Sciences Methods and Technologies International Journal (SciMeTech)

ISSN: 3085-5284, journal homepage: www.scimetech.com

AI-Driven Smart Energy Management in IoT-Enabled Buildings

ALOUACH Abdennour[1], BENKIRANE Said[2], GUEZZAZ Azidine[2], LAMAAKAL Ismail[1]

[1] Multidisciplinary Faculty of Nador, Mohammed Premier University, Oujda 60000, Morocco

[2] Higher School of Technology Essaouira, Cadi Ayyad University, Marrakech 40000, Morocco

How to cite

ALOUACH Abdennour, BENKIRANE Said, GUEZZAZ Azidine, LAMAAKAL Ismail., "AI-Driven Smart Energy Management in IoT-Enabled Buildings", Sciences Methods and Technologies International Journal (SciMeTech), Vol 2, Issue 1, p 116-123

Keywords:
IOT-enabled building
Smart Building
Energy Management
Time-Series Data
MQTT
Node-RED
InfluxDB
Prophet
Real-Time Feedback Control
Artificial Intelligence

Abstract

The presented study considers energy management in buildings which is a crucial issue that requires effective solutions. Specifically, it should be aimed at monitoring energy consumption, detecting potential problems and solving them automatically. In many cases, existing energy management systems are not able to make predictions regarding the possible development of circumstances and to perform actions automatically in response to abrupt changes in electricity consumption. In order to solve these shortcomings, this paper offers a holistic energy management architecture for an IoT-enabled building. It includes a multi-agent simulation using Node-RED which models realistic power consumption in functional rooms in the building, while InfluxDB is used for optimized storage of time series data. As for the reliability check, the process of dynamic anomaly injection, allowing for simulation of faults such as high-power spikes and blackouts, is used. Artificial intelligence forecasting model based on Prophet is used to predict energy consumption within six hours in the building while the threshold values for such analysis will be adjusted depending on season using Prophet's API. Once the deviation from predicted consumption is detected, real-time feedback control using MQTT will provide automatic isolation of circuits. According to findings, the system works fast enough, in about six seconds. The solution proposed in this paper can be implemented not only in smart buildings but also in the entire smart grid. In the future, reinforcement learning can be used to facilitate the work of a smart grid and to prevent faults in it.

I. Introduction

With growing energy demands, smart buildings seem to be the most efficient option when attempting to address climate change issues [1][2][3]. This is due to the high availability of IoT devices and their capabilities of conducting real-time observations. It is worth noting that there is a big difference between collecting and applying information. Modern buildings operate on technologies that help to track events from the past. However, the prediction of future incidents or even solving problems in a timely manner in order to maintain the integrity of the energy grid is currently impossible [4]. Hence, the significance of smart buildings and sustainable energy systems cannot be underestimated.

The design of smart buildings and green energy systems needs to conform to the legal requirements imposed on those systems. Smart buildings and sustainable energy grids should be created. The first problem concerns lack of knowledge of certain phenomena, namely, voltage fluctuations, blackouts, etc. They significantly affect intelligent buildings and energy systems. In this regard, lack of sufficient knowledge about such phenomena is considered one of the main impediments in the development of smart buildings and energy technologies.

In case machine learning technologies do not have opportunities to stress-test their algorithms, they will work poorly in the real world. What is the goal of developing these systems? It is important to create systems that would do something

other than simply visualize and solve existing problems. To this end, the following research offers a smart energy management system aimed at increasing autonomy and sustainability.

Three technical components were used in order to achieve this objective:

1. The Node-RED platform was used to simulate complex interactions of sensors and actuators in residential areas and to model the entire house.
2. Time-series database influxdb together with the Prophet algorithm was used to conduct forecasts and spot any anomalies.
3. Automatic protector Auto-Protector uses the MQTT protocol to activate the virtual breaker or load shedding processes [5][6].

This paper describes the process of self-learning to improve the efficiency of energy management. The process involves data collection, predictions, and further feedbacks to produce more accurate predictions. All steps of the process are illustrated in the following figure (Figure 1).

