



IJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85293>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Offline Emergency Communication Network Using ESP32 and LoRa Technology

Ayush Kalambe¹, Kshitij Dushing², Rutuja Bhusari³, Shreya Mahetkar⁴, Vedant Bavaria⁵, Prajkta Wankhede⁶,
Ruchi Kadwane⁷

^{1,2,3,4,5,6} Students, Department of Computer Science and Engineering, G.H. Rasoni University, Amravati, India

⁷ Assistant Professor, Department of Computer Science and Engineering, G.H. Rasoni University, Amravati, India

Abstract: Communication systems that depend on cellular networks or Internet connectivity may become unreliable or unavailable during disasters and emergency situations. This paper presents an offline emergency communication system that uses ESP32-based nodes and 433 MHz LoRa communication to transmit SOS information without requiring conventional network infrastructure. The system allows an emergency alert to be generated through either a physical SOS button or a local Wi-Fi web interface hosted by the transmitter node. The transmitter integrates a NEO-6M GPS module to obtain location information and combines it with emergency details such as the emergency type, number of people, and message before transmitting the SOS packet. The receiving node acknowledges the received packet and transfers the emergency information to a local computer through USB serial communication. The system also incorporates relay-based multi-hop forwarding to support communication beyond a direct transmitter-receiver link. On the computer side, a Python backend processes the received data, calculates emergency priority, stores the information in an SQLite database, and provides the data to a local rescue dashboard. The developed system integrates offline SOS generation, LoRa-based communication, location reporting, acknowledgement, emergency prioritisation, and local monitoring into a single communication framework for emergency scenarios where conventional connectivity is unavailable.

Keywords: LoRa, ESP32, Emergency Communication, SOS Transmission, Multi-Hop Communication, GPS Localization, Offline Communication

I. INTRODUCTION

During natural disasters and other emergency situations, communication infrastructure such as cellular networks and the Internet may become unavailable due to power failures, physical damage, or network congestion. This can make it difficult for affected people to send emergency information and for rescue teams to receive the location and status of people in need. Therefore, communication systems that can operate without conventional network infrastructure are important for disaster response [1], [2].

LoRa (Long Range) communication has received attention for such applications because it provides long-range, low-power wireless communication without requiring cellular or Internet connectivity. Several studies have explored LoRa for emergency communication using direct communication, mesh networks, and multi-hop forwarding. For example, existing systems have demonstrated LoRa mesh communication for disaster situations and have evaluated its ability to deliver emergency information through multiple nodes [3], [4]. LoRaWAN has also been investigated for allowing people affected by disasters to report their status to emergency authorities [5].

The addition of GPS can further improve the usefulness of these systems by providing the location of an emergency.

Previous research has combined LoRa with GPS and multi-hop communication to transmit both emergency messages and location information. Some systems have also introduced local web interfaces and offline maps so that users can interact with the network without Internet connectivity [6]. These studies show that LoRa, GPS, mesh communication, and local interfaces can provide useful capabilities for infrastructure-independent emergency communication.

However, the existing systems generally focus on particular aspects such as communication range, mesh routing, GPS tracking, or user interaction. Some recent systems have combined several of these functions, including LoRa, GPS, multi-hop communication, and monitoring [7]. However, there is still scope for a lightweight system that connects the complete emergency information flow, from generating a structured SOS message to processing and displaying the received information for rescue personnel.

To address this requirement, this paper presents an offline emergency communication system based on ESP32 and 433 MHz LoRa nodes. The system allows an SOS to be generated either through a physical emergency button or through a local Wi-Fi web interface hosted by the transmitter node.

The transmitter combines the emergency details with available GPS information and sends them as a structured SOS packet through the LoRa network. The receiving node acknowledges the packet and transfers the received information to a local computer.

On the computer side, a Python backend processes the received data, calculates emergency priority, stores the information in an SQLite database, and provides it to a local rescue dashboard. The system also supports relay-based multi-hop communication to extend the communication path beyond a direct transmitter-receiver connection.

