International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(15s) 2026 pp. 1131-1141
https://doi.org/10.70917/ijcisim-2026-4507
Article

An Edge-Based Intelligent System for Smart Building Energy Optimization

D. Sreevidya[1], Regana Kishore[2], Modepalli Meghana Chowdary[3], Nithya Siripurapu[4], Bhanuprakash Arekanti[5]

[1,2,3,4,5]Department of Computer Science and Engineering – Cyber Security, Data Science, and Artificial Intelligence & Data Science, VNR VJIET, Hyderabad, Telangana, India.

Abstract: The largest consumption of energy across the globe comes from residential building demographics. According to International Energy Agency nearly these buildings are responsible for around 30-40% of energy consumption across the world. One of the major issues related to energy waste in buildings is due to operation of traditional building systems at fixed timings without considering their occupancy in real-time. To address the above mentioned issue, the current paper proposes an Edge-enabled Digital Twin approach for optimal energy management in buildings. In the proposed framework, IoT-based sensors, edge computing, pre-diction of occupancy through machine learning, and use of digital twin dashboard helps in monitoring and optimizing energy usage through the indoor environment. All of the sensed data such as occupancy rate, temperature, and light intensity are processed locally using edge computing. Based on the above gathered information, HVAC systems and illumination systems are operated to save energy without making the occupant feel uncomfortable. The digital twin offers a visual representation of device status and building conditions which enables users to keep track of energy performance easily. The simulation results show that this approach can reduce approximately 20–30% of energy consumption compared to schedule-based methods. This high-lights the effectiveness of combining IoT, edge intelligence, and digital twin technology for smart building energy optimization.

Keywords: Digital Twin, Smart Buildings, Internet of Things (IoT), Edge Computing, Energy Optimization, Occupancy Detection, HVAC Control, Building Energy Management Systems (BEMS)

1. INTRODUCTION

At present, in many corporate and institutional settings, the HVAC and lighting systems are scheduled without regard to their usage by actual occupants.[33]

The best approach in solving such challenges is edge computing. In case data from sensors is analyzed within the building itself, then any changes in environmental conditions and occupancy will be captured, and action taken instantly without relying on cloud computing.[1]-[4]

This work introduces a Digital Twin-based framework for smarter energy optimization that combines IoT sensing, automated control mechanisms, and local edge processing. An edge device processes these readings to determine the best conditions for the building.[29],[30]

Contributions: The main contributions of this work are:

- Development of an IoT-based sensing framework to mon-itor occupancy and environmental conditions in indoor spaces.[5],[6]
- Design of an edge-enabled architecture which will per-form fast data processing and energy optimization using a Raspberry Pi 4 edge server with sensor ingestion based on MQTT protocol, an ML inference engine, and a FastAPI-based control mechanism. [1]-[4]

- Integration of an adaptive Exponentially Weighted Mov-ing Average (EWMA) occupancy prediction model that continuously improves accuracy as it collects observa-tions, along with a Random Forest model for longer-horizon demand forecasting.
- Implementation of a digital twin dashboard for real-time visualization of environmental parameters, energy consumption in the building, and device status, with asynchronous cloud logging to ThingSpeak.

2. PROPOSED SYSTEM

Rather than relying on time-scheduled events, the system is constantly monitoring the environment and adjusting the equipment used in buildings based on such findings.[1]-[4],[33]

1) *Real-Time Environmental Sensing*

The presence of people is detected using PIR sensors, whereas the room's thermal state and lighting levels are measured by special temperature, humidity, and illumination sensors.[5],[6]

2) *Edge-Based Data Processing*

The sensor values are then published using the MQTT protocol to the campus/Z0x/sensors channel and consumed by the hw_bridge component executing on the Raspberry Pi 4 edge server. The pre-processed data is then passed to the ML engine, where the adaptive EWMA occupancy model is applied, and the output is served via the FastAPI endpoint, using Nginx, without any reliance on the cloud.[5],[6]

3) *Predictive Energy Optimization*

The system uses not only the reactive response from the rules but also gathers historical data from sensors to build up the Random Forest model. This model predicts what changes will happen concerning occupant and power consumption over the coming intervals, thus helping the system be proactive rather than reactive.[11],[12]

4) *Digital Twin Monitoring*

In addition to feeding the control systems with the same streams of data, the twin updates a dashboard display showing the occupancy of zones, environmental data, state of equipment, and energy consumption in one glance.[29],[30]

