Journal of Knowledge Learning and Science Technology
ISSN: 2959-6386 (Online)
2024, Vol. 4, No. 1, pp. 61–70
DOI: https://doi.org/10.60087/jklst.v4.n1.007

Research Article

An Intelligent API Framework for Real-time Occupancy-Based HVAC Integration in Smart Building Management Systems

Sheriff Adefolarin Adepoju[1], David Olasunkanmi Segun[2]

[1]Department of Computer Science, College of Engineering, Prairie View A&M University, Texas, United States.
[2]Department of Computer Science, College of Physical Science, Federal University of Agriculture, Ogun State, Nigeria.

Abstract

Smart buildings require optimized heating, ventilation and air conditioning (HVAC) systems to improve energy efficiency and reduce operational costs. However, existing Building Management Systems (BMS) lack standardized interfaces for integrating real-time occupancy data with HVAC control. This study proposes an intelligent API framework that enhances interoperability and real-time data processing in smart building environments. The proposed framework consists of three layers: a data integration layer that harmonizes inputs from occupancy sensors and building automation systems, a processing layer that utilizes machine learning algorithms for occupancy prediction and control optimization, and a control layer that provides standardized interfaces for HVAC optimization. The results demonstrate that the intelligent API framework can attain reduction in energy use by systems compared to traditional BMS-based HVAC control. The proposed framework enables a seamless integration of emerging internet of things (IoT) technologies and facilitates the development of more sophisticated building control strategies.

Ethical Compliance: All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards.

Keywords

API, Building Management System, HVAC control, System integration, IoT, Energy Efficiency

1. Introduction

The rapid evolution of smart building technologies has fundamentally transformed modern building management systems. Today's global smart building market is predicted to reach $229.6 billion by 2025, at a Compound annual growth rate of 12.6% from 2020 (Esrafilian-Najafabadi & Haghighat, 2022).

*Corresponding author: Sheriff Adefolarin Adepoju1

Email addresses:

sadepoju1@pvamu.edu; seguno.22@student.funaab.edu.ng

Received: 01-11-2024; **Accepted:** 01-12-2024; **Published:** 25-01-2025

The increase in the demand for energy-efficient buildings has led to the growth of smart technology research. Modern smart buildings incorporate interconnected systems that support heating, ventilation and air conditioning (HVAC), lighting, security, and occupancy monitoring, generating tremendous amount of real-time data. Nonetheless, these technological developments have caused more complex problems in the management systems of these buildings. Current Building management system topologies need to integrate different protocols to manage real-time data.

This study explores the frameworks used in building automation systems and focuses on enabling connectivity between systems and devices. Frameworks which include Niagara, oBIX, BACnet, LonWorks and KNN are evaluated on their strengths and limitations, Architecture, and Components. A review of these frameworks highlights their applications in different sectors, such as like Healthcare, Energy Management, Cybersecurity, and residential system devices' control. The paper concludes by outlining the current and future trends of these frameworks and proposes a robust framework that can bridge the gap between interpolation and emerging technologies. Building Automation (BAS) is the "use of automation and control systems to monitor and control building wide systems, such as HVAC, lighting, alarms, and security access and cameras'" (cisco 2024).

A building automation system (BAS) consists of a system installed in a building that controls building services responsible for heating, cooling, ventilation, air conditioning, lighting, shading, life safety, alarm security systems, etc. Inserting these systems into a singular network infrastructure creates a smart building. Building automation started receiving attention in the 20th century as a result of the technological advancement of control systems to provide comfort to users, increase the efficiency of systems, and many more.

However, this technological advancement has introduced complex challenges to building management systems (BMS). The current BMS architectures struggle to integrate diverse protocols, manage real-time data streams, and coordinate multiple vendor-specific systems. The lack of standardization across different building automation systems has created significant barriers to achieving truly intelligent building operations. Building managers often face the challenge of managing multiple isolated systems, which lead to inefficient operations and increased maintenance costs.

The need for standardized frameworks has become increasingly critical as buildings incorporate more internet of things (IoT) devices and intelligent systems. While existing frameworks such as BACnet, Niagara, and oBIX have provided foundational capabilities, they were designed before the advent of modern cloud computing, IoT, and machine learning technologies. A standardized framework that can seamlessly integrate modern technologies while maintaining compatibility with existing systems is essential for the future of smart buildings.

