Internet of Things Fundamentals

Subject Project

BS AI 6th Smester SP-25 (AIE-3079)

Date: 26-06-2025					
Project Title:					
Smart Home Energy Analysis and Control					
Group Name/no.:					
MindMesh					
Team Members:					
Members	Registration no	Name	Signature		
Member-1 (Leader)	22-NTU-CS-1368	Muneeb Ur Rehman			
Member-2	22-NTU-CS-1338	Adifa Jahangir			
Member-3	22-NTU-CS-1377	Swaiba Shahid			
Member-4					

Contributions in % of each Team Members for each component					
		Member-1	Member-2	Member-3	Member-4
Distribution Components		Muneeb Ur Rehman	Adifa Jahangir	Swaiba Shahid	
Coding	ESP32- coding	Major Arduino Programming 70%	Web Pages & Web Server setup 15%	Wi-Fi Connection & PZEMT Sensor Reading 15%	
	Python Coding	Anomaly Detection model	Energy Forecasting Model	Streamlit Dashboard	
UI De	esign		GUI - Website		
Data	base	Influxdb			
Clo Integi				Realtime Database & Grafana	
IoT Ga	teway	ESP32 S3			
Ed Proce	_	Local Decision & Calculation			
Docume	entation		Yes		
Presen Des				Yes	
Hard Circuit	ware	Yes			

To be filled by the evaluator

Team-Based Evaluation (60 Marks)

Criteria	Obtained Marks	Out of
System Design & Architecture		10
Hardware Integration & Circuit Setup		10
IoT Gateway and Cloud Communication		10
Working Prototype Demonstration		10
Performance & Reliability Testing		10
Presentation		10
Total (Team-Based)		60

Individual-Based Evaluation (40 Marks per Member)

	Member 1	Member 2	Member 3	Member 4
Criteria				
Understanding of the Project & Role	/10	/10	/10	/10
Code Contribution and Explanation	/10	/10	/10	/10
Q/A VIVA	/10	/10	/10	/10
Documentation/Reporting & Communication	/10	/10	/10	/10
Total (Individual-Based)	/40	/40	/40	/40
Total Overall (60+40)	/100	/100	/100	/100
Weightage Lab Grade (50)				

Table of Contents

1. Abstract / Executive Summary	5
3. Introduction	5
Background & Motivation	5
Problem Statement	5
Project Goals	5
4. Literature Review (Optional)	5
Relevant IoT/ESP32 Concepts	5
Similar Projects/Research	Error! Bookmark not defined.
5. Methodology / System Design	5
5.1 Hardware Components	5
5.2 Software Design	6
6. Implementation	10
Step-by-Step Setup	10
Code Snippets	10
Challenges & Solutions	10
7. Results & Discussion	11
Screenshots/Output	11
Performance Analysis	11
Comparison with Expected Outcomes	14
8. Testing & Validation / Limitations	16
Test Cases	16
Limitations	16
9. Conclusion & Future Work	16
Key Takeaways	16
Potential Improvements	16
10 References	17

Table Of Figures

Figure 1 System Component Diagram	7
Figure 1 System Component DiagramFigure 2 : Data Flow DiagramFigure 3 : Mark in the Flow Diagram	8
Figure 3 : ML Integration Flow Diagram	9
Figure 4 : Voltage Stats Diagram	11
Figure 5 : Power Stats Diagram	11
Figure 6 : Frequency Stats Diagram	
Figure 7 : Energy (Units) Stats Diagram	12
Figure 8 : Current Stats Diagram	13
Figure 9 : Database Table & Values Diagram	13
Figure 10 : Login PageFigure 11 : Home PageFigure 12 : Room Control Page	14
Figure 11 : Home Page	14
Figure 12 : Room Control Page	15
Figure 13 : Home Stats page	15
Figure 14 : Report Page	16

1. Abstract / Executive Summary

This project presents a Smart Energy Monitoring and Control System using ESP32, designed to monitor, analyze, and control household electrical appliances. The system measures real-time electrical parameters (voltage, current, power, energy, frequency, power factor) using PZEM-004T and ACS712 sensors, and provides remote control and scheduling of appliances via a web dashboard. Data is logged to an InfluxDB time-series database, enabling advanced analytics and visualization (potentially via Grafana). The system aims to optimize energy usage, detect anomalies, and enhance user convenience.