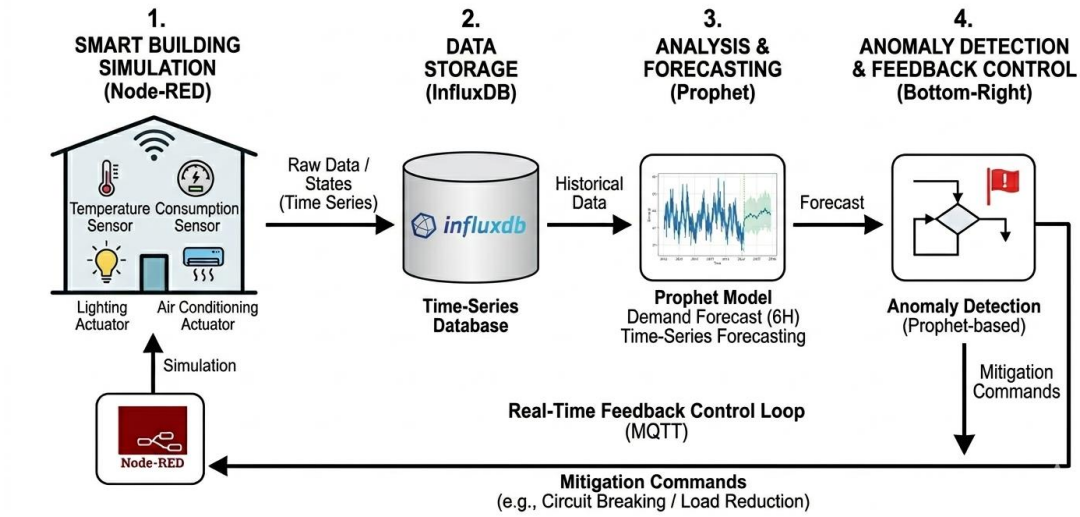

Figure 1. The global structure of the IoT energy management system and the four stages of the pipeline of data processing from simulations to feedback controller.

II. Methods

This part discusses the deployment of the recommended intelligent energy management system within four main areas: Node-RED simulator, management of the InfluxDB database including downsampling and retention policy, forecast using the Prophet model, and feedback controller using the MQTT protocol.

a. Node-RED Simulation Environment

The Node-RED framework is used to design a realistic home environment. Instead of creating random numbers, the Node-RED framework captures the timeline data from each individual zone, thus ensuring the creation of realistic usage of energy.

In particular, the framework implements a structure of the building consisting of six different zones, including the Living Room, Kitchen, Bedroom 1, Bedroom 2, Bathroom, and Guest Room/Hallway. Each of these zones includes certain sensors and control units, listed in Table 1 below. Realism in energy consumption in each zone is achieved by implementing temporal dynamics and internal mechanisms.

Table 1. Smart Home Zone Configuration - Sensors and actuators for each simulated room

Zone	Sensors	Actuators/Loads
Living Room	Temperature, Presence	HVAC (1000-3000W), Lighting (40W), TV (150W)

Zone	Sensors	Actuators/Loads
Kitchen	Fridge cycle, Oven state	Refrigerator (1000-2000W), Oven (2500W)
Bedroom 1	Temperature	Heater (750-2000W), Standby (15W)
Bedroom 2	Temperature	Heater (600-1200W)
Bathroom	Humidity	VMC (500W), Boiler (1500-4000W)
Guest Room	Passage detection	Heating (600-2000W), Lighting (10W)

The simulation engine works in steps of five seconds. It updates all the information from the sensors and the things that can be controlled. This is done to make the simulation look like what happens in life. For example the heating and cooling system in the Living Room changes how power it uses. It does this based on how hot or cold it's if someone is in the room. The system is on when people are usually in the room which's from 7:00 to 23:00. The Kitchen has a way of working for the refrigerator. It works for ten minutes using a lot of power which's 2000W and then it stops working for twenty minutes using less power, which is 1000W. The oven in the Kitchen works at certain times of the day which is from 18:00, to 20:00. When it is working it uses a lot of power which's 2500W and it does this for a certain amount of time which is ten cycles.

