An Edge-Centric IoT System for Smart Building Energy Management

João Silva[1], João Alves[1], Nuno Silva[1], Abdul Rauf[1], Victor Bongard[1],
Victor Rodríguez[2], Nuno Moreira[2],
Rui Pinto[1,3], Gil Gonçalves[1,3]
Dept. de Engenharia Informática, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal[1]
Dept. de Ciência de Computadores, Faculdade de Ciências, Universidade do Porto, Porto, Portugal[2]
SYSTEC, ARISE, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal[3]
Email: {up202108713, up202108670, up202005501, up202502193, up202502847}@edu.fe.up.pt
{up202105114, up202104873}@edu.fc.up.pt
{rpinto, gil}@fe.up.pt

***Abstract*—Rising energy costs and suboptimal environmental control in industrial and service facilities are frequently associated with limited real-time visibility, static control strategies, and low resilience in conventional building management systems. Many existing solutions rely on centralized architectures, which increases latency and reduces robustness under connectivity disruptions, while offering limited support for proactive optimization. This paper presents a modular, edge-centric Internet of Things (IoT) platform for multi-zone environmental monitoring and control, designed to preserve local autonomy under partial failure conditions. Environmental and occupancy data are acquired by microcontroller-based sensor nodes and disseminated via Message Queuing Telemetry Transport (MQTT) to a local edge gateway, where data validation, aggregation, and coordination are performed before selective synchronization with backend services for persistence and visualization. The platform integrates lightweight forecasting mechanisms that combine historical occupancy patterns with external weather information to support anticipatory control decisions. The proposed architecture enforces a clear separation between data and control planes, supports scalable multi-zone deployments, and enables energy-efficient operation with low deployment overhead, making it suitable for small- and medium-scale facilities.**

Keywords-IoT; CPS; MQTT; InfluxDB; SQLite; Grafana; Edge Computing; AI Forecast.

I. Introduction

The increasing pressure to reduce energy consumption and meet sustainability targets has intensified the need for more efficient management of lighting and Heating, Ventilation, and Air Conditioning (HVAC) systems in industrial and service facilities [1][2]. Despite advances in automation technologies, many buildings still rely on static schedules and coarse-grained control strategies, providing limited real-time visibility into environmental conditions and system behavior. Consequently, energy usage is often misaligned with actual occupancy and operational patterns, leading to waste and reduced efficiency [3]. Although Internet of Things (IoT) and Cyber-Physical Systems (CPS) approaches enable fine-grained sensing and adaptive control [4], many existing solutions remain centralized and cloud-dependent, reducing robustness under network disruptions and limiting suitability for resource-constrained environments. Moreover, fragmented deployments frequently result in transient or poorly persisted data, hindering long-term analysis and informed decision-making.

To address these limitations, this work proposes a resilient, edge-centric IoT/CPS platform for multi-zone environmental monitoring and control. The system connects sensors, microcontrollers, edge services, and backend components through a scalable architecture that separates telemetry from control execution. It supports real-time data acquisition, persistence, and visualization, exposes well-defined APIs for integration, and enables closed-loop control of lighting and HVAC systems. Lightweight forecasting mechanisms based on occupancy patterns and weather data allow proactive, energy-aware operation [5], while preserving local autonomy under connectivity failures.

The main contributions of this paper are threefold: (1) the design of a modular, edge-first IoT/CPS architecture that maintains operation during gateway or network disruptions; (2) the implementation of a clear separation between telemetry and control flows to enhance robustness and scalability; and (3) the integration of lightweight, edge-based decision support for occupancy- and weather-aware environmental control within a practical proof-of-concept deployment.

The remainder of this paper is organized as follows. Section II reviews related work and identifies research gaps. Section III details the proposed architecture and system design. Section IV presents implementation and evaluation results. Finally, Section V discusses limitations and outlines future research directions.

II. Related Work

Smart building energy management has evolved from traditional, proprietary Building Management Systems (BMS) toward IoT- and CPS-based architectures integrating distributed sensing, communication, and actuation. Conventional BMS solutions are typically centralized and offer limited interoperability and adaptability, motivating the adoption of modular IoT-based approaches as energy efficiency and sustainability requirements increase.

