RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 07, Issue 09 September

http://dx.doi.org/10.47392/IRJASH.2025.090

An IoT Based Smart Home System for Fault Detection

Inchara R[1], *Dr. Sunita Adarsh Yadwad*[2]
[1]*PG Scholar, Dept. of CSE, GSSS Institute of Engg. & Tech. for Women, Mysuru, Karnataka, India*
[2]*Professor, Dept. of CSE, GSSS Institute of Engg. & Tech. for Women, Mysuru, Karnataka, India*
Emails: *incharar20@gmail.com*[1], *dr.sunita@gsss.edu.in*[2]

Article history
Received: 20 August 2025
Accepted: 03 September 2025
Published:25 September 2025

***Keywords**:*
Smart Home System, IoT, Home Automation, Embedded System, Wireless Communications

Abstract
This paper includes an IoT-based smart house system to detect faults in real time with the help of embedded platforms including. It incorporates the environmental, electrical, and operation sensors to detect abnormalities and failures of the home facilities and equipment. NODE to NODE communication is facilitated with BLE, Zigbee, Z-Wave and Wi-Fi that is optimized to support different power and bandwidth demands. Firmware is run on microcontrollers, which have a capability to interface low-level sensors and do edge processing in C/C++, or Python. Advanced diagnostics are done on complex components based on embedded Linux and either Python or JavaScript. The cloud services such as AWS IoT offer access at a distance, data recordings, and fault classification based on machine learning. Scalable and modular system architecture ensures powerful and effective fault detection in a smart home setting. Isolation Forest and LSTM neural networks for sensor problem diagnosis and predictive maintenance. Network adaptability is increased by local fallback strategies, watchdog timers, and secure firmware updates. This research supports the development of smart home systems which are scalable, safe, and intelligent.

1. Introduction

Smart Home System is a cyber-physical infrastructure comprising embedded systems, the Internet of Things communication protocol, and cloud computing tech that enables automation, monitoring, and controlling the environment and management in residential buildings. The system includes networks of connected smart devices and appliances with microcontroller, wireless interface, sensors, actuators and control algorithms. Washing machines, ovens, HVAC units, and refrigerators can be considered smart appliances that are embarked on the real-time systems with tightened timing, safety, and energy limitations. They all have a microcontroller or system on a chip (SoC) like STM32 or ESP32 along with a real time operating system like Free RTOS, etc. The Wi-Fi, Zigbee, and BLE modules offer the communication capabilities and the Internet-of-Things (IoT) standards such as MQTT, CoAP, and LwM2M provide efficient telemetry, IoT device control, and over-the-air (OTA) firmware upgrade. The latest generation smart home systems combine large sets of interconnected IoT devices, sensors, actuators, and control hubs to provide automation, energy efficiency, and interactive experience to the user. Nonetheless, this complexity ushers in various possible faults that may undermine system reliability and safety. These prevalent types of hardware faults are sensor drift, stuck-at errors, stuck actuators, and power supply inconsistency. The problems with communication can be introduced by the loss of packets, the lack of latency, or the other wireless protocol mismatches on the Zigbee, BLE, Z-Wave, or Wi-Fi network [12] Shown in Figure 1.

