IJIE

INTERNATIONAL JOURNAL OF INTEGRATED ENGINEERING
ISSN: 2229-838X e-ISSN: 2600-7916

Vol. 18 No. 3 (2026) 424-441
https://publisher.uthm.edu.my/ojs/index.php/ijie

Edge Deployment of Behaviour-Based Smart Home Automation System Using LSTM on Raspberry Pi for Real-Time Electrical Load Control

Mochamad Susantok[1,2], Farhana Ahmad Poad[1]*, Ariffuddin Joret[1], Agus Urip Ari Wibowo[3], Joni Chang[4]

[1] *Faculty of Electrical and Electronic Engineering, University Tun Hussein Onn Malaysia, Parit Raja, Johor, 86400, MALAYSIA*
[2] *Telecommunication Network Engineering Technology Department, Politeknik Caltex Riau, Pekanbaru, 28265, INDONESIA*
[3] *Master of Applied Computer Engineering Department, Politeknik Caltex Riau, Pekanbaru, 28265, INDONESIA*
[4] *PT PCR Solusi Teknologi, Pekanbaru, 28265, INDONESIA*

*Corresponding Author: farhana@uthm.edu.my
DOI: https://doi.org/10.30880/ijie.2026.18.03.028

Article Info

Received: 14 October 2025
Accepted: 19 April 2026
Available online: 30 April 2026

Keywords

Smart Home Automation System (SHAS), Edge Computing, Long Short-Term Memory (LSTM), overload prevention, adaptive learning

Abstract

This paper presents the edge deployment of a behavior-based Smart Home Automation System (SHAS) designed for proactive electrical overload prevention. The term behavior-based refers to modeling appliance control decisions using historical usage patterns and contextual sensor data rather than predefined deterministic rules. The proposed system integrates IoT sensors and a Long Short-Term Memory (LSTM) model running on a Raspberry Pi 4 edge platform. The dataset was collected continuously over one week with a one-minute sampling interval, resulting in a real-world time-series dataset representing household energy usage behavior. Model performance was evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score, along with system-level evaluation using cumulative overload duration. Experimental results show a significant 59.6% reduction in total overload duration compared to baseline operation without adaptive control. This demonstrates the effectiveness of the proposed approach compared to conventional rule-based or non-adaptive systems. The system also maintains low-latency communication and stable resource utilization. Overall, this work demonstrates the feasibility of deploying deep learning-based energy management entirely on edge devices, providing improved responsiveness, reduced internet dependency, and enhanced data privacy in smart home environments.

1. Introduction

The rapid advancement of Artificial Intelligence of Things (AIoT) has significantly transformed the way Smart Home Automation Systems (SHAS) operate. While conventional SHAS often rely on deterministic logic and cloud-based platforms for remote functions, the integration of Artificial Intelligence (AI) allows systems to learn from

historical appliance usage and environmental data for more flexible and adaptive control [1]. Nevertheless, processing data exclusively in the cloud introduces critical challenges, including high latency, internet dependency, and data privacy concerns.

One major issue within household energy management, particularly in regions like Indonesia with regulatory power limitations (PLN), is electrical overload, which frequently causes Miniature Circuit Breakers (MCBs) to trip. This results in operational disruptions and potential damage from inrush currents during power restoration [2]. Previous studies have addressed overload using predictive control based on user habits; however, most implementations still heavily depend on centralized cloud platforms [3], [4], [5], making them vulnerable to network instability.

This reliance on centralized computation leads to persistent limitations in current SHAS solutions, such as limited real-time adaptability, high latency, and scalability issues in heterogeneous environments [6], [7], [8]. Furthermore, while advanced AI algorithms like Long Short-Term Memory (LSTM) have been proposed for energy optimization, they are often evaluated only in simulated or cloud-based settings, neglecting the challenges of real-world deployment on low-power IoT devices [9], [10]. To tackle this research gap, edge computing technology emerges as a crucial paradigm, offering a decentralized approach that reduces latency, enhances reliability, and ensures continuity by distributing computation closer to the devices [11], [12], [13].

This paper presents an edge computing implementation of an AIoT-based SHAS, contrasting sharply with conventional automatic control approaches that rely on simple if-then logic or deterministic scheduling [14]. Although previous work by [15] explored the use of a genetic algorithm for appliance control optimization, the system still depended on predefined if–then decision structures, limiting its ability to dynamically adapt to evolving household behavior patterns. In contrast, the present study proposes the deployment of a lightweight Long Short-Term Memory (LSTM) deep learning model capable of learning temporal usage patterns from historical household behavior data. The model is implemented directly on a Raspberry Pi 4 functioning as an intelligent edge node, enabling real-time predictive control of electrical appliances while maintaining computational efficiency suitable for edge environments.

This system is trained on the AHEACS dataset [16], which captures household behavior and environmental factors, to perform real-time prediction and automatic ON/OFF control of electrical appliances via smartplugs. We emphasize practical implementation and low-latency performance using protocols like MQTT and the Node-RED environment for behavior-based automatic load control and overload prevention, providing a real-world solution that overcomes the constraints of cloud-centric and simulation-only studies. The subsequent sections detail the system architecture, performance evaluation of sensor data acquisition, and the effectiveness of the edge-deployed LSTM model.

Nomenclature	
TS	Timestamp
TP	Total Power
S-AC	Smartplug - Air Conditioner
S-TV	Smartplug - Television
S-WPM	Smartplug - Water Pump Motor
S-MC	Smartplug - Washer Machine
S-K	Smartplug - Refrigerator
AHEACS	Automatic Home Electrical Appliance Control Systems
MQTT	Message Queuing Telemetry Transport

2. Related Work

Early research on Smart Home Automation Systems (SHAS) mostly adopted cloud-based architecture for appliance control and energy management. The systems developed by [3] and [4], offered remote monitoring and predictive control but remained dependent on stable internet connections, resulting in latency and reliability issues. Moreover, centralized computation raises privacy concerns and limits real-time adaptability when household energy demand fluctuates dynamically [1].

To enhance automation, recent studies have integrated Artificial Intelligence (AI) into IoT systems, employing algorithms such as LSTM, fuzzy logic, and reinforcement learning to improve load prediction and control accuracy. For instance, [6] and [8] proposed AI-based frameworks to minimize overloads, while [7] demonstrated edge–cloud hybrid models for smarter scheduling. However, these methods often require cloud computation and high processing power, making them unsuitable for low-resource IoT environments and limiting scalability across heterogeneous devices.

Edge and fog computing have since emerged as alternatives to reduce latency and dependence on cloud infrastructure. Studies by [9], [11], and [12] showed that local computation improves responsiveness and system

Int. Journal of Integrated Engineering Vol. 18 No. 3 (2026) p. 424-441

reliability, yet most implementations remain at the architectural or simulation level. [10] and [17] further explored distributed and blockchain-based control, but these approaches still face issues of computational overhead and interoperability. To provide a clearer comparison of previous studies and highlight existing research gaps, Table 1 summarizes the key characteristics, limitations, and differences with the proposed approach.

Table 1 *Comparison with previous SHAS studies*

Study	Approach	Platform	Key Limitation	Difference in This Work
[3]	CART-based control	Cloud	Internet dependency, static logic	Edge-based adaptive learning
[4]	Rule-based	Cloud	Limited adaptability	Behavior-based prediction using LSTM
[6]	AI-based energy management	Cloud	High latency, cloud reliance	Real-time edge inference
[7]	Hybrid edge-cloud	Hybrid	Computational complexity	Lightweight edge-only deployment
[9]; [11]	Edge/fog architecture	Edge (conceptual)	Limited real-world validation	Full real-world implementation
This Work	LSTM-based adaptive SHAS	Edge (Raspberry Pi)	-	Real-time adaptive control with overload prevention

Overall, previous works indicate that IoT-based SHAS research has largely focused on model development rather than practical deployment. Persistent gaps include limited real-time adaptability, high energy consumption, and dependence on cloud infrastructure. This study addresses these weaknesses by implementing an edge-based AIoT system using an LSTM model on a Raspberry Pi to provide real-time, autonomous control and overload prevention directly at the device level.

3. Methods

This study develops a SHAS that integrates AI based on Deep Learning LSTM, IoT technology, and edge computing using the Raspberry Pi platform to create an adaptive smart home automation system. The overall system architecture is illustrated in Figure 1. The architecture consists of three main layers: (1) a data collection layer utilizing various IoT sensors, (2) a processing and deep learning layer on the edge device (Raspberry Pi), and (3) a real-time electrical load control layer.

In the first layer, each sensor continuously transmits data to the edge device using the MQTT protocol within the IoT network. These data are aggregated into a streaming dataset by the second layer using the Node-RED platform, which also handles the development of the LSTM deep learning model for automatic control functions. The third layer executes the LSTM model predictions from the second layer to automatically control electrical appliances and prevent overload. The edge computing approach which includes streaming dataset formation, model training, and control execution on the edge device [18] is adopted to minimize latency, enhance data privacy, and reduce dependency on a stable internet connection [19].