The main objective of this work is to integrate emergency information generation, LoRa communication, GPS localization, acknowledgement, local data processing, priority assessment, storage, and rescue-side visualization into a single infrastructure-independent workflow. The proposed system is intended for emergency situations where conventional communication networks are unavailable or unreliable.

II. LITERATURE REVIEW AND RESEARCH GAP

A. Existing Emergency Communication Systems

Several studies have explored LoRa for emergency communication without relying on conventional cellular infrastructure. LoRa mesh and multi-hop approaches have been investigated for forwarding emergency messages between nodes [1], [2], [7]. Other works have combined LoRa with GPS to provide location information during emergencies [4], [5]. Offline user interfaces, monitoring systems, and disaster-response applications have also been integrated with LoRa-based communication [3], [6], [8]–[10].

These studies demonstrate the suitability of LoRa, GPS, multi-hop communication, and local monitoring for emergency applications. However, the reviewed systems generally emphasize communication, localization, routing, or specific monitoring applications rather than a single lightweight end-to-end emergency information workflow.

B. Research Gap

The reviewed literature shows that the individual technologies required for an offline emergency communication system have already been demonstrated. However, there is scope for integrating SOS generation, GPS-assisted structured messaging, LoRa communication, acknowledgement, relay-based forwarding, local data processing, emergency prioritisation, storage, and rescue-side visualization into one infrastructure-independent workflow.

C. Proposed Contribution

This work addresses this gap by developing an ESP32 and 433 MHz LoRa-based offline emergency communication system. The system supports both physical-button and local Wi-Fi-based SOS generation, GPS-assisted emergency packets, acknowledgement and multi-hop forwarding. The received information is processed locally using Python, prioritized using rule-based processing, stored in SQLite, and presented through a local rescue dashboard.

III. METHODOLOGY

A. System Architecture

The proposed system comprises two primary ESP32-based nodes: TX001 and RX001. It is also supported by optional LoRa relay nodes and a local computer-side software stack. TX001 serves as the emergency transmitter, offering both physical and web-based SOS generation options. RX001 acts as the primary receiving node and connects to the local computer via USB serial communication. When necessary, intermediate LoRa relay nodes can relay unseen emergency packets to extend the communication range.

The overall communication flow is as follows:

TX001 → 433 MHz LoRa → RX001 → USB Serial → Python Backend → SQLite Database → Rescue Dashboard.

For multi-hop communication, relay nodes forward eligible packets before they reach RX001, allowing the system to operate without relying on cellular networks or Internet connectivity (Fig. 1).