Recent surveys show that contemporary Building Energy Management Systems (BEMSs) increasingly rely on distributed IoT sensors, wireless protocols, and data-driven analytics to enable real-time monitoring and adaptive control. Reviews by Akbulut *et al.* and Poyyamozhi *et al.* highlight

TABLE I. ARCHITECTURAL COMPARISON WITH REPRESENTATIVE ENERGY MANAGEMENT PARADIGMS

Feature	Centralized IoT/BEMS	AI/DL Cloud HVAC	Microgrid EMCS	This Work
Primary processing location	Cloud	Cloud	Hierarchical/Central	Edge-first
Local autonomy under backend failure	Limited	Limited	Partial	Yes
Explicit data/control separation	Rare	Not emphasized	Layered control	Yes
Edge telemetry buffering	Uncommon	Uncommon	Application-specific	Yes
Acknowledgment-based actuation	Not typical	Not typical	Application-specific	Yes
Lightweight edge deployment	Variable	High compute	Grid-scale infra	Yes (RPi/MCU)
Edge-level predictive logic	Rare	Cloud-based	Optimization-based	Yes (lightweight)
Target deployment scale	Building	Building	Grid/Microgrid	Multi-zone building

the convergence of IoT, Artificial Intelligence (AI), and energy management, reporting consistent reductions in HVAC and lighting energy consumption while preserving occupant comfort [3][4]. These studies confirm the effectiveness of fine-grained sensing and analytics but also reveal a strong dependence on centralized processing infrastructures.

Occupancy-aware control has become a key research direction, replacing static schedules with dynamic adaptation based on inferred or predicted presence. Probabilistic occupancy prediction models have demonstrated improved HVAC optimization compared to binary detection approaches [5]. In parallel, predictive strategies such as Model Predictive Control (MPC) have shown measurable energy savings when combined with real-time sensor data and forecasts [2]. AI- and Deep Learning-based approaches further enhance adaptability under dynamic environmental and occupancy conditions [1]. However, many of these methods assume centralized computation, large datasets, and substantial processing resources, limiting applicability in lightweight or resource-constrained environments.

Energy-management optimization is also widely investigated beyond buildings. Microgrid Energy Management and Control Systems (EMCS) integrate layered control and optimization objectives for grid stability [6], while model-free wind-farm control and learning-augmented predictive control in hybrid and plug-in hybrid electric vehicles demonstrate the broader applicability of data-driven and predictive strategies across energy domains [7]–[9]. These cross-domain applications reinforce the trend toward predictive and model-based optimization but often target large-scale infrastructures rather than building-level deployments.

Despite significant progress in sensing, analytics, and predictive control, architectural limitations remain. Many IoT/BEMS solutions retain strong cloud dependency for persistence, analytics, and decision-making, increasing latency and reducing robustness under network disruptions [10]. Edge components are frequently treated as data relays rather than autonomous coordination layers, and explicit separation between telemetry and control flows is seldom emphasized [11]. Moreover, resilience mechanisms such as telemetry buffering, acknowledgment-based actuation tracking, and fault isolation across architectural layers are not systematically addressed in building-scale systems. While Digital Twin and AI-driven approaches represent the state of the art, they often introduce architectural complexity and computational overhead that hinder practical deployment in small- and medium-scale environments [12]. Additionally, reproducible and privacy-preserving evaluation strategies—such as the controlled use of synthetic data—remain underexplored in applied IoT/CPS prototypes [13].

Table I summarizes the architectural positioning of the proposed system relative to representative paradigms in the literature. In contrast to centralized BEMS and AI-driven HVAC solutions, which prioritize optimization but depend on continuous backend availability, the proposed architecture emphasizes edge autonomy, explicit separation of telemetry and control planes, acknowledgment and buffering mechanisms, and low-overhead deployment suitable for small- and medium-scale facilities.

III. PROPOSED SOLUTION

The proposed system is an IoT-based BEMS for intelligent environmental monitoring and control in indoor spaces. Although motivated by industrial and office scenarios, the architecture is generic and adaptable to residential or mixed-use environments. It combines distributed sensing, edge coordination, and backend services to automate lighting and HVAC control, targeting energy efficiency while maintaining comfort constraints.