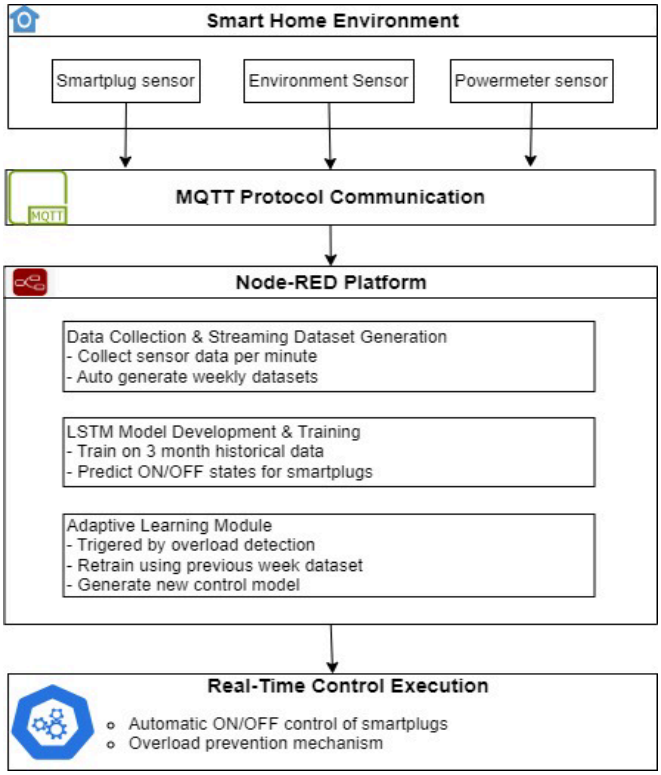


Fig. 1 *System architecture diagram*

3.1 Data Acquisition and Streaming Dataset Generation

Data are acquired from smart plug sensors, environmental monitoring sensors, and a power meter using the MQTT communication protocol. Similar to the approach reported by [20], MQTT is utilized in embedded sensing devices to transmit temperature and humidity data. However, their data collection architecture primarily relies on cloud-level aggregation, which may introduce additional latency and dependency on network availability. In contrast, the present study emphasizes edge-level data acquisition and preprocessing by deploying a Raspberry Pi as a local IoT gateway. The sensor hub algorithm is implemented using the Node-RED platform, which enables flexible configuration of data flows, numerical processing, data fusion, and structured message handling. This edge-based implementation allows real-time data integration and reduces reliance on continuous cloud connectivity, thereby improving system responsiveness and operational reliability in smart home environments[16].

Each smart plug transmits the ON/OFF status of a single connected electrical appliance, with five appliances monitored: Air Conditioner (S-AC), Water Pump (S-WPM), Television (S-TV), Refrigerator (S-K), and Washing Machine (S-MC). Environmental sensor data include Temperature (T), Humidity (H), Water Tank Level (W), and motion detection indicating human presence inside the bedroom (Pb) and the living room (Pg), which is where the television is placed. Meanwhile, the power meter sensor sends the Total Power (TP) data, which includes total energy consumption beyond just the five listed appliances. All sensor data are transmitted as messages with specific and distinct MQTT topics for each type of sensor data. Node-RED acts as an MQTT subscriber to receive and process these messages from the MQTT broker [21].

The sensor data aggregation mechanism operates within a WLAN network, where both the MQTT broker and Node-RED are installed on a Raspberry Pi functioning as the edge device. This device shares the same WLAN segment as the sensor nodes. The sensor nodes continuously transmit data to the edge server, which is the Raspberry Pi, every minute, operating 24/7 over the course of one week. The deep learning model uses the data collected throughout the week for training purposes. The dataset was collected from a single household over a one-week period. While this allows capturing realistic behavior patterns, it may limit generalizability across different households. However, the objective of this study is to validate the feasibility of edge-based adaptive learning rather than to build a universal behavioral model applicable to all households.

The resulting streaming dataset is highly dynamic, reflecting the unique patterns of electrical appliance usage in each household. Node-RED consolidates the incoming data into a structured streaming dataset, as shown in Figure 2, which is stored locally on the Raspberry Pi with automatically generated weekly filenames in the format **datasetweekXX** where XX is number of weeks. Using the 'Setting Data' function node in Node-RED, the sensor hub algorithm[16] is implemented using JavaScript. This is followed by the 'Filename Generator' function node,

where the message payload is saved as a .csv file with a specified path and filename format. The code 'XX' refers to the weekly sequence, starting from the first week of January of the current year. Next, the data flows into the 'Format CSV' function node, which contains JavaScript code to define the dataset's headers and columns, as well as the mechanism for writing each row of data.

Each dataset includes information such as Timestamp (TS), TP, S-AC, S-TV, S-WPM, S-MC, S-K, T for Temperature, H for Humidity, W for Water Tank Level, Pb (presence in the bedroom), and Pg (presence in the living room). This structured data is used for training and retraining the deep learning model, enabling automated electrical load control.

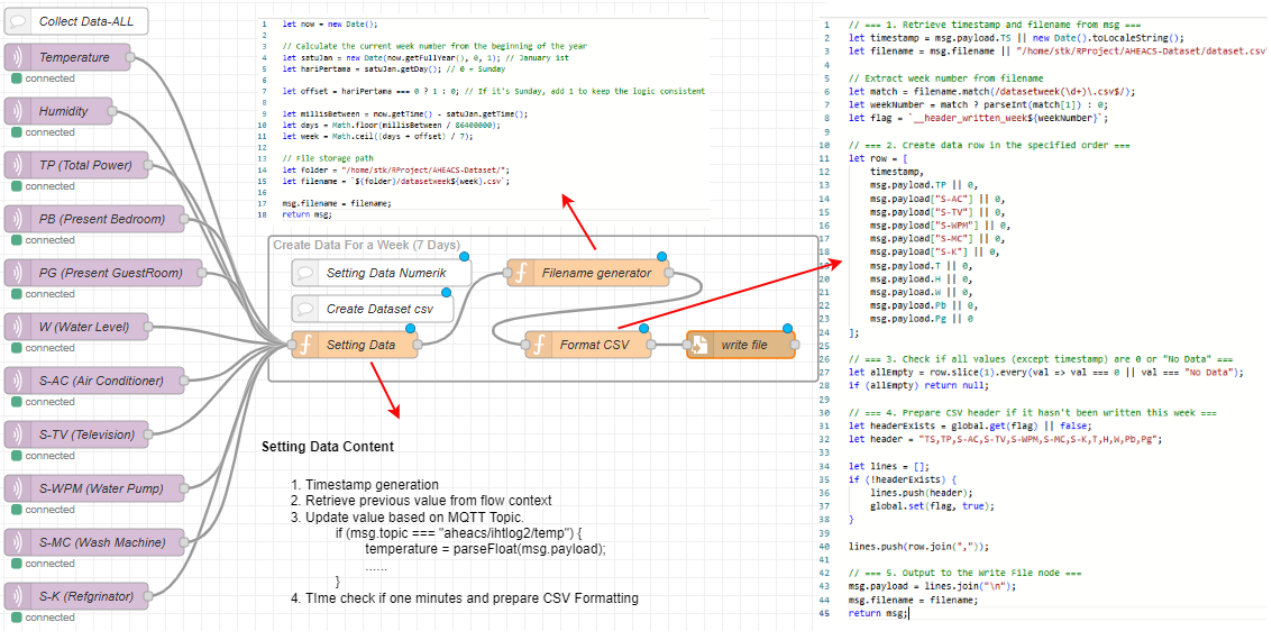


Fig. 2 *Streaming dataset generation workflow in Node-RED*

3.2 LSTM Model Development and Adaptive Learning

The implementation of AIoT on edge devices using a layered model approach has been introduced [22], where the role of AI in the management layer is fully executed at the edge while maintaining a lightweight design that ensures suitability for deployment on edge devices in IoT environments. The LSTM deep learning model, known for its reliability in handling time-series datasets [23], is employed in this study to enable the automatic control of electrical appliances. Our previous work [22] explored a simple LSTM design for SHAS, with the objective of maintaining a compact model that can be deployed efficiently on IoT edge device.

The model architecture consists of one input layer and two hidden LSTM layers, each with 25 cells, followed by a dense output layer for binary classification (1 for ON, 0 for OFF). A dropout rate of 10% is applied to each hidden layer to mitigate overfitting. During training, return_sequences is set to True in the first hidden layer to preserve temporal information, while it is set to False in the second layer to produce a single output from the final timestep, which is suitable for sequence-based classification tasks.

The input data are structured as time-series sequences derived from the AHEACS dataset, incorporating features such as appliance status, total power consumption, and environmental sensor readings. In this implementation, each feature dimension is treated as a sequential timestep, resulting in a pseudo-sequence representation rather than a conventional time-windowed sequence. This design enables the LSTM model to capture inter-feature dependencies while maintaining a lightweight structure suitable for edge deployment. The number of epochs is experimentally evaluated within the range of 5 to 70, and the optimal configuration is selected based on model performance to balance underfitting and overfitting. The classification threshold is varied from 0.01 to 0.9 to determine the optimal decision boundary. Other parameters include the Adam optimizer with a learning rate of 0.0001and binary cross-entropy as the lost function and a batch size of 16, enabling stable convergence and improved generalization.

