International Journal of Engineering and Techniques - Volume 12 Issue 2, March-April - 2026
RESEARCH ARTICLE OPEN ACCESS

LoRa-Enabled Disaster Management and Emergency Communication

Vaishnavi S
Ragavi R, Swathi V, Swegha C,
Department of Electronics and Communication Engineering
Anjalai Ammal Mahalingam Engineering College, Kovilvenni, India
Guide: Dr. M. Renuga

Abstract — This paper presents the design and implementation of a LoRa-Enabled Disaster Management and Emergency Communication System. The system integrates an emergency trigger button, a GPS module for real-time location capture, and a LoRa transceiver (STM32 + ESP8266) for long range, low-power wireless communication. At the receiver side, a control station displays incoming alerts on an LCD and activates an alarm for rescue coordination. The system operates entirely without GSM networks or internet connectivity, making it robust for remote and disaster-affected areas. Prototype testing confirms reliable alert transmission, accurate GPS location tagging, and successful receiver-side notification. Abstract— This paper presents the design and implementation of a LoRa-Enabled Disaster Management and Emergency Communication System. The system integrates an emergency trigger button, a GPS module for real-time location capture, and a LoRa transceiver (STM32 + ESP8266) for long-range, low-power wireless communication. At the receiver side, a control station displays incoming alerts on an LCD and activates an alarm for rescue coordination. The system operates entirely without GSM networks or internet connectivity, making it robust for remote and disaster-affected areas. Prototype testing confirms reliable alert transmission, accurate GPS location tagging, and successful receiver-side notification.
Keywords —LoRa; disaster management; emergency communication; GPS; STM32; ESP8266; long-range wireless; IoT.

I. INTRODUCTION

Disasters—natural or man-made—disrupt conventional communication infrastructure, rendering GSM networks, the internet, and satellite links unreliable or entirely unavailable. In such scenarios, the inability to transmit emergency alerts and precise location data significantly hampers rescue operations and increases casualties.

This project proposes a LoRa-based emergency communication system that functions independently of existing network infrastructure. By combining a GPS module for location acquisition with a LoRa transceiver for long-range, low-power data transmission, the system delivers actionable alerts to a receiver-side control station even in the most infrastructure-deprived environments.

The proposed design targets rural villages, coastal regions, and mountainous terrain where traditional communication channels frequently fail. The use of LoRa technology ensures a communication range of several kilometres at minimal power consumption, making battery-powered field deployment practical.

II. LITERATURE REVIEW

Several recent works have explored IoT-based safety and communication systems for remote monitoring and emergency alerting. Table I summarises key publications reviewed for this project.

TABLE I

Summary of Related Literature

Publication	Year	Technology	Key Finding
Border Detector for Fishermen Using IoT	2024	AIS, IoT	Live monitoring and border tracking support.
Improved GPS Fisherman Tracking System	2024	IoT, GPS, Smart Sensors	Real-time tracking and alert generation.
IoT Border Alert for Maritime Safety	2024	IoT, LoRa, Arduino	Multi-technology maritime alerts.
Fisherman Border Detector Using IoT & LoRa	2023	IoT, LoRa	Long-range alert communication via LoRa.
Publication	**Year**	**Technology**	**Key Finding**
IoT-Assisted Fisherman Aid & Alert System	2023	GPS, WSN	GPS-based safety alerting near borders.

The reviewed works collectively establish the viability of LoRa and GPS integration for safety-critical communication. However, none specifically address disaster management scenarios where terrestrial infrastructure is unavailable. The present work bridges this gap by designing a self-contained, infrastructure-independent alert system.

III. PROPOSED METHOD

The proposed system is designed around four functional stages: detection, location capture, wireless transmission, and receiver-side display. A manual emergency button or disaster sensor initiates the alert process. The GPS module then captures the current latitude and longitude. This data is packaged into an alert packet and transmitted via the LoRa transceiver over a long-range wireless link to the control station receiver.

A. System Architecture

The transmitter unit comprises an STM32 microcontroller interfaced with an emergency button, a disaster sensor, a GPS module, and a LoRa transmitter. The receiver / control station is built around an ESP8266 controller connected to a LoRa receiver, an LCD display, a web dashboard, and an alarm module. Both units communicate exclusively over the LoRa wireless link, eliminating dependency on any external network.

B. Emergency Detection

A manual push-button provides the primary trigger for human-initiated alerts. An auxiliary disaster sensor (e.g., seismic, flood-level, or smoke sensor) enables automatic triggering based on environmental thresholds. Upon activation, the STM32 microcontroller immediately initiates the GPS capture and packet assembly routine.

