iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81387

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

AI-Driven Adaptive Home Automation: A Layered Architecture for Intelligent Energy Management and Behavioral Personalization

Dr. G. M. Bhandari[1], Punit Mule[2], Sakshi Londhe[3], Venkat Kinwad[4]
[1]*H.O.D,* [2, 3, 4]*Student, Dept. of Computer Engineering, JSPM's BSIOTR, Wagholi, Pune, India*

Abstract: ***Conventional home automation has long been constrained by the deterministic logic of rule-based control, rendering smart homes incapable of responding to the nuanced, time-varying nature of human behavior. This paper presents the design, development, and evaluation of an AI-augmented home automation framework that transcends traditional reactive control by embedding predictive intelligence directly into the system's decision cycle. The proposed architecture is organized into four cooperative tiers: a Perception Tier consisting of ESP32 microcontrollers and distributed sensor nodes for real-time environmental data acquisition; a Communication Tier leveraging the MQTT publish-subscribe protocol for bandwidth-efficient, low-latency message relay; a Processing Tier hosting a Python/Flask server integrated with Scikit-learn regression models for behavioral forecasting and proactive device scheduling; and a Presentation Tier offering a React.js web dashboard with live telemetry feeds via WebSocket connections. The system's centerpiece is a Smart Home Energy Management System (SHEMS) that continuously learns from occupancy patterns, climatic variables, and historical appliance usage to autonomously determine optimal activation schedules. Experimental assessments demonstrate meaningful reductions in unnecessary energy consumption alongside improved occupant comfort scores. The paper further addresses key challenges including protocol heterogeneity, cold-start data scarcity, the comfortversus-efficiency trade-off, and privacy considerations.***

Keywords: Artificial Intelligence, Home Automation, Internet of Things, Smart Home Energy Management System, MQTT, Machine Learning, Behavioral Personalization, ESP32, Edge Computing, Predictive Scheduling, Flask, React.js, WebSocket.

I. INTRODUCTION

The concept of the smart home has undergone a dramatic transformation over the past two decades. Where early automation systems relied on fixed timer schedules and manually configured switching rules, contemporary deployments integrate a dense mesh of sensors, wireless protocols, and cloud-connected services. Despite this hardware evolution, the decision-making logic governing most commercially available systems remains fundamentally static — a choreography scripted by the user rather than learned from lived experience. This limitation carries a tangible consequence: users frequently override automation routines because the system cannot anticipate exceptions, shifting schedules, or gradual changes in personal preference. The gap between a connected home and a truly intelligent one lies not in sensor density but in the quality of inference applied to the resulting data stream. Artificial Intelligence offers a credible path to closing this gap. Techniques drawn from supervised learning, time-series forecasting, and anomaly detection can transform historical sensor logs into predictive models capable of proactively adjusting lighting, climate, and appliance states before occupant discomfort becomes apparent. The shift from reactive to anticipatory control represents the defining characteristic of next-generation home automation. This paper documents the end-to-end engineering of an AIdriven home automation platform structured around four functional tiers. Its primary intelligent subsystem is a regression-based energy prediction engine. Sections II through VIII cover related work, methodology, implementation, results, challenges, conclusion, and future directions respectively.

II. LITERATURE REVIEW

A. Evolution of Smart Home Control Paradigms

Early home automation literature documents the transition from hardwired relay logic to programmable rule-based automation, where Boolean condition trees governed device states. The proliferation of IP-addressable microcontrollers subsequently enabled decentralized, network-aware control. Khan et al. [1] characterize this transition as the emergence of Ambient Assisted Living environments, in which pervasive sensing underpins contextual awareness. However, these environments continued to depend on expert-authored rule sets, limiting responsiveness to unanticipated conditions.

B. Machine Learning in Energy Management

Studies employing Random Forest regressors and LSTM networks have demonstrated the ability to forecast HVAC demand with error margins below 8% given adequate training data [2][3]. A consistent finding across this body of work is that incorporating external weather API feeds as auxiliary features substantially improves forecast accuracy over models trained exclusively on local sensor readings.