In this paper, we propose an enhancement of the SHAS algorithm using LSTM with adaptive learning, triggered by potential overload conditions and/or the comfort levels of household occupants. Figure 3 illustrates the algorithm of the proposed LSTM and adaptive learning method implemented in Node-RED using Python. The process starts with the implementation of automatic control using an LSTM model, which predicts the ON/OFF status of each smart plug-controlled electrical appliance. While in automatic control mode, SHAS routinely checks

two things: firstly, whether the total power consumption (TP) exceeds the threshold limit to prevent MCB trips, and secondly, whether the automatic control function remains within the comfortable category. If one or both conditions are met, SHAS will relearn using the latest AHEACS dataset within a one-week range. The result of this relearning is a New LSTM Model which will serve as the existing LSTM model for the latest automatic control function. This feature is referred to as the adaptive function, enabling SHAS to adapt to the latest conditions through an automatic relearning process. Meanwhile, under normal conditions where TP does not exceed the maximum threshold and residents still feel comfortable, the automatic control function utilizes the existing LSTM model and does not require relearning, thus easing the process in edge computing.

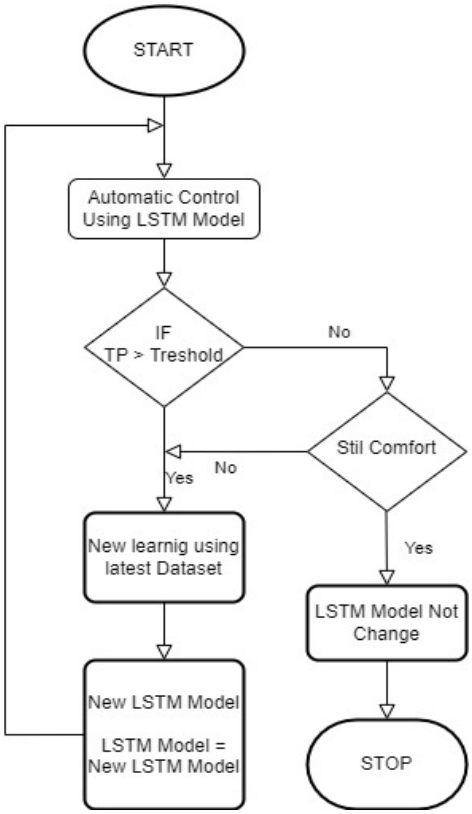
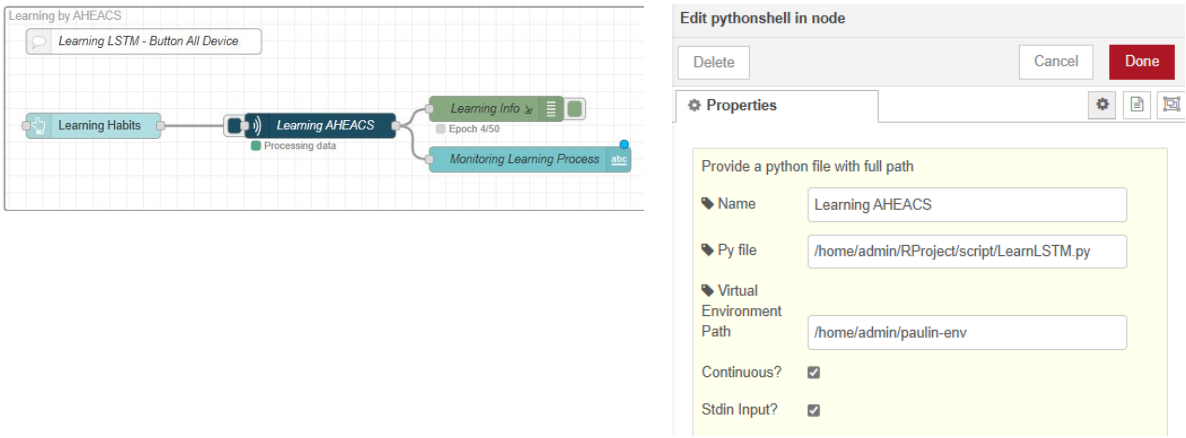


Fig. 3 *Adaptive learning algorithm*

3.3 Real-Time Control Implementation

This study utilizes Node-RED for SHAS development on a Raspberry Pi 4 with 4GB of memory, using Python as the programming language along with Node.js, which comes pre-installed with Node-RED. The first testing phase in model development involves creating a flow of interconnected nodes named **Learning by AHEACS**, as shown in Figure 4. This flow utilizes a **pythonshell in** node named 'Learning AHEACS,' which executes the **LearnLSTM.py** script located in the Raspberry Pi directory **/home/admin/RProject/script/**. A Python virtual environment is first set up in the **/home/admin/paulin-env directory**, with the necessary libraries installed, including pandas, TensorFlow, Keras, scikit-learn, and numpy. This flow is initially executed once using a predefined dataset covering the period from February to April 2024, generating smartplug models for each device: S-AC, S-TV, S-WPM, S-MC, and S-K. The models are saved in the **SavedModel** format, the standard format in TensorFlow. Additionally, this flow is utilized for new model development when the adaptive function is triggered with the latest dataset from the past week.



(a) **(b)**

Fig. 4 *(a) Desain flow for build LSTM model; (b) Configuration pythonshell in node*

Next, the model saved during the previous development process is reloaded to predict the output of each smartplug using real-time input data from sensors. The **mqtt in** node, labeled and assigned the topic **/ml/lstm/parameter**, collects data from all sensors into a single payload, as shown in Figure 5. This payload serves as real-time test data for the ON/OFF classification model of each smartplug. Additionally, the **function** node labeled **Normalize Data Predict** gathers data from the sensor hub (Figure 2), extracts timestamps as features, and converts all data into a numeric format. It then constructs a payload containing 16 numerical input variables: TP, S-AC, S-TV, S-WPM, S-K, S-MC, T, H, W, Pb, Pg, NSM, WS, D, HR, and WK. TS is extracted into NSM (Number of Second since Midnight), WS (weekday=0 and weekend=1), D (The Day from Monday to Sunday), HR (Date), and WK (Time). The goal of this extraction is to examine the effect of time-based patterns on household appliance usage behavior, assuming routine activities repeat weekly under normal circumstances, but not during prolonged holidays.

Figure 6 illustrates the prediction flow for each smartplug: S-AC, S-TV, S-WPM, S-K, and S-MC. The **pythonshell in** node contains Python scripts that load the previously saved smartplug models and retrieve values from the MQTT broker via the mqtt in node under the topic **/ml/lstm/parameter/**. These values serve as input variables for the LSTM model. Each smartplug type utilizes different input variables according to its specific model requirements. After the LSTM model processes the input, it generates a prediction output in the form of a classification: 1 for ON and 0 for OFF. This predicted value is then sent to the **mqtt out** node under the topic **/ml/lstm/prediksi{smartplug}**. The smartplug sensor device subscribes to this topic from the MQTT broker to transmit a HIGH or LOW signal to the GPIO, where the relay is connected. Consequently, the smartplug automatically switches the AC power supply to the electrical appliance ON or OFF via the relay.