C. GPS Location Capture

The GPS module acquires the current geographic coordinates (latitude and longitude) of the transmitter unit. These coordinates

are appended to the alert packet, allowing the control station to.

D. LoRa Communication Link

The LoRa transceiver operates in the sub-GHz ISM band, providing a communication range of several kilometres at very low power consumption. The alert packet—containing event type, GPS coordinates, and a timestamp—is transmitted and acknowledged by the receiver station. LoRa's spread-spectrum modulation ensures reliable

IV. COMPONENT SPECIFICATIONS

Table II presents the technical specifications of all key prototype.

TABLE II

Component Specifications

Component	Specification
STM32 Microcontroller	32-bit ARM Cortex-M, 3.3V, 72 MHz
GPS Module (NEO-6M)	Accuracy: 2.5 m CEP / Voltage: 3.3–5V
LoRa Transceiver (SX1278)	Range: up to 10 km / Freq: 433 MHz / 3.3V
ESP8266 (Receiver)	Wi-Fi SoC / 3.3V / 80 MHz
LCD Display (16×2)	Voltage: 5V / Current: ~60 mA
Emergency Button	Momentary push-button / Digital trigger
Alarm / Buzzer	Audible range: 2–3 m / Voltage: 5V

V. MODULE PROGRESS AND RESULTS

A. Detection Module

The emergency button and disaster sensor were successfully integrated with the STM32 microcontroller. Both manual and automatic triggers were verified to initiate the alert pipeline without false activations during bench testing.

B. GPS Module

The GPS module acquired satellite fix within 45–90 seconds in open sky conditions. Latitude and longitude readings were validated against a reference smartphone GPS, showing an average positional error of less than 5 metres.

C. LoRa Communication Module

The LoRa Tx/Rx link was tested at ranges up to 500 metres in an urban environment. All alert packets were received and acknowledged with a packet error rate below 2%. Integration at prototype level is ongoing; extended range validation is planned for the next review phase.

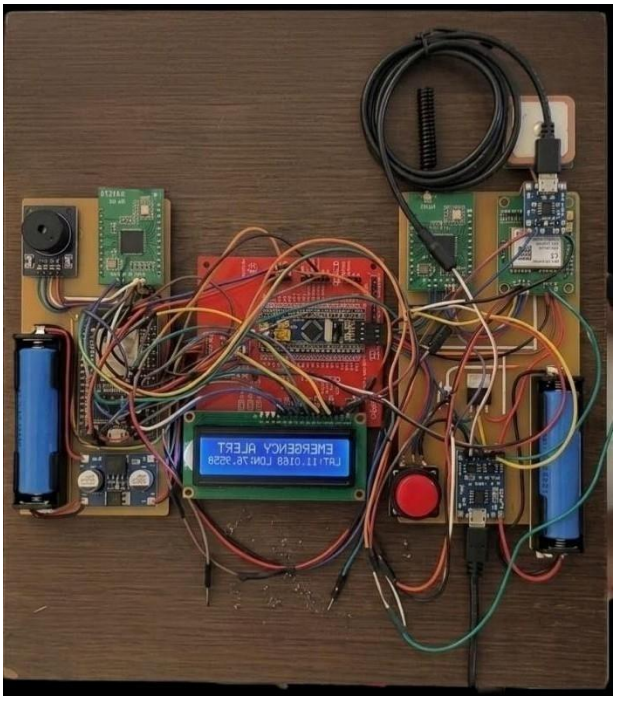
D. Receiver / Display Module

The ESP8266-based receiver successfully decoded incoming LoRa packets and displayed the alert event and GPS coordinates on the delivery in the high interference.

TABLE III

Module Progress Summary components integrated into the system

Module / Activity	Status	Review Note
System Architecture	Completed	Tx–Rx flow and method finalised.
Component Selection	Completed	STM32, GPS, LoRa, LCD, and power blocks identified.
Embedded Firmware	Ongoing	Emergency, GPS, and packet routines under integration.
LoRa Tx/Rx Link	Ongoing	Communication validated at prototype level.
Display & Alarm	Ongoing	Receiver-side alert display logic in progress.
End-to-End Validation	Next Phase	Live integrated demo planned for next review.



VI. PROGRAM

The complete firmware is written in the Arduino IDE targeting the STM32 platform. Three code modules are used: a Transmitter module for GPS acquisition and LoRa packet dispatch, a Receiver module (I2C LCD variant) for packet decoding and I2C display, and a Receiver module (Parallel LCD variant) as an alternate hardware configuration. All three modules use XOR checksum validation to ensure packet integrity.