3. Introduction

Background & Motivation

With the increasing demand for energy efficiency and smart home automation, real-time monitoring and control of electrical appliances have become essential. This project addresses the need for a cost-effective, scalable, and user-friendly solution for home energy management.

Problem Statement

Traditional energy meters lack real-time feedback and remote-control capabilities, leading to inefficient energy usage and higher costs.

Project Goals

- Real-time monitoring of voltage, current, power, energy, frequency, and power factor.
- Remote control and scheduling of appliances.
- Data logging for analytics and reporting.
- User-friendly web interface for monitoring and control.

4. Literature Review (Optional)

Relevant IoT/ESP32 Concepts

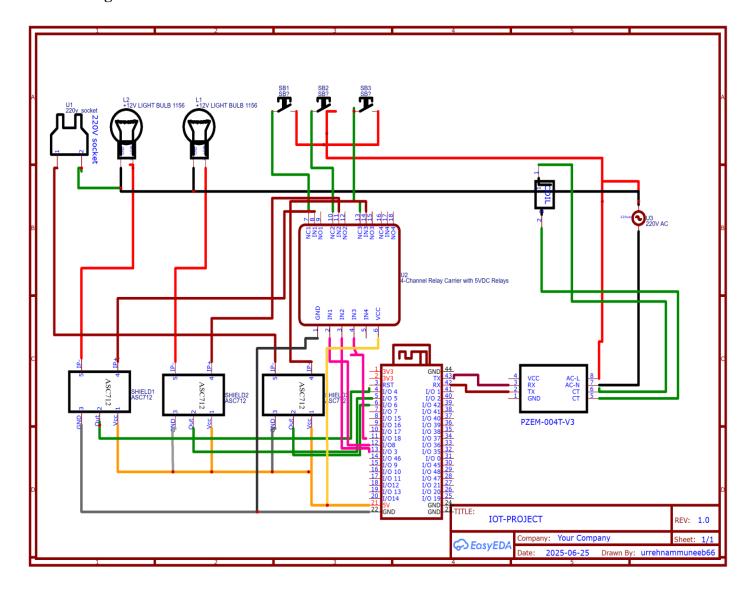
- ESP32 as a Wi-Fi-enabled microcontroller for IoT applications.
- Use of PZEM-004T for AC parameter measurement.
- ACS712 for current sensing.
- Web server and WebSocket communication for real-time updates.

5. Methodology / System Design

5.1 Hardware Components

- **ESP32**: Main controller with Wi-Fi capability.
- PZEM-004T v3.0: Measures voltage, current, power, energy, frequency, and power factor.
- ACS712: Measures current for individual appliances.
- **Relays**: Controls appliances (Light-1-Main, Light-2, Socket-1).
- OLED Display: Shows system status and IP address.
- Other: Resistors, Ac-wires, Jumper wires, breadboard/PCB.
- Switchboard: Built-in 4 button and 2 socket Switchboard.

Circuit Diagram



5.2 Software Design

System Architecture

- ESP32 runs a web server and WebSocket for real-time UI updates.
- Sensor data is read, processed, and sent to both the web dashboard and InfluxDB.
- Appliance control via relays, with scheduling and safety (low voltage) logic.
- Added anomaly detection and Energy Forcasting using ML.

