Culminating Activity Written Report

4BLAZERS GROUP

Elan, Kristiane Gwyn M.

Apolinario, Alaina B.

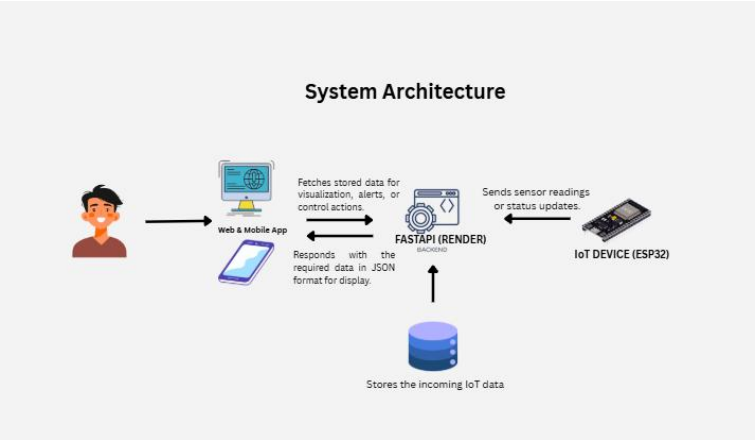
Lumay, Alyza Shane C.

Jimoya, Jude Adrianne B.

FlowTrack Energy: A Wide Electricity Consumption Monitoring Dashboard with IoT Integration

FlowTrack Energy is an advanced web/mobile application and monitoring dashboard designed to track real-time electricity consumption across various university buildings and campus-wide appliances. Through seamless integration with IoT devices, the app collects, stores, and visualizes data to give university administrators actionable insights into energy usage patterns.

This system aims to address the university's rising electricity costs, currently accounting for approximately 27 million pesos per month. By developing and deploying this system, the university will gain a powerful tool for managing energy usage, reducing wastage, and optimizing consumption to drive down costs.



Challenges

Wirings for IOT Connections

- Confusion in connecting to the correct GPIO pins on the ESP32.
- Risk of short circuits due to incorrect wiring.

Solutions: We refer to a labeled pinout chart from reliable sources & we use Breadboard & Dupont Wires to reduce soldering errors

Connecting the Iot Device to the App

- Difficulty configuring the ESP32 to send data to the FastAPI backend.
- Issues accessing the backend URL or IP from the mobile device.
- Problems handling JSON data sent between the ESP32 and FastAPI.
- Syncing data correctly between the backend and the React/React Native apps.

Solutions: We use HTTP POST requests with ArduinoHTTPClient or WiFiClientSecure. Additionally, we ensure Wi-Fi stability by either hardcoding credentials or using a configuration portal like WiFiManager. To facilitate communication, both the mobile device and ESP32 must be connected to the same Wi-Fi network. The ArduinoJson library is used for formatting and parsing JSON data. Since FastAPI expects dictionaries (dict), we validate the payload using Pydantic models.

Design Challenges

- Creating a consistent and user-friendly UI/UX between the mobile and web versions.

Solutions: To create a consistent and user-friendly UI/UX across web and mobile platforms, use component libraries like Material UI or TailwindCSS for web, and React Native Paper or NativeBase for mobile, while sharing design tokens such as colors and fonts.

Components or Materials Needed

- Missing basic components like sensors, jumper wires, resistors, or breadboard.

Solutions: To avoid delays caused by missing components, create a detailed Bill of Materials (BOM) listing all required parts with links and part numbers, use an ESP32 IoT starter kit with common sensors, maintain an inventory checklist before each session, and have fallback plans such as simulating sensor data in the code when hardware is unavailable.

Technologies & Tools Used

Iot Devices

- ESP32
- Split Core Current Transformer

Frontend Dashboard

- ReactJS & React Native

Backend API

- FASTAPI

Backend

- Sqlite

Hosting/Deployment

- Render
- Github
- Localhost

Tools Used

- ESP32
- Split Core Current Transformer
- Jumper Wire (Male to Female & Female to Female)
- Capacitor (10uf)
- Bread Board
- Voltage Sensor (ZMPT101B)
- Resistor 10k & 1k (1% tolerance)

Future Improvements or Features to Add.

In addition to the existing features, future improvements for the electricity consumption dashboard could include:

Energy Efficiency Recommendations: Provide suggestions for reducing energy usage per building or appliance based on patterns and benchmarks.

Real-Time Alerts and Notifications: Send automated alerts for unusual consumption spikes, outages, or system faults.