The design follows a layered IoT/CPS architecture that separates sensing, communication, processing, and user interaction. Embedded devices perform local data acquisition

and actuation; an edge gateway coordinates communication and autonomy; backend services provide persistence and visualization without being required for core operation. The architecture prioritizes resilience, modularity, and scalability, enabling continued local functionality under connectivity disruptions and incremental system expansion through standard protocols and well-defined interfaces. In addition to reactive control, the system integrates lightweight, edge-based decision support mechanisms—such as occupancy- and weather-aware logic—to demonstrate anticipatory control within a practical proof-of-concept deployment.

A. Layered IoT Architecture

The system adopts a layered IoT/CPS architecture (Figure 1) that separates physical, communication, processing, and user-facing concerns, aligning with CPS principles of sensing, computation, actuation, and feedback.

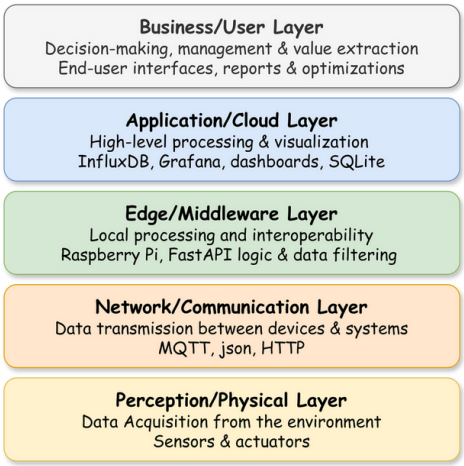

Figure 1. High-level overview of the adopted architecture.

- Perception/Physical Layer: ESP32-based devices connected to environmental sensors and actuators perform local data acquisition and immediate actuation.
- Network/Communication Layer: MQTT [14] supports publish–subscribe device-to-edge communication, while HTTP enables interaction among edge, backend, and frontend services.
- Edge/Middleware Layer: Implemented on a Raspberry Pi [15], this layer hosts the MQTT broker and services for device coordination, aggregation, buffering, and local autonomy.
- Application/Cloud Layer: Provides persistence and system state management. In the proof of concept, services are containerized and deployed locally but remain architecturally decoupled from the edge.
- Business/User Layer: Exposes visualization, configuration, and supervisory functions through a web-based interface.

Although the current implementation supports a single gateway, the architecture is designed to scale across multiple zones and multi-gateway deployments.

B. Data and Control Flow

System operation is structured around two explicit runtime paths (Figure 2):

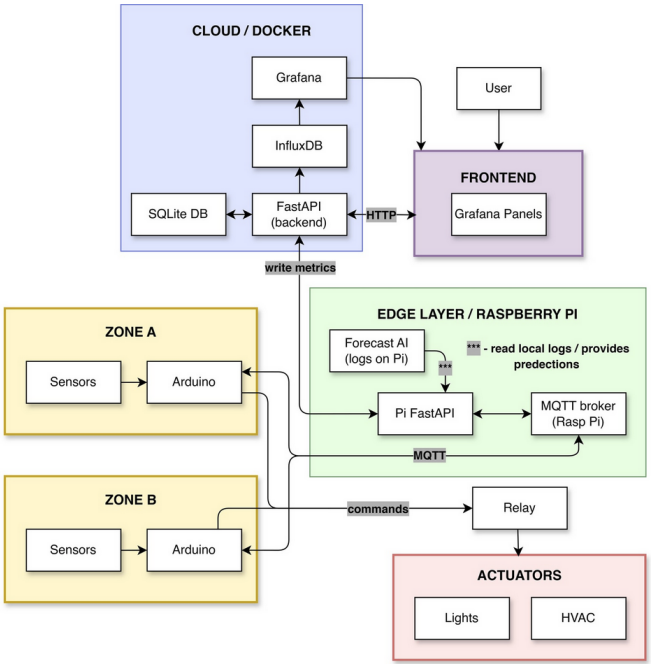

Figure 2. Overview of the data and control flows across system layers.

Data Flow. Embedded devices periodically publish telemetry via MQTT to the edge gateway. The gateway aggregates and enriches measurements before forwarding structured data to backend services for persistence and visualization. Time-series data and metadata are stored separately to support both historical analysis and state tracking. During backend unavailability, telemetry is buffered at the edge and synchronized upon recovery.