Libraries/Tools Used

- Arduino IDE
- ESPAsyncWebServer, ArduinoJson, PZEM004Tv30, Adafruit_SSD1306, EEPROM, NTPClient, WiFi, HTTPClient

5.3 System Diagrams

Component Diagram:

Smart Energy Monitoring - System Components

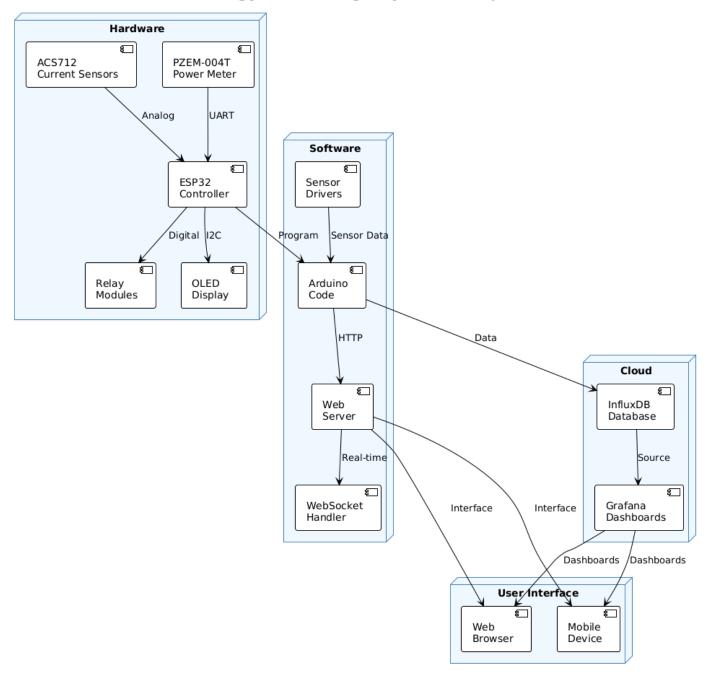


Figure 1 System Component Diagram

Flow Diagram:

Smart Energy Monitoring - Detailed Data Flow Process

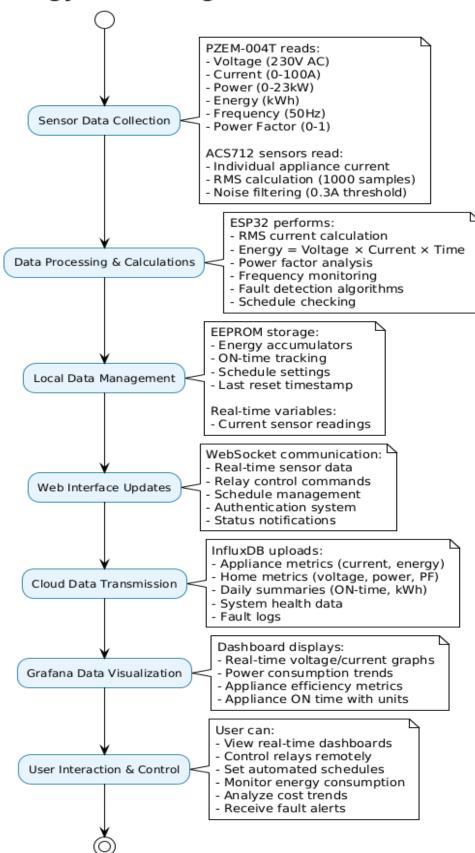


Figure 2: Data Flow Diagram

ML Flow Diagram:

ML Integration - Complete Workflow Initialize ML System Load configurations Setup InfluxDB connection Data Collection Phase fetch_daily_energy() Last 30 days of energy consumption Fetch Daily Energy Data fetch_training_anomaly_data() Last 7 days of sensor data Fetch Training Data for Anomaly Detection Validate Data Quality Yes Data Available? No Handle missing values Convert timestamps Calculate derived features Log Data Unavailable Process and Clean Data Ŏ Model Training Phase train_energy_model() Prophet time-series model Daily energy consumption Train Energy Forecasting Model Save Energy Model joblib.dump(model, "energymodel.pkl") train_anomaly_model() LocalOutlierFactor Unsupervised learning Train Anomaly Detection Model Save Anomaly Model joblib.dump(model, "anomaliemodel.pkl") Real-time Processing Phase fetch_last_10min_home_metrics() Voltage, current, power, power factor expected_power = voltage × current × powerFactor power_error = |expected_power - actual_power| joblib.load("energymodel.pkl") joblib.load("anomaliemodel.pkl") Load Trained Models forecast_energy() 7-day energy prediction Confidence intervals Generate Energy Forecast detect_recent_anomalies() \(\begin{square}{c} \ Anomaly score calculation \\ Outlier detection \end{square} Detect Anomalies Web Interface Phase Setup dashboard Configure tabs Initialize Streamlit App Live anomaly detection Real-time monitoring Display Dashboard Tab \downarrow Energy predictions Trend analysis Display Forecasting Tab Historical data Interactive charts Display Analysis Tab User Interaction Phase, Home (All Combined) Light 1 (Main) Light 2 Socket User Selects Appliance Trigger Anomaly Detection Red indicators Severity scores Detailed analysis Green indicators System health Display Normal Status Display Anomaly Alerts 7-day predictions Min/Max estimates Trend visualization Generate Energy Forecast Time-series charts Anomaly patterns Performance metrics Show Historical Analysis Continuous Monitoring Schedule Regular Updates Retrain Models Periodically Live charts Real-time metrics Update Visualizations

Figure 3: ML Integration Flow Diagram

Pseudocode (Example)

```
Apply to Smart_energr...

Loop:

Read sensor data (PZEM, ACS712)

Update energy and on-time accumulators

Check for scheduled relay toggles

Send data to InfluxDB

Update web dashboard via WebSocket

Save data to EEPROM periodically
```

6. Implementation

Step-by-Step Setup

- 1. **Wiring**: Connect PZEM-004T to ESP32 (Serial2), ACS712 sensors to analog pins, relays to digital pins, OLED to I2C.
- 2. Configuration: Set Wi-Fi credentials, InfluxDB endpoint, and other constants in the code.
- 3. Upload Code: Use Arduino IDE to upload the .ino file to ESP32.
- 4. **Web Dashboard**: Access via ESP32's IP address on the local network.

Code Snippets

```
• Wi-Fi Setup
```

```
Apply to Smart_energr...
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) { delay(500); }

• Sensor Reading
    Apply to Smart_energr...
        float voltage = pzem.voltage();
        float current = pzem.current();

• WebSocket Event Handling
        Apply to Smart_energr...
        ws.onEvent(onWsEvent);

• InfluxDB Data Push
        Apply to Smart_energr...
        influxSender.sendHomeMetrics(voltage, current, power, energy, frequency, pf);
```

Challenges & Solutions

- Wi-Fi Instability: Implemented periodic reconnection logic.
- Sensor Noise: Calibrated ACS712 offsets and used RMS calculation.
- **EEPROM Wear**: Limited write frequency to every 30 seconds.

