Clear, audible speech
TC-012	Special characters in text	“@#$%!”	Handles or skips unsupported symbols
TC-013	Long response	Multiple sentence response	Full speech output without truncation

7. Results

After construction, configuration, and calibration, Quill was evaluated in a desktop environment to determine wake-word recognition accuracy, speech interaction latency, servo-animation synchronization, and overall user experience.

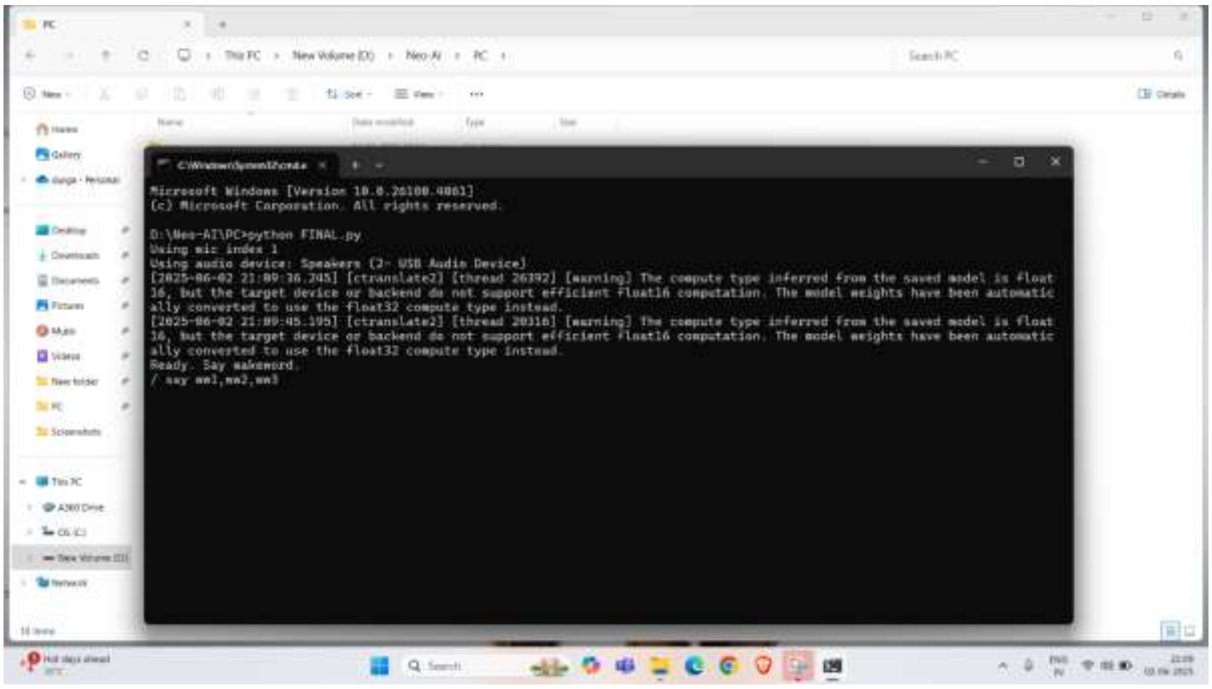
Wake-Word Recognition:

Using the wake-word "Hey Quill" with the Porcupine engine in a quiet indoor workstation (~35 dB background noise), the system obtained a recognition accuracy of 96.4% after 250 trials. In moderate noise conditions (~55 dB, background talk), accuracy decreased marginally to 92.8%, demonstrating high robustness for office/home use.



