Smart Multi-Building Energy Monitoring System

Saurabh Saluja, Colin Prochnow, Grant Couper, Filip Zivko, Xinrui Yu, Mikhail Gromov, and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract*—This paper presents an intelligent system for monitoring energy usage in smart buildings. It functions as a bridge between the smart grid and smart buildings. A web-based platform is designed to monitor sensor data and track power consumption on the grid. AI technology provides tips for energy conservation based on factors such as power usage patterns and weather conditions. The adaptability of our approach allows for the tuning of automated protocols based on evolving power usage patterns and grid conditions. Data is transmitted between the client devices and the central server via a Zigbee network organized in a star formation, allowing for mutual communication. Our implementation results in the improvement of the overall energy efficiency of supported infrastructure while ensuring continuous and optimized operation in case of a bulk grid blackout.**

Keywords—Smart Grids, Smart Buildings, IoT, Artificial Intelligence, Power Management

I. INTRODUCTION

The growing integration of renewable energy sources and smart devices presents both immense potential and novel challenges in optimizing energy efficiency. This work tackles the pivotal issue of real-time power monitoring and adaptive management for infrastructures utilizing smart grids and smart home systems. The proposed solution is a smart energy monitoring system for infrastructures utilizing smart home devices. By integrating real-time sensor data across smart devices with advanced AI-driven analysis, this system enables cutting-edge automation and optimization of energy consumption. The adaptability of our approach allows for the tuning of automated protocols based on evolving power usage patterns and grid conditions. This establishes a foundation for maximizing efficiency in everyday operations and resilience during grid failures.

Our system builds on recent advances in smart home energy management, renewable energy integration, and real-time power monitoring [1-2]. However, it uniquely combines these capabilities into an adaptable framework optimized for infrastructure-level deployment. The monitoring interface and customizable automation protocols facilitate both user-friendly operation and system-wide coordination. The proposed system also highlights a secure system to protect data throughout the smart grid. Robust security is vital for this system, as a breach could allow attackers to manipulate energy usage data or settings, disrupt operations, or use the infrastructure for malicious purposes. Implementing defense-in-depth with multiple layers of protection provides resilience against security threats [3-4]. The system features a Zigbee-based personal area network in a star topology, which ensures seamless two-way communication between the client devices (Arduino smart homes) and the central server (Raspberry Pi). This allows for fast energy-saving protocols to be implemented at all times to reduce load and therefore stress on the smart grid. A workflow overview of our system is shown in Fig. 1.

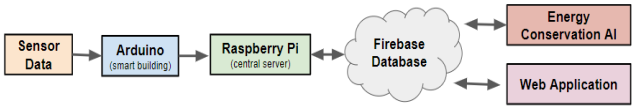

Fig. 1. Workflow Overview of the Smart Multi-Building Energy Monitoring System

Monitoring of typical power consumption patterns will enable the system to take extreme precautionary measures in instances of severe weather and/or bulk grid blackout to maximize the duration the smart grid may operate in island mode when relying solely on available stored energy. Currently, market-available products that may accomplish a similar task are smart meters that thoroughly communicate data from the meter to the grid company to closely monitor energy consumption or smart appliances that automatically have built-in power-saving features [5]. Our paper aims to take advantage of these existing products and further optimize energy consumption as well as smart grid battery life during blackouts or natural disasters. Rather than having individual smart devices scattered throughout infrastructures [6-7], we combine sensor data from numerous smart home devices working in conjunction with the sensor feedback from our power monitoring system to ensure best-case energy efficiency both throughout the day and in the instance of severe weather or blackout.