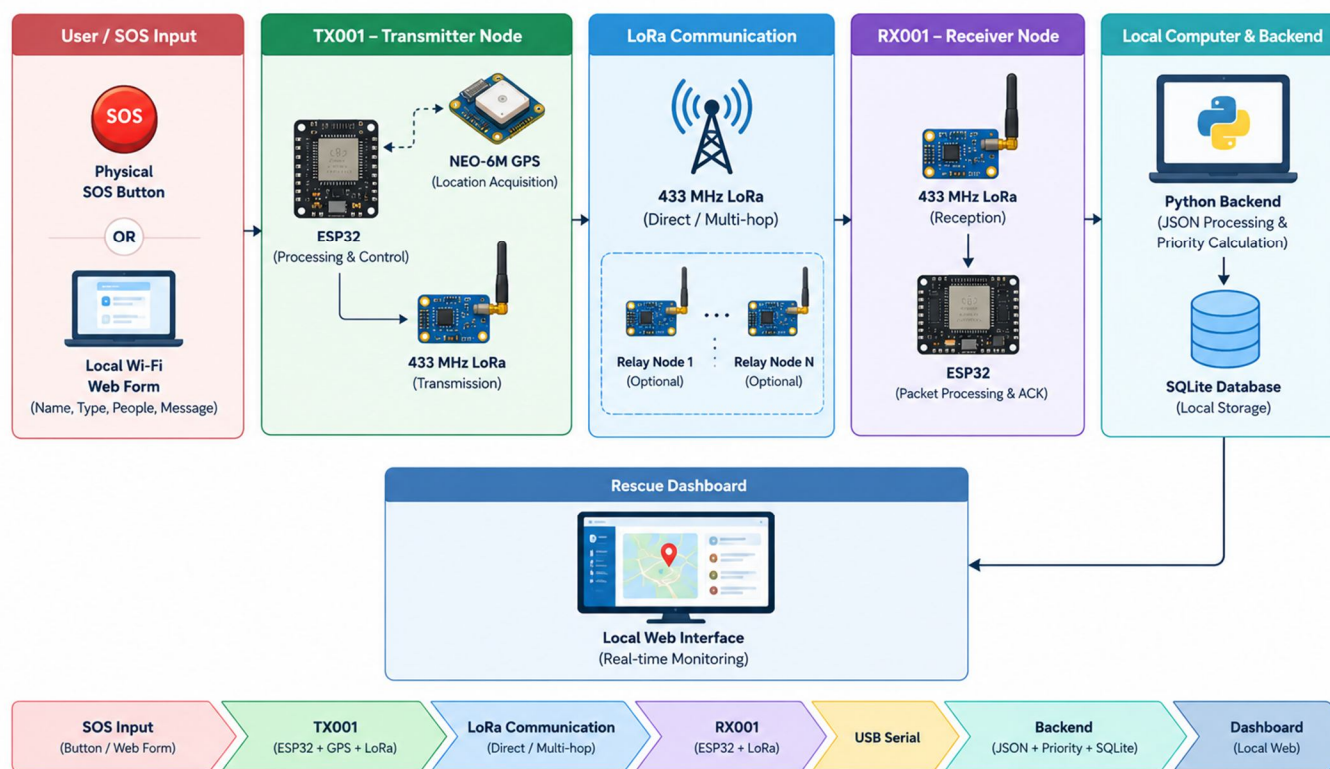


Fig. 1 Overall System Architecture of the Proposed Emergency Communication System

B. SOS Generation

An emergency alert can be generated at TX001 using either a physical SOS button or a local Wi-Fi web interface hosted by the ESP32. The web interface allows users to enter details such as their name, type of emergency, number of individuals involved, and an emergency message. This option provides an alternative to the physical button when additional information is necessary. After receiving the input, TX001 prepares the emergency information for wireless transmission. It includes a unique packet identifier and a transmitter node identifier to facilitate packet tracking and handling of duplicates (Fig. 2).

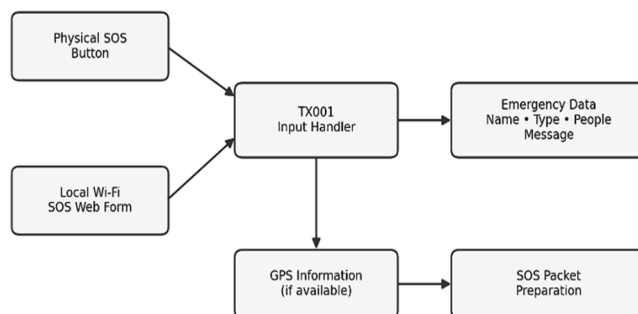


Fig. 2 SOS generation and emergency information preparation workflow

C. GPS Localization and Packet Formation

TX001 is connected to a NEO-6M GPS module to obtain the location of the emergency. When a valid GPS fix is available, latitude, longitude, and satellite information are included in the SOS data.

The transmitter forms a structured SOS packet containing the packet type, packet identifier, node identifier, emergency details, GPS information, and hop-control information. This packet is then transmitted using the 433 MHz LoRa module (Fig. 3).