Control Flow. Control commands originate either from user interactions or edge-level decision logic. Commands propagate from frontend to backend and then to the edge gateway, which dispatches them to devices via MQTT. An acknowledgment-based mechanism tracks commands with a pending status and confirms completion upon execution, enabling reliable actuation and safe retry under transient failures.

This explicit separation improves robustness, fault isolation, and scalability.

C. Edge Gateway and Reliability Mechanisms

The edge gateway plays a central role in coordinating communication, enforcing local autonomy, and supporting resilience under partial failures. By hosting critical coordination and control logic at the edge, the system reduces dependence on continuous backend availability and avoids single points of failure. At runtime, the gateway hosts the MQTT broker and executes services for device management, data aggregation, buffering, and command dispatch. Sensing and actuation continue locally during backend disruptions, with telemetry and logs stored at the edge until synchronization is restored.

Reliability is further enhanced through acknowledgment-based command handling. Control commands are applied immediately at the edge and confirmed back to the backend, ensuring that delayed or missed transmissions can be retried without duplication. Complementary mechanisms include telemetry buffering during disconnections, periodic resynchronization after recovery, and basic reconnection strategies at the device level. Collectively, these mechanisms enable continued operation under common failure scenarios while preserving consistency and recoverability.

D. Intelligent Decision Support at the Edge

The system integrates lightweight, edge-based decision support mechanisms to enable anticipatory control without imposing significant computational overhead.

Weather forecast data retrieved from the OpenWeatherMap API [16] are used to estimate short-term temperature trends, including direction and magnitude. These trends inform pre-conditioning decisions, mitigating thermal inertia effects and improving comfort while reducing peak energy demand.

Occupancy forecasting is formulated as a probabilistic classification problem, producing continuous occupancy likelihood estimates rather than binary labels. Due to the absence of real occupancy traces, models are trained using synthetically generated behavioral data that capture temporal patterns and stochastic variability. This approach supports reproducibility and privacy-preserving evaluation. At runtime, the trained model is used for inference at the edge, generating near-future occupancy probabilities that are integrated into control decisions.

Control logic follows a hybrid strategy combining predictive inputs with deterministic, rule-based constraints. Thermal comfort is managed through context-dependent temperature bands (e.g., occupied, unoccupied, peak tariff periods), while pre-conditioning decisions are based on predicted occupancy probability, deviation from comfort targets, and external temperature trends. Stability mechanisms such as minimum action intervals are employed to prevent oscillatory behavior and reduce actuator wear.

IV. Evaluation and Results

A. Implementation Scope and Experimental Setup

The system was implemented as a proof of concept using lightweight, widely adopted technologies suitable for IoT/CPS prototyping. ESP32 microcontrollers [17] were used for sensing and actuation, while a Raspberry Pi served as the edge gateway. Communication relied on MQTT, with edge and backend services implemented using FastAPI [18]. System configuration and state metadata were persisted in SQLite [19], while time-series telemetry was stored in InfluxDB [20] and visualized through Grafana dashboards [21]. A web-based frontend implemented in React [22] provided supervision and interaction. Lightweight decision support was implemented using a logistic regression model trained on synthetically generated occupancy data.

The evaluation focused on validating architectural decisions and runtime behavior rather than production readiness or quantitative energy savings. The system was deployed locally, without reliance on external cloud infrastructure, and supports a single edge gateway managing multiple logical zones. While the architecture is designed to scale to multi-zone and multi-gateway deployments, these aspects were not fully exercised in the current implementation. Figure 3 illustrates the implemented supervisory interface.