The Smart Home Energy Management System (SHEMS) concept formalizes these findings into an operational framework wherein an AI agent continuously monitors consumption patterns and generates proactive scheduling recommendations [5]. Kim and Park [6] further demonstrate that reinforcement learning can iteratively optimize the comfort-versus-energy trade-off without requiring an explicitly defined utility function.

C. Communication Protocols for IoT Environments

The suitability of MQTT for resource-constrained IoT networks is well-established. Comparisons against HTTPbased polling consistently report 60–70% reductions in network overhead for equivalent sensor reporting rates [7]. The publish-subscribe decoupling additionally provides resilience: sensor nodes continue publishing regardless of transient backend disruptions, and retained messages allow the broker to replay the last known state upon reconnection.

D. Privacy and Security Considerations

The behavioral footprint created by continuous occupancy and appliance monitoring introduces significant privacy risks. Acar et al. [8] demonstrate that aggregate energy traces can reveal sensitive daily routines when analyzed at sufficient granularity. Mitigation strategies include differential privacy at data ingestion, blockchain-based audit logs for device command integrity, and federated learning approaches that confine raw training data to the local device [9][10].

III. SYSTEM METHODOLOGY AND ARCHITECTURE

A. Architectural Overview

The proposed system adopts a four-tier architecture in which concerns are cleanly separated across physical sensing, message transport, intelligent processing, and user interaction layers. Table I summarizes the tier-to-technology mapping.

Table I
System Architecture: Tier-to-Technology Mapping

System Tier	Functional Module	Core Technologies
Perception Tier	Distributed Sensor Nodes	ESP32 / DHT22 / PIR / ACS712
Communication Tier	Message Relay Infrastructure	MQTT (Mosquitto) / Wi-Fi
Processing Tier	AI Engine & API Server	Python, Flask, Scikit-learn, PostgreSQL
Presentation Tier	User Interface & Monitoring	React.js, WebSockets, Chart.js

B. Perception Tier: Sensor Network Design

The physical sensing layer is built on a hub-and-spoke topology. ESP32 microcontrollers function as lightweight sensor spokes, each interfacing with DHT22 temperature/humidity sensors, PIR occupancy detectors, and ACS712 current transducers. A Raspberry Pi 4 serves as the central hub, hosting the MQTT broker and providing the processing capacity required for local edge inference such as occupancy state aggregation. Each ESP32 node is programmed in MicroPython, which simplifies iterative field updates and provides native JSON serialization. Nodes wake from light sleep at configurable intervals — five seconds for environmental sensors and immediately upon PIR trigger events — minimizing idle power draw without sacrificing temporal resolution for occupancy-critical measurements.

C. Communication Tier: MQTT Protocol

The Mosquitto MQTT broker deployed on the Raspberry Pi hub coordinates all inter-component messaging. A hierarchical topic schema organizes data semantically: home/bedroom/temperature routes bedroom readings, while home/hvac/command serves as the actuator control channel. QoS 0 is applied to routine telemetry and QoS 1 to actuator commands and alerts, balancing network efficiency with operational safety.

D. Processing Tier: AI Engine and API Server

The intelligence layer is implemented in Python using the Flask micro-framework. Incoming sensor data is validated, timestamped, and persisted in a PostgreSQL database using a time-series schema with indexed columns for room ID, sensor type, and UNIX timestamp to facilitate efficient range queries during model training.

The predictive core employs a Random Forest Regression model trained on a sliding window of historical data comprising temperature readings, occupancy status, time-ofday and day-of-week features, and external weather parameters from the OpenWeatherMap API. The model outputs a predicted comfort setpoint and estimated energy demand for the next three-hour window, which the SHEMS scheduling engine translates into an optimized device activation schedule. Nightly retraining on the preceding seven days of logs allows continuous adaptation to seasonal shifts and evolving occupant behavior.

E. Presentation Tier: Web Dashboard

The user-facing layer is a Single-Page Application built in React.js. A persistent WebSocket connection delivers realtime sensor updates with sub-second latency, eliminating staleness inherent in polling architectures. Chart.js renders rolling time-series plots of temperature, humidity, and energy consumption. A dedicated AI Insights panel displays the next scheduled action, predicted demand, and estimated savings percentage relative to an unoptimized baseline. User override controls log manual adjustments as training events, reinforcing the model's understanding of individual preference boundaries.