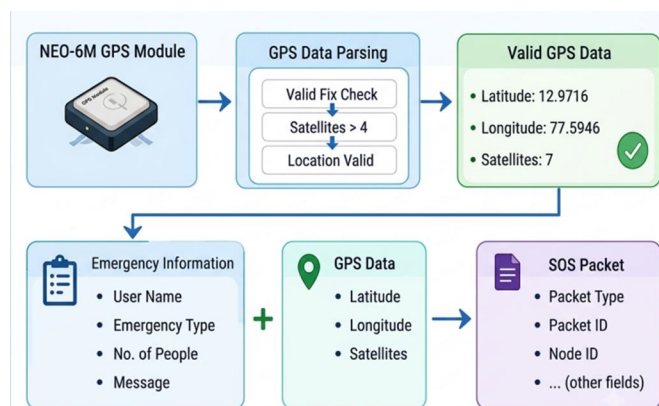


Fig. 3 GPS localization and structured SOS packet formation

D. LoRa Communication and Acknowledgement

The system utilizes 433 MHz LoRa communication between the ESP32 nodes. The implemented LoRa configuration uses a spreading factor of 12, a bandwidth of 125 kHz, a coding rate of 4/5, an 8-symbol preamble, and a transmission power of 17 dBm. The system supports both direct and relay-based communication. In direct communication, the SOS packet is transmitted from TX001 to RX001. In multi-hop operation, relay nodes forward previously unseen packets according to the configured hop-control mechanism. Duplicate packet detection prevents the repeated forwarding of the same emergency message. Upon receiving an SOS packet, RX001 processes it and sends an acknowledgment packet to indicate successful reception. The receiver also extracts communication information such as Received Signal Strength Indicator (RSSI) and Signal-to-Noise Ratio (SNR) to monitor the received LoRa signal (Fig. 4).

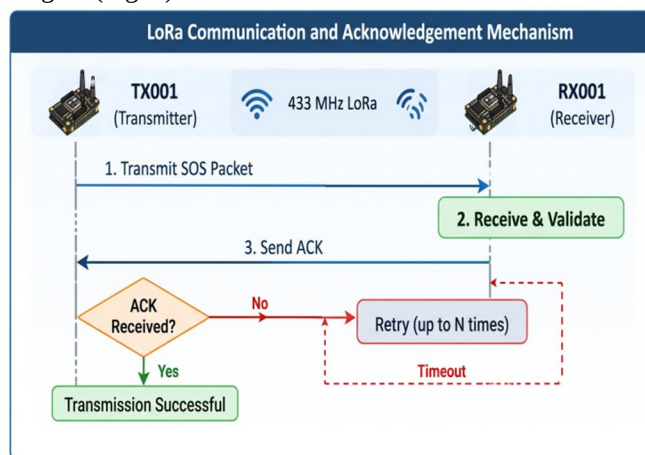


Fig. 4 LoRa communication and acknowledgement sequence

E. Backend Data Processing

RX001 transfers the received emergency information to the local computer via USB serial communication. A Python backend receives and processes the incoming data, converting it into a structured JSON representation. The backend calculates the priority of the emergency based on the received information and stores the processed records in an SQLite database. This approach provides persistent local storage without the need for a cloud database or Internet connectivity.

F. Rescue Dashboard

The processed emergency records are made available through a local HTTP interface to the rescue dashboard. This dashboard presents the received emergency information in a format suitable for monitoring, including details such as emergency type, message, number of affected individuals, location information, location pointing on offline map, and calculated priority. Overall, this methodology integrates emergency generation, GPS-assisted packet formation, LoRa communication, acknowledgment and relay handling, local data processing, database storage, and rescue-side visualization into a cohesive offline workflow.

IV.RESULTS AND DISCUSSION

The developed system was evaluated as an integrated offline emergency communication framework, with emphasis on the complete flow of emergency information from the user interface to the rescue-side monitoring application. The evaluation considered SOS generation, GPS-assisted packet formation, LoRa communication, acknowledgement, relay-based forwarding, local data processing, database storage, and dashboard visualization.

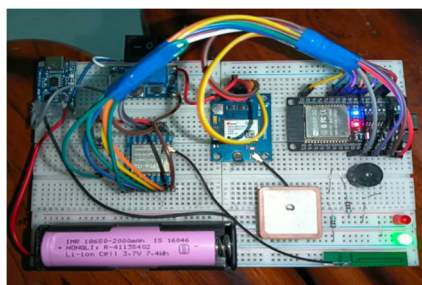
A. End-to-End System Implementation

The complete prototype integrates the TX001 transmitter, RX001 receiver, optional relay nodes, and the local computer-side software stack. TX001 provides the emergency input mechanisms and prepares the SOS information for transmission. RX001 receives the wireless packet and transfers the received information to the computer through USB serial communication. The computer-side application subsequently processes, stores, and displays the emergency information.

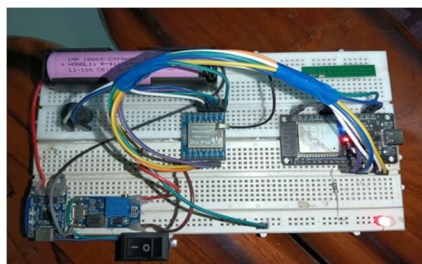
The implemented end-to-end flow is:

SOS Input → TX001 → 433 MHz LoRa → RX001 → USB Serial → Python Backend → SQLite Database → Rescue Dashboard

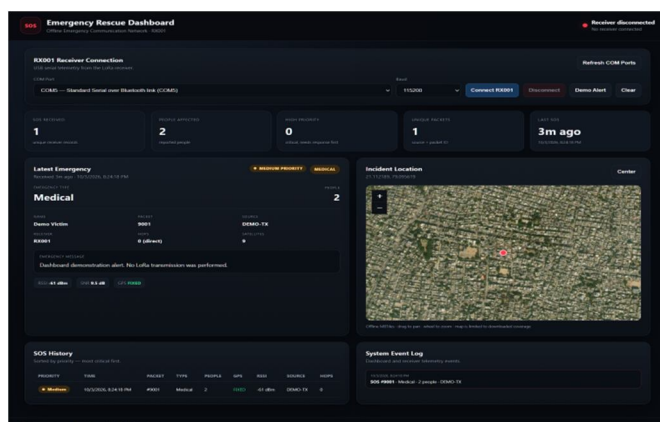
This integration demonstrates that the system is not limited to wireless message transmission. The received emergency information continues through a local processing pipeline until it becomes a structured alert available to the rescue operator.



(a) Transmitter Node



(b) Receiver Node



(c) Rescue Dashboard

Fig. 5 Real implementation of the developed emergency communication system: (a) TX001 transmitter, (b) RX001 receiver, and (c) rescue dashboard

The developed prototype was physically implemented using the TX001 transmitter, RX001 receiver, and the local rescue monitoring software. Fig. 5 presents the real implementation of the system, including the transmitter and receiver hardware and the rescue dashboard. These components demonstrate the integration of the embedded communication layer with the computer-side emergency information processing and monitoring layer.

B. Emergency Information Generation and Transmission

The system provides two mechanisms for generating an SOS alert. The physical emergency button provides a direct method for initiating an alert, while the local Wi-Fi web interface hosted by TX001 allows the user to enter additional emergency information. The web interface accepts the user's name, emergency type, number of people, and emergency message.

The web-based input provides greater flexibility than a predefined emergency message because the user can supply information specific to the current situation. After the input is received, the transmitter generates a structured SOS packet containing the emergency information together with packet identification and communication-control fields.

This approach allows the same communication infrastructure to support both rapid emergency activation and detailed emergency reporting. It also removes the requirement for a mobile application or Internet-based service at the point where the SOS is generated.

C. GPS-Assisted Emergency Information

The NEO-6M GPS module integrated with TX001 provides location information for the emergency alert when a valid GPS fix is available. The obtained latitude and longitude are included directly within the SOS packet along with the available satellite information.

The integration of location information with the emergency message is important because the receiving side does not need to obtain the user's position through a separate communication mechanism. Instead, the emergency information and its associated location are transported as part of the same structured packet.

The receiver architecture does not require a separate GPS module because location acquisition is performed at the transmitter. This reduces the hardware requirements of the receiving node while allowing location information to remain available to the rescue-side application when a valid GPS fix has been obtained.

D. LoRa Communication and Acknowledgement

The developed system uses 433 MHz LoRa communication for transferring emergency packets between the embedded nodes. The implemented configuration uses a spreading factor of 12, 125 kHz bandwidth, coding rate of 4/5, an 8-symbol preamble, and a transmission power of 17 dBm.

The receiver identifies the incoming packet and processes its emergency information. An acknowledgement packet is generated by RX001 after reception of an SOS packet. The use of packet identification and duplicate detection helps the receiver distinguish individual emergency messages and avoid repeatedly processing the same packet.

The receiver also provides RSSI and SNR information for received LoRa packets. These parameters provide useful information for subsequent evaluation of the wireless link under different environmental and deployment conditions.

The acknowledgement mechanism adds a feedback stage to the communication process. Therefore, the transmitter does not have to treat the initial transmission as the final state of the communication process. This is particularly relevant in emergency applications where confirmation of reception is useful for determining whether an alert has reached the receiving node.

E. Multi-Hop Communication and Packet Handling

In addition to direct communication between TX001 and RX001, the implemented firmware supports relay-based multi-hop forwarding. Intermediate LoRa nodes can forward eligible emergency packets when a direct communication path is not available.

Packet identification and duplicate detection are used during forwarding to reduce repeated processing of the same emergency packet. Hop-control information is also incorporated into the packet structure to regulate relay behavior.

The multi-hop capability extends the system architecture from a single transmitter-receiver link to a network that can contain intermediate nodes. However, the existence of the implemented relay mechanism should not be interpreted as a measured improvement in communication range or reliability. Quantitative multi-hop performance requires controlled field experiments with defined node spacing, packet counts, environmental conditions, and repeated trials.

F. Local Backend Processing and Emergency Prioritisation

After reception by RX001, the emergency information is transferred to the local computer through USB serial communication. The Python backend processes the received information and converts it into a structured JSON representation.

The backend then applies the implemented rule-based emergency priority calculation. This allows received alerts to be categorized according to the emergency information rather than presenting all received messages as identical events.

The processed emergency records are stored in an SQLite database. Local database storage provides persistent access to emergency information without requiring a cloud database or Internet connection. This is consistent with the overall infrastructure-independent objective of the system.

The processing pipeline can therefore be represented as:

RX001 → USB Serial → JSON Processing → Priority Calculation → SQLite

This stage is an important part of the proposed architecture because the system continues to process emergency information after the wireless communication stage instead of terminating at packet reception.

G. Rescue Dashboard and Operational Output

The processed information is made available to a local rescue dashboard through a local HTTP interface. The dashboard provides a centralized representation of the received emergency alerts and presents information such as emergency type, message, number of affected people, location information, and calculated priority.

The dashboard converts the structured data generated by the backend into an operator-oriented view. This reduces the need for the rescue operator to interpret raw serial data or LoRa packets directly.

The combination of SQLite storage and dashboard visualization also provides two complementary functions: the database maintains the emergency records, while the dashboard provides an immediate interface for monitoring them.

The dashboard therefore represents the final stage of the implemented emergency information workflow:

Wireless Alert → Data Processing → Priority Assessment → Local Storage → Rescue Visualization

H. Functional Verification

The major functional outcomes of the implemented system are summarized in Table I.

TABLE I
FUNCTIONAL VERIFICATION OF THE DEVELOPED SYSTEM

Evaluation Area	Implemented Outcome
SOS generation	Physical button and local Wi-Fi web interface
Emergency information	Name, emergency type, people count, and message
GPS localization	Latitude and longitude included when a valid GPS fix is available
Packet formation	Structured SOS packet with identification and communication-control information
LoRa communication	433 MHz wireless SOS transmission
Acknowledgement	Receiver-generated acknowledgement packet
Multi-hop	Relay-based forwarding of eligible packets
Duplicate handling	Packet identification and duplicate detection
Backend processing	Python-based processing and JSON representation
Priority assessment	Rule-based emergency priority calculation
Data storage	Local SQLite database
Rescue monitoring	Local HTTP-based dashboard
Internet dependency	Core emergency workflow operates without cellular or Internet connectivity

The evaluation demonstrates that the individual modules operate as interconnected components of a common emergency information workflow. In particular, the system combines user interaction, location information, wireless communication, acknowledgement, local processing, persistent storage, and visualization rather than treating them as independent functions.

V. CONCLUSION

This paper presented an offline emergency communication system designed to provide structured emergency information without depending on cellular networks or Internet connectivity. The developed system combines ESP32-based transmitter and receiver nodes with 433 MHz LoRa communication, GPS localization, acknowledgement handling, and relay-based multi-hop communication.

The transmitter provides two methods for generating an SOS: a physical emergency button and a local Wi-Fi web interface. Emergency information is combined with available GPS coordinates and transmitted as a structured SOS packet. At the receiving side, the emergency packet is acknowledged and transferred to a local computer, where a Python backend performs data processing and emergency priority calculation. The processed information is stored in an SQLite database and presented through a local rescue dashboard.

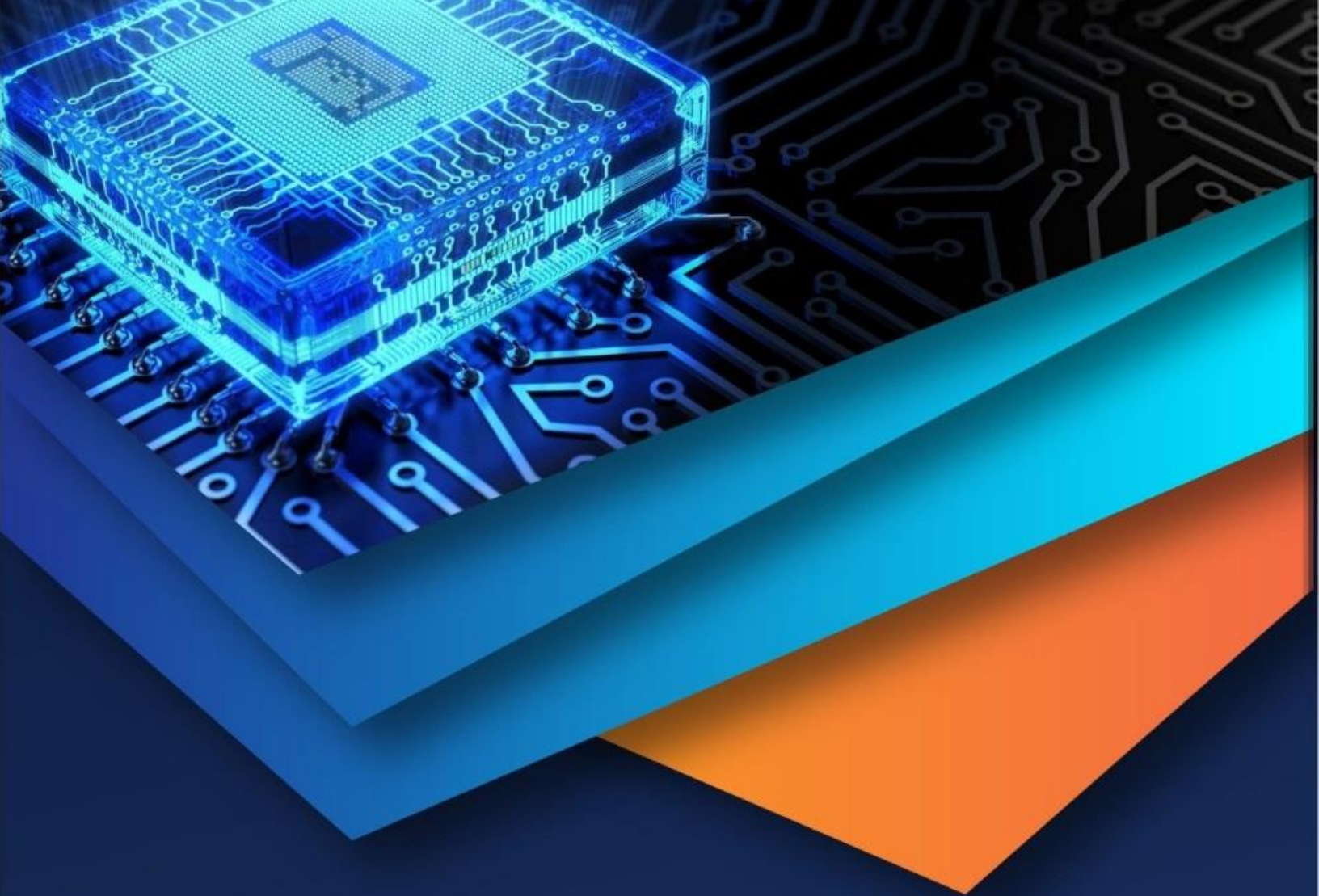
The main contribution of the developed system is the integration of the complete emergency information flow, from user-side SOS generation and wireless transmission to local processing, storage, and rescue-side visualization. This makes the system suitable as an infrastructure-independent communication framework for situations in which conventional communication networks are unavailable or unreliable.

A. Future Scope

Future work will focus on evaluating the system's communication performance and improving the reliability of multi-hop data transmission.

REFERENCES

- [1] U. G. Sharma, R. A. Avhad, B. B. Takale, and R. R. Katkar, "A LoRa-Based Mesh Network Architecture for Disaster-Resilient Emergency Communication and GPS Localization," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 8, no. 6, Jun. 2026, doi: 10.56726/IRJMETS101091.
- [2] T. R. Lakde, H. Shelar, C. Sahu, and S. V. Shelke, "Comparative Study of LoRa Mesh-Based Emergency Communication Systems," *International Journal of Engineering Development and Research*, vol. 14, no. 2, pp. 193–196, Jun. 2026.
- [3] A. Rokade, A. Ritund, K. Gosavi, and S. Bhosale, "Design and Implementation of a Decentralized LoRa Mesh Network for Off-Grid Emergency Communication," *International Journal for Research in Applied Science and Engineering Technology*, vol. 14, no. 5, pp. 3173–3176, May 2026, doi: 10.22214/ijraset.2026.82380.
- [4] K. R. Rekha, C. Narayan, H. Keshav, P. H. S., and D. S. N., "Solar-Powered LoRa Mesh Network for Emergency Communication and Tracking During Disasters," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 14, no. 12, Dec. 2025, doi: 10.17148/IJARCC.2025.141266.
- [5] U. Kumaran, B. Umah, P. Praneeth Reddy, P. Sai Shruthi, and P. Tharneesh, "Development of a Robust Emergency Communication System Utilizing LoRa and GPS with Multi-Hop Routing," in *Proc. 2nd Int. Conf. Self-Sustainable Artificial Intelligence Systems (ICSSAS)*, 2024, doi: 10.1109/ICSSAS64001.2024.10760890.
- [6] M. P. Manuel, M. Faied, and M. Krishnan, "A LoRa-Based Disaster Management System for Search and Rescue Mission," *IEEE Internet of Things Journal*, vol. 11, no. 20, pp. 34024–34034, Oct. 2024, doi: 10.1109/IJOT.2024.3435734.
- [7] B. Poke and S. Uludag, "Empirical Evaluation of a LoRa Mesh Network for Emergency Communication Systems," in *Proc. IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2023, doi: 10.1109/UEMCON59035.2023.10316084.
- [8] R. Pueyo Centelles, F. Freitag, R. Meseguer, L. Navarro, S. F. Ochoa, and R. M. Santos, "A LoRa-Based Communication System for Coordinated Response in an Earthquake Aftermath," *Proceedings*, vol. 31, no. 1, Art. no. 73, 2019, doi: 10.3390/proceedings2019031073.
- [9] B. Sangeeth Kumar, S. Dhanush Singh, N. Jha, Y. Madhumathi, and N. Phaneendra, "LoRa-Based Offline Emergency Communication System," *International Journal of Scientific Research in Engineering, Applied Sciences and Technology*, vol. 6, no. 4, pp. 162–166, Jul.–Aug. 2026, doi: 10.59256/ijreat.20260604019.
- [10] M. N. Luqman, M. Fikry, and Y. Afrillia, "Integrated Emergency Communication System for Disaster Areas Using Long Range," *Journal of Applied Informatics and Computing*, vol. 10, no. 3, pp. 2228–2241, Jun. 2026, doi: 10.30871/jaic.v10i3.12753.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)