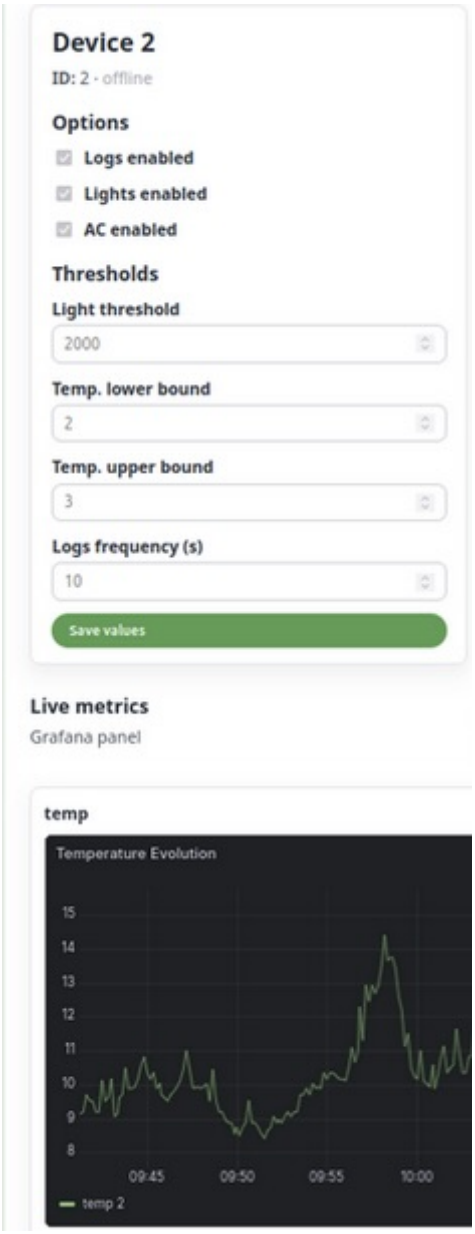

Figure 3. Frontend implementation (device page).

B. Validation Methodology and Scenarios

Evaluation was structured around end-to-end validation of the system's data and control paths under representative operating conditions. The data pipeline—from MQTT ingestion

at the edge to persistence in InfluxDB and visualization in Grafana—was validated using both real sensor data and mocked telemetry streams. Dashboard correctness was verified across different time ranges and filtering conditions, and displayed values were cross-validated against stored records.

Control paths (frontend → backend → edge → MQTT → devices) were exercised to confirm correct propagation of commands and state updates. Command handling relied on acknowledgment-based mechanisms between backend and edge components, enabling verification of reliable actuation and safe retry behavior under transient failures. All services were deployed locally using containerized environments [23], enabling functional validation while abstracting network latency and remote availability effects.

Two complementary validation scenarios were considered. First, a physical edge deployment was evaluated using one Raspberry Pi and two ESP32-based nodes, each representing an independent zone. Real-time telemetry (temperature, luminosity, and binary state variables) was published via MQTT, and control actions were issued from the frontend. In the absence of physical HVAC or lighting equipment, actuation was emulated using onboard indicators, allowing verification of message routing, state transitions, and bidirectional communication without hardware-specific dependencies.

Second, a mixed-data scenario was used to assess scalability and observability under higher logical load. Multiple logical devices were emulated using mocked telemetry to validate MQTT topic handling, backend routing, and dashboard filtering. This scenario enabled evaluation of persistence, visualization, and system behavior independently of physical device availability.

Edge-level decision logic integrating weather forecast data was evaluated by inspecting inference outputs and control decisions generated on the gateway. Due to abstracted actuation and the coexistence of manual and automated control paths, this component was assessed primarily in terms of computational correctness and system integration rather than observable physical effects.

C. Resilience Observations and Results

Table II summarizes the main resilience-related observations obtained during evaluation. The results confirm that the proposed architecture supports reliable telemetry ingestion, persistence, and visualization, maintaining consistent supervisory views once time-series data is available.

From a resilience perspective, the system demonstrated effective fault isolation across device, edge, and backend layers. Backend restarts did not affect previously persisted data, and buffered telemetry was successfully forwarded after recovery. Similarly, edge restarts resulted in brief service interruptions, after which devices automatically reconnected and normal data and control flows were restored. These behaviors validate the edge-centric design objective of preserving local autonomy under partial failures.

Observed data loss was confined to the device–edge boundary during edge downtime, reflecting an intentional design trade-off favoring lightweight embedded devices without persistent buffering. Importantly, such losses did not propagate upstream, preserving system stability and observability. Embedded dashboards consistently reflected the recovered system state following disruptions, reinforcing the suitability of the platform for supervisory monitoring.

TABLE II. OBSERVED RESILIENCE-RELATED BEHAVIORS DURING EVALUATION.