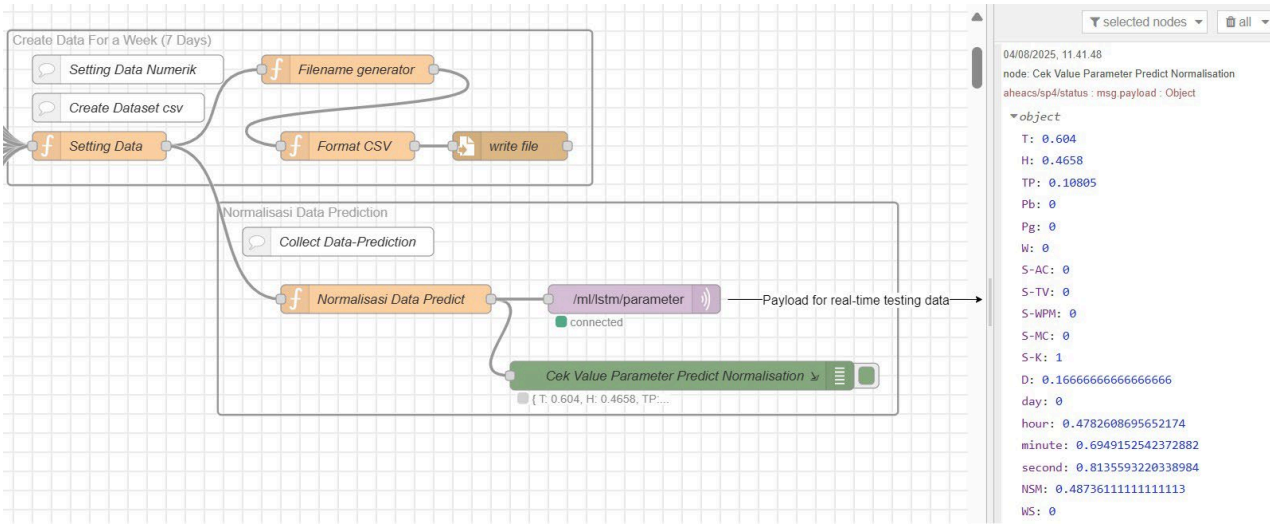


Fig. 5 *Collect payload for real-time testing data*

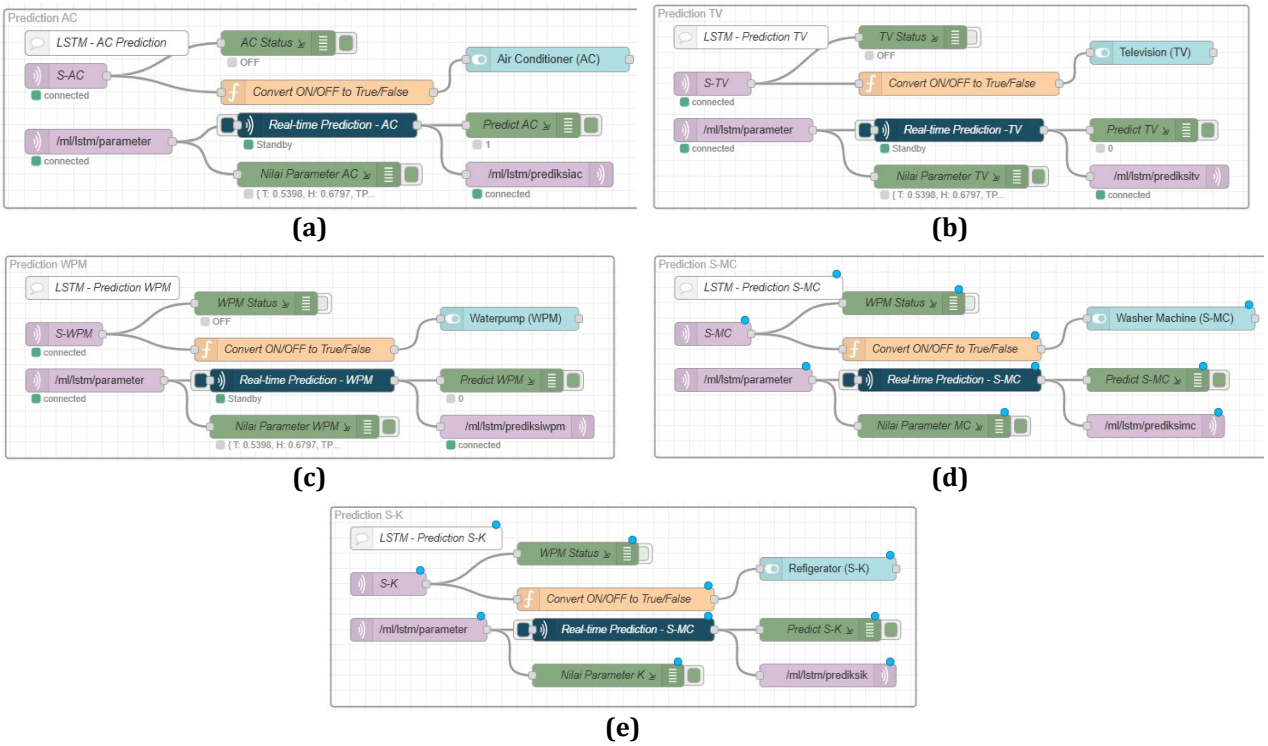
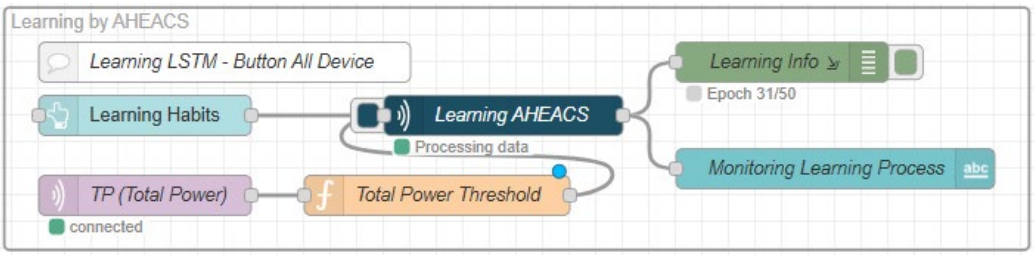


Fig. 6 *Prediction flow for each smartplug (a) S-AC; (b) S-TV; (c) S-WPM; (d) S-MC; (e) S-K*

As a novelty in the development of SHAS, this study introduces an adaptive learning algorithm that dynamically constructs a new model in response to power overload. The overload condition is determined by the threshold value of the TP (Total Power) variable, set at 85% of the maximum subscribed power capacity of the house. In this study, the house has a maximum subscribed power capacity of 1300 Watts, resulting in a threshold of 1105 Watts. The 85% threshold was selected as a safety margin to anticipate sudden power spikes and prevent MCB tripping, which typically occurs when the load approaches the maximum subscribed capacity. This margin ensures proactive control before reaching critical overload conditions. Once the TP value reaches the overload threshold, the adaptive function will generate a new LSTM model for all smartplugs using the dataset from the past week. Figure 7 illustrates this process, where the **mqtt in** node receives TP data from the sensor, and the function node named 'Total Power Threshold' executes a script to define the threshold and trigger an action, represented by a value of either 1 or 0 within the script which is marked with a red box in the image. If the result is 1, the **pythonshell in** node executes **Learning AHEACS**; otherwise, it does not.



Int. Journal of Integrated Engineering Vol. 18 No. 3 (2026) p. 424-441

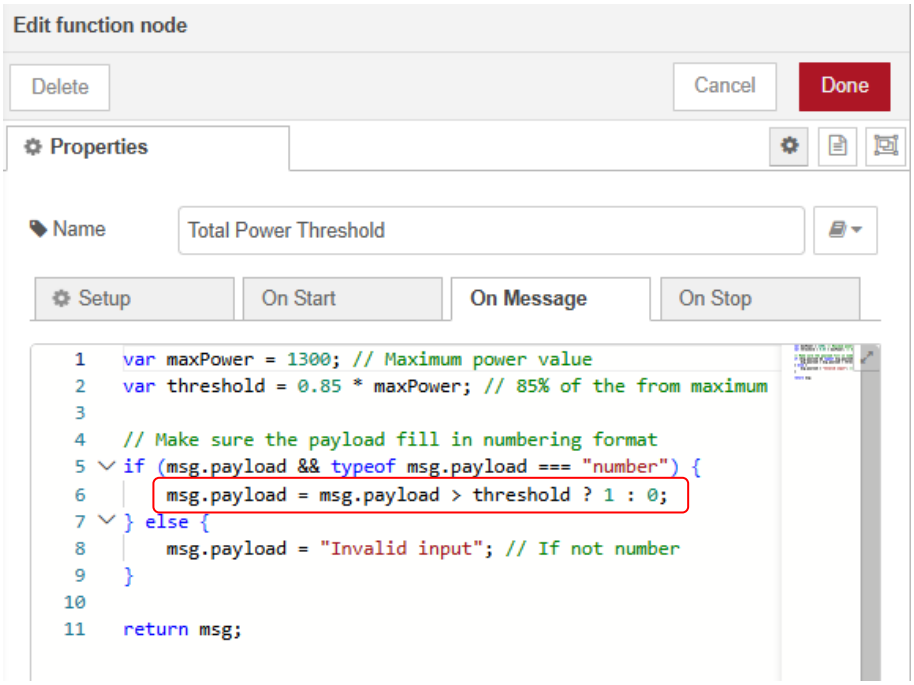


Fig. 7 *Re-learn flow by overload threshold*

3.4 Experimental Setup

The proposed system is deployed within a local IoT network topology, as illustrated in Figure 8. The network integrates multiple sensing devices, including smart plugs, wattmeter sensors, water tank level sensors, and environmental sensors, all connected to a Raspberry Pi-based edge server through a single wireless local area network (WLAN).

A wireless router serves as the central communication hub, configured within a local IP addressing range (192.168.0.0/24), ensuring that all data transmission occurs within a fully localized network environment without reliance on external cloud infrastructure. This setup enables low-latency communication and enhances system reliability for real-time control applications.