Economic and environmental implications of efficient building management are substantial. Buildings account for approximately 40% of global energy consumption and 30% of greenhouse gas emissions. Studies have indicated that smart building technologies can reduce energy consumption by 20-30% through optimized operations. However, these benefits can only be fully realized with integrated systems that can effectively coordinate different building components and respond to real-time conditions. The financial impact extends beyond energy savings and includes reduced maintenance costs, improved asset utilization, and enhanced occupant productivity.

These frameworks have been in use but have lacked some important features that could accommodate current technologies. Some of these features include easy scalability, modern cyber-attacks, machine-learning incorporation capability, and less cumbersome integration. These shortcomings have fueled this research to update the current framework and lay the necessary foundation for future demands such as edge computing and growth in the number of connected devices.

Its clear that a new approach must be conceived, one that builds upon the foundations laid by these venerable systems while embracing modern technological capabilities.

2 Background and Context

We take a quick deep dive into the historical journey of building automation.

- 1970. Birth of Building Automation

Building Automation was conceived in the 1970s with the introduction of a control system to manage Heating, ventilation, air conditioning and automated HVAC systems. These systems were mostly based on pneumatic controls; although functional, they were very limited and offered little or no flexibility to the system

- 1980. Digital Revolution and Direct Digital Control

Microprocessors were introduced during this period, which marked a significant leap in Building Automation Systems. This addition allowed for more control over the system moving from pneumatic control to a more rigid based control.

- 1990. Age at System Integration

This period brought about the introduction of communication protocols such a BACnet and LonWorks, which enables different devices and systems to communicate with each other. Progress during this period laid the groundwork for the more advanced systems in the 21st century.

- 2000. Rise of IOT (Internet of Things (IoT) and smart buildings

This era introduced the advent of IOT (Internet of Things) devices being integrated into building systems. Wireless technology also gained prominence during this period.

- 2010. Shift to Predictive and Adaptive Control

Wireless technology, Artificial Intelligence and Machine Learning have begun to enhance the BAS. This shift allowed for predictive and adaptive controls, where systems could anticipate changes and adjust operations in real time, rather than merely reacting to predefined rules.

- 2020. Emergence of Autonomous Buildings

Artificial Intelligence controls the complex operations of the systems with little Human Intervention, offering autonomous decision-making capabilities.

- 2024 and beyond. Future of Smart Buildings

As we look to the future, building automation is expected to become even more integrated with smart city initiatives, where buildings will operate as part of a larger, interconnected urban ecosystem. Technologies such as quantum computing, artificial intelligence (AI), and 5G will further enhance the capabilities of building systems, enabling them to manage and maintain buildings. 2.2. Analysis of Current Frameworks

2.1 Evaluation of the Niagara Framework

The Niagara Framework is a software infrastructure that addresses the challenges of device-to-enterprise applications, albeit with certain limitations inherent to its design. It is a robust framework developed by Tridium, and the most venerable framework has been a cornerstone since its inception. Its innovative platform serves as the skeleton of smart BMS, providing an open solution for integrating various building systems and protocols." (Liang et al., 2023).

Architecture and Components

Key aspects of the architecture and components of the framework include the core control Engine, Unified Data Normalization, Custom UI and Integration Capabilities.

Strengths and Limitations

The framework provides seamless integration of building systems. The architecture approach distinguishes it from traditional building automation systems. Niagara boasts a user-friendly web-based interface, intuitive visualizations, and dashboards (Liang et al., 2023). The framework was designed to effortlessly scale. Its flexible architecture accommodates the unique needs of various buildings.

There are certain limitations of the Niagara framework that might impact its implementation and usage, and the initial implementation of the framework might require technical expertise and licensing, deployment, and maintenance could be expensive, especially for small-scale applications. The implementation of the framework's advanced feature require training to effectively utilize it.

Market Adoption and Use

Today, there are over one million instances of Niagara work in hundreds of thousands of projects worldwide. It is also widely used in healthcare, education, and manufacturing sectors. Its uses include HVAC (heating, ventilation, and air conditioning), security systems, lighting, access control, fire safety, and energy management. [https://www.tridium.com/us/en/Products/niagara].

