IOT Project: SHEGA(Smart Home Environmental Guardian Assistant)

Name: Sebah Tewodros Tesfatsion ID: 7405796

Introduction

The SHEGA project(Smart Home Environmental Guardian Assistant) was developed as a simulation of how IoT technology can improve household safety. Many homes face hidden risks: rising carbon monoxide levels, poor air quality, or stove overheating. SHEGA addresses these by integrating environmental monitoring, stove safety management, secure communication, and AI-driven guidance into one cohesive system.

SHEGA was designed as a complete ecosystem. Each home in the system includes two devices:

1. AirNode, which monitors air quality and provides alarms.
2. StoveNode, which ensures that the stove cannot be operated under unsafe conditions.

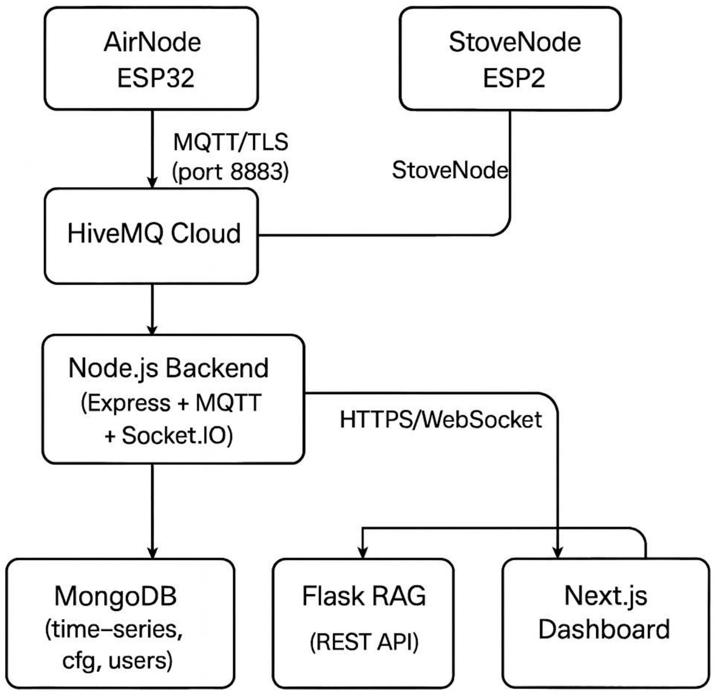
The devices communicate via HiveMQ MQTT (TLS-secured) to a Node.js backend with MongoDB storage. A Next.js dashboard built with TailwindCSS and Recharts provides visualization and controls. Finally, users can interact with a bilingual RAG assistant, which translates raw measurements into safety advice in Amharic and English.

Objectives

The goals of SHEGA were:

- Monitor CO_2, CO, particulate matter, temperature, and humidity in real time.
- Automatically lock the stove and trigger alarms when unsafe conditions are detected.
- Provide audio and visual warnings to occupants.
- Allow users and admins to view and manage their devices through a dashboard.
- Support role-based access: admins see all devices; users see only their own.
- Provide contextual, bilingual safety guidance through an AI assistant.
- end-to-end security across devices, communication, backend, and dashboard.

System Architecture



AirNode

The AirNode acts as the home's environmental monitor. It includes:

- SCD30/SCD41 → CO_2 concentration
- ZE03-CO → Carbon monoxide levels
- PMS5003 → Particulate matter (PM1.0, PM2.5, PM10)
- DHT22 → Temperature and humidity
- Buzzer actuator → Immediate audio alarms

All of these were implemented in WokWi using custom chips, which I authored in C. For each sensor, I defined:

- Pin mappings (VCC, GND, communication pins).
- A slider control (e.g., temperature or ppm value) to simulate live sensor changes.
- A C backend that mimics real sensor behavior over SPI, UART, or I^2C.

This allowed me to faithfully simulate unavailable hardware inside WokWi, ensuring realistic readings.

StoveNode

The StoveNode manages stove safety. Its key components are:

- Thermocouple (MAX31855) → Stove surface temperature.
- Fan actuator → For automated ventilation.
- Servo-controlled valve → Physically prevents the stove from turning on in unsafe states.

