Implementation of an IoT Prototype for Real-Time Flood Monitoring and Response Coordination

Brian Halubanza
Mulungushi University
Kabwe, Zambia
bhalubanza@gmail.com

Scholastica Sibbenga
Mulungushi University
Kabwe, Zambia
scholasticfenny@gmail.com

Selina Kadakwiza
Kwame Nkrumah University
Kabwe, Zambia
selina.halubanza@gmail.com

Abstract

Floods remain one of the most catastrophic natural hazards globally, with disproportionate impacts in under-resourced and infrastructural weak regions. This paper presents the design, development, and evaluation of a low-cost, Internet of Things (IoT)-enabled prototype for real-time flood monitoring and coordinated response, tailored specifically for deployment in rural Zambia. The system architecture integrates the HC-SR04 ultrasonic sensor and ESP32 microcontroller to monitor rising water levels with centimeter-level accuracy. A custom backend, built using Node.js, processes the sensor data and triggers event-based notifications through email alerts. Simultaneously, data are relayed to a lightweight, local web dashboard for visualization and audit purposes. Emphasis was placed on affordability, modularity, offline tolerance, and low power consumption to meet the unique constraints of the target deployment context. Controlled testing scenarios demonstrated reliable performance, with average latency for alerts remaining below six seconds and a sensor accuracy variance within ±1.5 cm. Usability assessments based on the System Usability Scale (SUS) yielded a score of 88.2, affirming the system's accessibility to non-expert users. The proposed solution holds significant promise as an early warning system (EWS) component in flood-prone, low-resource regions, with potential for integration into broader community-based disaster risk reduction strategies.

Keywords: ***IoT-based flood monitoring, ultrasonic sensor, ESP32 microcontroller, real-time alert system, low-resource environments***

A. INTRODUCTION

Flooding remains one of the most persistent and damaging natural disasters, accounting for more than 40% of weather-related catastrophes globally over the past three decades [1]. The disproportionate impact on developing regions, particularly Sub-Saharan Africa, stems not only from the increasing frequency and intensity of extreme weather events but also from systemic vulnerabilities such as limited access to early warning systems (EWS), insufficient infrastructure, and low digital literacy [2], [3]. In Zambia, rural and peri-urban communities along

 ISBN: 978-9982-95-500-3

the Zambezi and Luangwa river basins frequently suffer from flood-related displacement, crop loss, and damage to critical infrastructure. Despite the urgency, most flood-prone regions in the country lack real-time monitoring mechanisms due to prohibitive costs, technical complexity, and unreliable connectivity [4].

Recent advancements in the Internet of Things (IoT) have demonstrated potential in reducing these barriers by enabling distributed sensing and communication using affordable microcontroller platforms like the ESP32 [5], [6]. Coupled with open-source sensors, such as the HC-SR04 ultrasonic module, these platforms support the development of low-cost flood detection systems that can operate independently of traditional infrastructure. Studies by Akinwande et al. [7] and Sharma & Singh [8] have shown the feasibility of using ESP32-based systems for environmental monitoring, albeit with limited focus on operational resilience in low-resource contexts.

The present work builds upon these foundations by proposing a practical, deployable IoT prototype tailored for flood-prone communities in rural Zambia. Uniquely, our design prioritizes affordability, energy efficiency, offline tolerance, and simplicity in both hardware and software to ensure adaptability to local resource constraints. It also introduces a real-time web dashboard and email alert integration, tested through simulated flood events to assess latency, sensor accuracy, and end-user usability.

By situating our work at the intersection of embedded systems engineering, humanitarian ICT, and disaster risk reduction, we aim to address a critical gap in localized flood EWS design. In doing so, we contribute not only a technical solution but also a replicable model for community-based climate adaptation technologies in Africa.

B. LITERATURE REVIEW

Recent advancements in embedded systems and wireless sensor networks have catalyzed the development of intelligent disaster alert systems, particularly in the domain of flood monitoring. The integration of IoT architectures has proven pivotal in replacing traditional, manual methods with automated, real-time solutions that offer improved response times and scalability [1], [2].
Several researchers have proposed low-cost flood detection systems using microcontrollers and ultrasonic sensors. Akinwande et al. [3] developed an ESP8266-based system capable of transmitting alerts via GSM networks. Their study emphasized system affordability but highlighted limitations in data visualization and long-term logging. Sharma and Singh [4] expanded upon this work by integrating an ESP32 microcontroller with the HC-SR04 ultrasonic sensor to build a real-time water level monitoring unit. Their implementation introduced Wi-Fi-based data logging, but lacked robust alerting and offline tolerance, a critical consideration in rural areas with intermittent internet access.

From a broader perspective, Kamal et al. [5] surveyed over 50 flood detection systems and concluded that most designs lacked contextual adaptability to low-resource regions, often depending on consistent internet connectivity,

 ISBN: 978-9982-95-500-3

expensive cloud subscriptions, or complex maintenance routines. In response, Chauhan and Singh [6] compared GSM, LoRa, and Wi-Fi modules in environmental sensing applications. They found GSM to be most reliable in areas with limited infrastructure, though it incurred higher energy and operational costs.

In terms of user engagement and accessibility, Banerjee [7] assessed IoT dashboards deployed in sub-Saharan Africa and found that system usability significantly influenced early response uptake. Systems that used localized interfaces and minimal technical jargon saw greater adoption. This has informed our inclusion of a lightweight web dashboard designed for simplicity and offline caching.