2.2 Evaluation of the oBIX Framework

Open Building Information Exchange (oBIX) is an industry-wide initiative to define XML- and Web service-based mechanisms for building control systems. oBIX will instrument the control systems for the enterprise" (Chen & Li, 2023).

Architecture and Components

The oBIX architecture is based on the principles of object model (Chen & Li, 2023). A concise object model is used to define all oBIX information, a simple XML syntax is used to express the object model, URIs are used to identify information within the object model, and a small set of verbs is used to access objects via their URLs and transfer their state via XML.

Strengths and Limitations

The strengths of the oBIX framework include its ability to achieve semantic interoperability, web-based services, and support real-time data access. However, its limitations include scalability challenges, which may affect its performance in large-scale deployments, and a dependency on XML as a format such as JSON might be preferred.

Market Adoption and Use

The adoption of the oBIX framework has been steady compared with more established communication protocols, such as KNX and BACnet. It uses cases in healthcare, IOT applications, and other automation systems.

2.3 Evaluation of BACnet Framework

"Building Automation and Control Networks (BACnet) is a network protocol used in building automation systems (BAS) to control the data exchange between different devices and components, commonly used to control HVAC and lighting, manage security and access, monitor energy usage, and allow inter-operation with other building management systems." (Aste et al., 2017).

Architecture and Components

BACNet is built on three primary components. These are representations of information and devices as an object, communication between BACnet devices (communication protocols and selection of network technologies): some of the technologies include Ethernet, IP, Lon Talk, ZigBee, ARCnet, and MS/TP.

Strengths and limitations

One of the main strengths of the BACnet protocol is that there are many network options to choose from such as Ethernet, and ZigBee. BACnet is an open standard that allows for transparency and no licensing fees. A major limitation of the BACnet network protocol is that large scale interoperability might be an issue.

Market Adoption and use cases

BACnet is primarily used for automated heating, ventilation and air-conditioning (HVAC) systems, providing optimal and consistent remote control of HVAC and lighting which can also be managed more effectively. Consequently, the BACnet protocol has become essential for the interoperability of building management systems.

2.4 Comparative Analysis

A comparative view of the frameworks reviewed gives us an idea of how all frameworks perform against one another (Table 1).

Table 1. Comparative Table of current API Frameworks

Feature/Aspect	Niagara Framework	oBIX	BACnet
Architecture	Component-based - Proprietary core (Tridium) - Module-based extensibility	Web services based - XML-centric - RESTful architecture	Client-server - Object-oriented - Network-centric
Data Model	Proprietary (Normalized Object Model) - Fixed hierarchy	XML schema - Contract-based - Limited flexibility	Object-based - Standardized properties - Limited extensibility
Integration Capabilities	- Drivers required - Limited IoT support	HTTP/SOAP based - Web-centric	Network-specific - Protocol conversion needed
Real-time Processing	Poll-based - Limited event handling	Request-response model - Not real-time focused	COV (Change of Value) - Limited analytics
ML/AI Support	Third-party integration required - Limited native support	No native support - External systems needed	No native support
Security	Role-based access - SSL/TLS	Basic authentication - SSL support	Network security - Limited application security
Scalability	Limited horizontal scaling - Supervisor-based	Web service limitations - Single server model	Network segment limitations - Router-based
Development API	Java-based - Proprietary tools	XML/SOAP tools - Web services	- Various vendors - Protocol-specific
Cost Model	License-based - Vendor-dependent	Open specification - Implementation costs	Protocol license - Implementation costs

3. Proposed Framework Methodology

Table 2. provides a general overview of the proposed framework, which reevaluates the features. This section gives an in-depth look at how we came about the proposal.

Developing new frameworks for an API involves reevaluating features to optimize functionality, resilience, and scalability. The methodology proposes replacing traditional frameworks with new features that enable multiple users to plug without blockers.

Architecture

On the market solutions, a couple had challenges with scalability, while the only one that had scalability was restricted to proprietary solutions. There was a dependency on XML as a format that modern advances such as JSON could better serve. For this study, we seek modern architecture that combines several contemporary software design paradigms to create an adaptable system. It comprises three layers: the cloud, edge, and device. The cloud layer implements free microservice architecture using Podman containers and a GraphQL API gateway for flexible data queries. The edge layer handles the protocol translations and provide cache capabilities and local processing. This layer houses the proposed security level, which will be discussed later in the security feature. The Edge layer

also manages the device registration and monitoring.