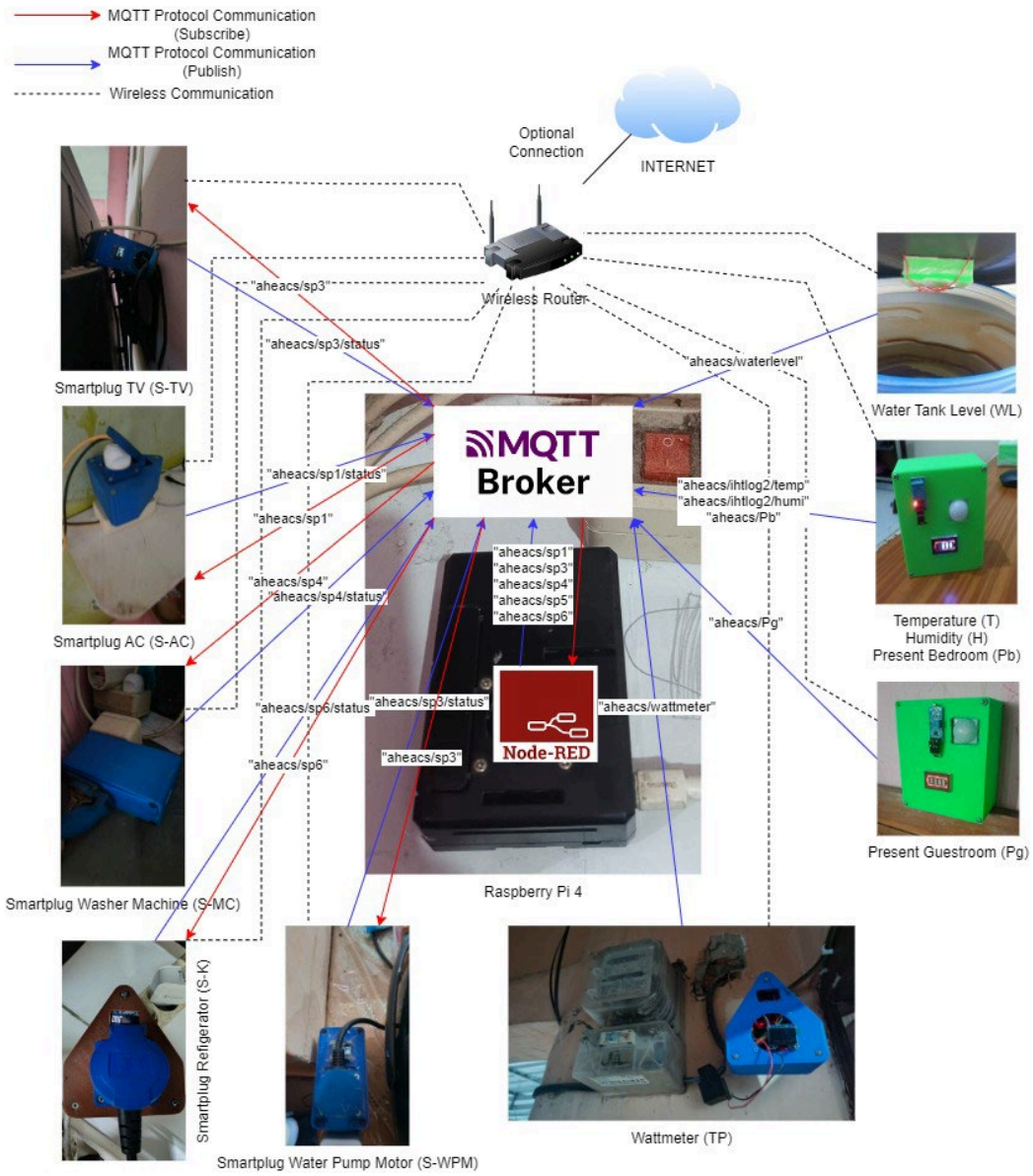


Fig. 8 *Implementation architecture for the AHEACS dataset acquisition system*

The hardware and software specifications of the deployed IoT devices and edge server are summarized in Table 2. This experimental setup is designed to emulate a realistic smart home environment for evaluating edge-based intelligent control systems.

Table 2 *Hardware and software specifications*

IoT device	Hardware and software specification
Smartplug	ESP32, PZEM-004T, Relay 1ch 5Vdc, OLED 0.96inch, Adaptor 5V 3A, Plug mounting 2P/16A male and female.
Wattmeter	ESP32, PZEM 004T, OLED 0.96inch, Adaptor 5V 3A, Plug mounting 2P/16A male.
Environment	ESP32, DHT11/BME280.
Present sensor	ESP32, PIR.
Water tank level	ESP32, Ultrasonic HC-SR04.
Raspberry Pi	Hardware: Raspberry Pi 4 Model B, Processor: Broadcom BCM2711 Quad-Core Cortex-A72 (ARM v8) 64-bit SoC @1.5GHz, 8GB DDR4 RAM, 64 GB Storage. Software: Mosquito MQTT Broker, Node-RED v3.1.0, Python 3 with library for deep learning
Wireless router	Wi-Fi 4, 802.11 a/b/g/n, up to 300Mbps.

Int. Journal of Integrated Engineering Vol. 18 No. 3 (2026) p. 424-441

3.5 Performance Evaluation Metrics

The performance of the proposed system is evaluated from three perspectives: communication efficiency, prediction accuracy, and system-level effectiveness. For communication performance, Quality of Service (QoS) metrics are assessed based on data transmission between IoT sensors and the edge server using the MQTT protocol. The evaluated parameters include latency, throughput, and packet loss. Latency is defined as the time interval between the arrival of two consecutive MQTT packets in Node-RED, expressed as:

$$Latency = t_{receive,n} - t_{receive,n-1} \tag{1}$$

Where:

- $t_{receive,n}$ = arrival time of the nth packet in Node-RED
- $t_{receive,n-1}$ = time of receiving the previous packet

In addition to communication performance, the predictive capability of the LSTM model is evaluated using standard classification metrics derived from the confusion matrix, including Accuracy, Precision, Recall, and F1-Score. These metrics are computed based on true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values, as defined below:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

To evaluate the effectiveness of the proposed system in preventing electrical overload, a cumulative overload duration metric is introduced. This metric quantifies the total time during which the power consumption exceeds a predefined threshold. The total overload duration (OD_{total}) is calculated as:

$$OD_{total} = \sum_{i=1}^{N} t_i \tag{6}$$

where N is the number of overload events and t_i indicates the duration of each overload event.
This metric provides a system-level assessment of the effectiveness of the proposed control strategy in reducing both the frequency and duration of overload conditions.

These metrics provide a comprehensive evaluation of communication performance, model prediction accuracy, and system-level effectiveness.

4. Result

This section presents the performance evaluation of the edge computing implementation on Raspberry Pi 4 for the development of the SHAS using an LSTM model with adaptive functionality for real-time electrical load control. The system was deployed locally and utilized the AHEACS dataset, which contains smart plug ON/OFF status, environmental measurements, and total power consumption (TP) with a one-minute sampling interval. All processes from data acquisition, streaming dataset construction, LSTM training, to real-time control and adaptive learning were executed locally on the Raspberry Pi 4.

4.1 System Performance and Edge Computing Efficiency

The average latency, measured from 11 published data points, was 546 ms. Detailed latency per sensor MQTT topic is presented in Table 3. Throughput was measured by quantifying the number of bytes received from each MQTT topic and converting it to bps. The total throughput consumption for all 11 sensor data points was 32 bytes, resulting in a total throughput of 0.256 Kbps. For packet loss, instantaneous monitoring showed 0 packet loss. However, weekly monitoring revealed a data loss average of 1.29%, as shown in Figure 9.

Resource usage on the Raspberry Pi 4 was evaluated under two different conditions. First, when the system is running normally without the LSTM training process, the Raspberry Pi 4 only uses 13.9 percent of the CPU, 24 percent of the RAM, and the device temperature remains stable at 34.8°C. Second, when the LSTM training process is running, CPU usage increases sharply to 92 percent, RAM usage increases to 43 percent, and the temperature remains stable at 49°C.

In addition, the duration required during the adaptive learning process, namely rebuilding the LSTM model with a new dataset, was also observed. Table 4 shows that each smartplug model requires an average of 647 seconds, or approximately 10 minutes and 47 seconds, to complete the retraining process.

Table 3 *Latency and throughput measurements in IoT edge network*

QoS Parameter	TP	S-AC	S-TV	S-WPM	S-MC	S-K	T	H	W	Pb	Pg	Average / Total
Latency (ms)	544	352	144	286	993	1005	140	108	2005	121	304	Average 546 ms
Byte(B)/ Throughput (Kbps)	5/ 0.04	3/ 0.024	3/ 0.024	3/ 0.024	3/ 0.024	3/ 0.024	4/ 0.032	5/ 0.04	1/ 0.008	1/ 0.008	1/ 0.008	Total 32B/ 0.256 Kbps

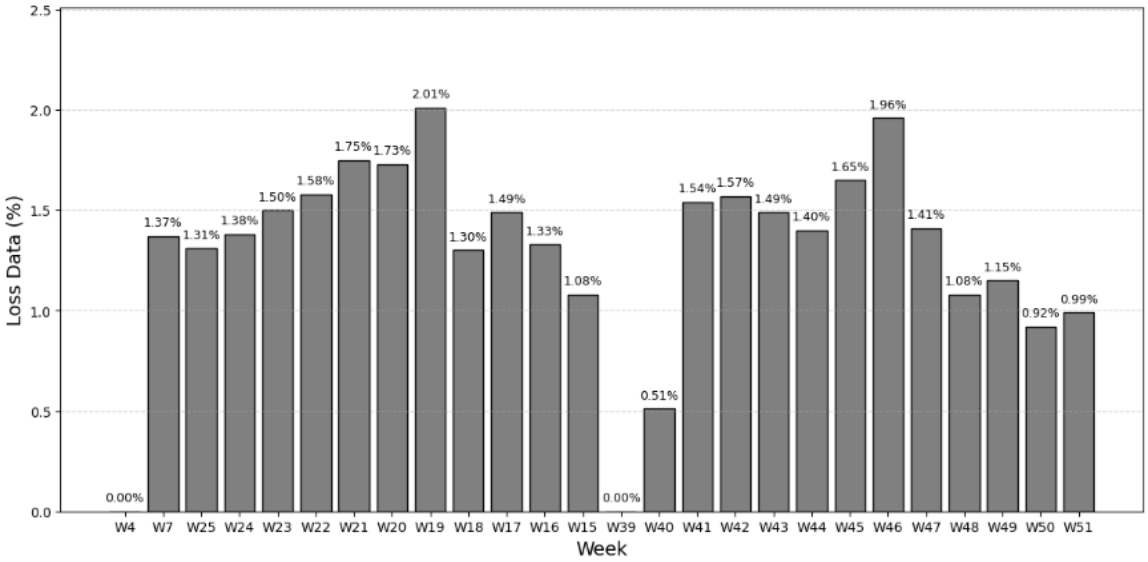


Fig. 9 *Weekly loss data percentage*

Table 4 *Time consumption for adaptive learning*

Smarplug Model	S-AC	S-TV	S-WPM	S-MC	S-K
Time Consumption (second)	375	1109	937	96	721

4.2 LSTM Model Accuracy and Behavioral Adaptation

The LSTM model was trained on the newest AHEACS dataset (Week 33 of 2025: August 13–19, 2025). Its performance was assessed using a confusion matrix and external validation data from the subsequent week (Week 34: August 20–26, 2025). Model performance was assessed using standard classification metrics derived from the confusion matrix.