3. SYSTEM ARCHITECTURE

The physical and computational elements of the proposed system are arranged into a layered architecture from sensing at the field level to visualisation at the application level.[1]-[4],[33]

1) *Perception Layer (Sensing Layer)*

This is the data-collection tier, where three categories of ESP32-based sensor nodes measure the physical state of each building zone:

- **ESP32 + DHT22 + BH1750** — measures temper-ature, humidity, and ambient light.
- **ESP32 + PZEM-004T** — monitors voltage, current, and energy consumption.
- **ESP32 + PIR + IR beam** — detects occupancy through passive infrared motion sensing and beam-break counting.[5],[6]

All nodes publish their readings to MQTT broker under the topic pattern campus/Z0x/sensors.

2) *Network Layer*

Sensor nodes deliver their readings to the edge device through wireless communication protocols. The proto-cols supported include:[5], [6].

- Wi-Fi (primary, used by all ESP32 nodes)
- Bluetooth Low Energy (BLE)
- Zigbee

Each protocol was selected to balance transmission reliability with energy efficiency.[1], [3], [4]

3) *Edge Processing Layer*

A Raspberry Pi 4 edge server [2][4]performs local data processing and control logic through three main components:[2]–[4].

- **hwbridge** — subscribes to the MQTT broker, performs signal filtering and normalization, and forwards structured records to the ML engine.
- **ML engine (EWMA · detect)** — runs the adaptive EWMA occupancy detection algorithm and generates control setpoints.
- **FastAPI (setpoints · API)** — exposes a REST interface through which actuators and the digital twin dashboard query the latest setpoints; served by Nginx at 192.168.10.100.

4) *Control Layer*

Control commands are published to campus/Z0x/commands topic and executed by two classes of actuator:

- **Relay / TRIAC module** — switches lights and fans on or off.
- **IR blaster** — transmits infrared codes to control split-type air-conditioning units.

5) *Devices Layer*

The actuators govern campus zone equipment including HVAC units, lighting circuits, fans, and other electrical loads.

6) *Cloud Logging (ThingSpeak)*

In addition to local edge storage, aggregated zone data is forwarded asynchronously to ThingSpeak every 15 seconds (one channel per zone, eight data fields) for long term trend analysis and remote dashboarding.[29,[30]

Fig. 1 shows the complete layered system architecture. Three ESP32 sensor node types at the perception layer com-municate over MQTT to the Raspberry Pi 4 edge server, which processes data and issues commands through the relay/TRIAC and IR blaster control modules to campus zone equipment. An asynchronous 15-second link forwards summarized data to the ThingSpeak cloud for long-term logging.

4. IOT SENSOR TECHNOLOGIES

This section describes the sensors selected for the task and the rationale behind their deployment.

A. Sensors Used in the System

The system consists of four basic sensors in its sensing unit[5][6]:

- **PIR Sensor** — senses the presence of a person in order to determine occupancy; used in combination with infrared beam counters to estimate crowd level.
- **Temperature / Humidity Sensor (DHT22)** — monitors the surrounding environment's temperature and humidity level for HVAC control.
- **Ambient Light Sensor (BH1750)** — This sensor mea-sures ambient illuminance levels in lux for automatic lighting control.
- **Power Meter (PZEM-004T)** — Measures real-time volt-age, current, and total energy consumed in each zone.

B. Sensor Specifications

Table I summarizes the key technical specifications of the sensors used in the proposed system.[5][6]

TABLE I

SENSOR TECHNICAL SPECIFICATIONS

Sensor	Parameter	Range	Accuracy
PIR Sensor	Motion / Occupancy	5–7 m detection	High

LM35	Temperature	−55 to 150 ◦C	±0.5 ◦C
DHT22	Temperature / Humidity	−40 to 80 ◦C	±0.5 ◦C
BH1750	Illuminance	1–65535 lux	±20%
PZEM-004T	Voltage / Energy	80–260 V AC	±0.5%
LDR	Light Intensity	1–1000 lux	Medium