Finally, the device layer will support multiple building protocols and abstractions. This will support plug-and-play device discovery and manage data aggregation.

Data Model

Current data models use object-based models and XML schema. This feature is placed in the cloud layer using a dynamic schema that supports GraphQL and real-time data streaming for on-the-spot decisions (El Kalach et al., 2024). This model is important for AI and ML.

The proposed data model combines modern principles to create an efficient, developer-friendly system to manage querying data.

Integration Capability

Under the integration capability of the table, we combined the support for IoT devices across protocols to process data at the edge of a network. This feature creates seamless connectivity in real-time for an IoT ecosystem.

Real-time Processing

After integration, the capability depends on how the data is processed. For this feature, we propose integrated steam processing to enable a system capable of managing continuous data flow for immediate insight. Real-time processing is adequate for systems that require rapid responsiveness.

ML/AI Support

AI and ML supports are the most prominent absentee features in the existing API framework. With machine learning and artificial intelligence, adding an ML/AL feature to the proposed framework is appropriate. This support will provide AL and ML capabilities for automatic optimization of the system.

Security: The proposal combines a Zero-Trust Architecture, OAuth 2.0, and JWT tokens to enable a secure system, making it a widely adopted feature in many industries (Sengupta & Lakshminarayanan, 2021).

Scalability

The proposed scalability is popular in the technology industry. This feature combines containerization with the proposed architecture to handle operations from edge devices in a cloud environment.

Development API

One of the critical aspects of this new proposal is to enable software developers to seamlessly integrate their IoT devices into the system. This framework combines WebSocket and RESTful/GraphQL APIs to create a solution for consuming APIs.

Cost Model

The last aspect of the proposed feature is the combination of open-source and a service-based models for rooms to monetize. This feature offers users free access to legacy and provides room to generate income if needed.

3.1. Theoretical Justification

3.1.1 Performance Evaluation

The total response time model considers the cumulative latency across all the architectural layers. The edge processing latency measures the time taken for the initial data processing and protocol translation at the edge devices. The network transmission time accounts for data transfer delays between the edge and cloud layers, including network congestion and packet loss. Cloud processing time represents the duration of data analysis, storage, and response generation in the cloud infrastructure. The system overhead accounts for the additional time required for security checks, data synchronization, and system management tasks.

Response Time Model:

$R_{total} = R_{edge} + R_{network} + R_{processing} + R_{overhead}$

Where:

- R_{edge} = Edge processing latency
- $R_{network}$ = network transmission time
- $R_{processing}$ = cloud processing time
- $R_{overhead}$ = system management overhead

3.1.2. Scalability Evaluation

The system scale factor equation provides a theoretical measure of how well system can handle increased load. The contention factor (α) represents system degradation due to resource competition and communication overhead (Table 2). As the number of nodes (N) increases, the contention factor becomes more significant, potentially limiting the linear scalability. This model helps predict the system behavior under various load conditions and identifies potential bottlenecks in the architecture.

System Scale Factor:

$S(N) = N \times (1-\alpha)$

Where:

- N = Number of nodes
- α = System contention factor (0-1)

3.3 Comparative Framework Analysis

3.1.3 Comparative Analysis of Frameworks

Framework comparison metrics

Architecture Flexibility (AF) The flexibility score measures the system's ability to adapt to different requirements and integration scenarios. Protocol support evaluates the range of automation protocols that can be integrated (Table 1). Integration capabilities assess the ease of connecting existing systems with third-party services. Extensibility options measure framework's ability to accommodate new features and technologies without major modifications.

Performance Characteristics (PC)

- Response time
- Throughput
- Resource utilization

Security Features (SF) Security features are evaluated across three main dimensions

- Authentication robustness (weighted 40%)
- Authorization granularity (weighted 30%)

Data protection mechanisms (weighted 30%)

Component Score = Implementation Level × Security Strength × Industry Standard Compliance

Where:

- Implementation Level: 1-5 scale
- Security Strength: 0.1-1.0
- Compliance: 0.8-1.2 multiplier

Comparison Matrix

Score = (AF × 0.4) + (PC × 0.3) + (SF × 0.3)

Validation Through Case Studies

Case Study Selection Criteria

- Different building types
- Varying the automation levels
- Multiple protocol requirements
- Diverse security needs