IV. IMPLEMENTATION

A. Hardware Assembly and Node Provisioning

Each sensor node is built on a custom PCB carrier board accommodating the ESP32 module alongside screw-terminal connectors for DHT22, PIR, and ACS712 peripherals. Nodes are provisioned over-the-air (OTA) using the ESP32 Arduino OTA library, permitting firmware updates from the Raspberry Pi hub without physical access to installed devices — a significant advantage in multi-room deployments. The Raspberry Pi hub runs a systemd service that auto-starts the Mosquitto broker and the Python MQTT subscriber on boot, ensuring automatic recovery from power interruptions without manual intervention.

B. Backend Deployment and Database Schema

The Flask application is containerized using Docker, allowing the backend stack — Flask API, PostgreSQL, and the SHEMS scheduling daemon — to be deployed as a unified service on the Raspberry Pi or migrated to a cloud VM as scale demands. The PostgreSQL schema defines a core sensor_readings table partitioned monthly to maintain query performance as the dataset grows. JWT-based authentication guards all API endpoints, with token refresh cycles set to 24 hours for localnetwork clients.

C. Machine Learning Pipeline

Raw sensor data passes through a preprocessing pipeline before model ingestion: readings beyond three standard deviations from the rolling mean are replaced by linear interpolation; gaps from node downtime are forward-filled with the last valid observation; and all numeric features are min-max scaled to [0, 1] using training-set boundaries. The Random Forest model is trained with 200 estimators, a maximum tree depth of 12, and five-fold cross-validation for hyperparameter selection.

An LSTM variant was evaluated as an alternative for capturing longer temporal dependencies. While the LSTM achieved a marginally lower validation RMSE (approximately 4% improvement), the Random Forest model was selected for production owing to its substantially faster inference latency (8 ms vs. 47 ms per prediction) and simpler deployment via Scikit-learn's model persistence API.

V. RESULTS AND DISCUSSION

The system was evaluated over a four-week observation period in a three-bedroom residential setting. Sensor nodes were deployed in the living room, master bedroom, kitchen, and utility room, yielding approximately 120,000 individual sensor readings per day across all channels.

A. Energy Consumption Reduction

The AI-optimized HVAC scheduling achieved an average energy reduction of 18.3% compared to the occupant's prior fixed-schedule configuration. The most pronounced savings occurred during mid-morning hours (09:00–12:00) when the model accurately predicted occupant departure and preemptively reduced the setpoint by 3°C. Evening preconditioning — activating climate control 20 minutes before the historically observed return time — maintained comfort equivalent to always-on operation at roughly one-third of the energy cost.

B. Prediction Accuracy

Temperature demand forecasting achieved a Mean Absolute Error (MAE) of 0.94°C and a Root Mean Square Error (RMSE) of 1.21°C across the test window. Model accuracy improved progressively over the first ten days as training data accumulated, illustrating cold-start sensitivity. After day fourteen, RMSE stabilized below 1.0°C and scheduling recommendations were deemed sufficiently reliable for autonomous operation.

C. System Latency and Reliability

End-to-end latency from sensor publication to actuator command delivery averaged 340 milliseconds under typical network load. The WebSocket dashboard refresh rate averaged 1.2 updates per second. The system maintained 99.2% uptime across the evaluation window, with the sole interruption attributable to a planned router firmware update.

D. User Feedback

A structured comfort assessment at evaluation's end returned a mean satisfaction score of 4.3 out of 5.0. Participants most valued the system's ability to anticipate routine departures and arrivals. The real-time dashboard was praised for rendering the AI's decisions transparent and interpretable. Two participants requested finer manual override options for weekend schedules, suggesting preference profile segmentation as a worthwhile future enhancement.