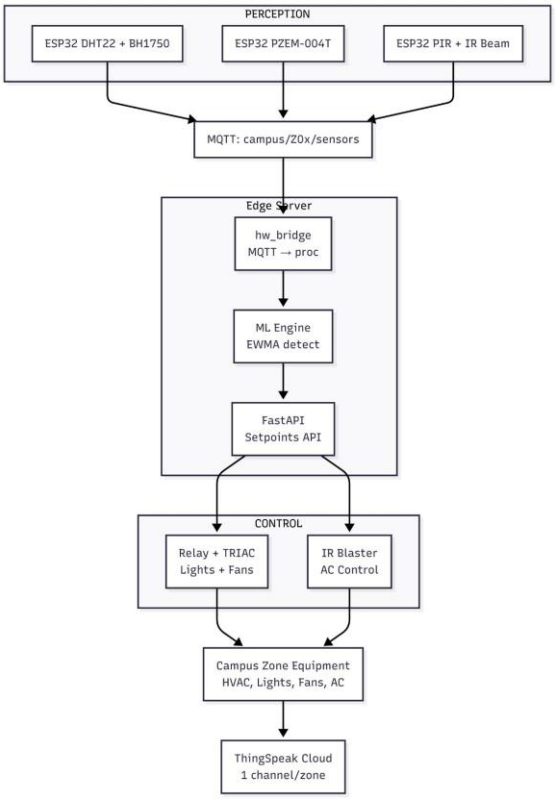

Fig. 1. Layered system architecture. Three ESP32 sensor node types (DHT22 + BH1750; PZEM-004T; PIR + IR beam) publish data over MQTT to the Raspberry Pi 4 edge server. The edge server runs an hw_bridge, an EWMA-based ML engine, and a FastAPI endpoint (Nginx, 192.168.10.100). Control commands flow via relay/TRIAC and IR blaster to campus zone equipment. A 15-second asynchronous link mirrors data to ThingSpeak (1 channel/zone, 8 fields).

C. Sensor Cost Analysis

The unit cost estimates of the sensor hardware are provided in Table II.

TABLE II

SENSOR COST ESTIMATION

Sensor Type	**Function**	**Approx. Cost**
PIR Sensor	Motion Detection	$2–$5
LM35 Temp. Sensor	Temperature Monitoring	$2–$4
DHT22 Sensor	Temperature / Humidity	$5–$8
BH1750 Lux Sensor	Illuminance Measurement	$2–$5

PZEM-004T	Energy Monitoring	$8–$12
LDR Sensor	Light Detection	$1–$3

D. Sensor Data Fusion

Each individual sensor provides information about a single aspect of the indoor environment, and thus, all information is combined to get an overall situational understanding:[5][6]

- Detecting room occupancy using PIR sensors and IR beam counters.
- Monitoring the thermal state of the environment using DHT22 temperature and humidity sensors.
- Determining the illumination level of the room using the BH1750 illuminance sensor.
- Monitoring real-time energy consumption using the PZEM-004T energy meter.

5. ENERGY OPTIMIZATION APPROACH

Control logic is encoded at the edge device and executes in real time as sensor readings arrive[1]-[4],[33]

A. Optimization Logic

Optimization proceeds through four inter-related steps:

1) *Occupancy Detection*

- PIR sensor detects motion signals.
- If motion is detected → Room status = Occupied.
- If no motion for a predefined interval (e.g., 10 min)

→ Room status = Unoccupied.

2) *Lighting Optimization*

- BH1750 sensor measures ambient illuminance.
- If illuminance < threshold (e.g., 300 lux) → Lights ON.
- If illuminance ≥ threshold → Lights OFF.

3) *HVAC Optimization*

- DHT22 measures room temperature.
- If temperature > 26◦C → AC ON.
- If temperature within comfort range → AC LOW / OFF.

4) *Idle Energy Reduction*

- If room status = Unoccupied → AC OFF.
- Lighting systems are switched OFF automatically.

B. Energy Consumption Estimation

Energy savings will be determined by comparing the actual energy usage to that of a base case representative of conventional scheduling-operation:[9]–[15], [33]

Energy Savings = Ebaseline − Eoptimized

where Ebaseline is energy used in conventional control mode and Eoptimized is energy used in occupancy-based control mode.