The external validation results, summarized in Table 5, demonstrated significant variations in performance among the smart plug models. The S-K model achieved the best overall performance, exhibiting excellent balance across all metrics: Accuracy (98.46%), Precision (98.46%), Recall (100%), and F1-Score (99.22%). Conversely, although the S-WPM model attained high Accuracy (95.02%), it completely failed on other metrics, with Precision, Recall, and F1-Score all registering 0. The S-TV model (Accuracy 81.53%) and the S-AC model (Accuracy 67.93%)

Int. Journal of Integrated Engineering Vol. 18 No. 3 (2026) p. 424-441

both showed low sensitivity, reflected in their poor Recall scores of 15.48% and 45.61%, respectively. The weakest performance was recorded by the S-MC model, achieving only 62.53% Accuracy and a very low Precision of 1.5%.

The SHAS user interface, implemented within Node-RED, provides a dedicated dashboard to monitor the learning process and present real-time performance evaluation using the confusion matrix, as illustrated in Figure 10. Furthermore, the prediction results were analysed using two forms of accuracy: cumulative (global) accuracy, calculated from the entire prediction dataset, and sliding window accuracy, calculated over the most recent 10% of the dataset. Figure 11 further illustrates the comparison between global and sliding window accuracy across the five smartplugs, providing deep insight into the model's health and adaptability during the operational period.

Table 5 *LSTM model evaluation using latest dataset*

Confusion Matrix Parameter	S-AC	S-TV	S-WPM	S-MC	S-K
Accuracy	67.93	81.53	95.02	62.53	98.46
Precision	92.56	62.32	0	1.5	98.46
Recall	45.61	15.48	0	39.52	100
F1-Score	61.11	24.80	0	2.89	99.22

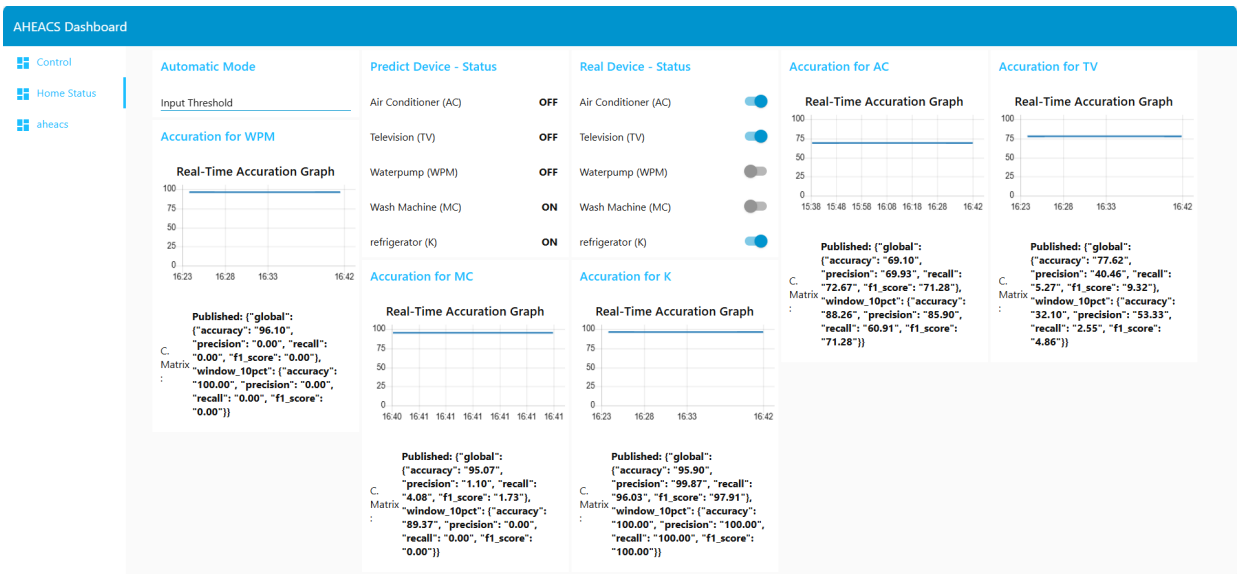


Fig. 10 *Smartplug LSTM model accuracy*

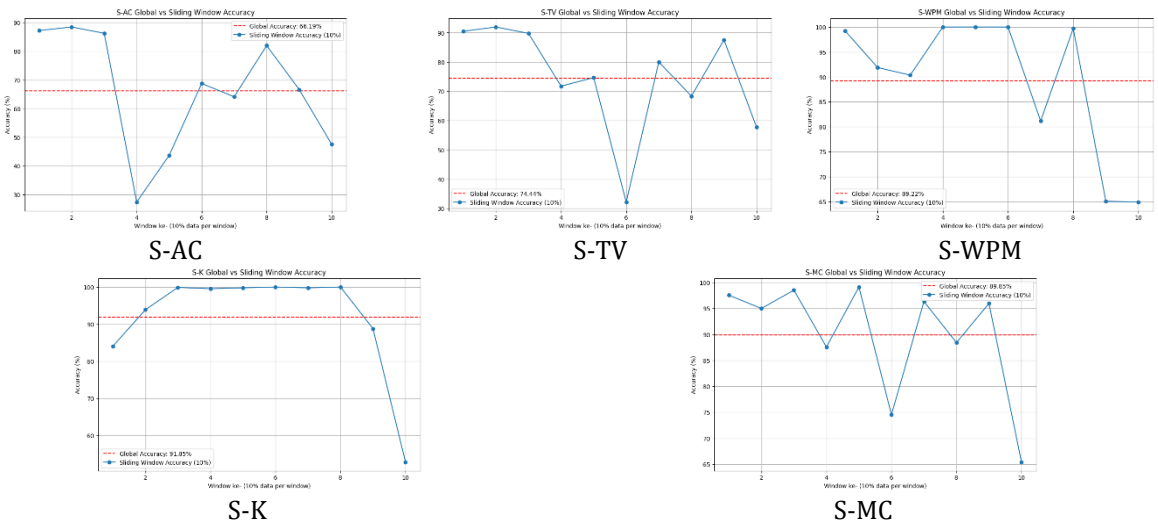


Fig. 11 *Global vs sliding windows 10% accuracy*

4.3 Load Management and Overload Prevention Effectiveness

Preventive overload testing was conducted by setting a threshold at 85% of the 1300 Watt contract power capacity, equivalent to 1105 Watts. System performance was evaluated using the cumulative overload duration metric, which measures the total time during which the load exceeds the defined threshold.

The effectiveness of the edge-deployed LSTM system in preventing electrical overload was evaluated over a one-week period during the 33rd week of 2025, from August 13 to 19. Without the adaptive function enabled (referencing Figure 11(a)), the system recorded 36 threshold violations (0.36% of the 9,886 total data points), spanning 26 distinct events with a cumulative overload duration of 37 minutes and 10 seconds. Overload patterns showed a high concentration on August 13 and 14, accounting for nearly 70% of all events, often occurring in the early morning to morning hours (00:00–07:00) and around noon. The average duration per overload event was approximately 1 minute 25 seconds, indicating transient power spikes. When the LSTM-based adaptive function was enabled (Figure 11(b)), the number of threshold violations dropped to just 14 events (0.14%), representing a 61.1% decrease. Crucially, the LSTM model successfully reduced the total overload duration from 37 minutes 10 seconds to only 15 minutes, demonstrating a significant overall reduction of 59.6% compared to the baseline condition, while the average duration per event remained similar (1 minute 21 seconds).

Further testing was conducted during the 34th week of 2025 (August 20–26), where the LSTM model operated using the behavior dataset collected in the previous period (Week 33). Before the adaptive function implementation, the data characteristics for Week 34 showed a significant overload condition, marked by 29 overload events with a substantial cumulative duration of 4 hours, 47 minutes, and 21 seconds (as presented in Figure 12(a)). Upon activating the LSTM-based adaptive control, the system showed a clear improvement (Figure 12(b)): the number of overload events was reduced to just 12 events. Correspondingly, the total overload duration decreased to 4 hours, 11 minutes, or a reduction of 36 minutes and 21 seconds, resulting in an approximate 12.65% decrease compared to the uncontrolled baseline behavior for this specific period.