II. BACKGROUND

The current energy infrastructure is faced with various challenges, including cyber threats and natural disasters that pose a risk to the power grid. Smart grids provide a solution to these challenges by offering both conjoined and island modes of operation. Energy storage technology such as photovoltaic arrays and wind turbines, although crucial to the functioning of smart grids, have limited windows of operation. To address these limitations, this paper proposes a smart energy monitoring system for smart buildings. The system is designed to monitor and level current distribution to support infrastructure, allowing for self-healing grid capabilities. The data acquired through this system will be used to optimize energy usage with AI-powered smart devices. The system utilizes Zigbee technology to create a

secure personal area network in a star topology, facilitating two-way communication between the client devices (Arduino-powered smart homes) and the central server (Raspberry Pi). Access to the system is restricted to authorized users through authentication and access control measures. The transmitted data is also protected through encryption, ensuring the security of both transmitted and stored data. The AI component of the system will optimize energy usage by learning energy consumption patterns, predicting future energy consumption using historical data and weather forecasts, and determining when to use saved renewable energy versus energy from the grid. Overall, this paper aims to revolutionize the way energy is monitored and managed in smart buildings, providing a comprehensive solution for energy management.

III. HARDWARE

The smart multi-building energy monitoring system prototype consists of three main categories of hardware components, namely sensors, Xbee modules, and relays. An overview of the hardware components is shown in Fig. 2. Also, a test circuit is created to perform tests with the prototype. The said circuity is shown in Fig. 3.

A. Sensors

The smart building prototype utilizes several sensors to gather data about the environment including temperature, light levels, and electrical current. A DHT22 temperature sensor measures temperature in a range of -40 to 80°C and humidity to calculate heat index [8]. A light sensor takes analog inputs (0-1023) to determine light thresholds of "light", "dim", and "dark" to optimize a smart blind system [9]. A voltage-current sensor measures electrical current in the test circuitry, scales the analog output, and sends the data via Zigbee to inform an AI model and web application about real-time energy usage [10]. These sensors enable the gathering and visualization of environmental data to create a rudimentary smart building system.

B. XBee Modules

This is the main communication network for the system [11]. The XBee modules run a wireless protocol called Zigbee, and it would be running in a star network topology. This enables two-way communication to and from the central system and smart homes. The Zigbee module was correctly set up between a Raspberry Pi and a singular Arduino building. All of the temperature, light, and voltage/current sensors were connected and data was transferred. Moreover, two-way communication was achieved in our system so that the user could control the operation of certain smart home appliances via the web application. An enable signal is sent from the web application to the Firebase real-time database then the Raspberry Pi program takes it from the Firebase and sends the signal to all Arduino smart buildings. If the signal corresponds to a certain building ID and component connected to that building, the status of the component will be changed. In terms of network management, QoS policies, and traffic prioritization have been added to the system to ensure time-sensitive control commands and alerts take precedence over routine data transmissions during periods of network congestion. Alerts are generated for failed devices or degraded links to facilitate rapid response.

C. Relays

On the simulated smart home side, relays are going to be used to turn on and off "loads". These loads would consist of LEDs and resistors simulating different appliances within a home. The relays digitize the control of these through the Arduino control unit in the smart homes allowing for remote control of the smart home appliances via the web application. When an enable signal is sent through our network topology, the enable signal of the corresponding relay switch will be toggled and change the status of that particular load.

