Smart Home for the Aged

Final Project Report

Team 37

J Adithya (2024101012)

Aryan Patel (2024111029)

Sree Chandana (2024101029)

1. Project Overview

The "Smart Home for the Aged" system enhances elderly users' safety and comfort through fall detection, voice command recognition, and automated environmental controls. This intelligent home system creates a safer living environment for elderly residents while providing convenience through hands-free operation.

2. Backend Code Functionality

Key Responsibilities:

- Polls the OM2M audio container (`/in-name/voice_command/audio_upload`) every 4 seconds for new base64-encoded audio recordings
- Reassembles WAV files from multipart Base64 messages (AUDIO_START, AUDIO_CHUNK, AUDIO_END)
- Uses FasterWhisper for speech-to-text transcription and SentenceTransformer for semantic matching against predefined commands
- Applies a similarity threshold (0.2) to choose the best command, attaches confidence and recognized text
- Executes commands by POSTing content instances back to the appropriate OM2M resources (LED, solenoid, fan)
- Tracks processed sessions via hashing to avoid duplicates
- **Data Persistence:** A separate Python script periodically fetches OM2M content instances (audio, fall, sensor data) and stores them in a local SQLite database

3. Main Node Code Overview

The ESP32 "main node" firmware handles fall detection and voice recording, uploading both to OM2M:

Setup & Connectivity

- Initializes MPU6050 accelerometer and Wi-Fi
- Provides audible success/error tones

Fall Detection

- Reads accelerometer, detects free-fall (< 0.44 g) followed by impact (> 0.05 g)
- Plays alert tone and uploads event data to OM2M

Touch-Triggered Recording

- On capacitive touch, records 3 seconds of 8 kHz audio via MAX9814 microphone
- Applies simple signal processing for improved clarity

WAV Packaging & Upload

- Builds a 44-byte WAV header
- Splits audio into four Base64 chunks and sends to OM2M
- Provides feedback & reliability through tones at key stages
- Includes memory checks to ensure robust operation

4. LED & Gas Sensor Node Code Overview

The ESP8266 "LED node" controls an LED and monitors a gas sensor:

Setup & Connectivity

- Configures pins for the main LED (D2) and gas indicator LEDs (D1 green, D3 red)
- Establishes Wi-Fi connection

LED Control

- Polls `/in-name/led/la` every 2 seconds
- Parses "ON"/"OFF" commands and toggles the LED only on change

Gas Monitoring

- Reads analog gas value
- If value > 400, turns on red LED and posts the value to `/in-name/gas_sensor/data`

5. Fan & Solenoid Node Code Overview

This ESP8266 node mirrors the LED node but handles fan speed and solenoid lock:

Setup & Connectivity

- Assigns GPIOs for solenoid lock and PWM fan output
- Connects to Wi-Fi

Command Fetching

- Polls `/in-name/fan/la` and `/in-name/solenoid/la` endpoints every 2 seconds

Actuation Logic

- **Fan:** Parses a 0–3 speed value and sets PWM duty cycle (0 = off, 1–3 = low/med/high)
- **Solenoid:** Controls door lock based on "LOCK"/"UNLOCK" commands

State Tracking

- Remembers last commands to avoid redundant writes
- Logs actions for debugging

6. Workflow Description

1. **Wearable Fall & Voice Event:** ESP32 detects a fall or touch, records audio or fall data, and uploads to OM2M
2. **Backend Processing:** Python script polls OM2M, reassembles audio, transcribes via FasterWhisper, semantically matches commands
3. **Command Dispatch:** Backend sends a content instance back to the relevant OM2M container (LED, fan, solenoid)
4. **Node Actuation:** ESP8266 nodes poll their respective containers, parse commands, and actuate hardware accordingly
5. **Data Persistence:** A dedicated Python script fetches OM2M entries and stores them in a local database

7. Challenges Encountered

- **Microphone Calibration & Noise:** Original sound sensor failed to capture clear voice; replaced with MAX9814 for better quality
- **Fall Sensor Calibration:** Tuning free-fall and impact thresholds on the MPU6050 was challenging; required extensive testing
- **Back-Voltage on Actuator Nodes:** Fan and solenoid back EMF caused spurious triggers; addressed with diodes and filtering capacitors
- **Audio Chunking & Memory Constraints:** ESP32 RAM couldn't hold full audio; implemented multi-part

transmission solution

- **Semantic Command Matching:** Adjusting the SentenceTransformer similarity threshold to 0.2 balanced flexibility with accuracy

8. Future Improvements

- Add a web/mobile dashboard for live monitoring and manual override
- Enable dynamic addition of new voice commands at runtime
- Integrate noise-reduction preprocessing on the backend for cleaner transcriptions
- Provide audio/LED feedback upon successful command execution at each node

---

Submitted: May 5, 2025