Technical Report – <u>Team Fusion</u>

Real-time IoT Energy Management System with Diagnostics and Reinforcement Learning

Team Members: Tanay Shah, Anirudhha Dalal, Dhup Thumbadiya
Date: October 2025

Github Link - https://github.com/ChikkiPikki/VidyutAI_team_Fusion

---

1. Introduction

The growing integration of **renewable energy sources (solar, wind)**, **battery energy storage systems (BESS)**, and **electric vehicle (EV) charging infrastructure** demands intelligent, data-driven control systems. Conventional SCADA-based systems, although robust, lack the flexibility and learning capability required for dynamic, distributed microgrids.

This project presents a **cloud-based Energy Management System (EMS)** capable of **real-time IoT data ingestion**, **scalable time-series processing**, **intelligent diagnostics**, and **Reinforcement Learning (RL)-based energy dispatch optimization**.

The system is designed with the following goals:

- Achieve **real-time situational awareness** across heterogeneous energy subsystems.
- Provide **data-driven diagnostics and predictive maintenance insights** using continuous sensor analytics.
- Enable **self-learning energy scheduling** through Deep Reinforcement Learning to reduce operational cost and carbon footprint.
- Utilize **industry-grade visualization** via Grafana for scalable monitoring and decision support.

The project demonstrates an **end-to-end architecture** combining IoT, data engineering, analytics, and AI — ready for industrial deployment.

---

2. Problem Statement

Modern distributed energy systems face five major pain points:

1. **Fragmented Data Ecosystem:**
   Energy assets (solar, battery, EV chargers, inverters) operate on isolated control systems, generating unstandardized telemetry data.
   → Result: delayed decision-making and lack of unified monitoring.
2. **Scalability & Data Velocity Issues:**
   Traditional relational databases cannot handle high-frequency sensor streams (1,000+ sensors @ 1s interval).
   → Need for a time-series database optimized for write-heavy IoT workloads.
3. **Reactive Maintenance & Fault Diagnosis:**
   Current systems rely on post-fault logs instead of proactive health indices.
   → Leads to equipment degradation and energy loss.
4. **Suboptimal Energy Dispatch:**
   Static rule-based control (e.g., fixed battery schedules) ignores real-time variations in solar output, demand, and tariff.
   → Results in economic inefficiency and battery wear.
5. **Visualization & Human Bottleneck:**
   Custom dashboards require coding and are not scalable.
   → Non-technical users struggle to interpret raw sensor data.

---

3. Objectives

The primary objectives of the project are:

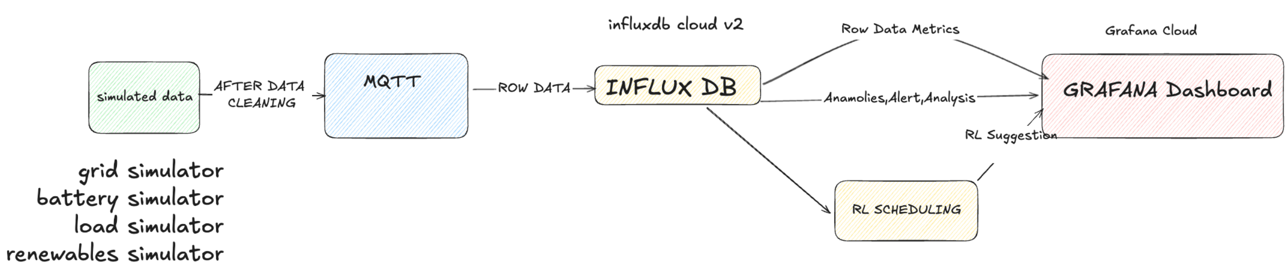
1. Build a **cloud-enabled data ingestion and storage pipeline** for real-time IoT data.
2. Design a **Diagnostics and Alert Engine** that detects anomalies and generates meaningful, actionable insights.
3. Implement a **Reinforcement Learning (RL) agent** for optimal energy dispatch and scheduling.
4. Provide an **industry-grade visualization layer** using Grafana dashboards.
5. Ensure **scalability, modularity, and real-time performance** across all components.