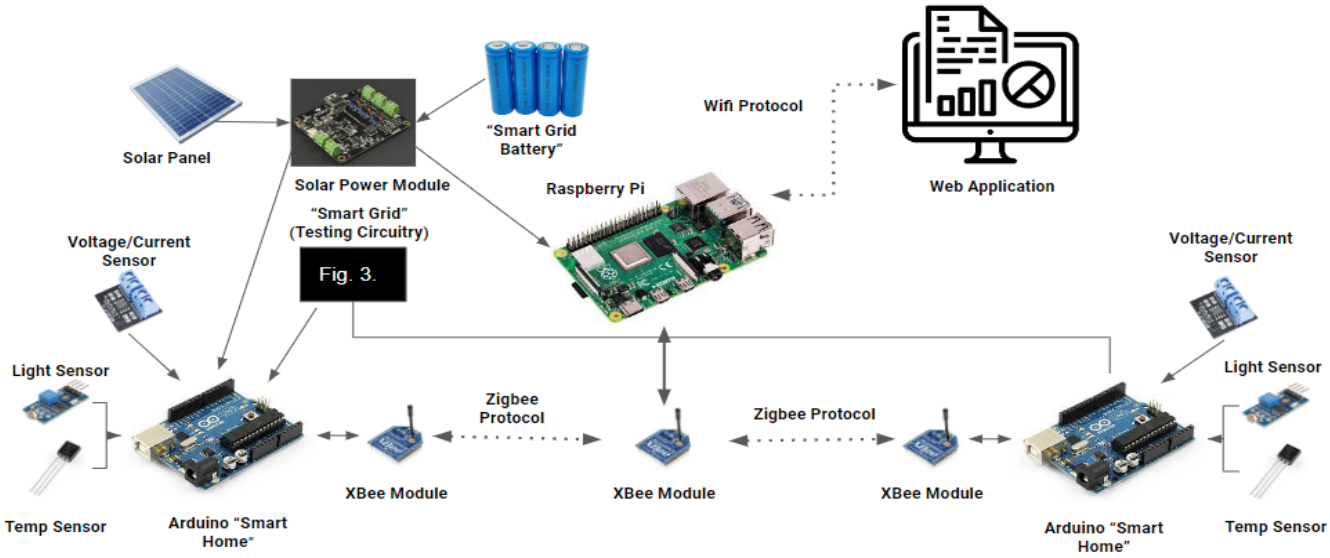

Fig. 2. Smart Multi-Building Energy Monitoring System Prototype Overview

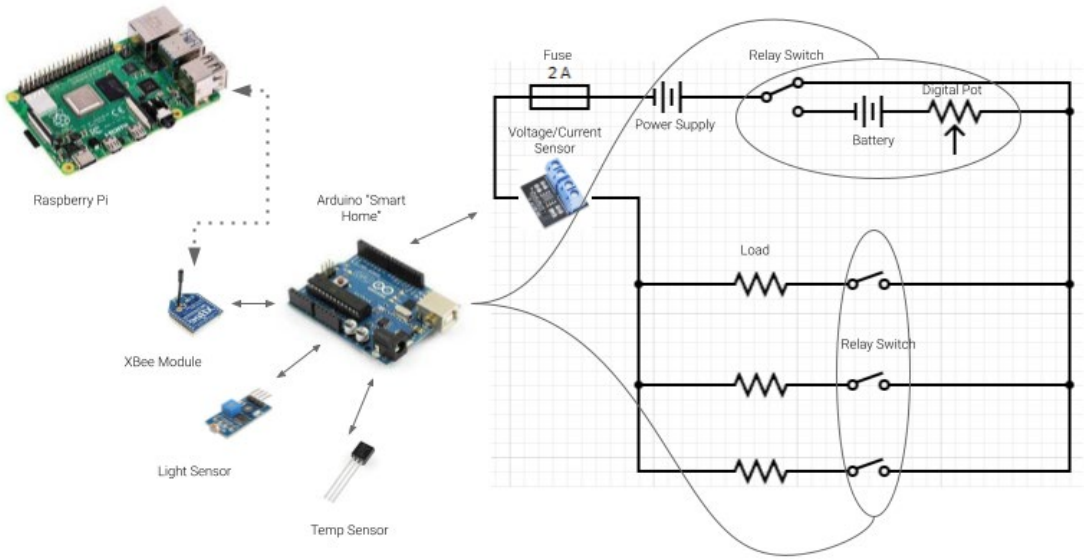

Fig. 3. Layout of the Test Circuit for Prototype

IV. SOFTWARE

A. *Web Application*

The design of the web app for this paper should prioritize user experience and ease of use as there will be many pieces of information displayed. The main features on the web page include energy monitoring, AI suggestions, and energy savings.

The main dashboard displays real-time energy usage as well as AI-generated suggestions for energy savings. Users can only view their own usage and adjust their own preferences as a building without interacting with other buildings. The main dashboard features a control panel for adjusting settings, a status panel displaying current energy usage, alerts for AI-generated suggestions, and a power graph visualizing energy consumption patterns. The main dashboard is shown in Fig. 4.