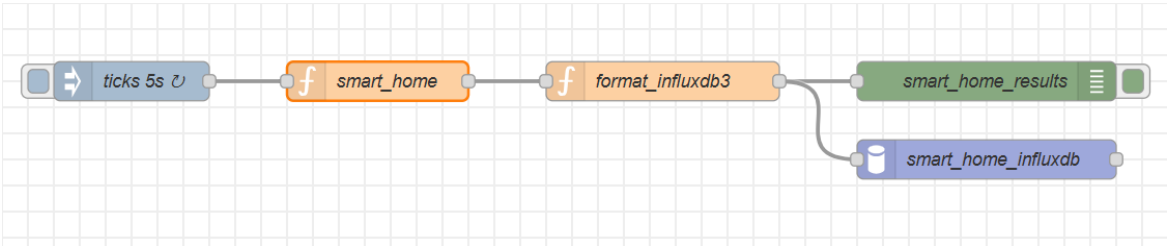

Figure 2. Node-RED Simulation Pipeline - Visual representation of the simulation flow including the 5-second ticker, smart home calculation node, and InfluxDB storage integration with flow context state management.

b. Anomaly Injection

In order to test the fault detection process, synthetic faults will be inserted into the simulation during the start of every tenth minute (when minute mod 10 = 0 & second < 2). The following fault scenarios are introduced with their respective probabilities of occurrence:

- 70% - Spike High
- 20% - Blackout
- 10% - Sensor Error

In case of Spike High, there is an additional random load between 6,000 to 11,000 W added to a random room that causes a temporary increase in its contribution and hence an increase in global power above the normal baseline level. In Blackout scenario, the sum of all rooms becomes zero, simulating a disruption in supply. In case of Sensor Error, there is a negative value generated in the total power of a random room. For instance, total power of kitchen will be -100 W. This scenario replicates a failure of signal processing unit and/or current transformer. These kinds of values will be very useful while calibrating the IoT system during its idle phase.

c. Database Management using Downsampling and Retention

The InfluxDB is a database that is suitable for maintaining information overtime. This database shows great performance when it comes to handling huge amounts of rapidly ingested data, which makes it suitable for keeping data from sensors [7]. InfluxDB uses hierarchy for data management, as well as down-sampling for data analysis and storage policy. The measurements in the smart home system are made every five seconds, giving 17,280 records daily. To keep

track of those data points, while still having the ability to perform analysis, some strong data management techniques have to be employed.

Down-sampling merges data collected during a five-minute period into one value through averaging and finding the maximum value of the period. After that, this data is saved to another measurement called "smart_home_5min." This gives us a context of what is happening, and helps detect trends while reducing the size of the data collection. According to the storage policy, original data will be deleted after thirty days, while forecasted data will be deleted after seven days. With this, storage space can be optimized while still having enough data to train models. This database has a built-in scheduler, which allows you to execute certain processes on intervals. In this project, the processes were executed once a day.

d. Forecasting based on Prophet

The forecasting module uses the Prophet algorithm from the Prophet package developed by Facebook [8], which is the time series decomposition model that adds the seasonal component. It is best suited for the situations when the data has pronounced seasonal changes and many periods in the past. In the algorithm specification, seasonality at the level of one day (hourly consumption), seasonality per week (weekday vs. weekend consumption), and seasonality per hour with improved granularity using Fourier order equal to 5 were used. Multiplicative seasonality is used as a default.

In order to produce predictions, the system uses 168 hours of history from InfluxDB. The data is gathered for seven days and used to train the Prophet model to make predictions for 6-hour horizon, making updates every five minutes. The result will be the forecasts denoted as yhat with the prediction intervals defined by yhat_lower and yhat_upper values. These forecasts are saved to the database to be visualized and compared to thresholds. The major benefit of the forecast is the ability to set season-based thresholds for anomaly detection in case of fluctuations.

The dynamic threshold approach sets the thresholds depending on seasonal changes and historical maxima. The approach uses three different modifiers:

- Winter – multiplied by 1.5
- Mid-season – multiplied by 1.2
- Summer – multiplied by 1.0

It should be noted that the heating load increases in winter, thus some adjustments are needed. For example, it will decrease the chances of receiving false signals but will increase the risk of missing out on true anomalies. To find the limit, we take the value under which fall 95% of the energy usage history for the last seven days. The threshold found is used as a base threshold for the energy use, adjusted by season to receive the final limits.