7. Results & Discussion

Screenshots/Output Grafana & influxdb



Figure 4: Voltage Stats Diagram



Figure 5: Power Stats Diagram



Figure 6 : Frequency Stats Diagram

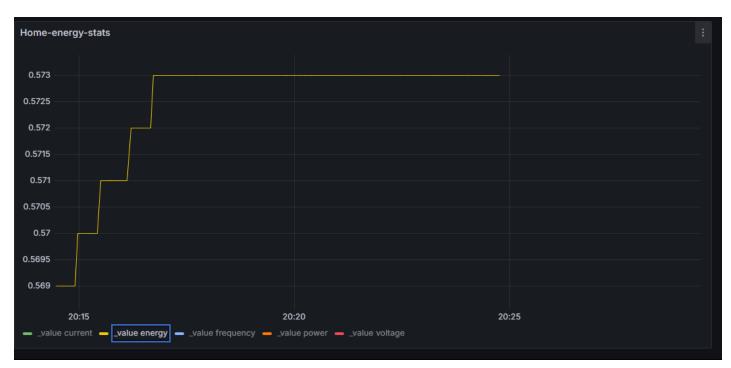


Figure 7 : Energy (Units) Stats Diagram



Figure 8: Current Stats Diagram

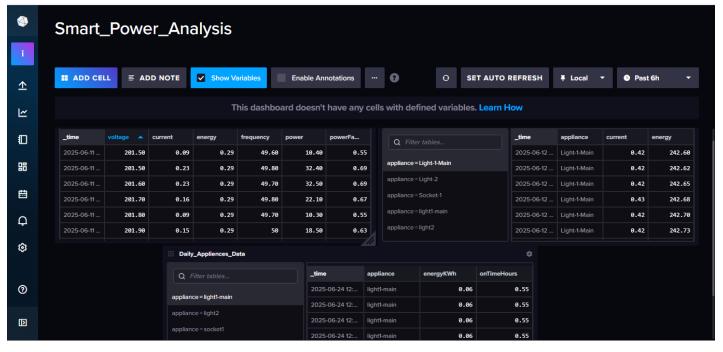


Figure 9: Database Table & Values Diagram

Performance Analysis

- Accuracy: Sensor readings are consistent with commercial meters.
- Latency: Real-time updates via WebSocket (<1s delay).
- Reliability: System recovers from Wi-Fi drops and power cycles.

Comparison with Expected Outcomes

• All core functionalities (monitoring, control, scheduling, reporting) are achieved.

