# Energy Monitoring

**INTRODUCTION**

This project is an **IoT-based energy monitoring system** that combines hardware (ESP32, PZEM-004T), backend (Node.js, PostgreSQL), frontend (HTML, CSS, JavaScript), and database technologies to monitor, store, and visualize energy data. The system enables real-time energy tracking, historical data analysis, and export functionalities.

**1. Purpose**

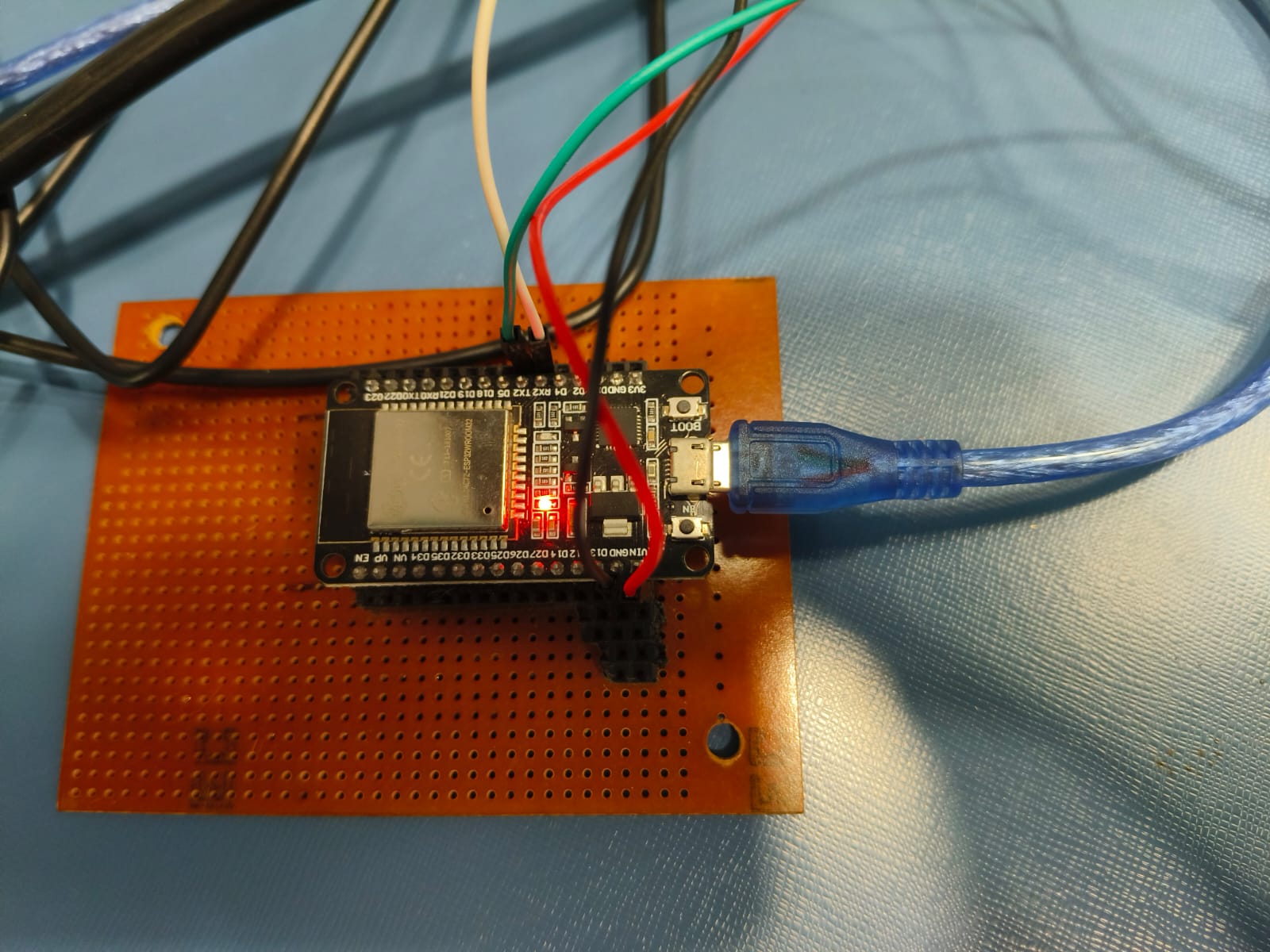
The project aims to:

1. Monitor electrical parameters (voltage, current, power, frequency, power factor, energy).
2. Store data for analysis and reporting.
3. Provide a user-friendly dashboard for real-time monitoring and historical data visualization.
4. Allow users to download data and configure the system.