TABLE III

ENERGY OPTIMIZATION CONTROL PARAMETERS

Parameter	Threshold	Control Action
Occupancy	No motion for 10 min	Lights OFF, AC OFF
Temperature	> 26 ◦C	AC ON
Light Intensity	< 300 lux	Lights ON

C. Data Processing Pipeline

The sensor readings undergo pre-processing along a hybrid pipeline that allows both control and learning.[7]–[10]

1) *Sensor Data Acquisition*

ESP32 nodes constantly read sensor data from PIR, DHT22, BH1750, PZEM-004T, and optional CO2 sen-sors, and publish them to the MQTT broker.[5], [6]

2) *Edge Pre-processing*

The hw_bridge component consumes MQTT mes-sages, removes noise, synchronizes timestamps, and performs normalization before pushing the data to the ML engine.[9], [26]

3) *Real-Time Rule-Based Control*

Trigger rules invoke actions directly when breached. In case of inactivity for a certain amount of time, lighting and HVAC controls are turned off. The control signals are broadcast via campus/Z0x/commands.[27], [33]

4) *Historical Data Storage*

Historical readings, inferred occupancy states, device configurations, and energy consumption are recorded in a local database for analytics purposes and machine learning models.[7], [8], [10]

5) *Machine Learning Model Training*

An adaptive EWMA algorithm operates persistently, updating its occupancy estimates every observation. A Random Forest modelis trained periodically on the history of temperature, illuminance, time, and energy readings.[11], [12], [20]

6) *Prediction and Adaptive Optimization*

The predictions made by the machine learning algorithm can be viewed using the FastAPI API endpoint and then used to update the system configurations.[2], [20]

7) *Digital Twin Update*

Values from all sensors, predictions, device statuses, and energy usage information are sent to the dashboard of the digital twin. [29][30]. A separate service replicates all data at ThingSpeak every 15 seconds.

D. Machine Learning Based Optimization

Firstly, an adaptive EWMA (Exponentially Weighted Mov-ing Average) model is implemented on the edge server. Since recent data points are given higher weight than older obser-vations, the model can adapt to shifts in occupancy patterns caused by timetable changes or seasonal variations without requiring complete retraining.[11], [12], [20]

As illustrated in Fig. 2, the prediction accuracy improves from approximately 51% on Day 1 to over 85% by Day 7, and stabilizes around 91% by Day 14.

In the second strategy, the approach employed is a Random Forest classifier that is periodically trained based on historical data kept in a local database. The model is fed with data based on various parameters like temperature, illuminance, time of day, day of week, and energy consumption. With such predictions, the HVAC setpoints can be adjusted prior to the peak demands.[11], [12]

6. IMPLEMENTATION AND EXPECTED RESULTS

The complete system is implemented using off-the-shelf IoT devices, an edge computing, and a digital twin dashboard. The implementation consists of three main components:[1]–[4], [29], [30].

1) IoT sensing infrastructure
2) Edge processing and control unit
3) Digital twin monitoring dashboard

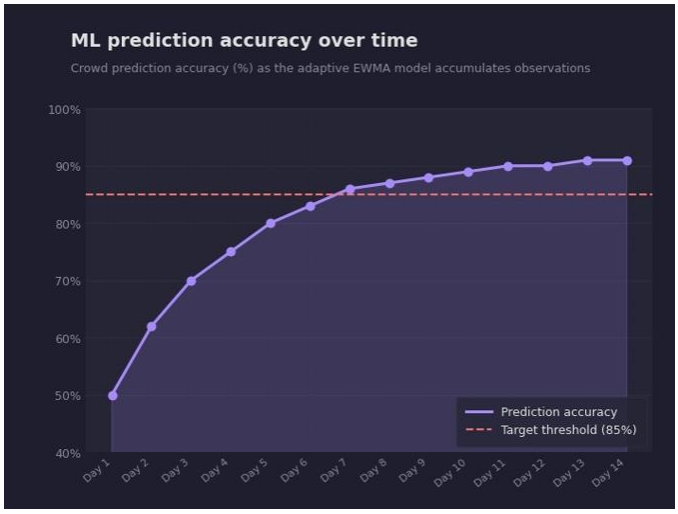

Fig. 2. Accuracy trend for ML prediction. The adaptive EWMA model begins with approximately 51% accuracy on Day 1, exceeds the 85% threshold (shown by the dashed red line) by Day 7, and converges to around 91% accuracy by Day 14.