Validation Parameters

The integration complexity model quantifies the effort and resources required for the system integration. Protocol weights reflect the relative difficulty of the different protocol implementations. Interface complexity measures the sophistication of the required data transformations and mapping. Transformation complexity accounts for data format conversion and business logic implementation.

$$IC = \Sigma (Wp \times Cp + Wi \times Ci + Wt \times Ct)$$

Reliability analysis uses the Mean Time to Failure (MTTF) and Mean Time to Repair (MTTR) to predict the system availability and maintenance requirements. This model helps estimate the system uptime and maintenance costs, which are crucial for building automation systems.

$$R = MTTF/ (MTTF + MTTR)$$

Return on Investment (ROI) calculations considered:
ROI = (Benefits - Costs)/Costs.

Proposed Standardized framework

After reviewing the analysis of a Standardized framework, the following table describes how all the characteristics are tied together in comparison with the frameworks available in the market.

Table 2. *Proposed Standardized framework*

Feature/Aspect	Niagara Framework	oBIX	BACnet	Proposed Standardized Framework
Architecture	Component-based - Proprietary core (Tridium) - Module-based extensibility	Web services based - XML-centric - RESTful architecture	Client-server - Object-oriented - Network-centric	Microservices-based - Cloud-native - Event-driven - Multi-layered abstraction
Data Model	Proprietary (Normalized Object Model) - Fixed hierarchy	XML schema - Contract-based - Limited flexibility	Object-based - Standardized properties - Limited extensibility	Dynamic schema - GraphQL support - Flexible data modeling - Real-time streams
Integration Capabilities	- Drivers required - Limited IoT support	HTTP/SOAP based - Web-centric	Network-specific - Protocol conversion needed	Native IoT support - Protocol-agnostic adapters - Edge computing support

Real-time Processing	Poll-based - Limited event handling	Request-response model - Not real-time focused	COV (Change of Value) - Limited analytics	Stream processing - Event-sourcing - Real-time analytics
ML/AI Support	Third-party integration required - Limited native support	No native support - External systems needed	No native support	Built-in ML pipeline - Online learning - Automated optimization
Security	Role-based access - SSL/TLS	Basic authentication - SSL support	Network security - Limited application security	OAuth 2.0 - JWT tokens - Zero-trust architecture
Scalability	Limited horizontal scaling - Supervisor-based	Web service limitations - Single server model	Network segment limitations - Router-based	Container orchestration - Cloud-native scaling - Edge-to-cloud architecture
Development API	Java-based - Proprietary tools	XML/SOAP tools - Web services	- Various vendors - Protocol-specific-	RESTful/GraphQL - WebSocket - Modern development tools
Cost Model	License-based - Vendor-dependent	Open specification - Implementation costs	Protocol license - Implementation costs	Open-source core - Service-based model

4. Results And Findings

Considering the comparative analysis of the previous section (Table 2), a theoretical statement can be postulated that connects all the features to improve the framework.

"In distributed edge-cloud systems, the overall system effectiveness (SE) can be expressed as a function of its dynamic operational characteristics, where:

System Effectiveness (SE) = f (P, S, F, Sec, I, R, C)

Where: P = Performance efficiency under load

S = Scalability with contention factor α

F = Flexibility score across protocols

Sec = Weighted security compliance

I = Integration complexity

R = Reliability ratio (MTTF/MTTR)

C = Cost-benefit coefficient

The components exhibit non-linear interdependencies, where improvements in one dimension often create trade-offs in others. For any given resource constraint RC, there exists an optimal balance point OB* where:

OB* = max (SE) subject to:

- Performance degradation ≤ acceptable threshold
- Security score ≥ minimum compliance
- Cost ≤ budget ceiling

5. Limitation

A limitation of this study is the inability to conduct extensive testing and validation of the proposed framework due to financial constraints.

Considering this limitation, the paper has been carefully designed as a template replicable for future researchers with funding to implement and validate the postulation. All proposed features are included in this paper for effortless reproduction and validation.

6. Discussion

The study showed that the proposed standardized framework is modern, scalable, and easier to use than existing frameworks. This proposed framework adopts microservice architecture that supports real-time events and incorporates machine learning and artificial intelligence with modern security. The developer-friendly API makes the system to be robust. A drawback noticed during this study was the system's complexity in using GraphQL and the proposed micro-service architecture. The proposed system will require a more sophisticated infrastructure and expertise.