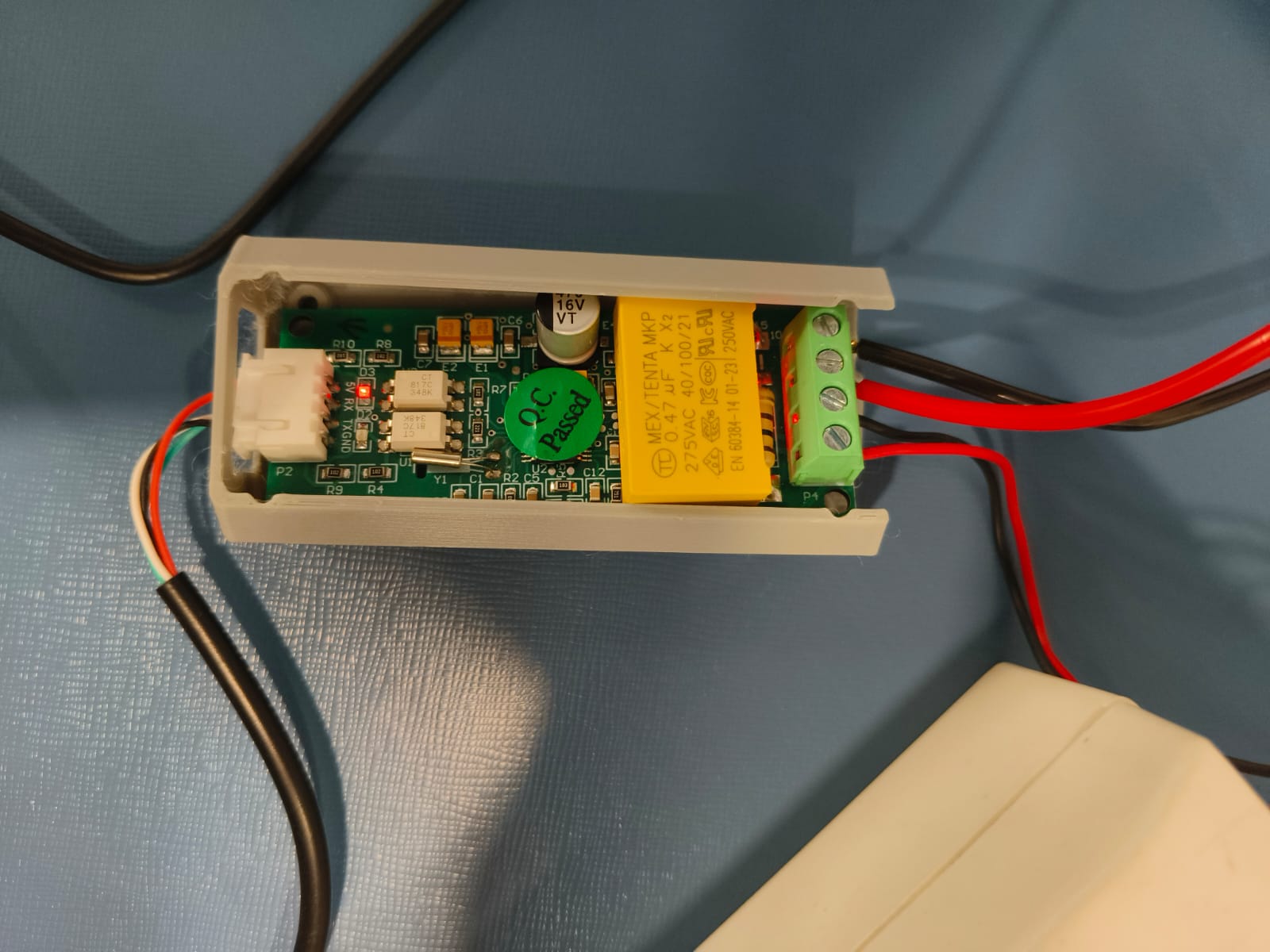
**2. Components**

#### ****Hardware****

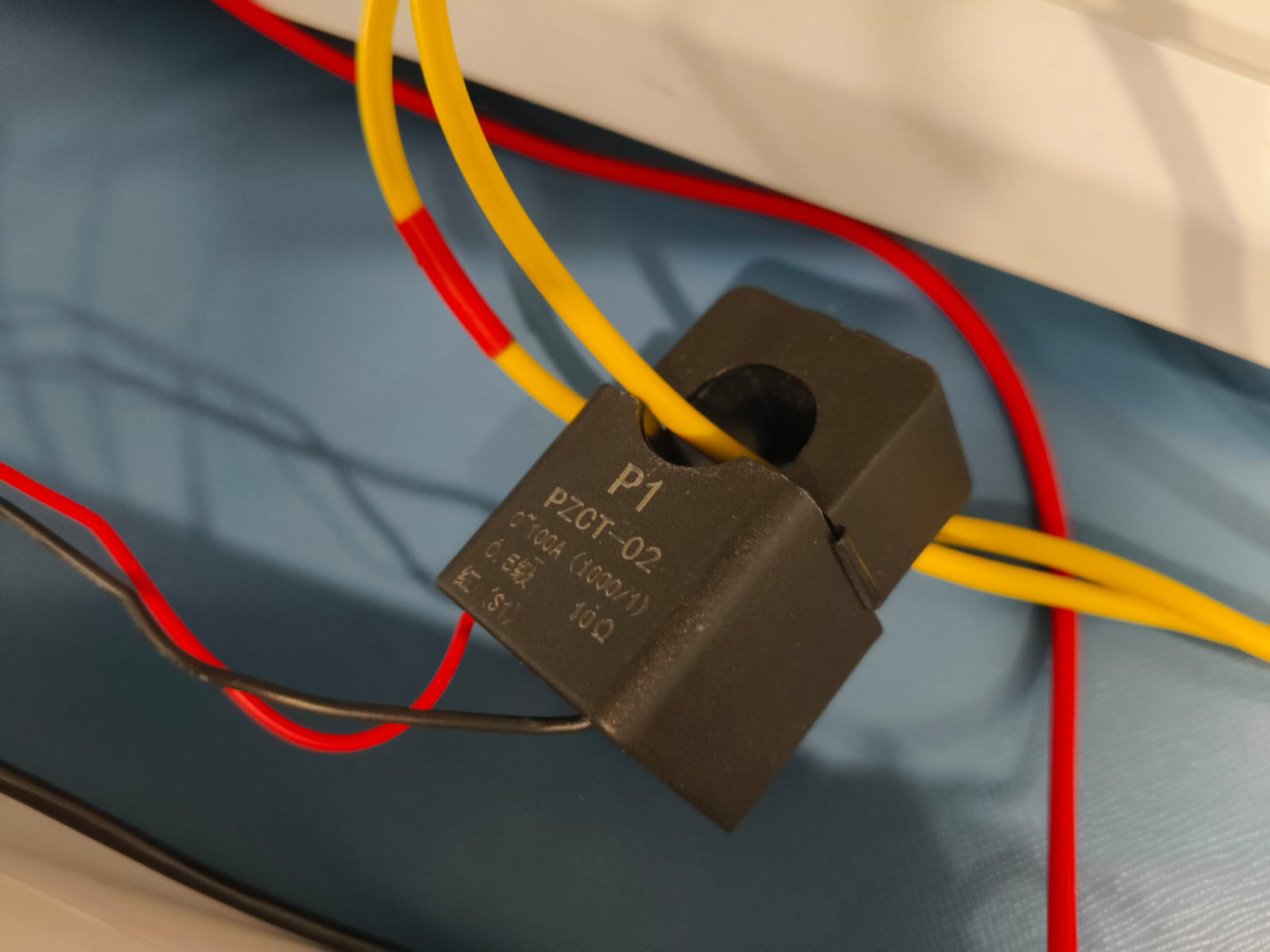
* **ESP32**:
  + Connects to the Wi-Fi network to transmit energy data.
  + Runs an HTTP server to serve live energy data in JSON format.



