Smart Robot Dog

Almaz Shinbay, Raiymbek Nazymkhan, Zhibek Rakhymbekkyzy,
Sultan Rzagaliyev, Aibar Ibrayev, Dr. Siamac Fazli[1]
Department of Computer Science
Nazarbayev University
Astana, Kazakhstan

***Abstract*—This project details the creation of a voice-interactive, 3D-printed replica of Pixar's WALL·E, capable of face recognition and animatronic movements. Using an open-source design, the team integrated a Raspberry Pi, Arduino, microphone array, and camera into a custom hardware/software system. The robot responds to offline voice commands and can identify and greet individuals. Over two semesters, the team assembled over 310 parts and developed the control software. The final robot met all project goals, showcasing a successful blend of robotics and AI in an engaging live demo.**

I. Introduction

WALL·E is a beloved robot character known for his charm and expressiveness. Recreating him as an interactive robot serves as both a tribute to the film and a hands-on engineering challenge, combining mechanical design, electronics, and artificial intelligence.

The goal of this project was to bring WALL·E to life as a personal robot capable of natural interaction. Unlike a static toy, our version responds to voice commands and recognizes individuals, demonstrating real-time human-robot interaction.

This work showcases the integration of advanced features—offline keyword detection and face recognition—into a DIY animatronic platform using hobbyist-scale hardware. The robot features fully articulated movement, driven by an Arduino for motor control and a Raspberry Pi for processing vision and voice commands. It can detect specific voice keywords and identify familiar faces, providing feedback through motion, sounds, and an LCD display.

The remainder of this report covers related work, system design, implementation details, and evaluation. We conclude with project outcomes, challenges, and potential directions for future development.

II. Related Work

Numerous hobbyists and researchers have attempted to recreate WALL·E due to the character's popularity. Our project builds on an open-source design by chillibasket (2019), which features a fully 3D-printed, servo-actuated WALL·E with seven joints and motorized treads. That version used an Arduino Uno for motor control and a Raspberry Pi running a Flask server to enable manual control via a web interface, offering animations, sound triggers, and remote driving. However, it lacked autonomous interaction capabilities.

[1]Advisor for the project

Our project advances this by combining voice and vision on a single platform. Unlike prior work, we enable keyword detection and real-time face recognition entirely on-device using a Raspberry Pi. For face recognition, we used TensorFlow Lite models on the Coral Edge TPU, including a MobileNet SSD v2 face detector and an embedding model based on FaceNet principles. This allowed the robot to identify individuals without cloud services, using embedding comparisons to recognize familiar faces.

For voice input, we used the ReSpeaker 4-Mic USB Array, which provides built-in noise reduction, voice activity detection, and direction-of-arrival estimation. These features enabled WALL·E to detect voice commands even in noisy environments and orient itself toward the speaker for more life-like interaction—similar to commercial assistants like Amazon Echo.

In summary, while prior work provided strong mechanical foundations and basic interactivity, our project is, to our knowledge, the first to integrate real-time voice and face recognition into a fully 3D-printed WALL·E. By running all AI components offline, we demonstrate a more autonomous and engaging robot that goes beyond remote control to enable natural human-robot interaction.

III. Project Approach

Our approach is divided into two main parts: (1) hardware integration, and (2) software architecture for control, voice, and vision. Below we detail the hardware design, tools used, and component roles.

A. Hardware Architecture

The robot's structure is based on a 3D-printed WALL·E design by chillibasket, with over 300 PLA parts assembled into a 40 cm tall frame. Key hardware components include:

- **Motion System:**
  - **Servos (7x):** TowerPro MG90S micro servos animate WALL·E's eyes (2), head pan (1), neck pitch and lift (2), and shoulder joints (2).
  - **Drive Motors (2x):** 12 V DC gear motors power skid-steering movement through the tracks.
- **Control Electronics:**
  - **Arduino Uno:** Handles low-level control: driving DC motors via an L298N H-bridge, and controlling servos through a PCA9685 12-channel PWM driver

over I^2C. Receives commands from the Pi via USB serial.
  - **Raspberry Pi 4:** Acts as the robot's "brain," running Python code for vision, voice interaction, and high-level behavior.
- **Sensors and I/O:**
  - **ReSpeaker 4-Mic Array:** Provides far-field voice input, noise reduction, and real-time direction-of-arrival (DOA) estimation, enabling WALL·E to turn toward the speaker.
  - **Logitech C270 Camera:** Mounted behind one eye, streams video for local face detection and recognition.
  - **16x2 LCD Display:** Placed behind the "solar charge level" label, used for text feedback (e.g., prompts during face learning).
  - **Speaker:** Outputs WALL·E sound effects and synthesized speech for interactive responses.
- **Power System:**
  - **Battery:** A 3-cell 11.1 V Li-Po battery powers the system. A buck converter provides 5 V for logic, while DC motors use the full 11 V.

All components are neatly integrated into WALL·E's frame. The Raspberry Pi and Arduino are housed in the torso, with careful cable routing through the neck and body to preserve joint mobility. This setup provides a robust platform for motion, voice, and visual interaction.