A. Hardware Components

Sensing is done using several low-cost IoT devices that collectively track motion, temperature, illuminance, and energy usage in each zone.[5], [6]

TABLE IV

IOT HARDWARE COMPONENTS FOR ZONE MONITORING

Device	**Function**	**Cost**
ESP32 DevKit	Sensor node with Wi-Fi and MQTT communication	$5–$8
PIR Motion Sensor	Occupancy detection	$2–$4
DHT22 Sensor	Temperature and humidity monitoring	$5–$8
BH1750 Lux Sensor	Ambient illuminance measurement	$2–$5
PZEM-004T	Real-time voltage and energy monitoring	$8–$12
4-Ch Relay Module	Switching HVAC and light-	$3–$6

	ing loads	
IR Blaster Module	Infrared AC unit control	$1–$3

Each sensor node send their respective readings to the edge gateway using MQTT.[5], [6]

B. Edge Computing Infrastructure

Processing the data on an edge node eliminates dependence on the availability of the cloud environment and ensures quick response times regardless of the WAN performance.[1]–[4], [33]

At this tier, the Raspberry Pi 4 edge server performs the following actions:[2]–[4].

- Managing the MQTT broker and aggregating sensor readings using hw_bridge.
- Occupancy detection in real-time using the adaptive

EWMA ML engine.

- Evaluating the environmental state and computing set-points.

TABLE V

EDGE PROCESSING HARDWARE

Device	**Role**
Raspberry Pi 4	Edge server: MQTT broker, EWMA + Random Forest ML inference, FastAPI, Ng-inx
ESP32	Sensor data acquisition and Wi-Fi / MQTT communica-tion
Relay Controller	Controls lights, fans, and HVAC devices
IR Blaster	Sends infrared codes to split-type AC units

- Generating control signals and broadcasting the same to

campus/Z0x/commands.

- Exposing the REST APIs using FastAPI (hosted on Nginx running at 192.168.10.100).

C. Software Technologies

The application is designed using a software stack com-prising the sensor processing component, machine learning inference component, energy management logic, and digital twin user interface.[2], [20], [29], [30]

TABLE VI

SOFTWARE STACK

System Layer	**Technology Used**
Frontend Dashboard	React.js
Backend Server	Node.js + Express.js
Database	MongoDB
Edge Processing	Python

ML Model (online)	Adaptive EWMA (Python)
ML Model (batch)	Random Forest (Scikit-learn)
Communication	REST API / MQTT
3D Visualization	React Three Fiber
Cloud Logging	ThingSpeak (15 s async)

The digital twin interface shows each building zone in 3D with the zone occupancy, trend predictions, and energy consumption statistics being updated with incoming read-ings from sensors. The dashboard is hosted on Nginx at 192.168.10.100 as a reverse proxy service.

D. System Operation Workflow

The flow through which the system processes a normal sensor reading proceeds in the following way:

1) The ESP32-based sensors monitor and publish occu-pancy, environmental, and energy parameters on the MQTT broker under campus/Z0x/sensors.
2) The hw_bridge application processes the raw MQTT payload.
3) The EWMA ML engine evaluates occupancy probability and environmental conditions; the Random Forest model provides longer-horizon demand forecasts.[11], [12]
4) Energy optimization logic determines the optimal device state and publishes setpoints via the FastAPI endpoint.
5) Control signals are sent to the relay/TRIAC mod-ule (lights, fans) and IR blaster (AC units) via campus/Z0x/commands.
6) System status, sensor readings, and energy metrics are visualized through the digital twin dashboard and mir-rored asynchronously to ThingSpeak every 15 seconds.[29], [30]

E. Energy Consumption Analysis

Fig. 3 compares the hourly combined HVAC and lighting load across the full 10-zone campus building over a 24-hour period for the proposed smart system versus a conventional schedule-based baseline. During the early morning (12 am–6 am), both systems draw comparable standby loads of approx-imately 8–13 kW because the campus is largely unoccupied. From approximately 7 am onward, as zones are occupied, the traditional schedule ramps loads to 41–43 kW and holds them flat until late afternoon regardless of actual room-level occupancy. In contrast, the smart system manages demand more accurately: its peak power draw amounts to just 30 kW at 9 am and 12 pm, and decreases gradually after zones become vacant in the afternoon, to about 8 kW by 10 pm. The dark-shaded region between the two curves illustrates the total amount of energy conserved due to occupancy-based control during the course of the day.[9]–[15], [33].