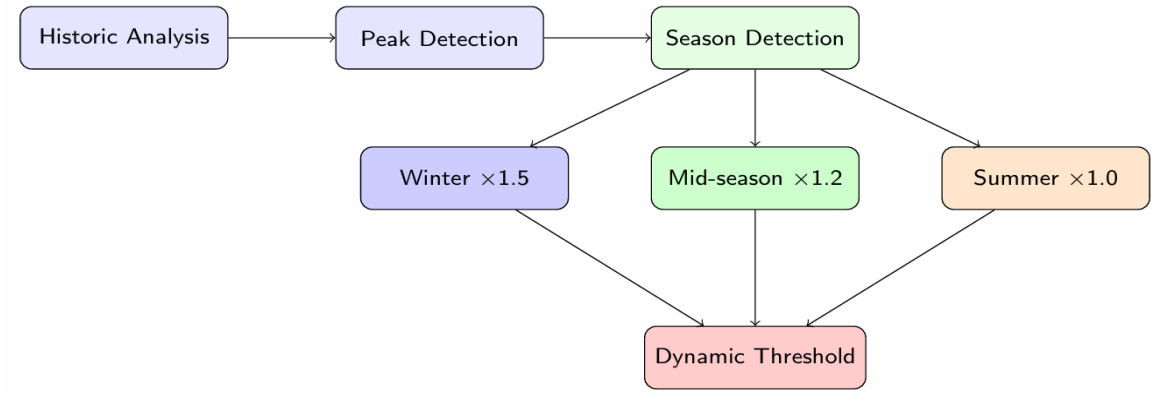

Figure 3. Real-Time Anomaly Detection using Dynamic Thresholding Algorithm

e. Real-Time Feedback Control via MQTT

The feedback control module, found in the feedback_control.py file, polls InfluxDB every five seconds. It analyzes the detected reading and finds out if it falls into any of the four categories: Normal, Sensor Error, Blackout, or Spike High. Specifically, the Sensor Error condition occurs whenever global_power < 0, while the Blackout is characterized by

global_power == 0, and Spike High corresponds to global_power > 6000 W. When detecting the Spike High condition, the feedback control module detects the room responsible for high power usage by analyzing six fields in the InfluxDB record where each field relates to one of the rooms. Afterward, the module sends a message to the MQTT topic home/control/command. The message is a JSON and tells the room that uses too much power and what should be done regarding the power (power_off).

Node-RED uses the MQTT-in node to subscribe to the specified topic and uses a function node called update_breaker_status to decode the incoming command. Thus, it sets the flag for the room based on which power should be switched off (true or false), updating the global circuit_breakers object in the meantime. This technique allows Node-RED to keep track of the status of the circuit breakers. At the next simulation tick, the main smart_home function will read this object and set zero total power for this room before calculating the new global consumption level. In this way, a closed loop controlling one of the components in the system is formed. However, its effect duration is limited by how often it performs check-ups. Since checking occurs every five seconds, the minimum required time to complete the process of executing the action and observing the results is ten seconds (two checks).

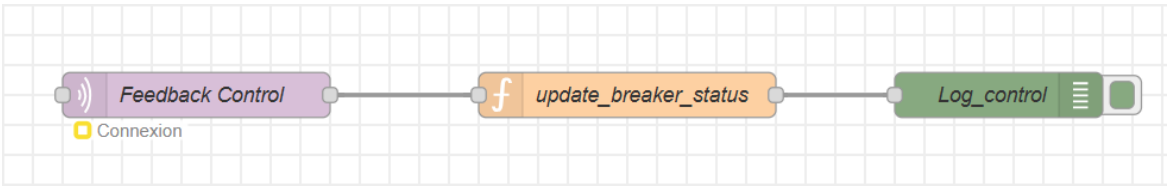

Figure 4. Node-RED feedback control pipeline: the MQTT subscriber passes commands to the circuit-breaker update function, which modifies the global actuator state.

f. Alert System

power_alert.py is a Python module that creates alerts when power consumption surpasses certain defined levels. This process happens independently and does not disrupt the feedback control loop. Whenever the power consumption becomes greater than the set level by Prophet, the module calculates the amount of deviation and sends an alert to the Node-RED node.

After receiving the alert in the Node-RED setup, the alert is tagged with a timestamp and a severity label before being logged and sent to the appropriate dashboard screens. This approach keeps the system simple and makes it easy to connect it to other alerting systems like email or SMS. In other words, the power_alert.py script acts as an alert creator, and alerting is handled separately from the rest of the systems. Thus, incorporating other alerting options like email and SMS into the process becomes easier.