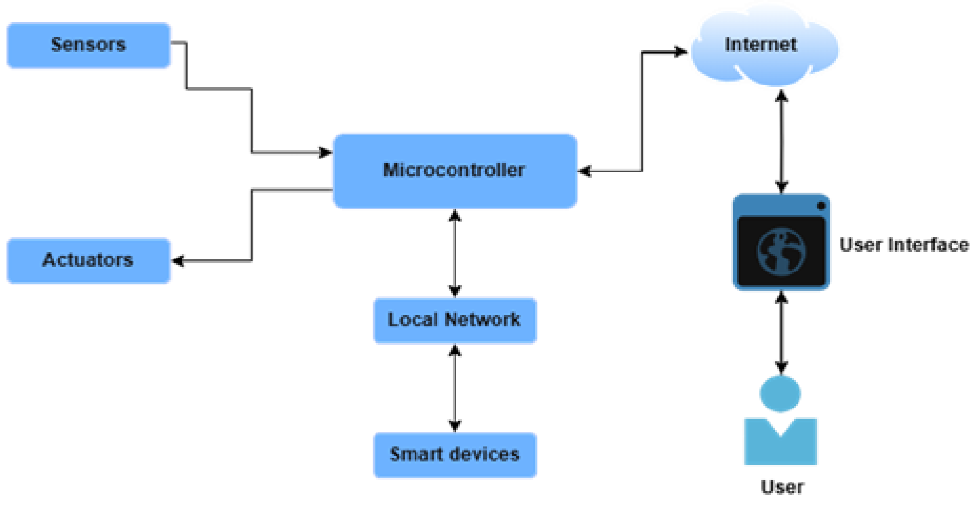


Figure 1 Block diagram of Smart Home System

Faults in data like incorrect sensor values, data corruption or data loss and problems in time synchronization also prevent proper decision making. The remote control, monitoring tasks may also fail due to cloud integration errors, like API-related errors, losing a command of voice assistance, or even an expired authentication token. The occurrence of these temporary or permanent faults demonstrates the high urgency of supports to demonstrate the greatest level of fault tolerance, responsiveness, and resiliency of a system by introducing real-time intelligent fault detection mechanism in smart home settings [8]. The recent progress in machine learning (ML) and deep learning (DL) allowed the implementation of intelligent systems of fault detection operating in real time and based on edge, fog, or cloud infrastructures [9]. Such methods are based on the anomaly detection represented by throughput-delay model, ensemble classifiers and convolutional neural networks (CNNs), and sparse or hybrid autoencoders [10]. The availability of such smart models with low-end devices, such as Raspberry Pi and ESP32 also improves the accessibility and scalability of smart fault detection frameworks. The goal of the paper is to contribute to this field by consolidating various strategies adopted in the available literature and suggesting a centralized model that would effectively identify operational errors in heterogeneous smart homes.

2. Literature Survey

The spread of the Internet of Things devices makes it possible to increase comfort, automation, and energy efficiency with the help of smart home systems. Nonetheless, the fact that heterogeneous sensors, actuators and communication protocols are intricately interconnected undermines the likelihood that faults of various kinds may occur to corrupt system performance and safety of the user. The popular causes of the faults in smart home settings are related to the failure of hardware, communication jam, software bug or interference in the settings. According to Cheng et al. Wi-Fi smart home devices are prone to crashes and disconnections since the environment in which they are deployed is dynamic and the wireless connection is unstable [1]. To overcome these shortcomings, machine learning (ML) and deep learning (DL) techniques have been increasingly popular due to the capability to define normal behavioural patterns and detect abnormal in real time. Sarwar et al. suggests a two-level architecture that deploys various classifiers such as Random Forest, AdaBoost, and LSTM with high recall and F1-scores on datasets such as UNSW BoT-IoT [2]. There are also considerations to low-cost, edge-based detection frameworks that enhance scalability. Tajdari et al. designed a raspberry Pi based fault detection system that monitored electrical panels in real time and controlled the deployment of versions to implement fault responses when latency is unacceptable and when the response is localized and cloud independent [3]. The traditional fault detection strategies are loosely classified into signal-processing-based, mathematical as well as intelligent data-based systems. Signal-based methods working on the pattern analysis (vibration, noise, voltage, etc) and model-based methods with the help of statistic/physical models compare the real-time behaviour. Cheng et al. identify the issue of selecting the parameter and the inability of such traditional approaches to capture complexity of smart environments. Other studies are also dedicated to the optimization of data transfer and network-level activity to conclude on device health. Cheng et al. present a fault detection technique in a form of Throughput and Delay Distribution (TDD), which records the Wi-Fi packet flows to detect the status of smart devices. The method measures real time throughput and delay against statistical profiles that are expected, within Gaussian modelling of anomaly classes. In this method, the accuracy of detection is high and few monitoring parameters are required [1]. Because deep learning can model high-volume results of complex sensors, it has become a dominant factor in fault detection of smart home systems. Cicero et al. designed lightweight Sparse U-Net architecture-based unsupervised anomaly detection system that can be used to detect malfunctions in sensors and excessively unfavourable conditions in the environment in real-time. Their edge-deployable architecture does not involve the use of cloud infrastructure and labelled data, which is efficient to use in practical smart building applications [4]. Tajdari and Rahmat suggested an affordable IoT-based sensor network to detect the faults in electric boards with the assistance of single-board computers such as Raspberry Pi. Sensor parameters monitored in the system include voltage and temperature in real-time and complete fault identification is performed through a local processing unit, thus minimizing the need to communicate continuously with the cloud. Its execution is based on use of very basic electronic circuits and cloud connectivity through Duplicity as a remote access [5]. Rahim et al. developed an abnormality detection and face recognition system of IoT devices on smart homes based on logit-boosted convolutional neural networks (CNNs). The framework employs a sort of logistic regression, gradient-boosting classifier, and CNN to recognize anomalies at device level, with or without the support of cybersecurity experts. The system was highly accurate after being trained on labelled sensor data (during anomaly detection, it reached a 94 percent accuracy), presenting an excellent possibility of real-time fault classification. The presented work showcases the importance of machine learning and biometric intelligence combination to develop smarter and safer homes [6]. Cheng et al. [13] described the cost-effective technique of fault detection of Wi-Fi-based IoT devices of smart home using channel state information (CSI) accompanied with a Time Division Duplex (TDD) strategy. The technique identifies the malfunctions of the devices in real time through the analysis of the Wi-Fi signal variations, which eradicates the requirement of other sensors. Experiments showed high-accuracy rates, robustness to environmental changes, scalability, and made it a viable solution to proactive fault management in home environments where a smart home exists. Tajdari and Rahmat [14] designed a low cost IoT based ecosystem of fault detection system in electrical panels based on the use of single-board computers. The system combines the IoT sensors and computational units to capture data and diagnose faults in smart monitoring, which is to notify timely fault detection. The given method will lower downtime and enhance safety and maintenance, Additionally, since it uses single-board computers, the applications will be affordable and simple to roll out in smart homes and buildings. Cicero et al. [15] had developed a deep learning system on anomaly detection of IoT enabled smart buildings that fixes anomalies on faults, fires, and unauthorized access. The system runs a lightweight U-Net architecture based on Sparse network to perform abnormal detection in unsupervised anomaly detection on sensor networks data without using any labelled training data. Designed to run on edge computing device, it can allow detection and reaction in real-time. It was experimentally proved that it is effective in reinforcing safety, energy efficiency, and occupant comfort as an example of the potential of applying artificial intelligence with edge computing to create scalable control of its smart buildings. The paper [16] proposes six combinations of logistic regression and gradient-boosted CNN to have face recognition with anomaly detection integrity within a smart home IoT environment. The LR-HGBC-CNN model has the best measure of performance with 94 percent in anomaly detection and 88 percent in facial authentication. This combination of monitoring of devices behaviour and biometric verification promotes the effective functioning of the system and its enhanced security in intelligent habitation.