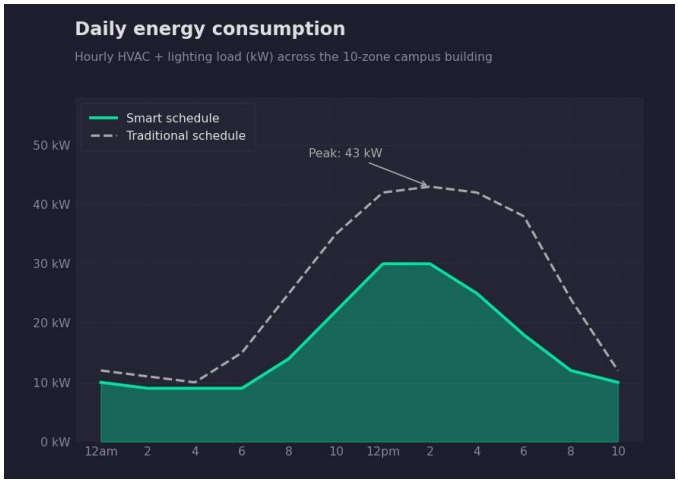

Fig. 3. Daily energy consumption: hourly HVAC + lighting load (kW) in the 10-zone campus building. The smart system (green, shaded) tracks actual occupancy and peaks near 30 kW, while the traditional schedule (dashed

grey) maintains 41–43 kW during occupancy hours regardless of demand, resulting in roughly 25–37% higher daily energy use.

Table VII contains a comparative analysis for a single seminar hall.

TABLE VII

ENERGY CONSUMPTION COMPARISON (SINGLE SEMINAR HALL)

System Type	Daily Energy Use	Reduction
Traditional System	16 kWh	–
Proposed System	10 kWh	37%

The savings result from the capability of the proposed system to turn off the lights and HVAC equipment as soon as occupancy drops to zero — an action impossible for a schedule-driven approach[9]–[15], [33].

F. Expected System Performance

From the analysis carried out, the performance indicators expected from the implemented system include:

- Reduction in energy usage: **25–40%** within all zones under observation.
- EWMA occupancy prediction accuracy: ≥**85%** from Day 7 on wards, converging towards **91%** by Day 14.
- Real-time PIR-based occupancy detection accuracy: **90–95%**.
- Automation of HVAC and lighting controls, with sub-second command execution times from the edge node.
- Lowered energy expenses and carbon emission footprint.
- Monitoring of the digital twin in real time with 15 seconds synchronization to ThingSpeak.

7. CONCLUSION

This paper has presented a design of a smart building energy optimization solution that takes into account occupancy-based sensing, environmental sensing, edge computation, and digital twin visualization. The use of ESP32-based sensor nodes fitted with PIR, DHT22, BH1750, and PZEM-004T sensors allows the creation of a complete picture regarding the state of each monitored zone.

Sensor input is taken through MQTT by a Raspberry Pi 4 edge server which runs an adaptive EWMA occupancy model along with a FastAPI control interface, responding to occupancy events and changing environment conditions with-out dependence on cloud infrastructure. The EWMA model exceeds the set target of 85% accuracy by Day 7 and reaches approximately 91% by Day 14.The HVAC setpoints prior to the occupancy peak. The study of energy consumption in the campus building with 10 zones proves that the smart system operates with a maximum power consumption of around 30 kW while the traditional one consumes 41-43 kW. It provides up to 25-40saving without any discomfort to the occupants. Future developments in the system will include the use of deep reinforcement learning in the generation of occupancy forecasts, improvements to multi-sensor data fusion for tighter detection accuracy, and expansion of digital twin into entire smart campus environments.

References

1. A. Dastjerdi and R. Buyya, “Fog computing for IoT applications,” Computer, vol. 49, no. 8, pp. 112–116, 2016.
2. W. Yu et al., “Edge computing for IoT systems,” IEEE Access, vol. 6,
3. pp. 6900–6919, 2018.
4. W. Shi et al., “Edge computing: Vision and challenges,” IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.
5. P. Mach and Z. Becvar, “Mobile edge computing: A survey,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628–1656, 2017.
6. A. Al-Fuqaha et al., “Internet of Things: A survey on enabling tech-nologies,” IEEE Communications Surveys & Tutorials, vol. 17, no. 4,
7. pp. 2347–2376, 2015.
8. J. Lin et al., “Survey on Internet of Things,” IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1125–1142, 2017.
9. M. Chen et al., “Big data analytics: A survey,” Mobile Networks and Applications, vol. 19, no. 2, pp. 171–209, 2014.
10. M. Marjani et al., “Big IoT data analytics,” IEEE Access, vol. 5,