Metric / Scenario	**Observed Behavior**
Physical devices tested	2 ESP32 + 1 Raspberry Pi
Mocked devices tested	Up to 10 logical devices
Backend restart recovery time	∼2 seconds
Edge restart recovery time	Few seconds (boot + services)
Data loss (ESP32 → edge)	During edge downtime
Data loss (edge → cloud)	Not observed; buffered data forwarded
Device reconnection	Automatic, without manual intervention
Command acknowledgment	Implemented cloud-edge only

Overall, the evaluation demonstrates that the proposed architecture achieves its primary objectives of modularity, observability, and resilience in a multi-zone, edge-centric setting, while highlighting device-level buffering and acknowledgment mechanisms as natural directions for future enhancement.

V. CONCLUSION AND FUTURE WORK

This work presented a modular, edge-centric IoT/CPS platform for multi-zone environmental monitoring and control, designed to preserve local autonomy while enabling scalable supervision and centralized observability. By combining distributed sensing with edge-level coordination and backend persistence, the proposed architecture supports reliable real-time and historical monitoring, explicit separation between data and control flows, and resilient operation under partial connectivity failures. The evaluation confirmed that the system maintains consistent behavior across device, edge, and backend layers, validating the architectural choices with respect to modularity, observability, and fault isolation.

While the proof-of-concept demonstrates the feasibility of the proposed approach, several limitations remain. Visualization relies on embedded dashboard components, which introduce authentication and browser policy constraints that may complicate operational deployment. Data quality is dependent on sensor calibration and timestamp consistency, and the current implementation does not include automated validation or anomaly detection mechanisms. In addition, although predictive components based on occupancy and weather context are integrated at the edge, their influence on control policies remains limited and has not yet been validated using real occupancy data.

Future work will focus on extending resilience through device-level buffering and backfilling mechanisms, strengthening authentication and access control across backend services and visualization components, and introducing automated monitoring and alerting. The decision-support layer will be further integrated into control logic and evaluated with real-world occupancy data. Finally, enhanced device management, calibration workflows, and deployment automation (e.g., continuous integration and continuous deployment (CI/CD) and container orchestration) will be explored to improve maintainability and support larger-scale deployments.

ACKNOWLEDGMENT

This work is financially supported by national funds through the FCT/MCTES (PIDDAC), under the Associate Laboratory Advanced Production and Intelligent Systems – ARISE LA/P/0112/2020 (DOI 10.54499/LA/P/0112/2020) and the Base Funding (UIDB/00147/2020) and Programmatic Funding (UIDP/00147/2020) of the R&D Unit Center for Systems and Technologies – SYSTEC.

REFERENCES

[1] S. A. Aghili et al., "Artificial Intelligence Approaches to Energy Management in HVAC Systems: A Systematic Review," *Buildings*, vol. 15, no. 7, Mar. 2025, ISSN: 2075-5309. DOI: 10.3390/buildings15071008. Accessed: Jan. 13, 2026.

[2] M. K. Pasupuleti, "Model Predictive Control for Smart HVAC Systems in Green Buildings," *International Journal of Academic and Industrial Research Innovations (IJAIRI)*, vol. 5, pp. 1–11, Jun. 2025. DOI: 10.62311/nesx/rphcrefcs1.

[3] L. Akbulut et al., "A Systematic Review of Building Energy Management Systems (BEMSs): Sensors, IoT, and AI Integration.," *Energies (19961073)*, vol. 18, no. 24, p. 6522, Dec. 2025, ISSN: 1996-1073. DOI: 10.3390/en18246522. Accessed: Jan. 13, 2026.

[4] M. Poyyamozhi et al., "IoT—A Promising Solution to Energy Management in Smart Buildings: A Systematic Review, Applications, Barriers, and Future Scope," *Buildings*, vol. 14, no. 11, Oct. 2024, ISSN: 2075-5309. DOI: 10.3390/buildings14113446. Accessed: Jan. 13, 2026.

[5] I. Khan, O. Zedadra, A. Guerrieri, and G. Spezzano, "Occupancy Prediction in IoT-Enabled Smart Buildings: Technologies, Methods, and Future Directions," *Sensors (Basel, Switzerland)*, vol. 24, no. 11, p. 3276, May 2024, ISSN: 1424-8220. DOI: 10.3390/s24113276. Accessed: Jan. 13, 2026.