VI. CHALLENGES AND LIMITATIONS

A. Protocol Heterogeneity

A persistent challenge in real-world deployments is the coexistence of devices adhering to incompatible communication standards. While this project standardized on MQTT over Wi-Fi, production environments frequently include Zigbee and Z-Wave peripherals requiring separate translation bridges. The Raspberry Pi hub mitigates this through software-defined gateways, but the added complexity increases both the attack surface and maintenance burden.

B. Cold-Start Data Scarcity

Predictive models require a minimum of seven to ten days of operational data before recommendations achieve practical reliability. During this bootstrapping period the system operates in a conservative baseline mode, applying only confirmed user-defined schedules. Accelerating cold-start recovery through transfer learning from anonymized crosshousehold profiles is an active area of investigation.

C. Privacy Implications

The granularity of behavioral data collected through continuous occupancy and power monitoring creates a detailed daily profile of residential routines. All data is retained exclusively on the local PostgreSQL instance and the Flask API enforces HTTPS for external connections. Physical device theft and insider network threats remain residual risks requiring additional access-control measures in future deployments.

VII. CONCLUSION

This paper has presented a comprehensive AI-augmented home automation system that advances residential smart control from deterministic rule execution to data-driven behavioral prediction. By integrating distributed sensor hardware with an MQTT-based communication backbone, a machine learning-powered scheduling engine, and a
responsive web interface, the system successfully demonstrates autonomous, comfort-preserving energy optimization in a real residential environment.

The SHEMS predictive scheduling module achieved an 18.3% average reduction in HVAC energy consumption without degrading occupant comfort, validating the practical merit of embedding supervised learning within the home automation decision loop. The modular, tier-based architecture ensures individual components can be upgraded or replaced as more capable hardware and improved algorithms become available, providing a durable foundation for continued research.