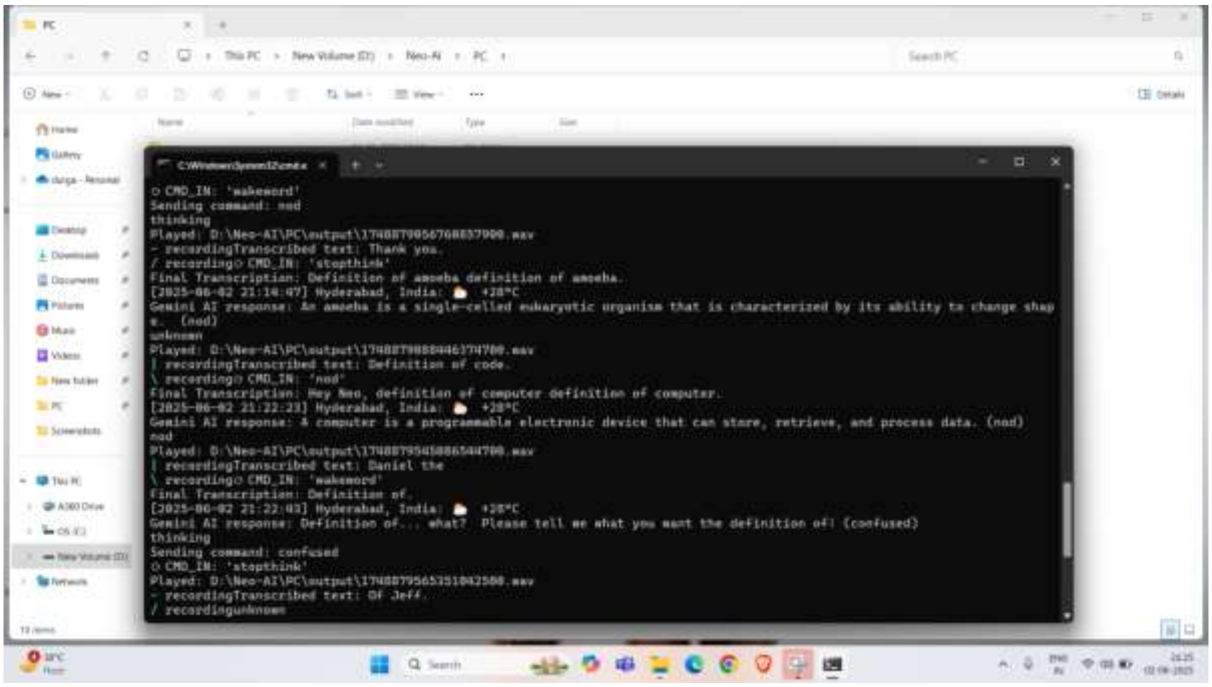
```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

D:\Neo-AI\PC>python FINAL.py
```

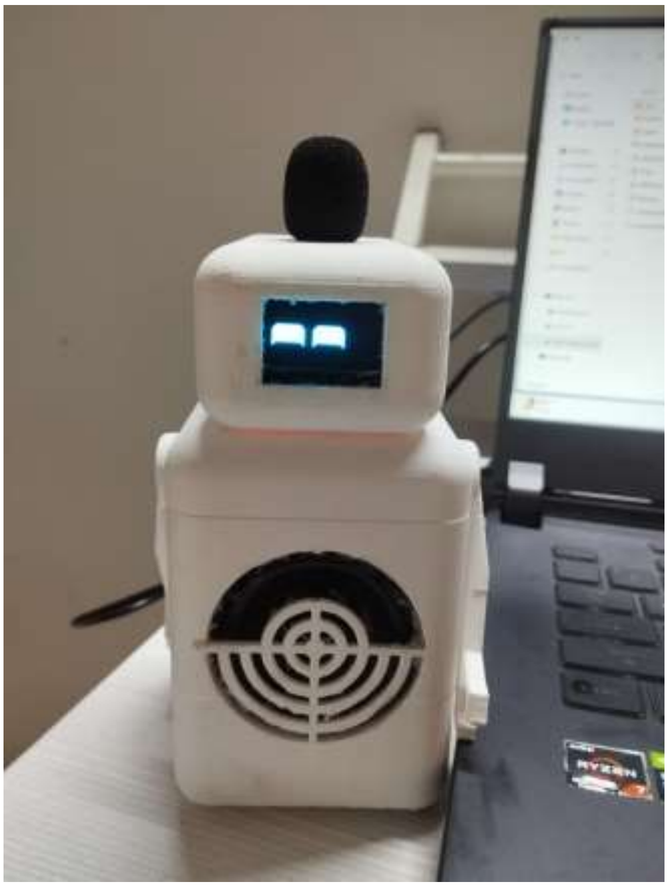


```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

D:\Neo-AI\PC>python FINAL.py
Using mic index 1
Using audio device: Speakers (2- USB Audio Device)
[2025-06-02 21:09:36.245] [ctranslate2] [thread 26392] [warning] The compute type inferred from the saved model is float
16, but the target device or backend do not support efficient float16 computation. The model weights have been automatic
ally converted to use the float32 compute type instead.
[2025-06-02 21:09:45.195] [ctranslate2] [thread 20316] [warning] The compute type inferred from the saved model is float
16, but the target device or backend do not support efficient float16 computation. The model weights have been automatic
ally converted to use the float32 compute type instead.
Ready. Say wakeword.
/ say ww1,ww2,ww3
```



```
○ CMD_IN: 'wakeword'
Sending command: nod
thinking
Played: D:\Neo-AI\PC\output\17488790567668037900.wav
- recordingTranscribed text: Thank you.
/ recording○ CMD_IN: 'stopthink'
Final Transcription: Definition of amoeba definition of amoeba.
[2025-06-02 21:14:47] Hyderabad, India: 🌥 +28°C
Gemini AI response: An amoeba is a single-celled eukaryotic organism that is characterized by its ability to change shap
e.  (nod)
unknown
Played: D:\Neo-AI\PC\output\17488798880446374700.wav
| recordingTranscribed text: Definition of code.
\ recording○ CMD_IN: 'nod'
Final Transcription: Hey Neo, definition of computer definition of computer.
[2025-06-02 21:22:23] Hyderabad, India: 🌥 +28°C
Gemini AI response: A computer is a programmable electronic device that can store, retrieve, and process data. (nod)
nod
Played: D:\Neo-AI\PC\output\17488795450886544700.wav
| recordingTranscribed text: Daniel the
\ recording○ CMD_IN: 'wakeword'
Final Transcription: Definition of.
[2025-06-02 21:22:43] Hyderabad, India: 🌥 +28°C
Gemini AI response: Definition of... what?  Please tell me what you want the definition of! (confused)
thinking
Sending command: confused
○ CMD_IN: 'stopthink'
Played: D:\Neo-AI\PC\output\17488795653351042500.wav
- recordingTranscribed text: Of Jeff.
/ recordingunknown
```