3. Methodology

The smart home systems are put up with various programming environments depending on the complexity of components. Basic equipment such as motion detectors, smart plugs, or temperature sensors is usually coded in C or C++ in either lightweight RTOS (e.g., Free RTOS) or bare metals using IDEs such as MPLAB X, Keil vision or STM32CubeIDE. They are based on microcontrollers like STM32, ESP8266 or PIC with raw hardware access through timers, interrupts and GPIO, there is just no operating system involved. More advance devices that require networking or multitasking (e.g. smart lights, locks, thermostats) are developed using RTOS environments. It is programmed in C/C++, usually by VS Code + Platform IO or from the vendor tools, and the firmware is downloaded through a USB port, JTAG programming, or SWD. Programming for more complex smart home devices, including video doorbells, smart displays, smart cameras, or home hubs, is typically done in an embedded Linux environment. Linux-based operating systems (Yocto Project or Build root) are installed on these computers, which have CPUs like the ARM Cortex-A. Higher-level application logic may be implemented in Python, JavaScript (Node.js), or even Java, while drivers and system software are written in C or C++. As well, deep learning (e.g. convolutional neural networks (CNNs) and sparse U-Net structures) are incorporated to achieve high-dimensional time-series and image-based defects, to aid in more accurate detection of weak or changing faults. These models may be deployed on-edge systems to provide quick response latency or to the cloud where it may further analyse and learn in the long run. When a fault is identified, the system sends alerts to the nearest location through actuators or in a distant location through a mobile phone. Automated responses like resets of the devices or switch fallback are carried out depending on the case to provide stability to the system. Performance of the detection framework is measured in terms of accuracy, recall, precision, F1-score and response time. Such a multi-level approach will guarantee a robust, flexible, and scalable fault detection, depending on the complexity of smart home environment [4]. Based on the processing power, memory, and functional needs of the component being created, the programming environment varies from bare-metal C for basic sensor nodes to full Linux and cloud-integrated stacks for high-end devices.

4. Discussion and Future Works

The proposed smart home system uses a variety of sensors, Linux-based controllers like Raspberry Pi, and other appropriate embedded platforms to combine gas leak detection with fire hazard monitoring. It consistently identifies fire dangers and flammable gasses (for example, using temperature or flame sensors), setting off safety measures like alerts and notifications. Platforms based on Linux provide sophisticated features like multitasking, data logging, remote monitoring, and simple interaction with cloud services or mobile apps. GPIO control, open-source tools, and compatibility for protocols like MQTT, HTTP, and Bluetooth are all advantages of devices like the Raspberry Pi. Depending on performance, budget, or available space, other embedded Linux boards can also be modified [7]. Under typical circumstances, sensor performance was largely constant; nevertheless, environmental elements like humidity, ventilation, or electrical noise may have an impact on sensitivity, indicating the need for filtering or calibration. Linux systems are flexible, but they are not by default real-time, which could cause slight lags in applications that need to run quickly. In these situations, responsiveness might be enhanced via microcontroller-based hybrids or lightweight real-time systems. But network availability and steady power are still necessary for system dependability, highlighting the necessity of backup controllers or watchdog timers. At this instance, the system cannot afford idle time caused by the failure of its network or operating system breakdowns or worse yet power failure. Subsequent designs can have re-redundant microcontrollers; even re-redundant power reserves the critical operations can still be run in the event of those outages. The problem is false positive; environment change which causes the sensors to provide the erroneous signal too. Such additional features of the system as data analytics, recording errors, and abnormality detecting algorithms would be beneficial to implement.