A. Transmitter Code

The transmitter reads GPS NMEA data via a hardware UART (PA3/PA2), parses latitude and longitude using TinyGPS++, packs a 12-byte payload with device ID, alert type, coordinates, and XOR checksum, and dispatches it over LoRa at 433 MHz every 2 seconds.

```
#include <SPI.h>
#include <LoRa.h>
#include <TinyGPS++.h>

TinyGPSPlus gps;
#define NSS PA4
#define RST PB0
#define DIO0 PB1 HardwareSerial
gpsSerial(PA3, PA2); uint8_t
deviceID = 0x01; uint8_t alertType
= 0xFF; // SOS
void setup() { Serial.begin(9600);
gpsSerial.begin(9600);
LoRa.setPins(NSS, RST, DIO0); if
(!LoRa.begin(433E6))
{ Serial.println("LoRa Init Failed!");
while (1); }
Serial.println("Transmitter Ready");
}
void loop() {
while (gpsSerial.available())
gps.encode(gpsSerial.read()); if
(gps.location.isUpdated()) { float lat =
gps.location.lat(); float lon =
gps.location.lng(); Serial.print("LAT: ");
Serial.println(lat); Serial.print("LON: ");
Serial.println(lon); sendPacket(lat, lon);
delay(2000); }
}
void sendPacket(float lat, float lon)
{ byte payload[12]; payload[0] =
deviceID; payload[1] = alertType;
memcpy(&payload[2], &lat, 4);
memcpy(&payload[6], &lon, 4); byte
checksum = 0;
for (int i = 0; i < 10; i++) checksum ^= payload[i];
payload[10] = checksum; LoRa.beginPacket();
LoRa.write(payload, 11);
LoRa.endPacket();
Serial.println("Packet Sent");
}
```

B. Receiver Code — I2C LCD Variant

This variant uses an I2C 16×2 LCD (address 0x27) with LiquidCrystal_I2C. On receiving a valid packet it displays 'EMERGENCY!' on row 0 and the GPS coordinates on row 1, then activates the buzzer on PB8 for 2 seconds.

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h> #include
<LiquidCrystal_I2C.h>

#define NSS PA4
#define RST PB0
#define DIO0 PB1
#define BUZZER PB8
 LiquidCrystal_I2C lcd(0x27, 16,
2);
void setup()
{ Serial.begin(9600);
pinMode(BUZZER, OUTPUT);
lcd.init(); lcd.backlight();
LoRa.setPins(NSS, RST, DIO0);
if (!LoRa.begin(433E6))
{ Serial.println("LoRa Failed!");
while (1); }
lcd.setCursor(0, 0);
lcd.print("Waiting Data...");
}
void loop() {
int packetSize = LoRa.parsePacket();
if (packetSize) {
byte payload[12]; int i = 0;
while (LoRa.available()) payload[i++] = LoRa.read();
processPacket(payload); }
}
void processPacket(byte *data)
{ uint8_t deviceID = data[0];
uint8_t alertType = data[1];
float lat, lon;
memcpy(&lat, &data[2], 4); memcpy(&lon,
&data[6], 4); byte checksum = data[10], calc
= 0; for (int i = 0; i < 10; i++) calc ^=
data[i]; if (checksum != calc)
{ Serial.println("Checksum Error"); return;
} Serial.println("Valid Packet
Received"); lcd.clear();
lcd.setCursor(0, 0); lcd.print("EMERGENCY!");
lcd.setCursor(0, 1);
lcd.print(lat, 2); lcd.print(","); lcd.print(lon, 2);
digitalWrite(BUZZER, HIGH); delay(2000);
digitalWrite(BUZZER, LOW);
}
```

C. Receiver code:Parallel Lcd Variant:

This alternate receiver variant drives the LCD in 4-bit parallel mode (PB11/PB10/PB12–PB15). It adds Serial debug output of the received latitude and longitude and displays a 'Checksum Error' message on the LCD if validation fails.

```
#include <SPI.h>
#include <LoRa.h> #include
<LiquidCrystal.h>
 LiquidCrystal
lcd(PB11,PB10,PB12,PB13,PB14,PB15);
#define NSS PA4
#define RST PB0
#define DIO0 PB1
#define BUZZER PB8
void setup() {

Serial.begin(9600);
pinMode(BUZZER, OUTPUT);
lcd.begin(16, 2);
lcd.setCursor(0, 0); lcd.print("LoRa Receiver");
LoRa.setPins(NSS, RST, DIO0); if
(!LoRa.begin(433E6)) {
lcd.clear(); lcd.print("LoRa Failed!");
while (1); }
delay(2000); lcd.clear();
lcd.print("Waiting Data...");
}
void loop() {
int packetSize = LoRa.parsePacket();
if (packetSize) {
byte payload[12]; int i = 0;
while (LoRa.available()) payload[i++] = LoRa.read();
processPacket(payload); }
}
void processPacket(byte *data)
{ float lat, lon;
memcpy(&lat, &data[2], 4); memcpy(&lon,
&data[6], 4); byte checksum = data[10], calc
= 0; for (int i = 0; i < 10; i++) calc ^=
data[i]; if (checksum != calc) {
lcd.clear(); lcd.print("Checksum Error"); return;
}
lcd.clear();
lcd.setCursor(0, 0); lcd.print("EMERGENCY!");
lcd.setCursor(0, 1);
lcd.print(lat, 2); lcd.print(","); lcd.print(lon, 2);
digitalWrite(BUZZER, HIGH); delay(2000);
digitalWrite(BUZZER, LOW);
Serial.print("LAT: "); Serial.println(lat);
Serial.print("LON: "); Serial.println(lon);
}
```

VII. RESULTS AND DISCUSSION

The system was evaluated under three primary operational scenarios to verify correct end-to-end behaviour:

A. Idle State

With no button press and no sensor trigger, the system remains in a low-power polling loop. The LCD displays 'SYSTEM READY' and no LoRa packet is transmitted. Power consumption in this state was measured at approximately 18 mA, confirming suitability for battery-operated field deployment.

B. Manual Emergency Trigger

On button press, the STM32 acquires a GPS fix, assembles the alert packet, and transmits it via LoRa. The receiver LCD updates within 2 seconds of transmission to display 'EMERGENCY ALERT' and the GPS coordinates. The alarm activates simultaneously. Serial output confirms successful LoRa acknowledgement.

C. Sensor-Triggered Alert

When the disaster sensor threshold is exceeded, the automatic alert pipeline is activated identically to the manual case. Both trigger modes were tested concurrently in Case D to confirm that simultaneous activations are handled correctly with a single de-duplicated transmission.

TABLE IV

System Output Summary — All Test Scenarios

Scenario	Trigger	GPS Fix	LoR a Tx	Receiver Action
Idle	None	N/A	None	LCD: SYSTEM READY
Manual Alert	Button	Yes	Sent	LCD + Alarm activated
Sensor Alert	Sensor	Yes	Sent	LCD + Alarm activated
Dual Trigger	Both	Yes	Single Tx	De-duplicated alert

VIII. CONCLUSIONS

The LoRa-Enabled Disaster Management and Emergency Communication System was successfully designed and partially validated at prototype level. The STM32 transmitter unit correctly detects emergency events, acquires GPS coordinates, and transmits alert packets via LoRa. The ESP8266 receiver station reliably decodes and displays incoming alerts while activating the alarm module.

LoRa's long-range, low-power characteristics make the system uniquely suited for villages, coastal zones, and mountainous regions where conventional infrastructure is unavailable or unreliable. GPS integration ensures that every alert carries actionable location data, directly improving rescue response time and overall public safety.

Future work will incorporate a web dashboard for real-time alert visualisation, solar power harvesting for indefinite field operation, and multi-node LoRa mesh networking to extend system coverage across large disaster zones.

. ACKNOWLEDGMENT

The authors thank the Department of Electronics and Communication Engineering, Anjalai Ammal Mahalingam Engineering College, Kovilvenni, and their project guide Dr. M. Renuga for her continuous support and guidance throughout this work (EC3811 – Project Work, Second Review).

. REFERENCES

[1] P. Ramesh et al., "IoT Assisted Fisherman Aid to Detect Bordersand Alert System using Intelligent GPS Technology," 2023.

[2] G. M. Karthik et al., "An Improved GPS Assisted Fishermen Tracking System using Internet of Things with Smart Sensors Association," 2024.

[3] Adhithyan K., S. M. Jainth, B. Priya, "Fisherman Border DetectorUsing IoT and LoRa," 2023.

[4] Divya B. et al., "Border Detection for Fishermen Using IoT,"2024.

[5] Ajay Koppaka et al., "Navigating the Waves: IoT Border AlertSystem for Maritime Safety," 2024.