7.1 User Interface

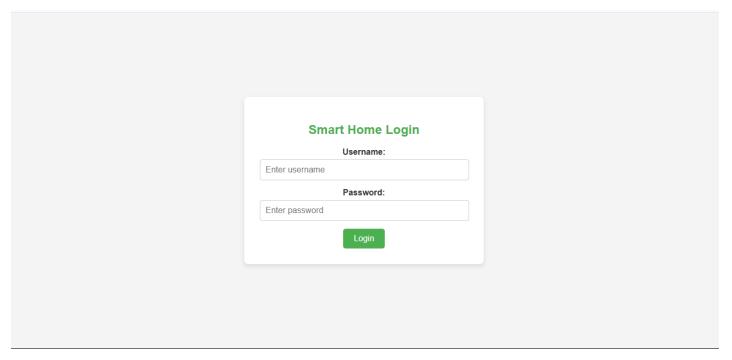


Figure 10: Login Page

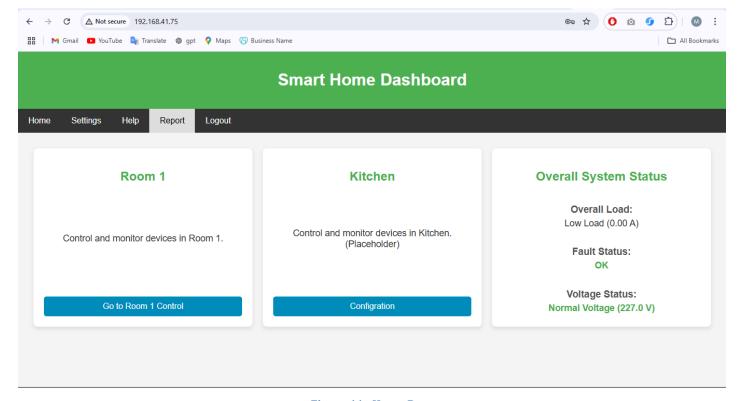


Figure 11: Home Page

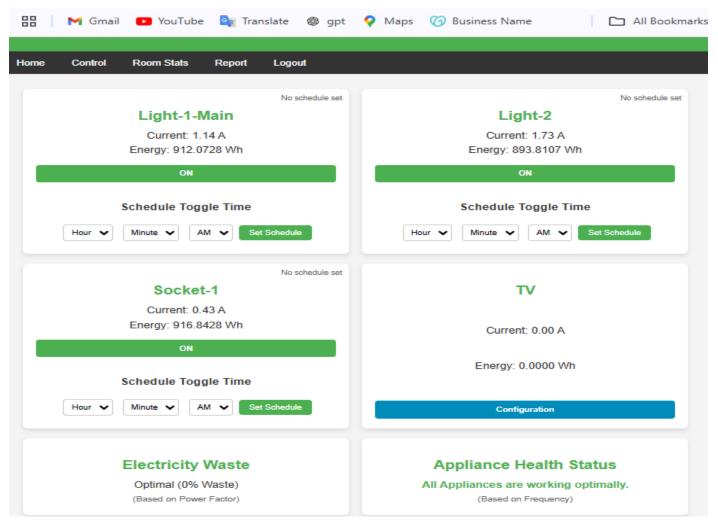


Figure 12: Room Control Page

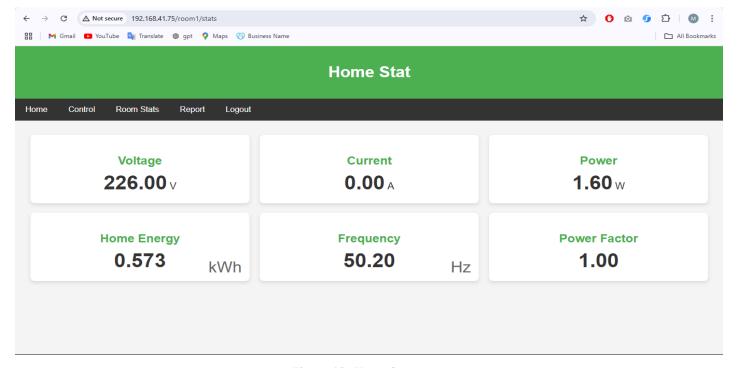


Figure 13: Home Stats page

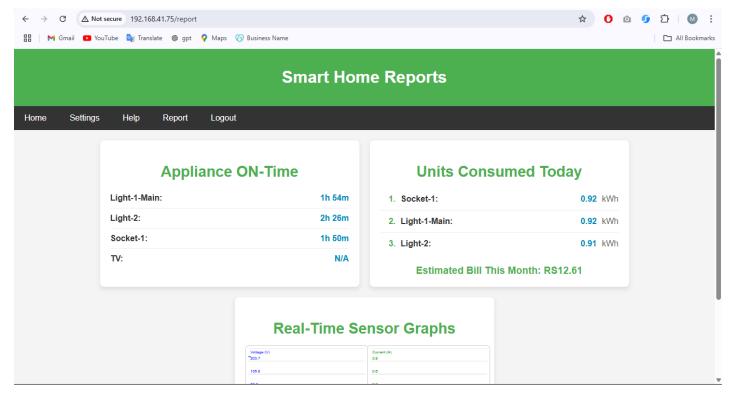


Figure 14: Report Page

8. Testing & Validation / Limitations

Test Cases

- Appliance ON/OFF control via dashboard.
- Schedule-based relay toggling.
- Data logging and retrieval from InfluxDB.

Limitations

- No direct cloud (Grafana Cloud) integration in firmware; requires external Grafana setup.
- Only three appliances monitored; scalable with hardware changes.
- Security: Basic authentication on web dashboard.

9. Conclusion & Future Work

Key Takeaways

- Demonstrated a robust, real-time smart energy monitoring and control system.
- Achieved reliable data logging and user-friendly control.

Potential Improvements

- Direct integration with Grafana Cloud or other cloud analytics platforms.
- Expand to more appliances and add mobile app support.
- Enhance security (OAuth, HTTPS).

10. References

- ESP32 Documentation
- PZEM-004T Datasheet
- InfluxDB Documentation
- Arduino Libraries

11. Link

Project GitHub link:

https://github.com/Muneeb-0/IoT-Project-Smart-Power-Monitoring-and-Control-System

Presentation link:

https://www.canva.com/design/DAGraLzGQ0o/Md5quqQ_RNv8zjO0zP9SMA/view?utm_content=DAGraLzGQ0o&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlId=h4a8f7812a0

Project Video Link:

https://drive.google.com/file/d/1rsZ98TMQz7kzg94lnAF9xFH8vMDiqLJo/view?usp=sharing