* **PZEM-004T**:
  + Measures energy parameters such as voltage, current, power, frequency, and power factor.
  + Connects to ESP32 via hardware serial communication.



#### ****Current Coil****

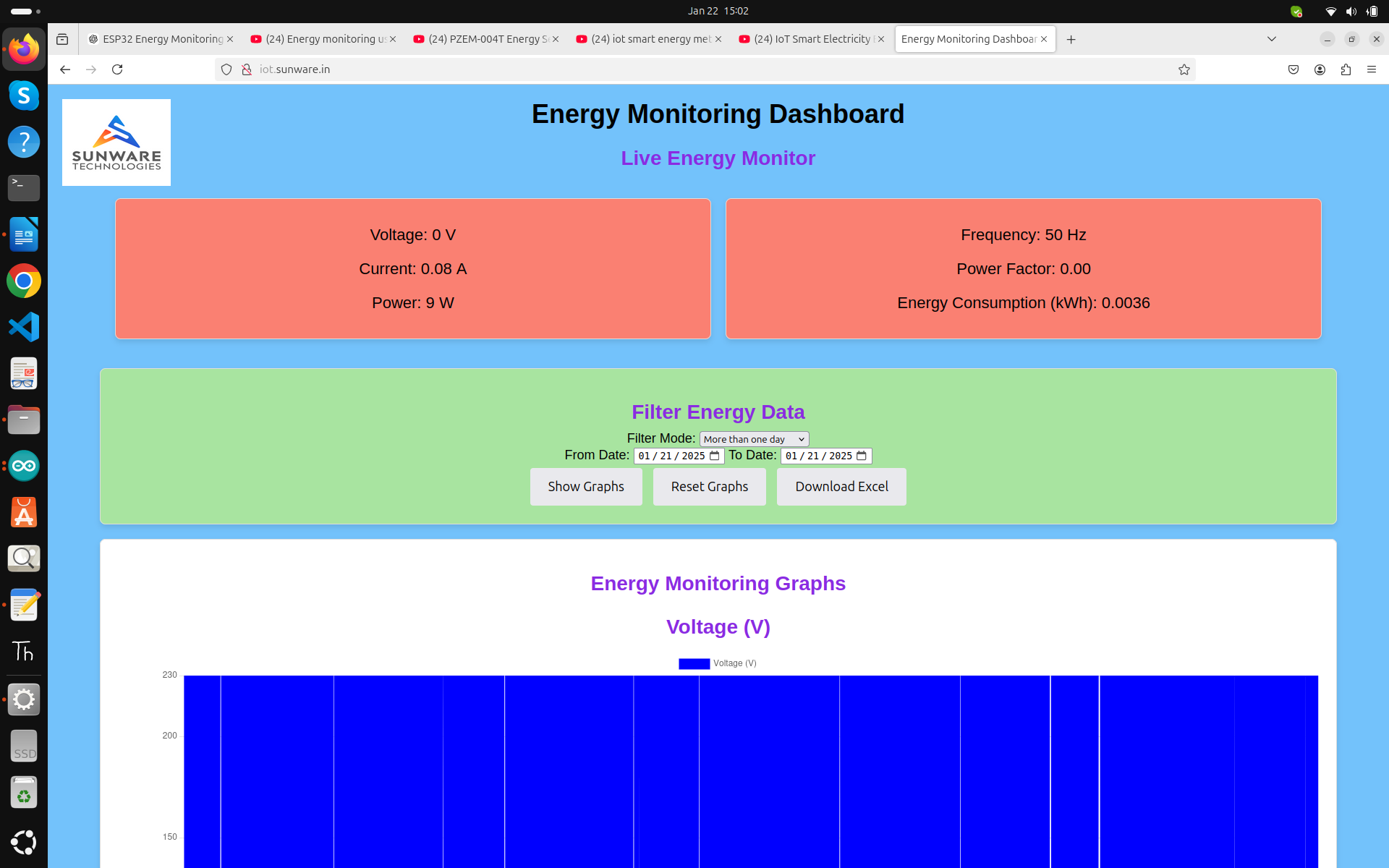


#### ****Software****

* **Node.js Backend**:
  + Handles API requests and responses.
  + Fetches live data from ESP32 and stores it in a PostgreSQL database.
  + Provides endpoints for retrieving filtered data, resetting Wi-Fi, and downloading data.
* **PostgreSQL Database**:
  + Stores energy monitoring data, including voltage, current, power, frequency, power factor, energy (kWh), and timestamps.
  + Allows querying of historical data based on filters.