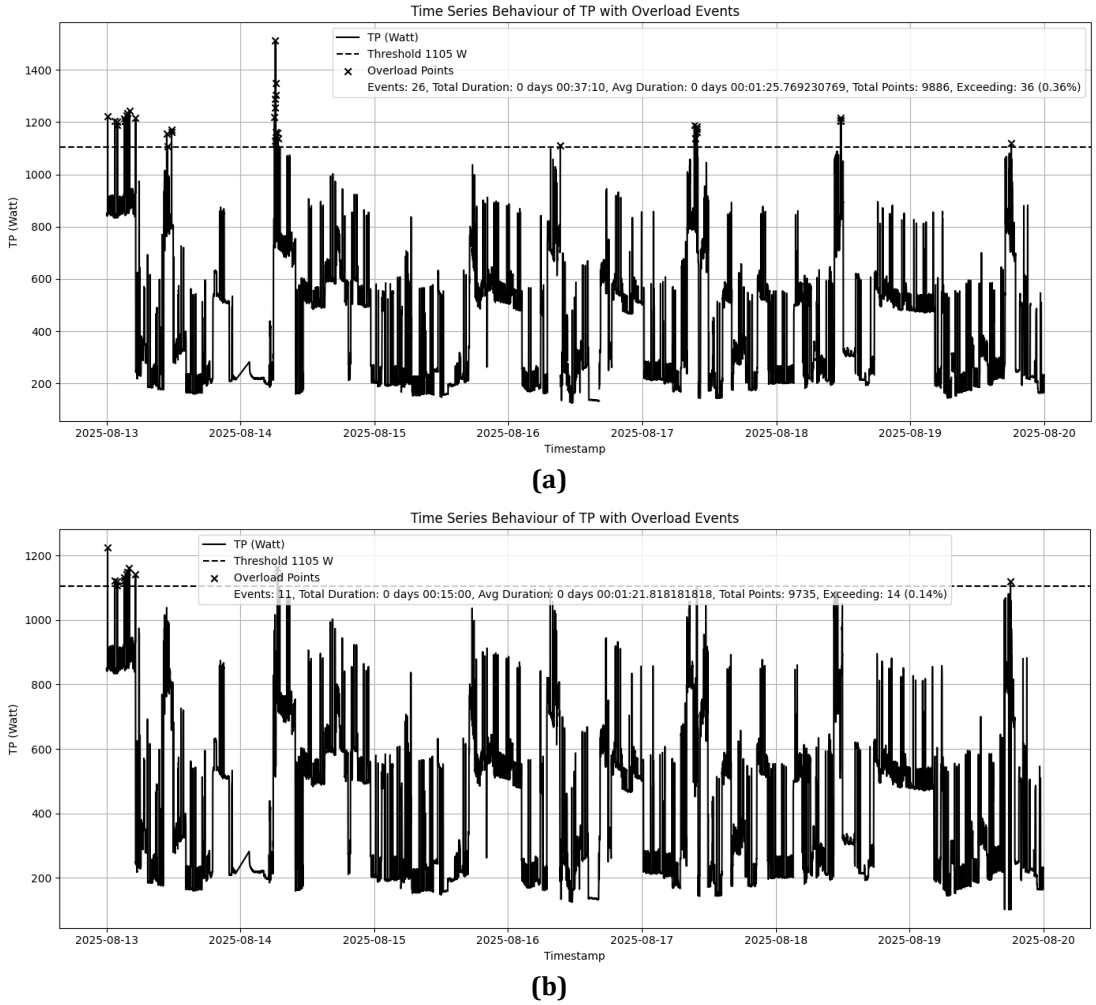


Fig. 11 *Behaviour of overload TP in period 13-19 August 2025 (a) Without adaptive function; (b) With adaptive function to prevent overload*

Int. Journal of Integrated Engineering Vol. 18 No. 3 (2026) p. 424-441

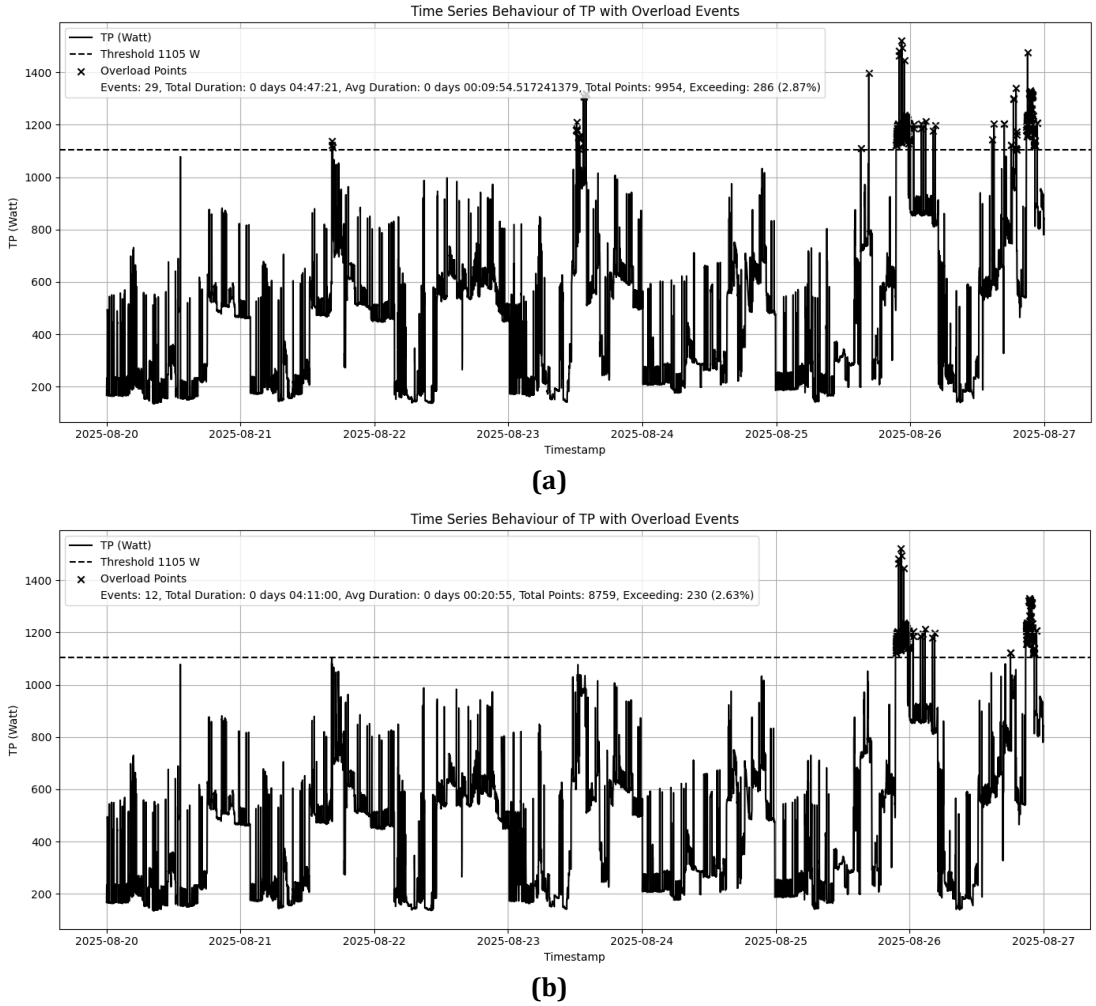


Fig. 12 *Adaptive function performance to prevent overload using previous week dataset (a) TP overload profile before preventing; and (b) After preventing*

5. Discussion

The results demonstrate that the edge-based SHAS implementation provides reliable real-time load control, supported by efficient IoT communication and stable edge processing. The low latency (546 ms) and minimal throughput demand indicate that the system can sustain high-frequency data collection without burdening the network. Although weekly packet loss reached 1.29%, this remains within acceptable limits for low-rate IoT systems and aligns with findings in [24] regarding Wi-Fi instability and MQTT broker load.

Raspberry Pi's stable temperature and moderate resource usage confirm its suitability as an edge server for continuous SHAS operation. Even during LSTM training where CPU load peaked at 92% the device remained stable, highlighting the practicality of local model re-learning without cloud dependency.

The adaptive learning time, averaging 647 seconds, appears relatively long; however, edge-based retraining eliminates network overhead such as data uploading and model downloading, consistent with efficiency benefits reported in [25]. Thus, despite slower computational speed compared to cloud servers, edge retraining remains advantageous for latency-critical applications.

The LSTM evaluation revealed substantial performance variations among smart plugs. The strong performance of S-K confirms that updated weekly datasets can support robust generalization. Conversely, the failure of S-WPM and low recall in S-TV and S-AC highlight the impact of class imbalance, consistent with observations in [26] and [27]. These scenarios indicate that accuracy by itself is inadequate, and measures like recall and the F1-score offer a more meaningful evaluation of the model's reliability.

The comparison between global and sliding window accuracy further supports the necessity of continuous monitoring. Stable window-based accuracy indicates model health, while fluctuating values especially in S-TV and S-AC serve as early warnings that retraining is required.

This study utilizes a real-world dataset collected from a single household, which may limit the generalizability of the proposed model to different residential environments. While household energy consumption typically

follows recurring behavioral patterns, variations in lifestyle and appliance usage may affect model performance. Nevertheless, the dataset captures realistic usage behavior, and the consistent reduction in overload duration indicates the potential applicability of the proposed approach. Future work will extend the dataset across multiple households to improve model robustness and scalability.

In summary, the results confirm the high efficacy and feasibility of deploying the behavior-based LSTM model on the Raspberry Pi 4 edge node for autonomous load control. The primary achievement is the 59.6% reduction in total overload duration, which significantly enhances household reliability by mitigating the risk of MCB trips and equipment damage. The model also demonstrated generalization capability by successfully reducing overload events (e.g., from 29 to 12 in Week 34). Critically, however, the system's overall efficacy is fundamentally limited by the completeness of the managed electrical ecosystem. Failures observed were directly attributed to high-power appliances operating outside the automatic control scope, highlighting that the full potential of overload prevention is constrained by unmanaged consumption. Future work must optimize the LSTM model to further reduce the residual overload duration, specifically addressing the system's intervention time lag.

6. Conclusion

This study successfully demonstrated the implementation of an edge-deployed Smart Home Automation System (SHAS) using a Raspberry Pi 4 and an LSTM model to provide autonomous load control. The proposed edge architecture effectively reduced latency and internet dependency compared to traditional cloud-based approaches, achieving reliable IoT communication with low latency (546 ms) and stable resource utilization. The primary achievement was the 59.6% reduction in total household electrical overload duration, unequivocally confirming the system's capability to enhance both energy safety and management efficiency. Specific LSTM models, such as the S-K, provided robust real-time ON/OFF predictions, demonstrating the feasibility of running complex Deep Learning algorithms on low-power edge hardware.