B. Software Architecture

The control software runs on two processors: an **Arduino Uno**, which manages real-time motor control, and a **Raspberry Pi 4**, which handles perception, behavior, and high-level coordination.

1) Arduino Firmware: The Arduino firmware, written in C++, interprets high-level serial commands from the Raspberry Pi to control WALL·E's servos and drive motors. It uses a simple protocol with single-character and multi-byte commands to trigger gestures and movements. The code maintains an internal animation queue with easing functions for smooth, lifelike motion. Each servo is calibrated for consistent behavior across sessions.

For full source code and implementation details, refer to the original repository: https://github.com/chillibasket/walle-replica.

2) Raspberry Pi Software: The Raspberry Pi runs a Python 3 application structured into three main threads that operate independently via thread-safe queues:

a) Voice Interface: Audio from the ReSpeaker mic array is processed using *Picovoice Rhino*, an offline speech-to-intent engine. Recognized intents (e.g., "move forward", "raise arms") are passed to the intent handler. The mic array also provides the angle of arrival, allowing WALL·E to rotate toward the speaker using `rotate_to_voice()` before acting.

b) Intent Handler: Each voice command maps to a motor command or a sequence of Arduino instructions. Some behaviors also trigger synchronized audio playback using `simpleaudio`. New actions can be added by extending the intent-to-command mapping.

c) Face Recognition: A dedicated thread handles real-time face detection and recognition using TensorFlow Lite models accelerated by the Coral USB TPU. If a face matches an existing profile, WALL·E greets the user by name via `pyttsx3`. If not, it prompts "Who are you?" on the LCD, records the spoken name (validated against a whitelist), and adds a new face embedding to memory.

IV. Project Execution

We began this project at the start of the Fall 2024 semester (Senior Project I) and completed it by April 2025 (Senior Project II), following a planned timeline with some adjustments along the way. The execution can be described in phases: initial research and design, hardware manufacturing, software development (iterative), integration, and testing/refinement. Below we outline the timeline, reflect on what went well or poorly in each phase, and discuss how we managed team dynamics and decision changes throughout the project.

A. Phase 1: Research & Planning (Aug 2024 – Dec 2024)

We began by outlining the desired functionality, components, and architecture for our smart robot dog. Our key goal was to build an offline-capable system, which led us to focus on optimizing and integrating all required ML models onto a Raspberry Pi. We explored pretrained, lightweight models for vision and voice recognition, and researched both cloud-based and offline solutions.

On the vision side, we initially tested OpenCV's LBPH face recognizer with a laptop webcam. However, its limitations under varying conditions led us to consider neural network-based solutions accelerated by the Coral USB TPU. For voice recognition, we began with OpenAI's Whisper small models but found them too large and slow. We then tried Whisper tiny and added keyword-matching algorithms for better performance, but inference time remained a concern. Ultimately, we chose Picovoice Rhino for its customizable, phrase-based recognition.

B. Phase 2: 3D Printing and Assembly (Jan 2025 – Mar 2025)

We began 3D printing in mid-January 2025 after finalizing the STL files. The chillibasket design includes around 310 parts, with 210 being small tread links.

Printing took longer than expected due to cracking in thin parts and warping in some assemblies, requiring several reprints and causing delays.

While some team members managed the printing process, others continued software integration to avoid bottlenecks. Assembly was done in parallel, starting with the head/eye gimbal, arm joints, and tread bogies—each tread consisting of 70 plates and 140 pins.

By mid-March, we had installed the chassis, tracks, servos, and DC motors, and completed the wiring, preparing the robot for electronics bring-up.

C. Phase 3: Software–Hardware Integration, Testing & Refinement (Mar 2025 –Apr 2025)

With the mechanical build complete, effort shifted to a two-step sprint: *(i)* wiring and software integration, followed by *(ii)* intensive user-style testing.

a) Integrated control loop.: A Python script on the Raspberry Pi verified serial control of the Arduino (`w/a/s/d/q` for tracks and single-letter servo codes). We then embedded the Picovoice *Rhino* SDK, compiled an offline intent model containing ten core commands plus two triggers (`WALL-E` and `who_am_I`) [4]. Using the ReSpeaker array's DOA API, each recognised command is preceded by a head/body turn toward the sound source [3]. By mid-March, voice commands reliably produced Arduino actions.

b) Face-recognition thread.: The Coral USB Edge-TPU runtime was installed and tested with a quantised SSD-MobileNet-v2 face detector and a MobileNet-based embedding model [5]. Running detection at 320×240 kept inference at $\sim$15 fps while leaving CPU free for Python threads. Embeddings are stored in RAM (and mirrored to JSON) and matched with a light k-NN search; a 30 s timeout aborts stalled enrolment.

c) Concurrent testing and polish.: From late March we exercised the full stack with scripted regression tests and ad-hoc trials by classmates. Feedback added synonyms (e.g. "come here"$\rightarrow$*move_forward*) and confirmed average command latency of $\approx$ 1 s and face-ID latency of $\approx$ 2 s. Noise handling was tuned so unrecognised speech is ignored, flashing the mic-array LED as feedback. Servo easing plus a small neck counter-weight kept motions smooth even under the head's mass. By mid-April the robot could execute every voice command, greet known users, and learn new faces without supervision, completing the integration milestone.