#### ****Frontend Dashboard****

* Built with **HTML**, **CSS**, and **JavaScript**.
* Features a live data display, historical data filters, and graph visualization using Chart.js.
* Provides options to reset graphs, filter data, and download results.



### ****3. Functionality****

#### ****Data Monitoring****

1. **Live Data**:
   * The ESP32 collects energy data from the PZEM-004T sensor and serves it via an HTTP endpoint.
   * The data includes voltage, current, power, frequency, power factor, and total energy (kWh).
2. **Historical Data**:
   * The backend fetches historical data from PostgreSQL.
   * Users can filter data by date range or specific time intervals.

#### ****Visualization****

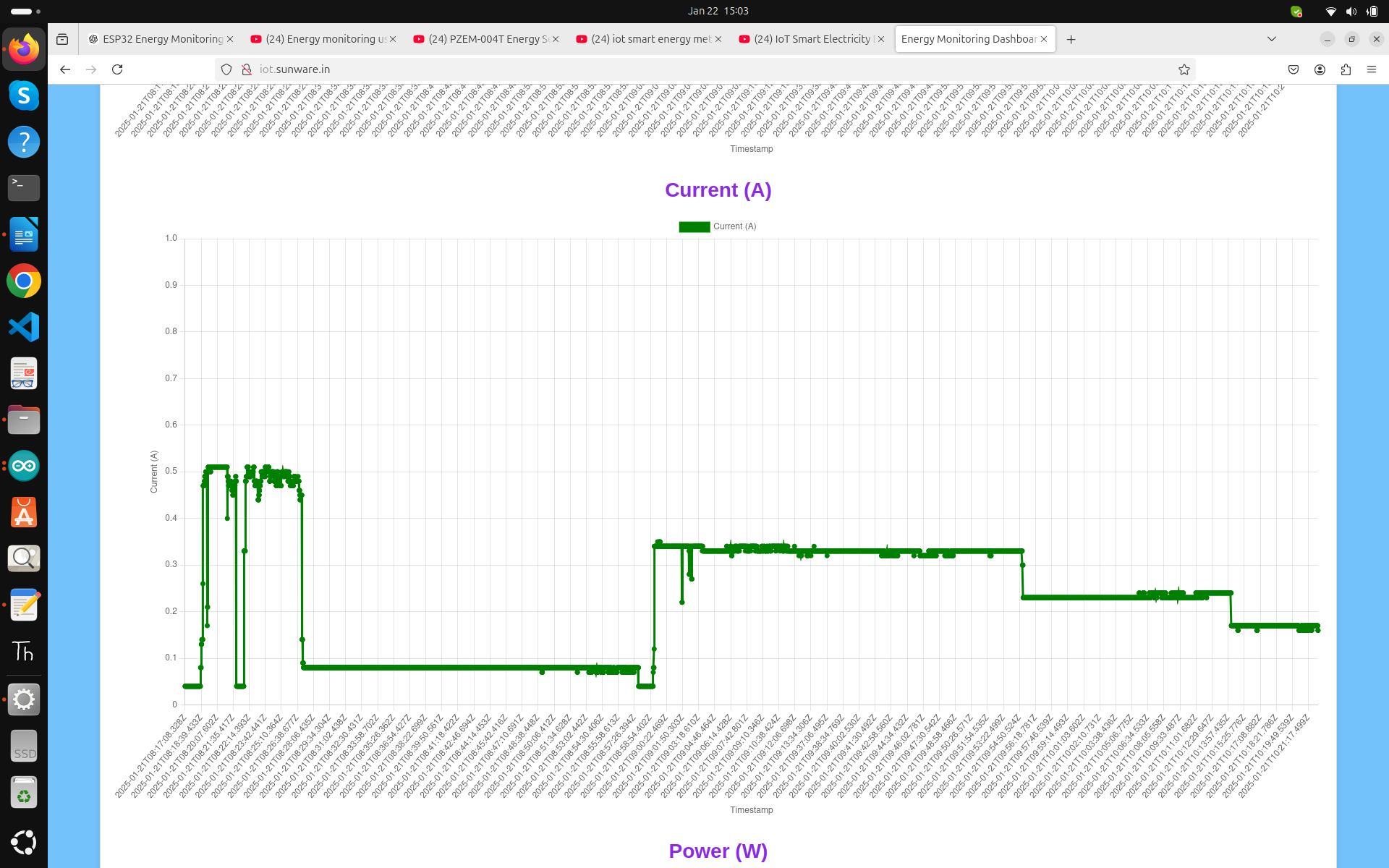
* The frontend renders graphs for voltage, current, power, power factor, and energy consumption.
* Dynamic filters allow users to adjust the data range.