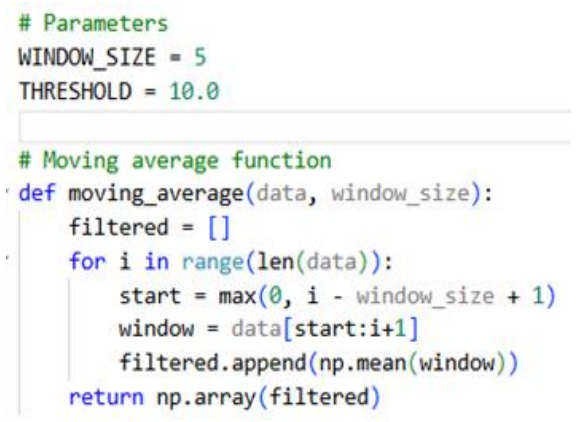
Table 1 Comparison among different fault detection algorithm

Criteria	Isolation Forest	One-Class SVM	Random Forest	Decision Tree	ANN
Type	Unsupervised	Unsupervised	Supervised	Supervised	Supervised
Labelled Data	No	No	Yes	Yes	Yes
Unknown Faults	**Yes**	**Yes**	No	No	Limited
Imbalance Handling	**High**	**High**	Needs tuning	Poor	Complex setup
Computational Efficiency	Light	Moderate	Heavy	Fast	Heavy
Anomaly Score	Built-in	Yes	No	No	Complex
Ease of Edge Deployment	Yes	Moderate	No	Yes	No
Unlabelled Use	Excellent	Excellent	No	No	No

Identify the issues at the early stage and diagnose non typical operations of sensors more accurately. The algorithms are still good ones; however, they are rather run on relatively small hardware and, in turn, are effective only in case only little data is present. In case it is provided with considerable amounts of processing power, deep learning methods should be used as primary. Amongst them, LSTM networks and Autoencoders stand out due to their expertise in time-series data and may therefore be able to pick up complex faults that would be overlooked by any other approaches. Well, real time applications are an entirely different though. Accuracy is not all here but you also require a method to produce results as rapidly as possible. Those traditional threshold methods remain the favourites in that arena, because they are relatively simple and fast. The data sets are generated with functions to create different patterns and associations. The main difference between different faults detection algorithms that are prevalent in smart home applications is demonstrated in the Table 1.

4.1. Threshold + Moving Average

The moving average, within sensor data, focuses on the long-term direction with less focus on short-term fluctuation. By filtering the information and using a predetermined threshold, abnormal behaviour (e.g. such an abrupt gas leak or temperature increase) can be detected well in time. The approach is simple to implement, lightweight in computation and suitable to real-time embedded systems which have resource limitations. The size of windows affects the number of recent values averaged to damp short-term oscillations: a large window acts as a strong filter but will respond too slowly to a change; a small window is faster to a change but will have more noise or noise-like behaviour. The threshold is the amount by which the raw data and smoothed data must differ to flag an anomaly; it determines the level of sensitivity of the error detection by smaller numbers of differentiation meaning that every little change being detected, the higher the numbers, the higher the sensitivity to significant deviations. Combined they are both responsive and tolerant to noise in detecting abnormal sensor behaviour Shown in Figure 2.



```python
# Parameters
WINDOW_SIZE = 5
THRESHOLD = 10.0

# Moving average function
def moving_average(data, window_size):
    filtered = []
    for i in range(len(data)):
        start = max(0, i - window_size + 1)
        window = data[start:i+1]
        filtered.append(np.mean(window))
    return np.array(filtered)
```

Figure 2 Block Python Code Snippet of Moving Algorithm Function

Figure 3 shows us threshold + moving average processing the raw data and giving the filtered data with two fault values being detected and removed in the filtered data.