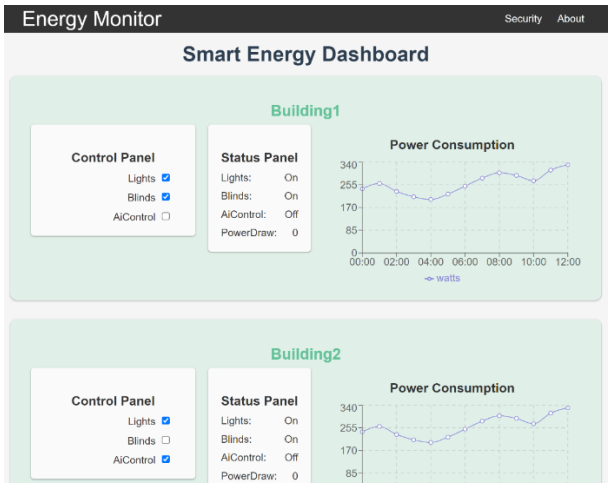

Fig. 4. Web Application: Image of Main Dashboard

On a separate security tab, authorized users can access and control the system. This section includes performance monitoring of individual devices, access management, and other security features, ensuring the system operates safely and securely. Administrators can also handle connected devices and manage authentication for users granting access only to authorized individuals. The security tab is shown in Fig. 5.

Another section of the app, the "About" tab, provides historical data and energy consumption patterns. This information helps users understand their energy usage habits and make informed decisions about energy management. The About tab briefly explains the paper's purpose, its features, and the team members involved in its creation. The web app combines functionality, security, and user-friendliness to deliver an effective energy management platform.

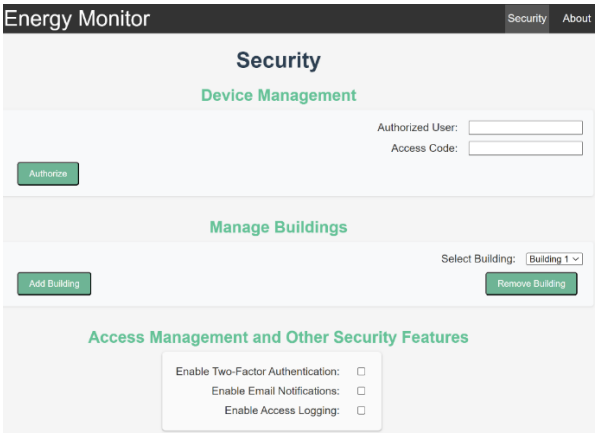

Fig. 5. Web Application: Image of Security Tab

B. *Artificial Intelligence (Predictive Decision Making)*

To predict the energy usage within our system, we have to use a machine learning model that is appropriate for the usage case. A regression model such as Random Forest is well suited for this task. A Python script that we coded employs a Random Forest Regressor model [12] to forecast power consumption based on various environmental factors. The Random Forest is an ensemble learning technique that builds multiple decision trees during training and outputs the average prediction (regression) or most common (classification) of the individual trees. By averaging results from multiple trees, the model enhances predictive accuracy and controls overfitting. The script imports the RandomForestRegressor from the sklearn.ensemble module and sets the number of trees (n_estimators) to 100.

The script starts by importing required libraries, such as pandas for data manipulation, requests for API calls, and datetime and pytz [13] for date and time handling. It also imports relevant modules from sklearn for model training, evaluation, and data partitioning.

We also make use of weather forecasts to take preventative measures. For example, if it is known that it will be sunny and warm out on a particular day based on the forecast, the system will respond accordingly. This can include actions such as lowering the blinds to block out some of the heat from the sun and beginning to start the air conditioning units earlier in the day to decrease overall costs.

To implement this, the function get_weather_data() retrieves weather information from the OpenWeatherMap API [14] for a specified city and datetime. Another function train_and_evaluate_rf_model() trains a Random Forest model on the input data and assesses its performance using the Mean Absolute Error (MAE) metric from the sklearn.metrics module.

To train this model, we had to create data that simulates the energy usage of several buildings. This simulated data contains patterns such as spikes at certain intervals (for example a spike at 8 am, when many users are turning on their computers and starting their workday). The model can predict these regular patterns in energy usage and respond accordingly.

Historical data is loaded from a CSV file, preprocessed, and divided into training and testing sets using train_test_split from the sklearn.model_selection module. The RandomForestRegressor model is then trained on this data and evaluated on the test set. The historical data used for training and testing the Random Forest model consists of the following features: building ID, date, time, current draw, light level, indoor temperature, cloudiness, humidity outside temperature, and power_usage. For this system, data over a week was collected for training and testing the model. As the system is used more often the data collected will continuously strengthen the model.

The user is prompted to input a date, time, and other feature values for a power usage prediction. The script acquires the weather data for the specified datetime, extracts pertinent information, and combines it with the user inputs to create a data frame. The Random Forest model predicts power consumption, and the result is shown to the user.

The next section includes a daily model update process, which entails fetching the current day's weather data, loading new data from a CSV file, dividing it into training and testing sets, and updating the model with the new data. The updated model's performance is assessed, and the new data is saved back to the CSV file after updating the outside_temperature column with the latest data from the API.

C. *Real-Time Decision-Making*

The system also can make decisions in real-time in response to sudden unpredictable changes in energy and environmental changes. As mentioned above, we have predictive measures in place to analyze, manage, and respond accordingly within the system. Of course, no predictive model is 100% accurate, so we must also have measures in place such that we can further improve the grid's overall efficiency. These actions would be situational, decision-based actions.

A Python script that we coded monitors real-time sensor data from a pair of buildings, obtains weather information via the OpenWeatherMap API, and dispatches alert notifications to Firebase when specific conditions arise. The script sets up a Firebase connection and establishes threshold values for various parameters, such as current, light intensity, and indoor temperature. It incorporates functions for acquiring weather data, identifying daytime, transmitting alert messages to Firebase, and evaluating sensor data for alert conditions. The primary loop consistently processes data for both structures and examines any alert situations. When a condition is satisfied, it transmits an alert notification to Firebase and pauses for 60 seconds before reevaluating.

D. *Firebase Database*

In this paper, Firebase acts as the backbone for managing and accessing data associated with energy monitoring for two structures. It links the web application, the AI algorithm, and the Raspberry Pi device, ensuring smooth interaction between all components. An overview of the wireless communication setup is shown in Fig. 6. Below is a summary of Firebase's role in each segment:

1) Web app: The web app employs the Firebase Realtime Database to store and exhibit real-time energy consumption data, AI-generated recommendations, and other pertinent information. The firebase.js file sets up the Firebase configuration, connects to the database, and exports the database object for utilization in other sections of the web app.

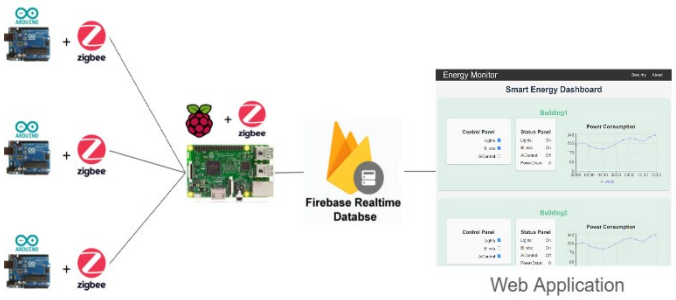

Fig. 6. Wireless Communication Setup Overview

2) AI algorithm: The AI code fetches sensor data from the Firebase Realtime Database and processes it to produce suggestions and notifications for energy management. It leverages external APIs (such as OpenWeatherMap) to collect relevant weather data for decision-making purposes. After processing the data, the AI algorithm sends alerts back to Firebase, which can then be shown on the web app. The historical code that utilizes Random Forest also uses values indirectly from Firebase, as the values at the end of the day will be grabbed from historical_data.csv.

3) Raspberry Pi code: The web app employs the Firebase Realtime Database to store and exhibit real-time energy consumption data, AI-generated recommendations, and other pertinent information. The firebase.js file sets up the Firebase configuration, connects to the database, and exports the database object for utilization in other sections of the web app.

V. SECURITY

To address security concerns, Firebase offers a variety of security features designed to safeguard data and maintain appropriate access control. With Firebase Authentication, users are authenticated before gaining access to the web app. This ensures that only authorized individuals can view and modify data. To protect against unauthorized access, our web application and central server implement multi-factor authentication using a combination of passwords, one-time codes sent to registered devices, and U2F security keys. This layered approach prevents threat actors from easily compromising user accounts. We have also set up Firebase Realtime Database Rules, where we can control the read and write access to our specific database. This means that users can only interact with data they are authorized to access (thus preventing unauthorized access or tampering). Database Rules can also validate incoming data, ensuring that only data meeting specific criteria can be stored in the database. This helps us maintain the data integrity of our system. Finally, Firebase employs SSL/TLS encryption to protect data transmitted between the client and server, ensuring that data in transit remains secure and uncompromised.

Next, the Zigbee protocol, which is used by XBee modules, has built-in security features designed to protect communication between devices. First, Zigbee leverages Advanced Encryption Standard (AES) with a 128-bit key to encrypt data transmitted between devices. This security measure ensures data is safe from eavesdropping and tampering during transmission. Next, Zigbee uses link keys to protect communication between individual devices, and network keys to protect communication within the entire network. These keys, combined with AES-128 encryption, provide secure communication channels. Additionally, Zigbee includes key management features that securely generate, distribute, and update keys. This guarantees that keys stay secure and up-to-date, minimizing the likelihood of unauthorized access. Finally, Zigbee devices can authenticate each other before establishing a secure communication channel. This stops unauthorized devices from joining the network or posing as legitimate devices.

The security tab on the website provides users with the ability to oversee encrypted data transfers between the Arduino units and the central Raspberry Pi, guaranteeing the accuracy and reliability of the exchanged data. Additionally, the tab offers user management options for authentication and access control, enabling access exclusively for authorized individuals. Users can create, modify, or revoke usernames and passwords as required. Moreover, the security tab allows administrators to handle connected devices within the system effectively.

We also have authentication and access control. What this means is that only authorized users have access to, and the ability to control the system. We require users to have a username and password to access our system, which will be behind the web application, which we described earlier in this paper.

Finally, we have device management. This allow us for example to remove an Arduino from the system if it has been compromised or stolen.

VI. RESULTS

From the goals our team set out to accomplish, we made significant progress and were ultimately able to show proof of concept and develop the structure for our system. Since the system is highly customizable and scalable, we wanted to make sure that the most vital components of our system were implemented and functioning.

The web application user interface was developed and communicated with both the AI algorithm implemented as well as with our Firebase real-time database to collect and visualize the sensor data. Each Arduino smart building was connected to a temperature/humidity sensor, current sensor, and photosensitive sensor, as well as to a router XBee module for two-way wireless communication.

Not only did our system push real-time sensor data to the Firebase via the Raspberry Pi coordinator device, but we were also able to implement relay switches so that a user could manually control the functions of smart home appliances through the web application. This was demonstrated by the model circuit's loads being turned on or off to decrease the current draw if the current went past a designated threshold. This was accompanied by the AI algorithm displaying suggestions/warnings on the web application for energy consumption optimization. Additionally, we had a portion of our system powered by a separate backup battery as well as charged by a photovoltaic array to mitigate the system going down in case of a blackout/brownout in the infrastructure's power grid.

Some challenges we faced included finding a common sampling rate for all of our sensor data to be sent at the same time which was ultimately constrained by the DHT22 sensor having a .5Hz sample rate as well as the fact that we could not flood the Zigbee with too much data all at once or we would run into packet collision errors and only a fraction of the data would be sent.

The same issue was faced in implementation of the two-way communication as packet collision in the other direction caused significant latency when turning off devices through the web

application. This is also because the XBee modules cannot transmit and receive data simultaneously and instead require some amount of time of separation between when it receives and transmits, mitigated by the utilization of delays within our Arduino program.

Moreover, we had challenges developing training data for the AI as it would require multiple weeks of the system running and collecting variable data to tune itself to make proper suggestions, however, the real-time portion was able to function properly and display suggestions on the web application.

VII. Conclusion

The smart energy monitoring system aims to improve energy efficiency in smart buildings by integrating with the smart grid and smart buildings. The system offers real-time power monitoring through a web-based interface and AI-generated recommendations for energy savings. A Zigbee network configured in a star topology ensures client and server communication.

As our system is highly customizable and scalable, many future changes could be made highly dependent on the infrastructure in which it will be implemented. Since we did not have access to a full-scale model, the current sensor is an Arduino-based current sensor meant for small electronic applications, however, the current sensor as well as the other sensors may need to be scaled up to higher quality versions to suit the needs of a particular application.

Additionally, more sensors could be added to test for other variables not tested for in our system for a more complete and detailed energy-optimizing system. The AI could be highly optimized in the future once a significant capacity of training data has been collected from infrastructures with smart home devices to pick up on energy consumption trends over a longer period. The web application will have to be optimized to visualize any further sensor data that gets implemented as well as customize the device control section to operate in conjunction with features available within smart home appliances. As of right now the feature simply turns on/off the device while in reality, we would like for the thermostat control feature for example to be able to adjust the temperature remotely or turn off or dim specific lights in portions of an infrastructure that are either unnecessary or being unused to save energy. Additionally, for practical implementation, strict network segmentation and micro-segmentation policies would have to be included to contain any potential breach. The smart home device networks would then be isolated from the central management network, limiting lateral movement opportunities for attackers. Additionally, firewalls should be put up so individual smart homes are protected from each other.

This system is poised to bring about a major change in energy management in smart buildings, due to its comprehensive and effective solution for smart energy monitoring.

References

[1] I. Priyadarshini, S. Sahu, R. Kumar, and D. Taniar, “A machine-learning ensemble model for predicting energy consumption in smart homes,” Internet of Things, vol. 20, p. 100636, 2022.

[2] C. Liu, B. Sun, C. Zhang, and F. Li, “A hybrid prediction model for residential electricity consumption using Holt-Winters and extreme learning machine,” 2020.

[3] W. Wang and Z. Lu, “Cyber security in the smart grid: Survey and challenges,” Computer Networks, vol. 57, no. 5, pp. 1344–1371, 2013.

[4] N. Abosata, S. Al-Rubaye, G. Inalhan, and C. Emmanouilidis, “Internet of things for system integrity: A comprehensive survey on security, attacks and countermeasures for Industrial Applications,” Sensors, vol. 21, no. 11, p. 3654, 2021.

[5] “Smart Home: The Smart Grid.” *Smart Home: The Smart Grid | SmartGrid.gov*, 16 Dec. 2019, https://www.smartgrid.gov/the_smart_grid/smart_home.html.

[6] M. Yesilbudak and A. Colak, “Integration challenges and solutions for renewable energy sources, electric vehicles and demand-side initiatives in smart grids,” 2018 7th International Conference on Renewable Energy Research and Applications (ICRERA), 2018.

[7] K. M. Tan, T. S. Babu, V. K. Ramachandaramurthy, P. Kasinathan, S. G. Solanki, and S. K. Raveendran, “Empowering smart grid: A comprehensive review of energy storage technology and application with Renewable Energy Integration,” Journal of Energy Storage, vol. 39, p. 102591, 2021.

[8] “DHT22 Temperature Sensor Datasheet”, https://wiki.dfrobot.com/DHT22_Temperature_and_humidity_module_SKU_SEN0137

[9] “Photosensitive sensor module”, http://www.energiazero.org/arduino_sensori/photosensitive_sensor_module.pdf

[10] “DKARDU Voltage Current Sensor Module“, https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf

[11] “Xbee Guide.” *XBee Guide - SparkFun Electronics*, https://www.sparkfun.com/pages/xbee_guide.

[12] “Sklearn.Ensemble.Randomforestregressor.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[13] Bishop, Stuart. Pytz, pypi.org/project/pytz/.

[14] “One Call API 3.0” *OpenWeatherMap*, https://openweathermap.org/api/one-call-3