---

4. System Overview

The **Smart EMS** consists of five core layers, each performing a specific role:

1. **IoT Sensor Layer:**
   Collects real-time electrical and environmental data (V, I, Power, SoC, SoH, Temp, Frequency) from solar arrays, BESS, EV chargers, and loads.
2. **Communication Layer:**
   Utilizes **MQTT protocol** for efficient message transmission with QoS guarantees.
   MQTT Broker handles multiple topics per asset — enabling decoupled, asynchronous data flow.
3. **Data Storage & Analytics Layer:**
   Data is processed and stored in **InfluxDB**, optimized for high write throughput and time-range queries.
   This layer supports retention policies, continuous queries, and downsampling.
4. **Intelligent Diagnostics Engine:**
   Processes data streams for anomalies using **statistical thresholds, time-series models**, and **LLM reasoning** to classify faults and provide root cause explanations.
5. **Reinforcement Learning Agent:**
   Uses **Deep Q-Learning** for adaptive scheduling — optimizing between solar usage, battery cycling, and grid imports.
6. **Visualization & Control Layer (Grafana):**
   Displays real-time metrics, health status, alerts, and RL recommendations with customizable panels.



5. Methodology

5.1 IoT Data Ingestion

- Each energy subsystem publishes data every second to the **MQTT Broker (Eclipse Mosquitto)** hosted on a cloud VM.
- The broker streams data into a backend collector written in **Python**, which validates and transforms messages into InfluxDB line protocol.
- Example:
- `measurement=solar_data,tag=panel_1 voltage=230,current=5.2,power=1196,time=timestamp`
- The ingestion pipeline supports **1,000+ concurrent sensor feeds** with batch writes to minimize latency.

5.2 Time-Series Data Management

- **InfluxDB 3.0** is used for efficient time-series management.
- Data is bucketed by measurement (solar, battery, EV, load) with appropriate retention policies.
- Queries are written in **Flux language**, enabling:
  - Temporal aggregation (mean, max, min)
  - Time-window anomaly analysis
  - Real-time alert condition checks
- **Grafana** directly connects to InfluxDB via a data source plugin for instant visualization.

5.3 Diagnostics Engine

- Designed as a modular microservice analyzing all incoming streams:
  - Performs **Z-score & moving average filtering** for anomaly detection.
  - Calculates **Health Indices (HI)** for each subsystem:

$$HI = 100 - \alpha_1 \times Degradation - \alpha_2 \times FaultRate - \alpha_3 \times TemperatureDeviation$$

  - Employs **rule-based expert logic** (e.g., SoC < 10% + Voltage < threshold → "Battery Critical").
  - Integrates a **LLM reasoning model** to convert numeric findings into natural-language recommendations.
  - Outputs are written back into InfluxDB for dashboard rendering.

5.4 Reinforcement Learning Scheduler

- The RL problem is formulated as:
  - **State (s):** SoC, solar power, load, grid tariff, EV demand, temperature.
  - **Actions (a):** Charge/Discharge battery, Grid import/export, Adjust EV charging.
  - **Reward (r):**

$$r = -(C_{grid} \times P_{import}) + (C_{solar} \times P_{solar}) - (Penalty_{SoC} + Penalty_{LoadUnmet})$$

- **Algorithm:** Deep Q-Network (DQN) with ε-greedy exploration.
- **Training:** Performed on simulated datasets generated from the physics-based environment.
- The trained agent outputs optimal dispatch decisions every 5 minutes and updates Grafana via backend APIs.

Reward Function (Simplified):

$$\text{Reward} = -\text{Net Cost} + \text{Bonus}$$

Net Cost Components: Energy Cost + Emission Cost + Battery Degradation + Unmet Demand Penalty – Revenue

Bonus Components: Renewable utilization, battery management, healthy SoC, smart exports

States (Input to RL Agent)	Actions (Decisions by RL Agent)	Reward Key Points
`time_step` – Current simulation step (15-min)	0 – Idle: Maintain balance with renewables + grid	Minimize net cost, ensure grid balance
`renewable_gen` – Current solar/wind output	1 – Export: Send surplus renewable energy to grid	Reward for renewable utilization, revenue from export
`battery_soc` – Battery State of Charge (0–1)	2 – Charge: Store energy in battery from renewables	Penalize overcharging, reward optimal SoC
`battery_soh` – Battery State of Health (0–1)	3 – Discharge: Use battery to meet load demand	Penalize degradation, reward healthy energy dispatch
`ev_demand` – Current EV charging demand	4 – Grid Buy: Purchase electricity to meet demand	Penalize high energy cost, unmet demand penalty
`load_demand` – Current microgrid load	5 – Grid Sell: Sell battery energy to grid if profitable	Reward revenue, penalize selling when battery low
`grid_price_buy` – Cost to buy electricity	6 – EV+: Aggressively charge EVs to meet demand	Balance EV demand vs cost & battery health
`grid_price_sell` – Revenue from grid sale	7 – EV-: Slow EV charging to save energy	Avoid unnecessary battery usage, save cost
`ev_rul` – Remaining Useful Life of EV batteries	–	Penalize actions that reduce EV lifetime
`emission_factor` – CO_2 per kWh	–	Penalize high emission usage
`predicted_demand_next` – Forecasted load	–	Reward proactive demand satisfaction
`predicted_renewable_next` – Forecasted renewable gen	–	Reward renewable utilization

5.5 Visualization & Alerts

- **Grafana Dashboard** serves as the unified monitoring platform.
- Features:
    - Real-time charts for each subsystem (solar, battery, EV, grid).
    - Health & Diagnostics panels showing subsystem HI and fault messages.
    - RL Recommendation panel showing suggested dispatch vs actual dispatch.