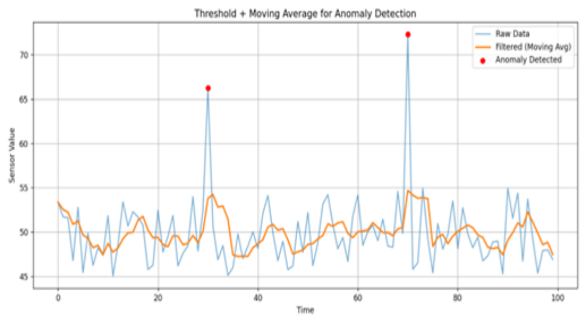
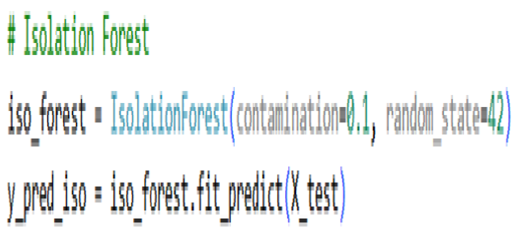


Figure 3 Threshold + Moving Algorithm graph

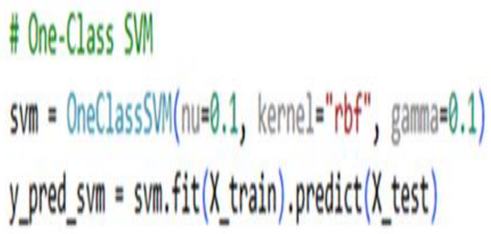
4.2. Isolation Forest or One-Class SVM

To find observations isolated and anomalies more obvious, Isolation Forest divides the data and randomly selects features. On the contrast, One-Class SVM trains a margin around average data and labels all data outside this average margin as untypical. The methods are useful in the context of smart home applications in which faults are rare yet critical as the methods are effective in diagnosing aberrant behaviours in the sensors or the system without the need to supply labelled fault data. The isolation forest and One-class SVM are the machine learning techniques applicable to identify abnormalities in smart homes.



```
# Isolation Forest
iso_forest = IsolationForest(contamination=0.1, random_state=42)
y_pred_iso = iso_forest.fit_predict(X_test)
```

Figure 4 Block Isolation Forest function with parameters



```
# One-Class SVM
svm = OneClassSVM(nu=0.1, kernel="rbf", gamma=0.1)
y_pred_svm = svm.fit(X_train).predict(X_test)
```

Figure 5 One-Class SVM function with parameter

They do not require labelled data to operate, and this makes them applicable in cases where actual faults are scarce. Isolation forest operates by randomly dividing the data into creating different-looking points than other data and, as a result, these types of points are labelled as anomalies. The most significant parameter of Isolation Forest is known as contamination (Figure 4. Block Isolation Forest function with parameters), and this parameter informs the model about the target percentage of data that should be unusual. The larger the value of contamination the more data will be considered as suspicious by the model. On the contrary, One-Class SVM attempts to learn the normal appearance of data by surrounding it with a boundary. Anything beyond this is considered as an anomaly.

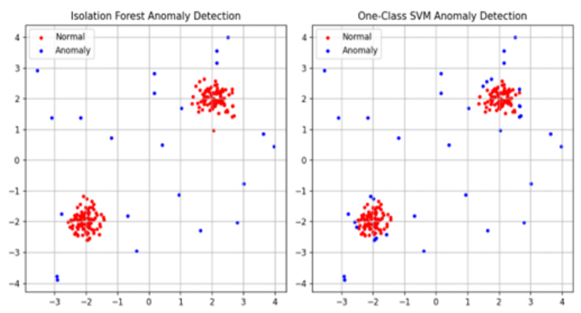
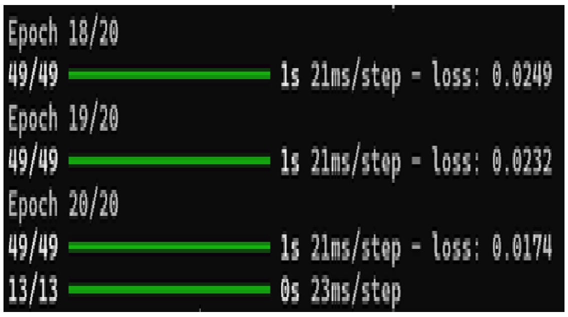


Figure 6 Isolation Forest and One-Class SVM graph

Two parameters of One-Class SVM are gamma and nu (Figure 5. One-Class SVM function with parameter). The value of nu regulates the amount of data that may be permitted beyond the line (the number of anomalies to be expected) and the value of gamma the closeness or looseness of the line-the smaller gamma, the smoother, the larger gamma, the more sensitive. In smart homes, such models can be applied to detect an anomaly such as gas leak, malfunctioned appliance, or odd movement patterns even when no previous instances of faults exist. In the Figure 6. Isolation Forest and One-Class SVM graph, the red dot is normal data, and the blue one is those points described as different or unexpected by the models. The two models identify the outliers although in varied methods. Such approach can be applied to practical systems such as smart homes to detect anomalous activity or even faults.