V. Project Evaluation

The project was evaluated through functional tests conducted by our team in the laboratory. No external users or formal questionnaires were employed.

A. Test Procedure

- **Voice–motion commands.** We issued the full set of ten predefined commands (*forward, back, left, right, stop, wave_your_hands, raise_both_arms, how_are_you, i_love_you, who_am_I*) ten times each, listening through the ReSpeaker array. The Arduino's serial log confirmed every command was received and executed; visually, the robot performed the corresponding movement or gesture on all 100 attempts.
- **Face recognition.** With our team members, we ran twenty recognition trials in normal lab lighting. All faces were detected; the known individuals were greeted by name in every case, and an unknown face was correctly labelled "Unknown" five times out of five before entering the learning dialogue.
- **Concurrent-load test.** To verify the Raspberry Pi could run audio capture, intent parsing, Coral-TPU vision and servo control simultaneously, we kept *face recognition* active while issuing random voice commands for fifteen minutes.

B. Results

All functional checks passed:

- Every voice command triggered the intended motion or animation.
- Face detection and identification worked reliably in the controlled indoor setting.
- The Pi handled audio, vision and motion threads together without noticeable latency or thermal throttling.

C. Limitations

The evaluation was confined to a quiet laboratory with even lighting; performance in noisy spaces or under harsh illumination was not measured. Only several faces were enrolled, so large-scale accuracy remains untested. Nevertheless, the results confirm that the integrated system performs its intended functions on the target hardware.

VI. Conclusion and Possible Future Work

A. Conclusion

In this project, we successfully designed and built a fully interactive WALL·E robot replica that merges advanced voice and vision capabilities with a proven 3D-printed animatronic platform. We began with an ambitious goal – to give a hobby-scale robot the ability to understand spoken commands and recognize individuals – and we achieved it through careful integration of hardware and software. The final robot can navigate and gesture on command, respond with character-appropriate sounds and movements, and even recall and greet people by name. These contributions demonstrate how modern AI technologies (like offline NLP and machine vision) can be deployed on compact, standalone systems to enhance human-robot interaction.

The project also provided invaluable learning experiences for our team. We gained hands-on skills in mechatronics (3D printing, circuitry, servo tuning), embedded programming (Arduino real-time control), and AI deployment on edge devices (optimizing models to run on a Raspberry Pi + TPU, and dealing with sensor data in real environments). We learned to manage a complex project over an extended period, coordinating between sub-teams. All objectives outlined in our project proposal were met, and the final deliverable was a working system demonstrated to faculty and peers. In conclusion, the project achieved its purpose of bringing WALL·E to life in both form and function. We have essentially created a small-scale social robot: one that can listen, move, see, and respond to people in an engaging way.

B. Possible Future Work

While the current implementation is fully functional, there are several avenues to extend and improve the project=. Here we outline some potential future work:

1) **Miniaturisation & efficiency.** Redesign the chassis and electronics layout to shrink overall size, cut weight and reduce idle power. A lighter platform would extend runtime and ease transport to outreach events.
2) **Bill-of-materials cost reduction.** Replace premium parts (e.g. Raspberry Pi 4, ReSpeaker) with lower-cost SBCs and MEMS mic arrays; investigate injection-moulded treads instead of 3-D-printed links. The goal is to cut total hardware cost by at least 30 %, making the robot affordable.
3) **Conversational dialogue engine.** Extend the current intent grammar with a lightweight on-device ASR+NLU stack (e.g. Vosk + Rasa) so users can ask open questions and receive natural replies rather than fixed keyword commands.
4) **Autonomous follow-me navigation.** Add a depth sensor or stereo vision module plus SLAM/path-planning so the robot can lock onto a user and follow while avoiding obstacles—essential for dynamic classroom demos.
5) **Sub-500ms response time.** Profile the end-to-end pipeline and migrate bottlenecks (e.g. audio pre-processing, serial round-trip) to faster implementations; target a maximum command-to-motion latency of 0.5 s for snappier interactions.

REFERENCES

[1] S. Bluett, "3D Printed WALL·E Robot," *wired.chillibasket.com*, Available: https://wired.chillibasket.com/3d-printed-wall-e/.

[2] S. Bluett, "*wall-e-replica*: Robot and controller code for a WALL·E replica robot," *GitHub*, Available: https://github.com/chillibasket/walle-replica.

[3] Seeed Studio, "ReSpeaker USB Mic Array," *Seeed Studio Wiki*, Available: https://wiki.seeedstudio.com/ReSpeaker-USB-Mic-Array/.

[4] Picovoice Inc., "Rhino Speech-to-Intent SDK Documentation," *picovoice.ai*, Available: https://picovoice.ai/docs/rhino/.

[5] Google Coral Team, "Get started with the USB Accelerator," *coral.ai*, Available: https://coral.ai/docs/accelerator/get-started/.