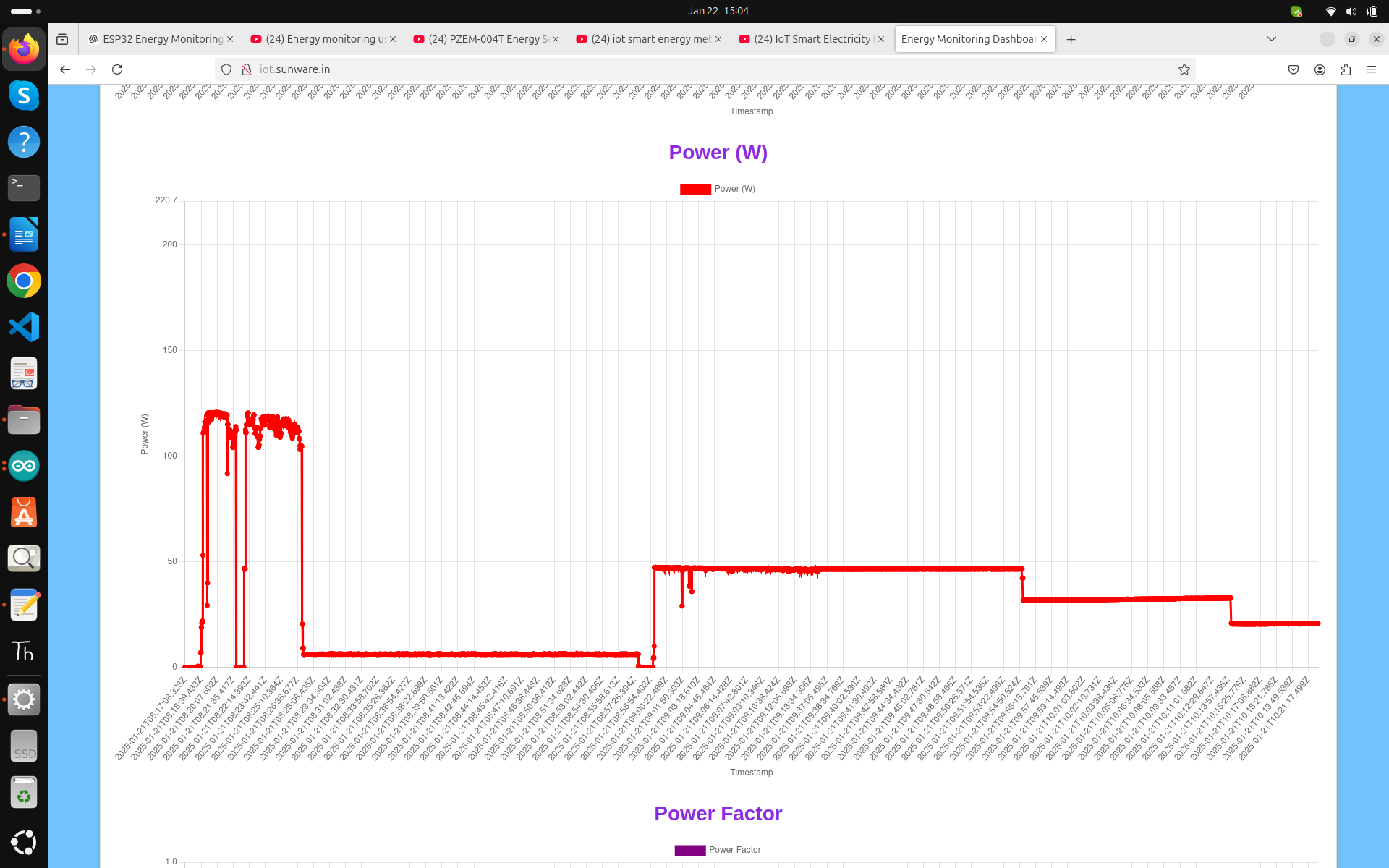
#### ****Data Export****

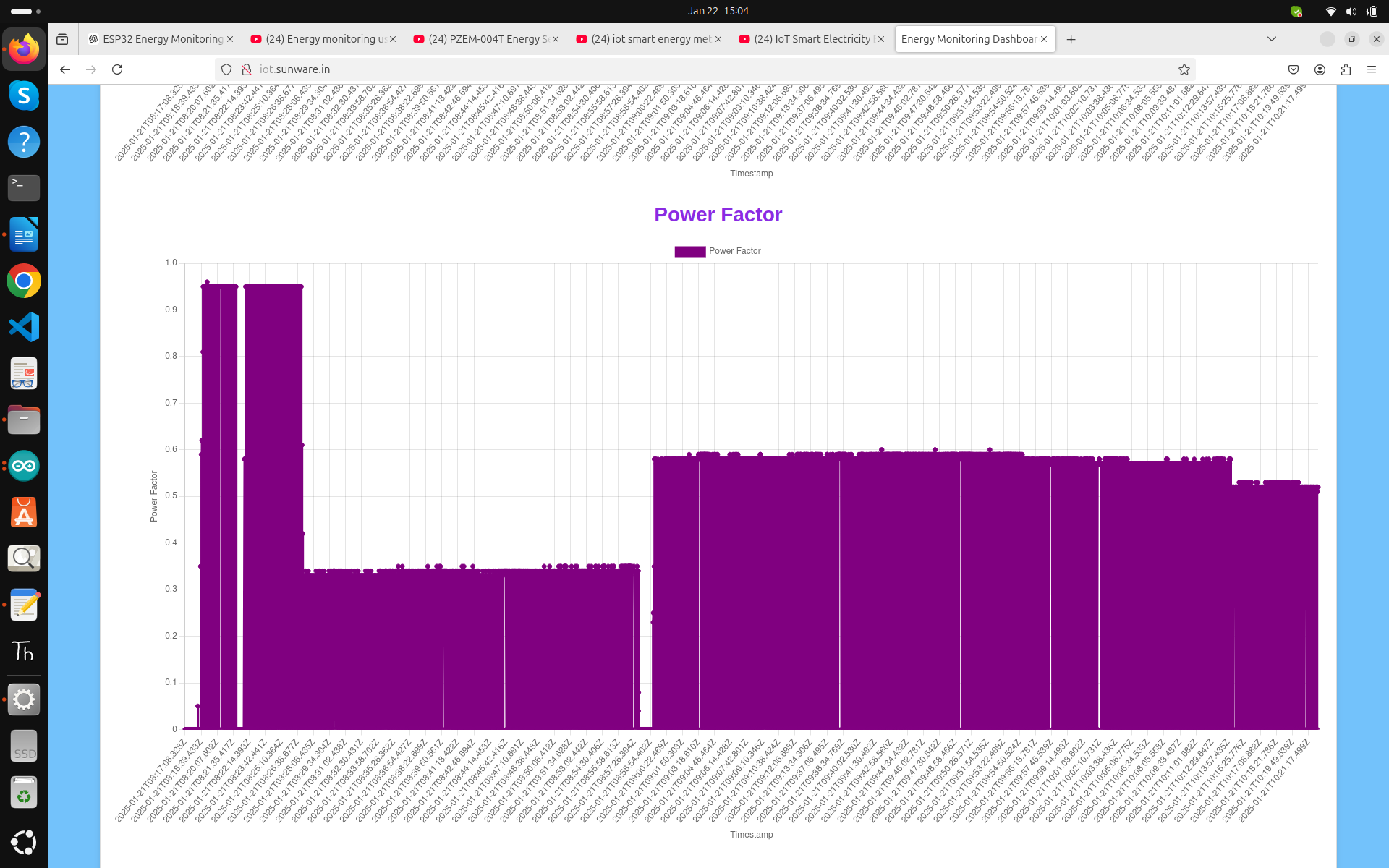