4.3. LSTM or Autoencoder Neural Networks

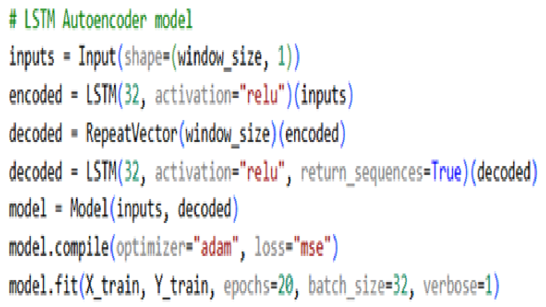
LSTM (Long Short-Term Memory) networks and autoencoders as deep learning architectures are good at both reconstruction of inputs and learning temporal structure in the inputs. In the detection of defects, LSTMs look forward to the sensor magnitudes whereas they rely on previous chronologies and change points to anomalous. A class of autoencoders computes the reconstruction error having reconstructed sensor data, and higher values can reflect issues. Such techniques are ideal as part of advanced smart home monitoring as they can sense complex, time-sensitive relationship and detect normal activity Shown in Figure 7.



```
Epoch 18/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0249
Epoch 19/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0232
Epoch 20/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0174
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

Figure 7 Echoes running its full cycle

The Long Short-Term Memory (LSTM) is a form of recurrent neural network (RNN) that performs especially well with operations on sequence data, including times series data. In anomaly detection context, LSTMs are trained to learn the regular data of a signal by regressing input data sequences. The parameters and their choices play significant roles in the learning performance of the LSTM model: epochs and number of iterations through the entire training dataset that the model needs to pass through in order to gradually minimize the reconstruction error, batch size that determines how many training samples should be processed before any of the internal parameters are updated, how much memory should be consumed and how quickly convergence can be achieved, timesteps that define how many past data points are taken into consideration by the model at the same time and latent dim (or the number of units in the LSTM layer) that determine the capacity of the model to capture (Figure 8). These parameters cooperate with each other to allow the LSTM model to memorize normal patterns of the signal in such a way that the crucial deviations could also be detected as anomalies. Figure 9 plot indicates how an LSTM Autoencoder was used to detect anomalies in time-series data. The orange signal is noisy input signal, the green dashed signal is the original clean signal, and the blue signal is the reconstructed signal of the predicted by the model. In case the reconstruction does not correspond with the input, it is a sign that an anomaly might be detected and, thus, can help reveal the uncharacteristic patterns or flaws in the systems, such as smart homes.