11. pp. 5247–5261, 2017.
12. Y. Liu et al., “Edge-enabled smart grid data preprocessing,” IEEE Internet of Things Journal, vol. 7, no. 10, pp. 10392–10402, 2020.
13. Y. Wang et al., “Review of smart meter data analytics,” IEEE Transac-tions on Smart Grid, vol. 10, no. 3, pp. 3125–3148, 2019.
14. T. Hong and S. Fan, “Probabilistic electric load forecasting,” Interna-tional Journal of Forecasting, vol. 32, no. 3, pp. 914–938, 2016.
15. Y. Sun et al., “Energy consumption forecasting,” IEEE Transactions on Smart Grid, vol. 10, no. 6, pp. 6981–6990, 2019.
16. H. Wang et al., “Distributed optimization in microgrids,” IEEE Trans-actions on Smart Grid, vol. 10, no. 3, pp. 2364–2376, 2019.
17. B. Zhao et al., “Energy management of microgrids,” IEEE Transactions on Power Systems, vol. 33, no. 6, pp. 6410–6421, 2018.
18. Y. Liu et al., “Distributed energy resources survey,” Renewable and Sustainable Energy Reviews, vol. 112, pp. 545–563, 2019.
19. K. Zhou et al., “Energy internet: Business perspective,” Applied Energy, vol. 178, pp. 212–222, 2016.
20. C. Zhang et al., “Peer-to-peer energy trading,” Applied Energy, vol. 220,
21. pp. 1–12, 2018.
22. M. Yu et al., “Deep reinforcement learning for smart homes,” IEEE Internet of Things Journal, vol. 7, no. 4, pp. 2751–2762, 2020.
23. M. Zhang et al., “Deep learning in wireless networks,” IEEE Commu-nications Surveys & Tutorials, vol. 21, no. 3, pp. 2224–2287, 2019.
24. M. Hosseini et al., “Deep learning with edge computing,” IEEE Access, vol. 8, pp. 11160–11174, 2020.
25. Q. Yang et al., “Federated machine learning applications,” ACM Trans-actions on Intelligent Systems, vol. 10, no. 2, 2019.
26. T. Alladi et al., “Blockchain applications in smart grids,” Sensors, vol. 19, no. 22, p. 4862, 2019.
27. P. Siano et al., “Distributed ledger technologies for energy systems,” IEEE Systems Journal, vol. 13, no. 3, pp. 3454–3466, 2019.
28. K. Yang et al., “Edge computing for smart grids overview,” IEEE Access, vol. 8, pp. 85847–85855, 2020.
29. X. Wang et al., “Data compression in edge computing,” Journal of Network and Computer Applications, vol. 162, p. 102637, 2020.
30. J. Chen et al., “Data quality in edge computing,” IEEE Access, vol. 8,
31. pp. 107184–107197, 2020.
32. B. Cheng et al., “Adaptive scheduling in fog computing,” Future Generation Computer Systems, vol. 108, pp. 512–525, 2020.
33. S. Singh and I. Chana, “Cloud computing resource scheduling survey,” Journal of Grid Computing, vol. 14, no. 2, pp. 217–264, 2016.
34. M. Fuller et al., “Digital twin: Enabling technologies and challenges,” IEEE Access, 2019.
35. T. Cioara et al., “Digital twins for smart energy grid applications,” IEEE Access, 2021.
36. Y. Liu et al., “Intelligent edge computing for smart cities,” IEEE Network, vol. 33, no. 2, pp. 111–117, 2019.
37. S. Wang et al., “Smart factory for Industry 4.0,” Computer Networks, vol. 101, pp. 158–168, 2016.
38. L. Zhang et al., “Edge-fog architecture for smart buildings,” IEEE Internet of Things Journal, vol. 9, no. 6, pp. 4208–4221, 2022.
39. A. Kumar, “Federated learning for energy management,” IEEE Trans-actions on Smart Grid, vol. 13, no. 5, pp. 3878–3889, 2022.
40. W. Chen et al., “Reinforcement learning for energy optimization,” IEEE Transactions on Intelligent Systems, vol. 38, no. 7, pp. 5124–5138, 2023.
41. J. Huang, S. Zhou, G. Li, and Q. Shen, “Real-time monitoring and optimization methods for user-side energy management based on edge computing,” Scientific Reports, vol. 15, Article no. 24890, 2025. DOI: 10.1038/s41598-025-07592-4.