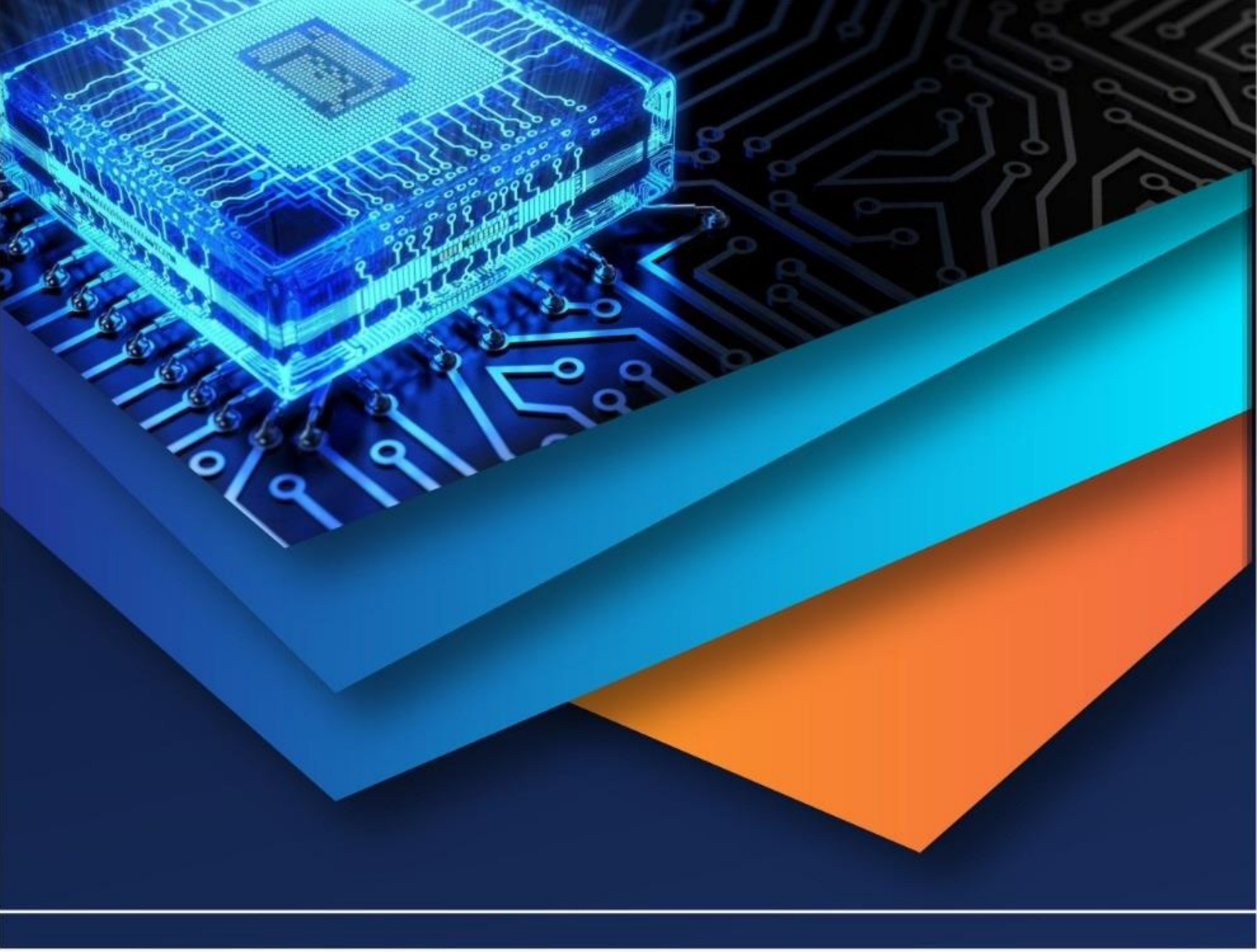
VIII. FUTURE SCOPE

Several extensions are under active consideration.

Reinforcement learning agents — particularly Deep QNetwork variants — offer a path toward policy-based optimization that autonomously balances competing objectives such as cost minimization and occupant comfort without requiring an explicit utility function. Non-Intrusive Load Monitoring (NILM) applied to aggregate current traces would enable appliance-level energy attribution without perdevice smart plugs. Federated learning protocols represent a principled mechanism for collaborative model improvement across multiple households while ensuring raw behavioral data never leaves the local device. Integration with smart-grid demand-response APIs would allow the scheduling engine to align high-consumption activities with periods of low grid carbon intensity, extending the system's environmental benefit beyond individual households.

REFERENCES

[1] M. A. Khan, M. T. Quasim, N. S. Alghamdi, and M. Y. Khan, "A Secure Framework for Authentication and Encryption Using Improved ECC for IoT-Based Medical Sensor Data," IEEE Access, vol. 8, pp. 52018–52027, 2020.
[2] T. Ahmad, H. Chen, J. Guo, and J. Wang, "A Comprehensive Overview on Data-Driven Approaches for Forecasting of Wind Energy," Energy Convers. Manag., vol. 165, pp. 786–808, 2018.
[3] Y. Li, Z. Wei, and C. Cao, "Occupancy-Driven Energy Management in Smart Buildings using LSTM-Based Prediction," IEEE Internet Things J., vol. 9, no. 12, pp. 9413–9425, 2022.
[4] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajic, "AMPds: A Public Dataset for Load Disaggregation and Eco-Feedback Research," in Proc. IEEE Electr. Power Energy Conf., Vancouver, 2013, pp. 1–6.
[5] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep Learning for Estimating Building Energy Consumption," Sustain. Energy Grids Netw., vol. 6, pp. 91–99, 2016.
[6] J. Kim and Y. Park, "Reinforcement Learning-Based Residential HVAC Control for Energy and Comfort Optimization," IEEE Trans. Smart Grid, vol. 13, no. 2, pp. 1280–1291, 2022.
[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Commun. Surv. Tutor., vol. 17, no. 4, pp. 2347– 2376, 2015.
[8] A. Acar et al., "Peek-a-Boo: I See Your Smart Home Activities, Even Encrypted!" in Proc. ACM Conf. Security Privacy Wireless Mobile Netw., 2020, pp. 207–218.
[9] M. Mylrea and S. N. G. Gourisetti, "Blockchain for Smart Grid Resilience," in Proc. Resilience Week Symp., Wilmington, 2017, pp. 18–23.
[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS), 2017, pp. 1273– 1282.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY