Kalman-Enhanced Wake-on-Fire System for Smart Home Fire Response with Network and Energy Optimization

Kirankumar Manivannan
School of Electronics Engineering
Vellore Institute of Technology,
Chennai Campus
Chennai, India
Kiransun5@gmail.com

Sumedh Kudale
School of Computer Science and Engineering
Vellore Institute of Technology, Chennai Campus
Chennai, India
https://orcid.org/0009-0003-3005-1538

Mukunda Hosangadi
School of Computer Science and Engineering
Vellore Institute of Technology, Chennai Campus
Chennai, India
https://orcid.org/0009-0004-8699-0101

***Abstract*— Fire hazards in residential environments pose serious threats to safety and property. Traditional fire alarm systems are limited by delayed detection and the absence of remote alert capabilities. This paper presents a smart home fire safety system leveraging the Arduino Mega 2560 microcontroller, multiple sensors, and IoT based communication to address these challenges. The system monitors environmental changes using an MQ 2 gas sensor and DHT22 temperature sensor. The proposed system uses ESP32 CAM for imaging while integrating a NEO 6M GPS module for geolocation and an ESP8266 WiFi module for real time alert transmission. Upon detecting a fire, the system activates a sprinkler and sends alerts containing sensor readings, location, and image data. Further software enhancements to the system include Kalman filtering for reliable noise reduced input from sensor, a Wake on Fire energy saving model for low power operation, and a simulation based evaluation of network routing protocols to assess fast and efficient propagation performance. Together, these improvements deliver a temperature accurate, energy efficient and network efficient embedded fire safety solution.**

***Keywords*— Fire detection, IoT, Arduino Mega 2560, Kalman filter, Wake-on-Fire, low-power systems, ESP8266, ESP32-CAM, smart home automation, real-time monitoring, MQ-2 gas sensor, DHT22 temperature sensor, NEO-6M GPS module, network simulation.**

I. Introduction

Fire outbreaks in residential areas often lead to catastrophic losses due to delayed, inaccurate detection and insufficient response mechanisms. Traditional fire alarm systems typically rely on basic smoke sensors and offer limited remote monitoring or suppression capabilities. With the advancement of embedded systems and the Internet of Things (IoT) [7], modern fire safety systems can now incorporate real-time electronic sensing, automated actuation, and network-based emergency transmission to cloud to significantly reduce response times and improve reliability [7][4].

This paper proposes a smart home fire safety system built around the Arduino Mega 2560 platform integrating an MQ-2 gas sensor, DHT22 temperature sensor, ESP32-CAM, NEO-6M GPS module, and ESP8266 Wi-Fi. The system periodically monitors ambient conditions, detects fire indicators like temperature and smoke, captures and transmits images, provides location data, and activates an actuator-based sprinkler system for immediate suppression.

Beyond conventional approaches, this work introduces three embedded software enhancements: Kalman filter-based sensor noise reduction to improve detection under noisy conditions [9], a power-efficient Wake-on-Fire (WoF) model that minimizes energy usage by keeping nodes in sense-sleep mode until a fire is detected, and a simulation of fire alert propagation across networked nodes to evaluate communication delays and protocol efficiency [10]. Together, these components create a precise, efficient, and deployable fire safety solution suitable for modern smart homes.

979-8-3315-1504-1/25/$31.00 ©2025 IEEE

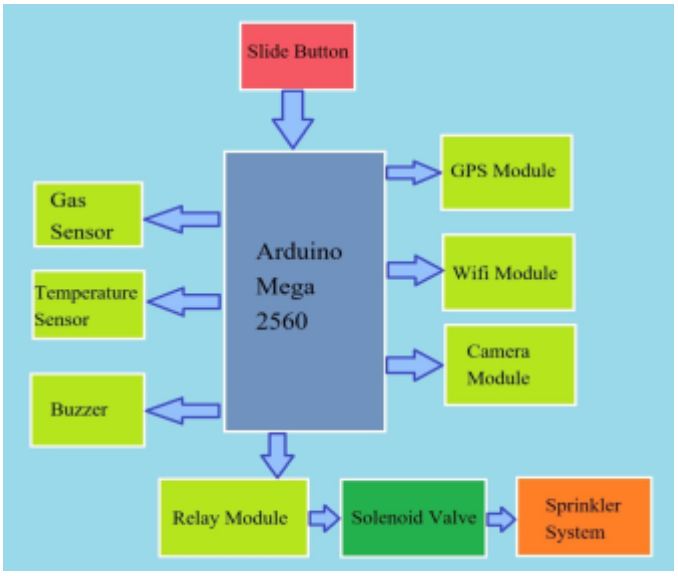

Fig. 1. Arduino based Fire Detection and Alarm Hardware Block Diagram

II. SYSTEM DESIGN

The smart fire safety system is built around an Arduino Mega 2560, which serves as the main controller for processing sensor inputs and executing programmed responses. Fig. 1 illustrates the system's block diagram, showing connections between sensors, actuators, and communication modules. Fig. 2 showcases the circuit diagram while Fig.3 explains the working of the system.

The MQ-2 gas sensor and DHT22 temperature sensor continuously monitor environmental conditions. If gas levels exceed 300 PPM or temperature surpasses 60°C, the system classifies the event as a potential fire. To improve temperature accuracy, readings are passed through a Kalman Filter configured for noisy environments, with the following parameters: 100 time steps, fire introduced at timestep 60, baseline temperature 27°C, fire temperature leading to 60°C, and noise level 2. The filter tracks both temperature and rate of change using a 2D state model with a process noise matrix Q and measurement noise R, initialized with moderate uncertainty.

A NEO-6M GPS module retrieves location data, while the ESP32-CAM captures an image of the affected area. For communication, the ESP8266 Wi-Fi module transmits fire alerts containing sensor data [3], GPS coordinates, and image captures to a secure cloud server [6] when it detects a fire. An actuator-controlled sprinkler system, managed via a KF-301 relay module, ensures immediate fire suppression. A buzzer also provides an auditory warning to occupants.

The Arduino Mega 2560 interfaces with all modules and communicates with the ESP32-CAM via UART. The system operates on a 12V power supply, with voltage regulators to support 3.3V modules like ESP8266 and GPS [5].

Power efficiency is addressed through a Wake-on-Fire (WoF) architecture. Before fire detection, nodes cycle between sleep (2μA), sensing (6mA), and alert (20mA) states. Simulations used the following parameters: 3.3V supply, sampling every 5 seconds. We simulate a timeframe of 1000 seconds with fire occurring at 620 seconds. A heatmap of time-state energy usage was generated from this simulation, seen in Fig.5. For comparison, a second model calculated total energy for WoF vs. Always-On operation with simulation conditions: fire starting at 620s, 1000s total time, 10s sampling interval, and power draws of 0.0001W (sleep), 0.01W (sense), and 0.05W (alert). Refer Table II and Fig.6 for output of the second model.

Network efficiency was evaluated using a simulation of 100 nodes randomly distributed in a 100x100 unit area, with a communication range of 30 units. In the first simulation, the fire source was node 0, and energy cost per transmission was set to 0.05 mWh. Refer Table II for output of the first simulation. For the second simulation, a protocol visualization was done for the number of transmissions per protocol. A simpler model was used here: 100 nodes in a 1.0 x 1.0 area with 0.2 communication range. Fig. 7 illustrates the bar chart comparing the number of transmissions. These simulations helped analyze network protocols in terms of latency, energy, and coverage efficiency.

Together, the hardware setup and simulation parameters provide a comprehensive embedded system model for smart fire detection, suppression, and communication.

979-8-3315-1504-1/25/$31.00 ©2025 IEEE

III. Working

The smart home fire safety system operates through sensor-based monitoring, automated suppression, and real-time communication. Upon startup, the Arduino Mega 2560 initializes all connected modules including the MQ-2 gas sensor, DHT22 temperature sensor, ESP32-CAM, NEO-6M GPS, ESP8266 Wi-Fi, and KF-301 relay. Each module is supplied with the required voltage through a regulated 12V power system, ensuring stable operation of 3.3V components like the GPS and Wi-Fi modules.

The MQ-2 gas sensor analyzes smoke and gas concentrations in real time, while the DHT22 records ambient temperature. The ESP32-CAM remains in standby and activates only when a fire is confirmed. When gas concentration exceeds 300 PPM or temperature surpasses 60°C, the Arduino initiates a verification routine. A short time window is used to cross-check readings and filter out transient anomalies.

To improve detection reliability, a Kalman Filter fuses sensor readings and smooths noise [9]. The state vector includes temperature and its rate of change and is represented as:

$$x = \begin{bmatrix} T \\ \dot{T} \end{bmatrix}, \quad F = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The filter updates the state estimate and uncertainty using equations:

$$\begin{aligned} \mathbf{x}_{k|k-1} &= \mathbf{F}\mathbf{x}_{k-1} \\ \mathbf{P}_{k|k-1} &= \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R})^{-1} \\ \mathbf{x}_k &= \mathbf{x}_{k|k-1} + \mathbf{K}_k(z_k - \mathbf{H}\mathbf{x}_{k|k-1}) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1} \end{aligned}$$

Once confirmed, the system initiates multi-layered response actions. The piezoelectric buzzer sounds a local alarm. The ESP32-CAM captures real-time fire images. Simultaneously, the NEO-6M GPS fetches location data. The ESP8266 transmits an alert—containing sensor values, GPS coordinates, and captured images—to a cloud server for emergency response and homeowner notification.

In parallel, the Arduino triggers the KF-301 relay, which powers an electric solenoid valve, initiating the water sprinkler system for localized suppression. The system continues monitoring fire parameters, deactivating suppression once values fall below safe thresholds. If fire persists, updated alerts are transmitted every 30 seconds.

The system also includes a Wake-on-Fire (WoF) energy management model. The microcontroller sleeps at 2 μA, wakes periodically at 6 mA to sense, and enters 20 mA alert mode during fire. Energy is calculated using:

$$E = V \times I \times t$$

with voltage at 3.3V, and current/time depending on state.

Network propagation is simulated over 100 nodes in a 100x100 unit field, with 0.05 mWh per transmission, comparing flooding vs. cluster-based alerting models. Refer Table III for simulation of energy comparison.

This integrated, layered approach combines accurate detection, autonomous suppression, real-time alerting, and low-power design, providing a comprehensive embedded fire response solution for smart homes.

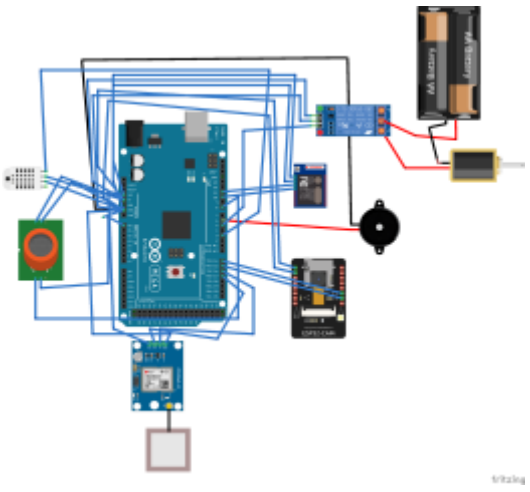

Fig. 2. Circuit Diagram

979-8-3315-1504-1/25/$31.00 ©2025 IEEE

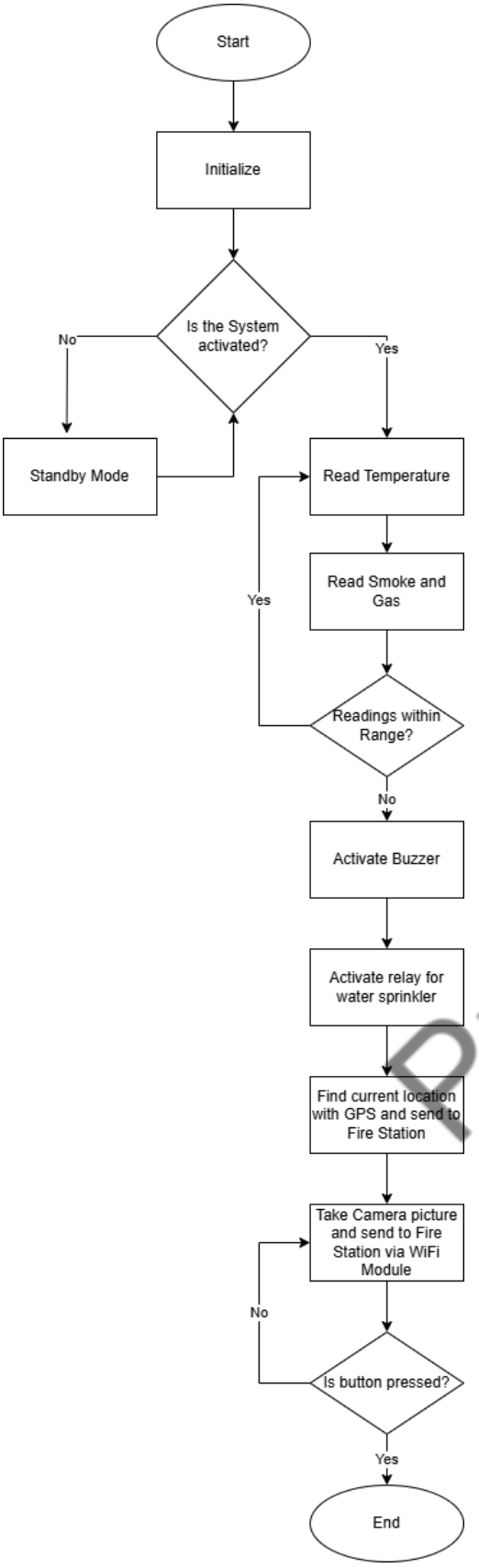

Fig. 3. System Flow Chart for Working of Hardware Components

IV. Results

The embedded software enhancements for the proposed system were simulated in Python 3.11, utilizing libraries such as SimPy for discrete-event simulation, NumPy for numerical operations, Matplotlib and Plotly for data visualization, and FilterPy for Kalman filtering. Results were categorized into three domains: sensor filtering accuracy, energy efficiency, and communication protocol performance.

A. Sensor Fusion Accuracy using Kalman Filter

Temperature sensor data was simulated under noisy conditions and evaluated using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Three scenarios were tested, which can be seen in Table I.

TABLE I. Calculated Errors for Kalman Filter Under 3 Scenarios

Scenario	MAE (°C)	RMSE (°C)
Original Kalman Filter	0.5021	0.6481
With Faulty Sensor	0.5468	0.7373
Tuned Kalman Filter	0.4667	0.6179

The plotted result of the simulation comparing reduction in noise can be seen in Fig.4.

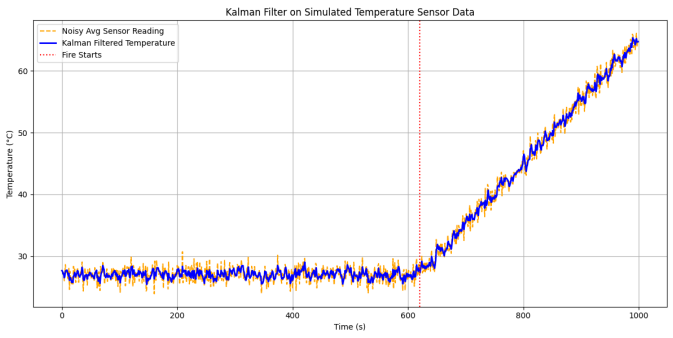

Fig. 4. Kalman Filter on Simulated Temperature Sensor Data

B. Energy Efficiency Analysis

The Wake-on-Fire model was simulated with realistic power draw values for three states: sleep (2 µA), sensing (6 mA), and alert (20 mA). Comparisons were made between Wake-on-Fire

979-8-3315-1504-1/25/$31.00 ©2025 IEEE

and a traditional always-on system over 1000 seconds.

TABLE II. Energy Comparison Between Always-On and Wake-on-Fire

System Type	Total Energy Used (mWh)	Energy Saved (mWh)	Reduction (%)
Always-On	7.22	-	-
Wake-on-Fire	5.74	1.48	20.56

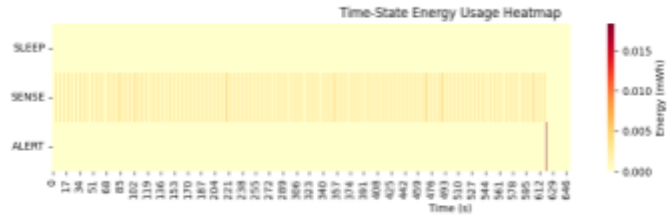

Fig. 5. Time-State Energy Usage Heatmap

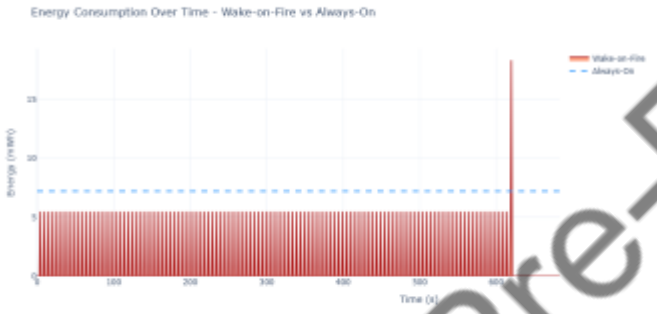

Fig. 6. Energy Consumption Over Time - Wake-on-Fire vs Always-On

C. Network Protocol Evaluation

A discrete event network simulation was conducted with 100 nodes in a 100x100 field, simulating fire alerts from node 0 using two protocols: flooding and cluster-based routing.

TABLE III. Network Protocol Parameters and Energy Comparison

Protocol	Latency	Transmissions	Energy Used (mWh)
Flooding	4	100	5.0
Cluster -Based	3	24	1.2

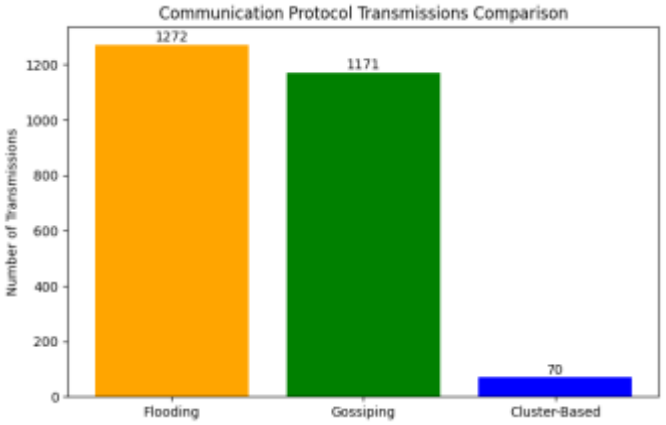

Fig. 7. Communication Protocol Energy Comparison

These results validate the effectiveness of Kalman filtering in noisy environments, confirm the energy-saving advantage of Wake-on-Fire architecture, and highlight the protocol trade-offs between speed, energy, while maintaining similar coverage. Flooding offers complete coverage at higher energy cost, whereas cluster-based approach reduces energy.

V. Conclusions

The proposed smart home fire safety system integrates multi-sensor detection, real-time image capture, GPS-based location tracking, and automated fire suppression into a unified embedded platform. Unlike conventional alarms, this system enables faster detection, immediate local response, and cloud-based alerting for emergency coordination.

Furthermore, this work introduces several embedded software enhancements to the proposed system: Kalman filtering for accurate fire classification under sensor noise, a Wake-on-Fire (WoF) architecture for power-efficient operation, and discrete event simulations for evaluating network-level communication delays. The system was developed and validated through Python-based simulations using libraries such as SimPy, NumPy, and FilterPy, demonstrating the benefits of accurate sensor readings, significant energy savings, and protocol-aware network efficiency.

979-8-3315-1504-1/25/$31.00 ©2025 IEEE

These results establish a foundation for deploying precise, energy-aware fire detection systems in smart homes. Future work may include integration with AI-based flame and smoke recognition, machine learning classifiers for adaptive thresholds, and deployment over real mesh-based IoT infrastructures for scalable smart safety networks.

References

[1] M. S. Dauda and U. S. Toro, "Arduino Based Fire Detection and Control System," *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 11, pp. 447-453, 2020.

[2] M. S. Bin Bahrudin, R. A. Kassim, and N. Buniyamin, "Development of Fire Alarm System Using Raspberry Pi and Arduino Uno," *2013 International Conference on Electrical, Electronics and System Engineering (ICEESE)*, Kuala Lumpur, Malaysia, pp. 1-6, 2013.

[3] S. Suwarjono et al., "Design of a Home Fire Detection System Using Arduino and SMS Gateway," *Knowledge*, vol. 1, no. 1, pp. 7-12, 2021.

[4] N. Mahzan et al., "Design of an Arduino-Based Home Fire Alarm System with GSM Module," *Journal of Physics: Conference Series*, vol. 1019, no. 1, pp. 1-8, 2018.

[5] A. R. Z. Abdul Latiff and Z. Mohammad, "The Development of Fire Detection and Automated Fire Extinguisher System by Using Arduino and NodeMCU ESP8266," *Malaysian Journal of Industrial Technology*, vol. 5, no. 5, pp. 1-7, 2021.

[6] A. R. Hutauruk et al., "Implementation of Wireless Sensor Network as Fire Detector using Arduino Nano," *2019 International Conference of Computer Science and Information Technology (ICoSNIKOM)*, Medan, Indonesia, pp. 1-5, 2019.

[7] F. S. Perilla et al., "Fire Safety and Alert System Using Arduino Sensors with IoT Integration," *Proceedings of the 2018 7th International Conference on Software and Computer Applications (ICSCA '18)*, pp. 199-203, 2018.

[8] S. Sharma et al., "Development of an Early Detection System for Fire Using Wireless Sensor Networks and Arduino," *2018 International Conference on Sustainable Energy, Electronics, and Computing Systems (SEEMS)*, Greater Noida, India, pp. 1-5, 2018.

[9] Welch, Greg, and Gary Bishop. "An introduction to the Kalman filter." (1995): 2.

[10] Silvani, Xavier, et al. "Evaluation of a wireless sensor network with low cost and low energy consumption for fire detection and monitoring." Fire Technology 51 (2015): 971-993.

979-8-3315-1504-1/25/$31.00 ©2025 IEEE