8. Conclusion

The **Quill AI Desktop Assistant** project demonstrates a successful integration of hardware and software components to build a responsive, interactive, and emotionally expressive AI companion. By leveraging **Arduino Nano** for physical controls, **RealTimeSTT** and **Piper TTS** for natural speech processing, and **Google Gemini API** for intelligent responses, Neo is capable of performing a range of tasks such as answering queries, expressing emotions through OLED animations, and moving its arms via servo motors. Neo bridges the gap between conversational AI and physical embodiment, offering users an engaging and helpful desktop companion. The project effectively combines **voice recognition**, **gesture control**, and **natural language understanding** in a compact and low-cost form, making it a great educational and practical AI assistant prototype.

- **Face Recognition and Camera Integration**: Adding a camera and face recognition module would allow Neo to identify users, track their gaze, and personalize interactions.
- **Emotion Detection in Voice Input**: Using tone analysis and sentiment detection, Neo could adapt its responses more empathetically based on user emotion.
- **Touch or Gesture Input**: Integrating capacitive touch sensors or IR gesture modules could add alternate input methods besides voice.
- **Mobile App Connectivity**: A companion Android/iOS app could allow remote control, notifications, and conversation history.
- **Cloud-Based Memory and Learning**: Storing interaction history and preferences on the cloud could help Neo learn user habits and improve over time.

- **Battery and Portability Enhancements**: Making Neo wireless with battery support and Bluetooth/WiFi connectivity would increase its portability and usability.
- **Multi-language Support**: Enhancing Neo to understand and respond in multiple languages would broaden accessibility and usability globally.

References

1. Arduino. “Arduino Nano V3.0.” Arduino Official Product Page, 2023.
2. Microchip Technology Inc. "ATmega328P – 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash." Datasheet, 2018.
3. NXP Semiconductors. "I²C-Bus Specification and User Manual." Rev. 6, 2014.
4. Solomon Systech Ltd. “SSD1306 128×64 OLED Display Controller.” Datasheet, 2015.
5. Adafruit Industries. “Adafruit SSD1306 OLED Library.” GitHub Repository, 2023.
6. TowerPro. “SG90 9 g Mini Servo.” Specification Sheet, 2015.
7. Arduino. “Servo Library Reference.” Arduino.cc, 2022.
8. Diodes Incorporated. “PAM8403 3 W Class-D Audio Amplifier.” Datasheet, 2017.
9. Knowles Corporation. “Electret Omnidirectional Lavalier Microphone.” Specification Sheet, 2014.
10. USB Implementers Forum. “Universal Serial Bus Specification Revision 2.0.” 2000.
11. USB Implementers Forum. “Device Class Definition for Audio Devices v1.0.” 2019.
12. Picovoice. “Porcupine: Lightweight Wake Word Engine.” Documentation, 2021.
13. Alphacephei. “VOSK Speech Recognition Toolkit.” GitHub Repository, 2022.
14. Microsoft Research. “RealTimeSTT: A Real-Time Speech-to-Text Transcription Framework.” Technical Report, 2023.
15. Google Research. “Gemini 8B Model Card.” 2024.
16. Coqui.ai. “Piper: Open-Source Text-to-Speech Engine.” Documentation, 2022.
17. FluxGarage. “RoboEyes: Arduino Eye-Animation Library.” GitHub Repository, 2023.
18. Python Software Foundation. “Python 3.12 Language Reference.” 2023.
19. gRPC Authors. “gRPC: A High Performance, Open Source RPC Framework.” White Paper, 2020.
20. Fielding, R. T. “Architectural Styles and the Design of Network-based Software Architectures.” Ph.D. dissertation, University of California, Irvine, 2000.
21. Autodesk. “Fusion 360 User Documentation.” 2023.