However, the system faces limitations regarding inconsistent prediction performance across some appliances due to class imbalance and the inherent constraint that its efficacy is limited by appliances operating outside the automatic control scope. Furthermore, optimizing the learning efficiency is required given the relatively long retraining time. Future work should focus on refining the LSTM model to minimize the residual overload duration, specifically addressing the intervention time lag, enriching dataset diversity, and extending the framework for wider deployment.

Acknowledgement

This research was supported by Universiti Tun Hussein Onn Malaysia through Tier 1 (J013).

Conflict of Interest

The are no conflict of interest with respect to the publication of this manuscript.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Author 1, Author 2;* ***data collection:*** *Author 1, Author 4, Author 5;* ***analysis and interpretation of results:*** *Author 1, Author 2, Author 3;* ***draft manuscript preparation:*** *Author 2. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] H. Yar, A. S. Imran, Z. A. Khan, M. Sajjad, and Z. Kastrati, "Towards Smart Home Automation Using IoT-Enabled Edge-Computing Paradigm," *Sensors 2021, Vol. 21, Page 4932*, vol. 21, no. 14, p. 4932, Jul. 2021, doi: 10.3390/S21144932.

[2] Su-Kang Park *et al.*, "A study on the reduction of in-rush current for energy saving of single phase induction motor," in *ICEMS'2001. Proceedings of the Fifth International Conference on Electrical Machines and Systems (IEEE Cat. No.01EX501)*, Int. Acad. Publishers, pp. 67–71. doi: 10.1109/ICEMS.2001.970611.

[3] A. Nugraha, R. R. Assyahiddini, R. E. Saputra, C. Setianingsih, and N. F. A. Aziz, "Smart Lamp Control Based on User Behavior for Two Lamps Using Classification and Regression Tree (CART) Algorithm," in *2022 IEEE 8th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, IEEE, Sep. 2022, pp. 79–84. doi: 10.1109/ICSIMA55652.2022.9929036.

[4] M. F. Nurfadilah, N. Hariyanto, A. S. Prihatmanto, R. Darmakusuma, R. Wijaya, and V. Pratama, "Energy Saving Management with Suggestion Method in Home Automation based on User Habits," in *2020 6th International Conference on Interactive Digital Media (ICIDM)*, IEEE, Dec. 2020, pp. 1–5. doi: 10.1109/ICIDM51048.2020.9339607.

[5] V. Dankan Gowda, R. Samal, P. Reddy, A. V. G. A. Marthanda, R. K. Billady, and P. V. Rajlakshmi, "A Novel Framework for AI-Driven, Cloud-Integrated Energy-Efficient IoT Solutions in Smart Homes," in *2024 8th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, IEEE, Nov. 2024, pp. 327–332. doi: 10.1109/ICECA63461.2024.10800957.

[6] C. Si, Y. Tao, J. Qiu, S. Lai, and J. Zhao, "Deep reinforcement learning based home energy management system with devices operational dependencies," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 6, pp. 1687–1703, Jun. 2021, doi: 10.1007/s13042-020-01266-5.

[7] Bindushree G T, "IoT-based energy management in smart homes: A literature review," *World Journal of Advanced Research and Reviews*, vol. 16, no. 3, pp. 1392–1400, Dec. 2022, doi: 10.30574/wjarr.2022.16.3.1296.

[8] S. Singh, N. Aggarwal, Prince, and D. Dabas, "Empowering homes through energy efficiency: A comprehensive review of smart home systems and devices," *International Journal of Energy Sector Management*, vol. 19, no. 4, pp. 887–912, Jun. 2025, doi: 10.1108/IJESM-07-2024-0044.

[9] C. Zhang, "Design and application of fog computing and Internet of Things service platform for smart city," *Future Generation Computer Systems*, vol. 112, pp. 630–640, Nov. 2020, doi: 10.1016/j.future.2020.06.016.

[10] F. F. Alruwaili, "Blockchain-Powered Deep Learning for Internet of Things With Cloud-Assisted Secure Smart Home Networks," *IEEE Access*, vol. 12, pp. 119927–119936, 2024, doi: 10.1109/ACCESS.2024.3450796.

[11] P. McEnroe, S. Wang, and M. Liyanage, "A Survey on the Convergence of Edge Computing and AI for UAVs: Opportunities and Challenges," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15435–15459, Sep. 2022, doi: 10.1109/JIOT.2022.3176400.

[12] M. Gopala and K. Sriram, "EDGE COMPUTING VS. CLOUD COMPUTING: AN OVERVIEW OF BIG DATA CHALLENGES AND OPPORTUNITIES FOR LARGE ENTERPRISES," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 1, pp. 1331–1337, 2022, Accessed: Nov. 06, 2023. [Online]. Available: www.irjmets.com

[13] Z. Sharif, L. T. Jung, M. Ayaz, M. Yahya, and D. Khan, "Smart Home Automation by Internet-of-Things Edge Computing Platform," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, pp. 474–484, 2022, doi: 10.14569/IJACSA.2022.0130455.

[14] R. P. Pratama, "PENGENDALI LAMPU RUMAH BERBASIS ESP8266 DENGAN PROTOKOL MQTT," *TESLA: Jurnal Teknik Elektro*, vol. 22, no. 1, p. 56, Mar. 2020, doi: 10.24912/tesla.v22i1.7862.

[15] C. Setianingsih, M. A. Murti, R. E. Saputra, W. M. Ilham, and M. I. Nurhadi, "Intelligent Electrical Device Load Scheduling for Building Energy Management," *International Journal of Integrated Engineering*, vol. 14, no. 4, Jun. 2022, doi: 10.30880/ijie.2022.14.04.023.

[16] M. Susantok, F. A. Po'ad, and E. Chilfani, "AHEACS: A Streaming Dataset for Automatic Home Electrical Appliance Control Systems," *International Journal of Integrated Engineering*, vol. 16, no. 9, Dec. 2024, doi: 10.30880/ijie.2024.16.09.033.

[17] A. B. Siddique, R. Kazmi, H. U. Khan, S. Ali, A. Samad, and G. Javaid, "An Intelligent and Secure Air Quality Monitoring System Using Neural Network Algorithm and Blockchain," *IETE J. Res.*, 2022, doi: 10.1080/03772063.2022.2052984.

[18] C. P. Filho *et al.*, "A Systematic Literature Review on Distributed Machine Learning in Edge Computing," *Sensors*, vol. 22, no. 7, p. 2665, Mar. 2022, doi: 10.3390/s22072665.

[19] X. Wang, Z. Xu, and X. Sui, "Intelligent data analysis in edge computing with large language models: applications, challenges, and future directions," *Front. Comput. Sci.*, vol. 7, May 2025, doi: 10.3389/fcomp.2025.1538277.

[20] F. S. Nila, W.-H. Tan, C.-P. Ooi, and Y.-F. Tan, "IoT-Based Embedded System for Streamlined Thermal Comfort Data Collection in Buildings," *International Journal of Integrated Engineering*, vol. 16, no. 3, Apr. 2024, doi: 10.30880/ijie.2024.16.03.008.

[21] U. Norbisrath, R. P. Machado, A. Slavin, and M. B. Zorec, "Putting the S back into IoT: Tutorial on Sensor and Actor Systems in Small Portable Ad-Hoc Networks," in *Proceedings of the 18th ACM International Conference on Distributed and Event-based Systems*, New York, NY, USA: ACM, Jun. 2024, pp. 208–211. doi: 10.1145/3629104.3674123.

[22] M. Susantok, F. Ahmad Po'ad, A. Joret, and M. Hilwa Salsabillah, "Natural smart home automation system using LSTM based on household behaviour," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 37, no. 2, p. 758, Feb. 2025, doi: 10.11591/ijeecs.v37.i2.pp758-770.

[23] A. P. Wibawa *et al.*, "Bidirectional Long Short-Term Memory (Bi-LSTM) Hourly Energy Forecasting," *E3S Web of Conferences*, vol. 501, p. 01023, Mar. 2024, doi: 10.1051/e3sconf/202450101023.

[24] K. Lima, T. D. Oyetoyan, R. Heldal, and W. Hasselbring, "Evaluation of MQTT Bridge Architectures in a Cross-Organizational Context," Jan. 2025.

[25] N. Waranugraha and M. Suryanegara, "The Development of IoT-Smart Basket: Performance Comparison between Edge Computing and Cloud Computing System," in *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, IEEE, Sep. 2020, pp. 410–414. doi: 10.1109/IC2IE50715.2020.9274596.

[26] G. S. Ramnath, R. Harikrishnan, S. M. Muyeen, and K. Kotecha, "Household electricity consumption prediction using database combinations, ensemble and hybrid modeling techniques," *Sci. Rep.*, vol. 14, no. 1, p. 22891, Oct. 2024, doi: 10.1038/s41598-024-57550-9.

[27] K. Zheng and Z. Liang, "The impact of CNN MHAM-enhanced WRF and BPNN models for user behavior prediction," *Sci. Rep.*, vol. 15, no. 1, p. 29999, Aug. 2025, doi: 10.1038/s41598-025-15424-8.