- Alert rules with **Prometheus Alertmanager**-style triggers integrated via Influx queries.
- Dashboards are **auto-scalable** — new sensors appear dynamically using template variables.

---

6. Simulation Environment

- A **custom physics-based simulator** was developed in Python to emulate realistic energy system behavior.
- Each component follows simplified but accurate mathematical models:
  - **Solar PV:**

$$P_{solar} = \eta_{panel} \times G \times A \times (1 - 0.005(T - 25))$$

  - **Battery Dynamics:**
    SoC(t+1) = SoC(t) + (I_{charge}/C_{nom}) - (I_{discharge}/C_{nom})
  - **EV Chargers:**
    Variable charging loads modeled by probability distributions.
  - **Grid:**
    Includes tariff-based dynamic pricing and voltage/frequency variations.
- Fault injection is supported for:
  - Solar soiling or shading.
  - Battery overheating or rapid degradation.
  - Grid voltage spikes and outages.
- The simulator publishes all values to MQTT in real-time, ensuring end-to-end testing.
- Over **1000+ virtual sensors** are simulated to emulate a realistic microgrid scenario.

---

7. System Architecture

The overall system architecture is modular and cloud-enabled, ensuring scalability and fault tolerance.

Components:

1. **IoT Sensor Network** – Data acquisition layer using MQTT.
2. **InfluxDB Backend** – Time-series data management and historical logging.
3. **Diagnostics and Health Engine** – Fault detection and predictive analytics.
4. **RL Optimization Engine** – Adaptive energy dispatch and control decisions.
5. **Visualization Layer (Grafana)** – Real-time dashboard with dynamic alerts and reports.

(Diagram Placeholder: System Architecture Block Diagram)

---

8. Validation and Testing

The system underwent comprehensive validation:

- **Data Pipeline Validation:** Verified real-time ingestion from simulated sensors and integrity in InfluxDB.
- **Diagnostic Validation:** Tested under multiple fault conditions such as solar drop, inverter overheating, and grid instability.
- **RL Performance:** The DQN agent demonstrated improved energy efficiency and reduced grid dependency in simulated environments.
- **Dashboard Validation:** Grafana panels tested for low latency, accurate visualization, and alert triggers.

---

9. Scalability and Flexibility

- The architecture is **modular**, allowing integration of new assets, sensors, or plants with minimal configuration.
- Thanks to **InfluxDB's schema flexibility**, new parameters can be added without database redesign.
- Grafana dashboards use **dynamic variables**, automatically recognizing new metrics and assets — eliminating the need for recoding visualization.
- The entire system supports **cloud-based scaling** for multi-site energy management.

---

10. Impact and Benefits

Technical Benefits:

- Real-time visibility and actionable insights for all subsystems.
- Intelligent fault detection and predictive maintenance capabilities.
- RL-based dispatch reduces operational cost and increases renewable utilization.
- Scalable and future-proof architecture suitable for industrial deployment.

Industrial & Societal Impact:

- Enables smarter, greener microgrid operations.
- Supports India's transition toward decentralized renewable systems.
- Provides an open, replicable framework for research and utility applications.

- Reduces downtime and maintenance cost through predictive alerts.

---

11. Future Scope

The system can be expanded to:

- Integrate **real hardware** from solar farms, EV charging stations, and microgrids.
- Employ **LSTM/Transformer models** for load and generation forecasting.
- Develop **multi-agent RL systems** for distributed microgrids coordination.
- Implement **edge computing** for low-latency, on-site control.
- Collaborate with **industrial and government partners** to deploy the platform at scale across India.

---

12. Conclusion

The project demonstrates a fully functional **IoT-driven Energy Management System** integrating cloud infrastructure, diagnostics intelligence, and AI-based optimization.
By leveraging **InfluxDB**, **Grafana**, **MQTT**, and **Reinforcement Learning**, the system delivers scalability, reliability, and operational efficiency.

It stands as a practical and deployable solution for modern smart grids and renewable energy networks — showcasing how data-driven intelligence can transform energy management for a sustainable future.