[6] S. Ahmad, M. Shafiullah, C. B. Ahmed, and M. Alowaifeer, "A Review of Microgrid Energy Management and Control Strategies," *IEEE Access*, vol. 11, pp. 21 728–21 757, Feb. 2023. DOI: 10.1109/ACCESS.2023.3248511.

[7] M. A. Ahmad, M. R. Hao, R. M. T. R. Ismail, and A. N. K. Nasir, "Model-free Wind Farm Control Based on Random Search," in *2016 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, Oct. 2016, pp. 131–134. DOI: 10.1109/I2CACIS.2016.7885302.

[8] J. J. Jui, M. A. Ahmad, M. I. Milla, and M. I. M. Rashid, "Optimal Energy Management Strategies for Hybrid Electric Vehicles: A Recent Survey of Machine Learning Approaches," *Journal of Engineering Research*, vol. 12, pp. 454–467, 2024. DOI: 10.1016/j.jer.2024.01.016.

[9] X. Sun, J. Fu, H. Yang, M. Xie, and J. Liu, "An Energy Management Strategy for Plug-in Hybrid Electric Vehicles Based on Deep Learning and Improved Model Predictive Control," *Energy*, vol. 269, 2023. DOI: 10.1016/j.energy.2023.126772.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. DOI: 10.1109/JIOT.2016.2579198.

[11] A. Sivanathan, H. Habibi Gharakheili, and V. Sivaraman, "Managing iot cyber-security using programmable telemetry and machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 60–74, 2020. DOI: 10.1109/TNSM.2020.2971213.

[12] A. Ghaemi, Y. Rezgui, I. Petri, T. Beach, and A. Ghoroghi, "Ai and digital twin applications in building energy management: A state-of-the-art review," in *2025 IEEE International Conference on Engineering, Technology, and Innovation (ICE/ITMC)*, 2025, pp. 1–11. DOI: 10.1109/ICE/ITMC65658.2025.11106601.

[13] N. Harth, C. Anagnostopoulos, and D. Pezaros, "Predictive intelligence to the edge: Impact on edge analytics," *Evolving Systems*, vol. 9, no. 2, pp. 95–118, Jun. 2018, ISSN: 1868-6486. DOI: 10.1007/s12530-017-9190-z. [Online]. Available: https://doi.org/10.1007/s12530-017-9190-z.

[14] A. Banks and R. Gupta, "MQTT Version 3.1.1," 2014, Accessed: Jan. 19, 2026. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[15] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, and N. K. Kumaravelu, "A comprehensive review on applications of raspberry pi," *Computer Science Review*, vol. 52, p. 100 636, 2024.

[16] OpenWeather Ltd., "Weather API - OpenWeatherMap," 2025, Accessed: Jan. 19, 2026. [Online]. Available: https://openweathermap.org/api.

[17] N. Cameron, "Esp32 microcontroller," in *ESP32 Formats and Communication: Application of Communication Protocols with ESP32 Microcontroller*. Berkeley, CA: Apress, 2023, pp. 1–54, ISBN: 978-1-4842-9376-8. DOI: 10.1007/978-1-4842-9376-8_1. [Online]. Available: https://doi.org/10.1007/978-1-4842-9376-8_1.

[18] M. Lathkar, "Getting started with fastapi," in *High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python*. Berkeley, CA: Apress, 2023, pp. 29–64, ISBN: 978-1-4842-9178-8. DOI: 10.1007/978-1-4842-9178-8_2. [Online]. Available: https://doi.org/10.1007/978-1-4842-9178-8_2.

[19] J. Kreibich, *Using SQLite*. " O'Reilly Media, Inc.", 2010.

[20] InfluxData, "InfluxDB Documentation," 2025, Accessed: Jan. 19, 2026. [Online]. Available: https://docs.influxdata.com/influxdb/.

[21] S. Kirešová et al., "Grafana as a visualization tool for measurements," in *2023 IEEE 5th International Conference on Modern Electrical and Energy System (MEES)*, IEEE, 2023, pp. 1–5.

[22] Meta Platforms, Inc., "React: A JavaScript Library for Building User Interfaces," 2025, Accessed: Jan. 19, 2026. [Online]. Available: https://reactjs.org/.

[23] I. Docker, "Docker Documentation," 2023, Accessed: Jan. 19, 2026. [Online]. Available: https://docs.docker.com/.