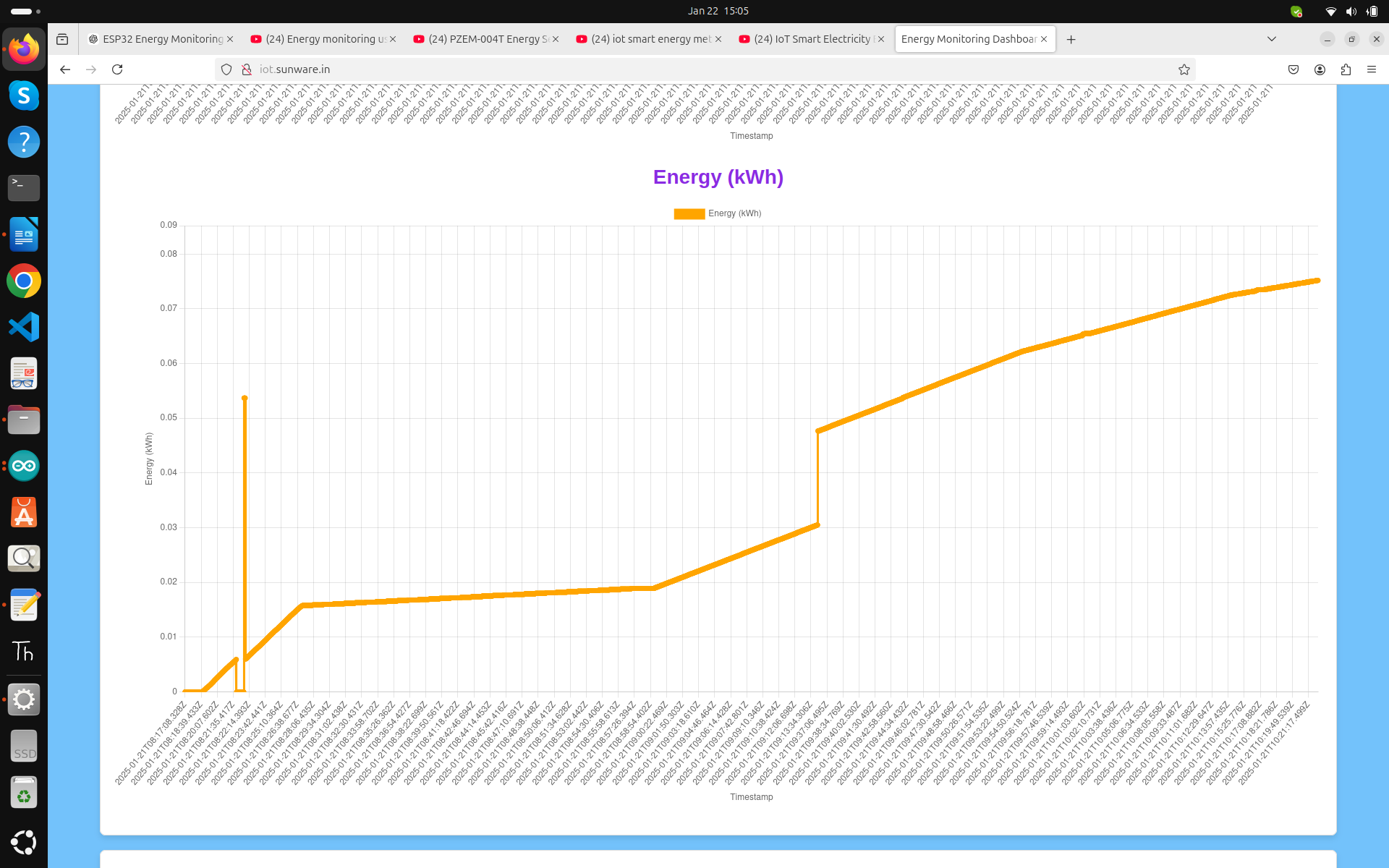
* Historical energy data can be downloaded as an Excel file for offline analysis.