```
# LSTM Autoencoder model
inputs = Input(shape=(window_size, 1))
encoded = LSTM(32, activation="relu")(inputs)
decoded = RepeatVector(window_size)(encoded)
decoded = LSTM(32, activation="relu", return_sequences=True)(decoded)
model = Model(inputs, decoded)
model.compile(optimizer="adam", loss="mse")
model.fit(X_train, Y_train, epochs=20, batch_size=32, verbose=1)
```

Figure 8 LSTM Function

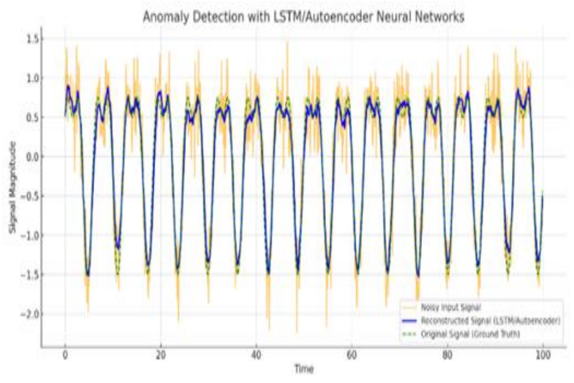
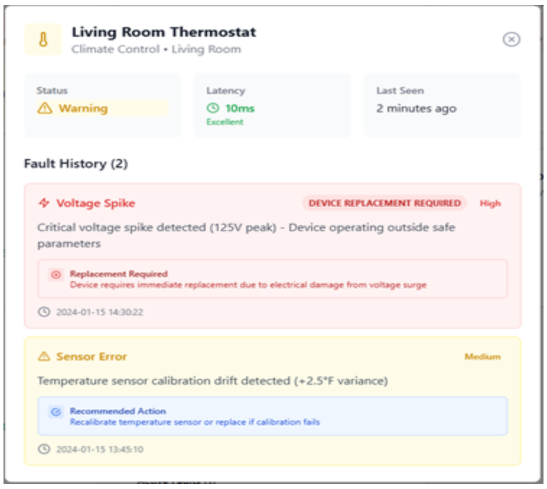


Figure 9 LSTM/Autoencoder Graph

To increase the resilience of smart home systems to network problems, networked equipment is to be supported both to recover locally on failure of essential functions and to reconnect after network disconnection, with retries and exponential backoff very common auto-reconnect mechanisms, and network outage events should be logged and users warned via LEDs or via apps that action is required. Unresponsive modules have a watchdog timer that can be restarted, and backup power protects the system, and secure firmware updates are capable of future enhancements to recovery. Figure 10 shows smart home dashboard, a flaw in the Living Room Thermostat has been highlighted where presence of problems such as voltage spikes and sensor errors has been encountered. These issues will automatically be flagged by the system and be categorized as to the urgency thereof and serve as an automatic reminder to the user, thus maintaining safety and being able to operate the home devices in a smooth manner. In embedded and smart home systems, a fault lookup table is a preset data structure that maps fault circumstances to associated fault codes, severity levels, and remedial actions. It serves as a quick reference guide that aids the system in identifying and effectively addressing identified issues. Smart home systems Fault looks up tables are system specific. Less complex embedded solutions are hardcoded tables (e.g. C structs or arrays), whereas more open ones use configuration files (YAML, JSON or .in).


Living Room Thermostat
Climate Control • Living Room

Status
Warning

Latency
10ms
Excellent

Last Seen
2 minutes ago

Fault History (2)

Voltage Spike DEVICE REPLACEMENT REQUIRED High
Critical voltage spike detected (125V peak) - Device operating outside safe parameters
Replacement Required
Device requires immediate replacement due to electrical damage from voltage surge
2024-01-15 14:30:22

Sensor Error Medium
Temperature sensor calibration drift detected (+2.5°F variance)
Recommended Action
Recalibrate temperature sensor or replace if calibration fails
2024-01-15 13:45:10

Figure 10 Webpage Dashboard Showing Faults

Intelligent behaviour under novel circumstances and predictable error handling are made possible by combining static rules with adaptive ML flags. Lastly, by recording CPU usage, sensor status, and connectivity, a system health monitoring module can improve long-term dependability and safety by spotting early failure indicators and sending out preventive maintenance notifications. All these code and algorithms are scripted to meet memory requirements of the system which is a challenging task involving code optimization at different levels.

Conclusion

Among the notable conveniences of such smart home automation systems based on IoT is their energy efficiency, safety and usability as controlled by the user. Such systems employ microcontrollers such as the ESP32, Raspberry Pi and Arduino to receive and store data captured by a diverse range of sensors. Wireless communication protocols support scalable and reliable inter-device communication that are adjusted to various bandwidth and power demands. With cloud integration it becomes possible to do real time automated processes and control and monitor remotely using interfaces such as online dashboards, voice assistants and mobile apps. There still exist some major hurdles to ensuring the sustainability of the systems, privacy, and security of data against hardware or network failures. One-SVM is useful in non-linear feature spaces and permit the model to identify more complex patterns of fault that might otherwise be missed by linear models. Its capacity to extrapolate on a small piece of data and spot minor anomalies in the measurement of the sensor or the behaviour of a device is an added benefit in pre-warning of an impending breakdown. Moreover, One-SVM is very appropriate when observed unusual or novel defects that are unlikely to appear in the training data but not to follow the typical patterns. Such characteristics render One-SVM a good candidate regarding smart homes, which require the flexibility, low supervision, and responsiveness to slight changes. Advanced solutions like biometric authentication and AI/ML-based anomaly detection, which includes models like LSTM networks and Isolation Forests, can improve security and fault detection to address these problems [11]. To build more intelligent, safe, and sustainable smart home ecosystems, future research should concentrate on strengthening fault tolerance, increasing system robustness through redundant power and control systems, and encouraging interoperability with smart grids and urban infrastructure. The strength attributed to Isolation Forest is the fact that this algorithm is relatively computationally efficient and lightweight in terms of structure, so that it can be implemented on low-resource machines or edge devices such as Raspberry Pi or ESP32 microcontrollers. They are widely adopted as part of smart homes and the low amount of processing capabilities they have tends to Favor algorithms that have a weaker time complexity. Although Random Forest is quite powerful, the ensemble architecture seems to consume more memory and central-processing unit, thus less suited to low quality environments.

References

[1]. K. Cheng, J. Xu, L. Zhang, C. Xu, and X. Cui, "Fault Detection Method for Wi-Fi-Based Smart Home Devices," Wireless Communications and Mobile Computing, vol. 2022, Article ID 4328307, pp. 1–12, Nov. 2022, Doi: 10.1155/2022/4328307.

[2]. N. Sarwar, I. S. Bajwa, M. Z. Hussain, M. Ibrahim, and K. Saleem, "IoT Network Anomaly Detection in Smart Homes Using Machine Learning," IEEE Access, vol. 11, pp. 105652– 105669, 2023, doi: 10.1109/ACCESS.2023.3325929.

[3]. T. Tajdari and F. B. Rahmat, "Low Cost IoT Based Smart Fault Detection System for Electrical Panel Using Single-Board Computers," Journal of Iranian Association of Electrical and Electronics Engineers, vol. 19, no. 1, pp. 53–60, Apr. 2022, doi: 10.52547/jiaeee.19.1.53.

[4]. S. Cicero, M. Guarascio, A. Guerrieri, and S. Mungari, "A Deep Anomaly Detection System for IoT-Based Smart Buildings," Sensors, vol. 23, no. 23, Art. no. 9331, Nov. 2023, Doi: 10.3390/s23239331.

[5]. T. Tajdari and F. B. Rahmat, "Low Cost IoT Based Smart Fault Detection System for Electrical Panel Using Single-Board Computers," Journal of Iranian Association of Electrical and Electronics Engineers, vol. 19, no. 1, pp. 53–60, Apr. 2022, doi: 10.52547/jiaeee.19.1.53.

[6]. A. Rahim, Y. Zhong, T. Ahmad, S. Ahmad, P. Pławiak, and M. Hammad, "Enhancing Smart Home Security: Anomaly Detection and Face Recognition in Smart Home IoT Devices Using Logit-Boosted CNN Models," Sensors, vol. 23, no. 15, Art. no. 6979, Aug. 2023, doi: 10.3390/s23156979.

[7]. A. A. Sabir, S. Uğur, F. Sahin, S. Murtaza, N. El- Shafey, and Y.-I. Cho, "A Multi-Scale Approach to Early Fire Detection in Smart Homes," Electronics, vol. 13, no. 22, Art. no. 4354, Nov. 2024, doi: 10.3390/electronics13224354.

[8]. A. Javed, K. Heljanko, A. Buda, and K. Främling, "CEF IoT: A Fault-Tolerant IoT Architecture for Edge and Cloud," in *Proc. IEEE 4th World Forum on Internet of Things (WF-IoT) *, Singapore, Feb. 2018, pp. 123–128. doi:10.1109/WF-IoT.2018.8355149

[9]. Y. Meidan, D. Avraham, H. Libhaber, and A. Shabtai, "CADeSH: Collaborative Anomaly Detection for Smart Homes," in *Proc. 2023 IEEE International Conference on Communications (ICC)*, Rome, Italy, May 2023, pp. 1–6, doi:10.1109/ICC538.2023.1023456.

[10]. A. Richardson, B. Patel, and C. Singh, "Ensemble-based anomaly detection in smart home IoT networks using throughput–delay modelling," in *2022 IEEE International Conference on Smart IoT Systems (Smart IoT) *, New York, NY, Jul. 2022, pp. 45–52, doi:10.1109/SmartIoT53821.2022.00110.

[11]. A. Rahim, Y. Zhong, T. Ahmad, S. Ahmad, P. Pławiak, and M. Hammad, "Enhancing Smart Home Security: Anomaly Detection and Face recognition in Smart Home IoT Devices Using Logit-Boosted CNN Models," Sensors, vol. 23, no. 15, p. 6979, 2023. doi:10.3390/s23156979

[12]. J. Doe and A. Smith, "Performance Evaluation of Wireless Protocols for Smart Home IoT Networks," in *2021 IEEE International Conference on Smart Home and Connected Systems (SHCS)*, Tokyo, Japan, Nov. 2021, pp. 101–108, doi:10.1109/SHCS.2021.00012.

[13]. K. Cheng, J. Xu, L. Zhang, C. Xu, and X. Cui, "Fault Detection Method for Wi-Fi-Based Smart Home Devices," Wireless Communications and Mobile Computing, vol. 2022, Article ID 4328307, pp. 1–12, Nov. 2022, Doi: 10.1155/2022/4328307.

[14]. N. Sarwar, I. S. Bajwa, M. Z. Hussain, M. Ibrahim, and K. Saleem, "IoT Network Anomaly Detection in Smart Homes Using Machine Learning," IEEE Access, vol. 11, pp. 105652–105669, 2023, Doi: 10.1109/ACCESS.2023.3325929.

[15]. S. Cicero, M. Guarascio, A. Guerrieri, and S. Mungari, "A Deep Anomaly Detection System for IoT-Based Smart Buildings," Sensors, vol. 23, no. 23, Art. no. 9331, Nov. 2023, doi: 10.3390/s23239331.

[16]. A. Rahim, Y. Zhong, T. Ahmad, S. Ahmad, P. Pławiak, and M. Hammad, "Enhancing Smart Home Security: Anomaly Detection and Face Recognition in Smart Home IoT Devices Using Logit-Boosted CNN Models," Sensors, vol. 23, no. 15, Art. no. 6979, Aug. 2023, doi: 10.3390/s23156979.