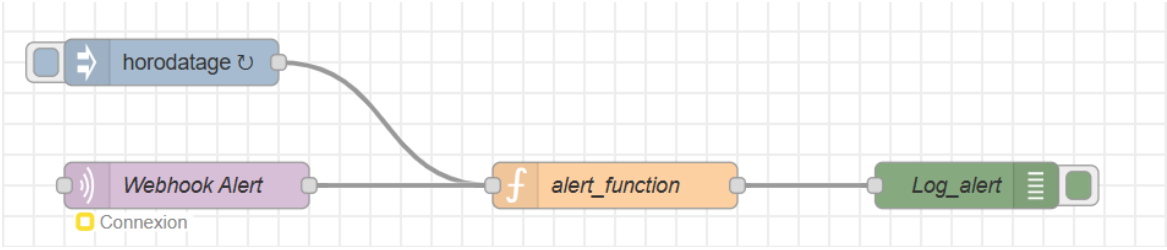

Figure 5. Node-RED alert pipeline: the webhook receives Python alerts, the alert_function formats and enriches the payload, and Log_alert records it persistently.

III. Results and discussion

The developed system shows successful implementation and integration of all components considered in the design, including simulation, prediction, and control phases. During testing lasting 7 days with continuous injection of anomalies, we have found the system consistently detected any problem and reacted swiftly. Below, the results of the analysis for each system component are described.

a. Real-Time Dashboard and Data Visualization

In order to have an operational view of what is happening in the smart building, the output of the system will be shown via a real-time dashboard (Figure 5). This acts as the main medium to monitor the amount of electricity that is consumed in individual zones as well as total demand in the building.

From the dashboard, we are able to see how much power is being consumed worldwide, which is (7955W). There is also information on power consumption in six different zones. From the dashboard, we see that the total power consumption is broken down into various components like the amount of power used by the kitchen, which is (2800W) and the amount of power used by the living room, which is (1190W). There are also clear indications about which zones are consuming more power from the dashboard. In this case, the kitchen zone consumes the power, and this is expected considering the amount of power used by the fridge and oven.

This amount of detail becomes very necessary when looking for the location where the Spike High problem is occurring, since it ensures that the system is able to address only the area affected by the problem and not the entire building. The power usage will help to achieve this.

Figure 6. Real-time monitoring dashboard displaying global and per-zone power consumption metrics.

b. Power Consumption Forecasting Results

The Prophet forecasting model successfully captured the daily consumption patterns characteristic of residential buildings. Table 2 presents sample forecast outputs demonstrating the model's ability to predict the evening consumption peak, a critical period for anomaly detection. The forecast shows increasing consumption from 15:05 (6120.5W baseline) to 15:50 (12500.6W peak), corresponding to typical evening activities including meal preparation and heating system activation.

Table 2. Power Consumption Forecast - Sample predictions for evening period showing upward trend

Time	yhat (W)	yhat_lower	yhat_upper	Trend
15:05:00	6120.5	6050.2	6195.8	Base
15:25:00	8100.2	8010.5	8195.6	Rising
15:35:00	9750.4	9635.7	9865.9	High
15:50:00	12500.6	12380.2	12625.9	Peak

c. System Latency and Resource Usage

The time it took for the whole process to happen. From when a weird value was added to InfluxDB to when the circuit-breaker flag was turned on in Node-RED. Was always between 5 and 10 seconds. We tried this 50 times. Got the same result. The main reason for this delay was the 5 second gap between checks by the Python feedback module. When we looked at how it took for MQTT messages to be delivered it was really fast. Less, than 20 milliseconds. When we tested it on our own computer. The Memory footprint for all Python processes combined was really small it stayed below 350 MB. The CPU utilization on a single-core test machine was also very low it stayed under 15% when things were normal. When the Prophet was retrained, which happens every 30 minutes the CPU utilization went up to 60-70% for a little while.

The InfluxDB downsampling task was very helpful it reduced the raw data volume by a lot, 60 times smaller for records that are older than seven days. The scheduled deletion of raw points which are 30 days old and forecast records, which are 7 days old helped keep the database size stable. The database size stayed under 200 MB after a simulated two-month deployment, which is a pretty long time. The Memory usage for all the Python processes in aggregate and the InfluxDB database storage requirements were reasonably controlled.

d. Comparison to Related Work

In contrast to energy management systems based on rules that depend on predetermined schedules or thresholds [9][10][11][12][13], our architecture is capable of demand forecasting that considers temporal and seasonal changes. As opposed to advanced deep learning models like Long Short-Term Memory (LSTM) networks used in anomaly detection [14][15][16], the proposed pipeline sacrifices some predictive power but gains significant efficiency. The use of MQTT as the actuation channel aligns with established IoT messaging standards and ensures interoperability [17] with physical hardware controllers that support the same protocol, making the transition from simulation to real deployment straightforward.