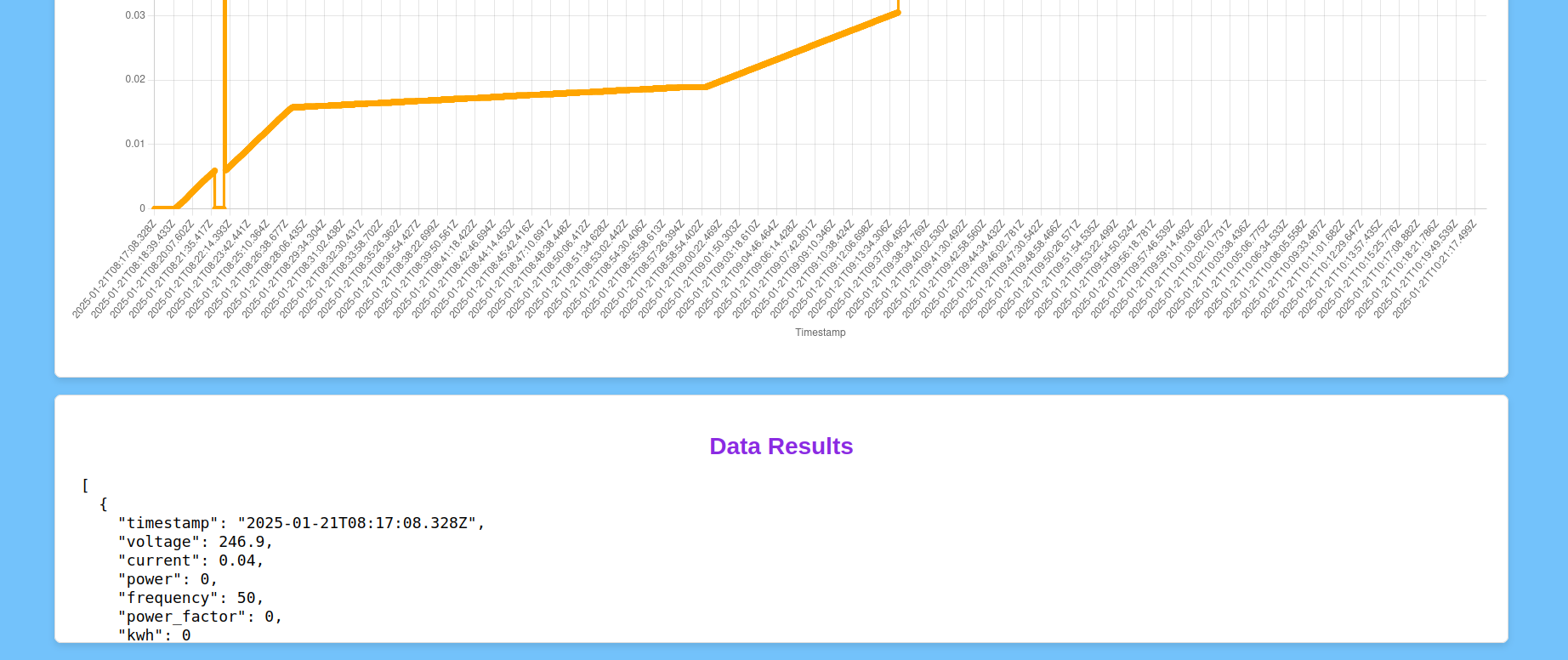
### 

## ****Energy Monitoring Graphs****



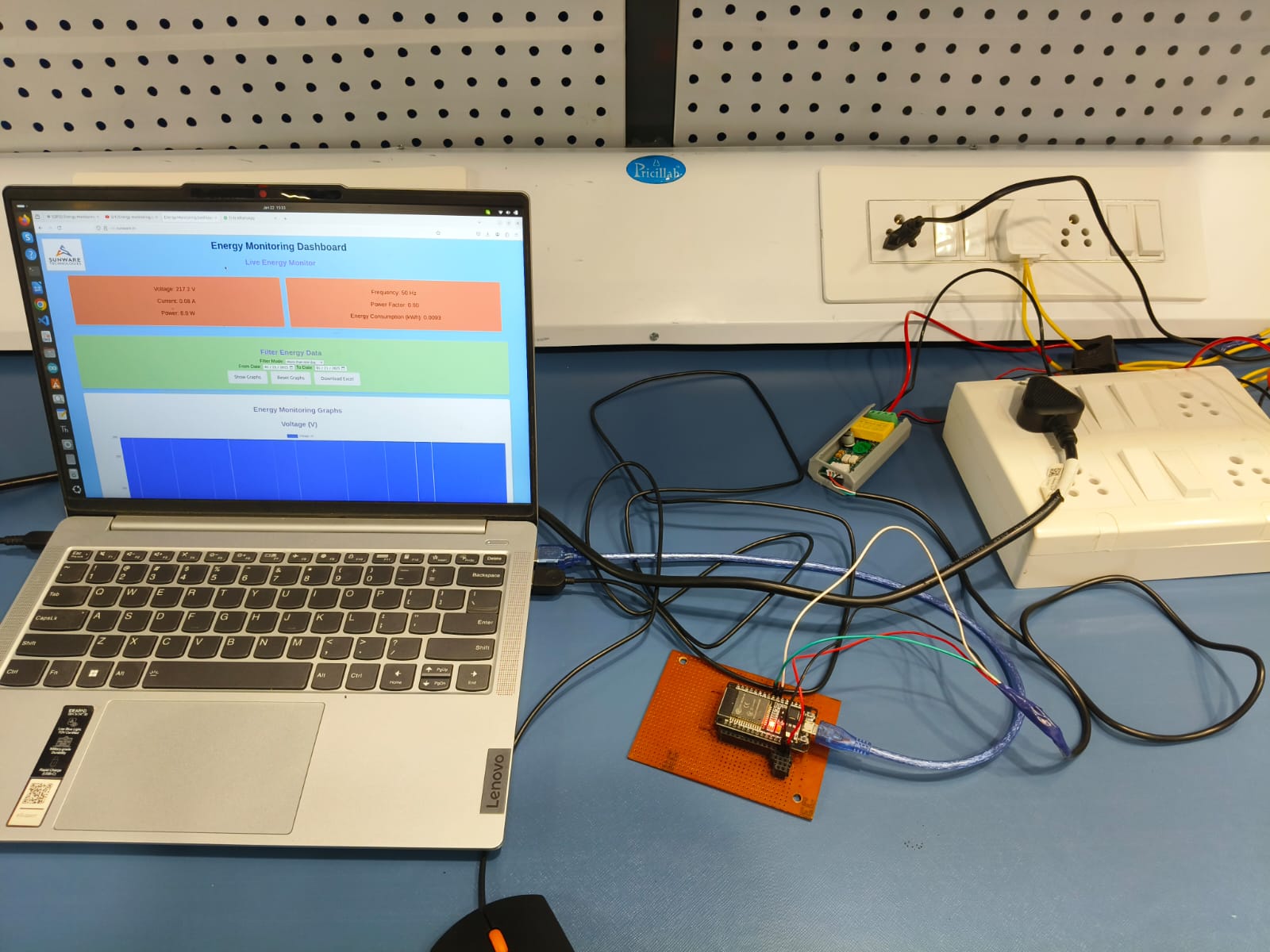




### ****4. Architecture****

1. **ESP32 (IoT Device)**:
   * Collects energy data.
   * Serves live data via an HTTP server.
   * Manages Wi-Fi credentials with WiFiManager.
2. **Backend (Node.js)**:
   * Acts as an API server.
   * Fetches live data from ESP32.
   * Stores and retrieves data from PostgreSQL.
3. **Frontend (Dashboard)**:
   * Displays real-time and historical energy data.
   * Provides graph-based visualization and interactive filters.
4. **Database (PostgreSQL)**:
   * Stores energy data for long-term analysis.



### ****5. Features****

1. **Real-Time Monitoring**:
   * Live data updates every second on the dashboard.
2. **Graph-Based Visualization**:
   * Displays energy trends with graphs for voltage, current, power, power factor, and energy (kWh).
3. **Data Filters**:
   * Users can filter data by date range or specific time intervals.
4. **Data Export**:
   * Historical data can be downloaded as an Excel file.
5. **Wi-Fi Management**:
   * Users can reset or update Wi-Fi credentials for the ESP32.
6. **Responsive Design**:
   * The dashboard is styled for a clean and user-friendly experience.

### ****6. Technical Implementation****

#### ****ESP32 Code****

* Configures a static IP and sets up Wi-Fi using WiFiManager.
* Communicates with the PZEM-004T sensor to collect energy data.
* Serves /live-data endpoint with JSON responses for voltage, current, power, frequency, power factor, and energy (kWh).

#### ****Backend API****

* **Technologies**: Node.js, Express, PostgreSQL.
* **Endpoints**:
  + /api/live-data: Fetches live energy data from ESP32 and stores it in the database.
  + /api/graph-data: Retrieves filtered historical data.
  + /api/reset-wifi and /api/save-wifi: Manages ESP32 Wi-Fi credentials.
  + /api/download-data: Generates an Excel file with energy data for a specified range.

#### ****Database Schema****

* **Table**: energy\_monitoring
* **Fields**:
  + voltage, current, power, frequency, power\_factor, kwh, timestamp.

#### ****Frontend****

* **HTML**: Structure for the energy monitoring dashboard.
* **CSS**: Provides styling for the dashboard, ensuring responsiveness and aesthetics.
* **JavaScript**:
  + Fetches live and historical data.
  + Updates graphs dynamically using Chart.js.
  + Handles user interactions like data filtering, Wi-Fi reset, and data export.

### ****7. Deployment****

1. **Frontend**:
   * Hosted on Nginx at /var/www/iot.sunware.in.
   * Accessible via http://iot.sunware.in.
2. **Backend**:
   * Runs on port 3000, proxied via Nginx.
3. **Database**:
   * PostgreSQL hosted locally with user authentication.

### ****8. Advantages****

1. **Real-Time Monitoring**:
   * Immediate insights into energy consumption and efficiency.
2. **Historical Analysis**:
   * Data storage allows for trend analysis and optimization.
3. **User-Friendly Dashboard**:
   * Intuitive interface for both technical and non-technical users.
4. **Customizable Filters**:
   * Flexible data querying for specific analysis needs.
5. **Portable and Scalable**:
   * Modular design enables scaling for multiple devices or locations.

### ****9. Potential Enhancements****

1. **Alert Mechanisms**:
   * Add alerts for abnormal energy usage.
2. **User Authentication**:
   * Secure access to the dashboard and API endpoints.
3. **Additional Sensors**:
   * Integrate more sensors for temperature, humidity, or device-specific energy monitoring.
4. **Cloud Integration**:
   * Push data to cloud services for remote monitoring and analytics.