7. Conclusions

While existing frameworks can still serve traditional home automation systems, the proposed framework is a solution for the future considering advancements in machine learning, artificial intelligence, and chip development. This study advancement aims to meet the constantly evolving demands of society. Being an open-source solution makes it worthy compared to currently available solutions for intelligent automation and building management systems.

8. Declaration Of Competing Interest

We declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this study.

Acknowledgments

I would like t to thank God for giving us the opportunity to start this project.

Author Contributions

Sheriff Adefolarin Adepoju[1]: Conceptualization, data curation, Formal Analysis, Funding, Investigation, Methodology, Project administration, software, Validation, Visualization, Writing- original draft, writing- review, and editing
David Olasunkanmi Segun[2]: literature review, Writing- original draft, Writing- review & editing.

Funding

This work is self-funded after noticing the gaps in the slowness of the new API development frameworks.

Data Availability Statement

The data is available from the corresponding author upon reasonable request.

Conflicts of Interest

There was no conflict of interest in the progression of this research.

References

Aste, N., Manfren, M., & Marenzi, G. (2017). Building automation and control systems and performance optimization: A framework for analysis. *Renewable and Sustainable Energy Reviews, 75*, 313–330. https://doi.org/10.1016/j.rser.2016.10.072

Chen, W., & Li, M. (2023). Standardized motion detection and real-time heart rate monitoring of aerobics training based on convolution neural network. *Preventive Medicine, 174*, 107642. https://doi.org/10.1016/j.ypmed.2023.107642

El Kalach, F., Solanki, J., & Todkar, A. (2024). A federated information system framework for vertical integration. *Manufacturing Letters, 41*, 1192–1199. https://doi.org/10.1016/j.mfglet.2024.09.145

Esrafilian-Najafabadi, M., & Haghighat, F. (2022). Impact of occupancy prediction models on building HVAC control system performance: Application of machine learning techniques. *Energy and Buildings, 257*, 111808. https://doi.org/10.1016/j.enbuild.2021.111808

G., O. (2015). A survey of ZigBee wireless sensor network technology: Topology, applications and challenges. *International Journal of Computer Applications, 130*(9), 47–55. https://doi.org/10.5120/ijca2015907130

Grzegorz, D., & Vala, D. (2024). KNX-ZigBee gateway. *IFAC-PapersOnLine, 58*(9), 85–90. https://doi.org/10.1016/j.ifacol.2024.07.376

Liang, X., Chen, K., Chen, S., Zhu, X., Jin, X., & Du, Z. (2023). IoT-based intelligent energy management system for optimal planning of HVAC devices in net-zero emissions PV-battery building considering demand compliance. *Energy Conversion and Management, 292*, 117369. https://doi.org/10.1016/j.enconman.2023.117369

Morales-Gonzalez, C., Harper, M., Cash, M., Luo, L., Ling, Z., Sun, Q. Z., & Fu, X. (2024). On building automation system security. *High-Confidence Computing, 4*(3), 100236. https://doi.org/10.1016/j.hcc.2024.100236

Sengupta, B., & Lakshminarayanan, A. (2021). DistriTrust: Distributed and low-latency access validation in zero-trust architecture. *Journal of Information Security and Applications, 63*, 103023. https://doi.org/10.1016/j.jisa.2021.103023

Biography

Sheriff Adefolarin Adepoju is a graduate student of Prairie View A&M University, Computer Science Department. He completed his master's degree in computer science in Summer of 2024 and his master's in architecture from University of Lincoln in United Kingdom. He currently works as a software Engineer in one of the top Engineering Companies in the US.

David Olasunkanmi Segun is currently a student of Federal University of Agriculture, Abeokuta, Ogun State, Nigeria.

Research Field

Sheriff Adefolarin Adepoju: Sustainable Energy -1, Time-Series Forecasting and Analysis -2, Machine Learning for Smart Building Systems -3, Graph-Based Neural Network Applications -4, Internet of Things (IoT) in Building Automation -5

David Olasunkanmi Segun: Sustainable Energy -1, Time-Series Forecasting and Analysis -2, Machine Learning for Smart Building Systems -3, Graph-Based Neural Network Applications -4, Internet of Things (IoT) in Building Automation -5