IV. Conclusion

This paper is about an energy management system for buildings. It is a system that uses the internet of things. The system has parts, including simulation, storage, machine learning, anomaly detection and feedback control. The people who made this system used Node-RED, InfluxDB, Prophet and MQTT. They worked together to make a system that can automate energy use. This system is special because it is free to use and does not need a lot of computer power. The system was. It worked very well. It can detect problems in, than 10 seconds. It does not make warnings when everything is working normally. The system can also store a lot of data for a time by using a special way of keeping only the important information.

There are ways we can take this project in the future. We could make the feedback system better by making it turn back on by itself after a while so people do not have to do it themselves when the power comes back on. The part of the system that makes predictions could use things like the temperature or how many people are in the building to make better guesses when the seasons are changing. If we put the system on hardware like using real current transformers and smart plugs we can see how it works in the real world with things like delays in the network and noise from the sensors. This would be an improvement over just simulating it with Node-RED. We could also use something called learning to let many buildings share information about how much energy they use but in a way that keeps the details private so we can make better predictions without needing to know everything, about each building.

References

[1] Ghaffarianhoseini, A., et al. (2016). The essence of future smart houses: From embedding ICT to adapting to sustainability principles. *Renewable and Sustainable Energy Reviews*, 68, 587–603.

[2] Al-Hussein, M., Al-Sakkaf, L., & Al-Habaibeh, A. (2025). AI-driven transformations in smart buildings: A review of energy efficiency and sustainable operations. *Energy Nexus*, 10, 100342.

[3] Muniandi, B., et al. (2024). AI-Driven Energy Management Systems for Smart Buildings. PowerTech Journal, 1, 280. DOI: 10.52783/pst.280.

[4] Fang, X., Misra, S., Xue, G., & Yang, D. (2012). Smart grid — The new and improved power grid: A survey. *IEEE Communications Surveys & Tutorials*, 14(4), 944–980.

[5] Banks, A., & Gupta, R. (2014). MQTT Version 3.1.1. *OASIS Standard*, 1–81.

[6] Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). MQTT-S — A publish/subscribe protocol for wireless sensor networks. *IEEE CCNC*, 302–306.

[7] O'Donovan, P., Leahy, K., Bruton, K., & O'Sullivan, D. T. J. (2015). An industrial big data pipeline for data-driven analytics maintenance applications. *Journal of Big Data*, 2, 25.

[8] Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37–45.

[9] Afram, A., & Janabi-Sharifi, F. (2014). Theory and applications of HVAC control systems – A review of model predictive control (MPC). *Building and Environment*, 72, 343–355.

[10] Ahmad, T., & Chen, H. (2019). Short and medium-term forecasting of cooling and heating load demand in buildings: A review. *Energy and Buildings*, 182, 25–37.

[11] Pipattanasomporn, M., Kuzlu, M., & Rahman, S. (2012). An algorithm for intelligent home energy management and demand response analysis. *IEEE Transactions on Smart Grid*, 3(4), 2166–2173.

[12] Kolokotsa, D., Pouliezos, A., Stavrakakis, G., & Lazos, C. (2009). Predictive control techniques for energy and indoor environmental quality management in buildings. *Energy and Buildings*, 43(2–3), 368–376.

[13] Wang, S., & Ma, Z. (2008). Supervisory and optimal control of building HVAC systems: A review. *Renewable and Sustainable Energy Reviews*, 13(6–7), 136–143.

[14] Mocanu, E., Nguyen, P. H., Gibescu, M., & Kling, W. L. (2016). Deep learning for estimating building energy consumption. *Sustainable Energy, Grids and Networks*, 6, 91–99.

[15] Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., & Zhang, Y. (2017). Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE Transactions on Smart Grid*, 10(1), 841–851.

[16] Marino, D. L., Amarasinghe, K., & Manic, M. (2016). Building energy load forecasting using deep neural networks. *Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Florence, Italy, pp. 7046–7051.

[17] Blackstock, M., & Lea, R. (2014). IoT interoperability: A practical approach. *IEEE World Forum on Internet of Things*, 1–6.