The thermocouple was also implemented as a custom WokWi chip, which outputs 32-bit SPI frames exactly like a real MAX31855 would. Through this, I could simulate overheating events and watch the system lock the stove, spin the fan, and trigger the AirNode alarm.

The fan is also implemented With LED that turns on and off when the fan is on or off

Communication Layer

Communication is handled by HiveMQ MQTT, running over TLS (port 8883) with username/password authentication. Each message is scoped with a homeId and deviceId, ensuring that multiple homes can coexist.

- shega/airnode/data → CO_2, CO, PM, temperature, humidity
- shega/stovenode/status → Stove safety data
- shega/control → Remote commands
- shega/events → System-generated safety events

Backend and Dashboard

The backend was built in Node.js + Express. It listens to device data via MQTT, applies threshold checks, and persists telemetry in MongoDB. It also exposes REST APIs for dashboard queries, authentication, and device management.

The dashboard was built with Next.js, styled with TailwindCSS, and visualized using Recharts. It provides:

- Live gauges and graphs for each household.
- Stove safety indicators showing when the stove is locked or overheated.

Role-based views:

- Admins → View all devices across all homes, add users.
- Users → View only their devices and interact with the RAG assistant.

RAG Assistant

Retrieval-Augmented Generation (RAG) Assistant

To complement raw telemetry data with meaningful explanations, I built a RAG service that combines lightweight retrieval with local language model generation.

- Framework & API :Implemented in Flask (Python) with REST + Server-Sent Events (SSE) for real-time streaming answers to the dashboard.
- Retrieval : Uses scikit-learn's TfidfVectorizer + cosine similarity for efficient document retrieval from a curated knowledge base (shega_rag.json). This ensures responses are always grounded in stored safety information.
- Generation : Integrates with Ollama running the phi3:mini model, instructed by a strict system prompt that encodes safety thresholds (CO_2, CO, PM2.5, CO, stove/room temperature).
- Bilingual responses : Automatically replies in Amharic or English based on the user's query language.
- Role-based access: Only household users can query the assistant; admins manage devices but do not use RAG.

Example: If CO_2 exceeds 1500 ppm, the assistant retrieves safety knowledge from the KB, interprets the live value against the threshold, and streams back a short explanation of health risks plus actionable advice (e.g., ventilate the room).

Security

Since the project controls critical appliances, I prioritized security:

Device Security

- HiveMQ communication uses TLS encryption.
- Devices authenticate with username/password credentials.
- Devices implement Last Will and Testament (LWT) to signal unexpected disconnections.

Backend Security

- Helmet.js for hardened HTTP headers.
- CORS restricted to dashboard origins.
- Rate limiting to prevent brute-force attacks.
- JWT authentication with strict role checks (admin vs user).

Data Protection

All telemetry linked to homeId and deviceId for strict isolation.

Users cannot access or modify data from other households.

This combination of device-level, communication-level, and backend-level security ensures the system simulates what a production IoT platform would require.

Results

At the end of implementation, the following results were achieved:

- Two simulated edge devices with custom WokWi chips (AirNode + StoveNode).
- Full TLS-secured MQTT communication through HiveMQ.
- Automatic safety enforcement: stove lockout, fan activation, and buzzer alarms.
- MongoDB persistence of all telemetry with household separation.
- A modern dashboard with role-based access, built on Next.js + Tailwind + Recharts.
- A bilingual RAG assistant to help users interpret readings and take action.

Conclusion

SHEGA demonstrates how IoT can go beyond monitoring and into active prevention and education. The project's main contributions include:

- A dual-device architecture for air quality and stove safety.
- Secure communication through HiveMQ with TLS.
- Custom WokWi chip simulations for otherwise unavailable sensors.
- A role-based dashboard with clear visualization.
- A bilingual RAG assistant that makes the system accessible and useful to everyday users.

In the future, SHEGA could be extended with SMS alerts, LPG gas detection, and mobile app integration. With real hardware deployment, it could serve as a practical safety companion for homes everywhere.