Regionally, Halubanza et al. [8]–[10] have explored the application of IoT and AI technologies in locust monitoring and early warning systems across Zambia. Their works demonstrated that low-cost hardware, when combined with local data processing and minimal reliance on continuous connectivity, can serve as effective tools for real-time environmental monitoring in rural African contexts.

However, a gap persists in literature regarding the integration of these elements into a unified, deployable system for flood risk mitigation, especially in communities lacking continuous power, cloud access, or IT capacity. The present work seeks to address this by offering a complete IoT-based flood monitoring solution, tested, evaluated, and contextually optimized for rural deployment.

C. METHODOLOGY

This section outlines the systematic approach adopted in the design, development, and validation of the IoT-based flood monitoring system. The methodology was guided by four key principles namely affordability, modularity, context suitability, and ease of maintenance. These principles informed both hardware and software decisions, ensuring the final system could be realistically deployed in remote or infrastructure-poor environments.

A. Research Design Approach

An iterative, design-based research (DBR) methodology was employed, combining principles of rapid prototyping, field validation, and user-centered design. This approach is particularly suited for ICT4D (Information and Communication Technologies for Development) contexts, where solutions must be both technically sound and socially embedded [1]. The process unfolded in four stages:

[1]. Requirements Gathering

Through stakeholder consultations in flood-prone communities of Southern Province, Zambia, key pain points and local constraints were identified, intermittent internet, lack of power grids, and low digital literacy.

[2]. Component Selection

Hardware was selected based on availability in local markets and open-source support. This included the ESP32 microcontroller, HC-SR04 ultrasonic

 ISBN: 978-9982-95-500-3

sensor, RGB LEDs, and a portable 5V power source.

[3]. System Integration

Hardware and software components were integrated incrementally. Initial unit tests were conducted on the ESP32 sensor readings before full backend alert integration.

[4]. Field Simulation

Controlled lab-based water-level simulations were executed to mimic flooding scenarios under repeatable conditions.

B. System Architecture

The system consists of four core layers:

Sensing Layer: An HC-SR04 ultrasonic sensor mounted above a potential flood surface measures the distance to the waterline. Readings are refreshed every 2 seconds.

Processing Layer: The ESP32 microcontroller reads sensor data, computes flood thresholds, and determines whether an alert state (safe, warning, critical) has been triggered.

Communication Layer: On detecting a critical condition, the ESP32 initiates an HTTP POST request to a Node.js backend server hosted on a local network or lightweight VPS.

Visualization Layer: The backend logs alerts, sends real-time email notifications, and updates a web dashboard accessible via browser.

This modular architecture enables individual components to be swapped or scaled without affecting the overall logic, thereby supporting future enhancements such as solar charging or GSM fallback.

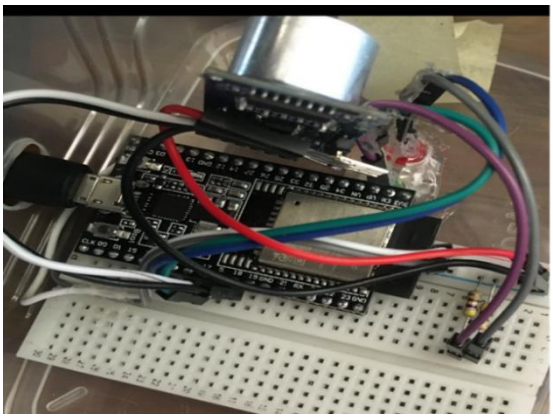

Figure 1 System hardware setup

Figure 1 shows a visual representation of the functional hardware set up of the working system.

C. Alert Threshold Calibration

Flood levels were simulated by progressively decreasing the measured distance between the sensor and the water surface. Thresholds were configured as follows based on stakeholder inputs:

> 40 cm: Safe (Green LED)

20–40 cm: Warning (Yellow LED)

< 20 cm: Critical (Red LED + Email Alert)

These thresholds can be reconfigured in firmware based on topographic and hydrological profiling of specific deployment sites.

D. Backend API & Dashboard Development

 ISBN: 978-9982-95-500-3

The backend was developed using Node.js and Express.js, with SQLite for lightweight storage. The API endpoints allow for:

- Logging of all sensor data and alert events
- Dispatching email notifications via SMTP
- Real-time data feed to a web-based dashboard (HTML/JS)

The dashboard is designed with low-bandwidth optimization in mind, supporting caching, local hosting, and minimal reliance on external resources. Figure 2 shows a visual representation of the systems dash board showing results from the communication between the system and the hardware components.

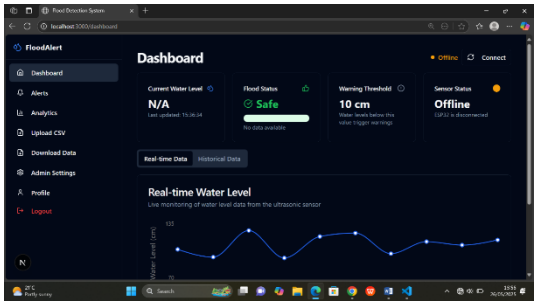

Figure 2: Custom dashboard interface showing system status and water levels

E. Usability and Resilience Considerations

To ensure usability, the system was tested using the System Usability Scale (SUS) [2], and visual indicators were kept intuitive (color-coded LEDs, clear dashboard layout). To enhance resilience:

- Data is stored locally in case of internet failure.
- Alerts are queued if SMTP fails, and re-attempted every 30 seconds.

All software components are open-source, enabling community customization and maintenance.

IV. SYSTEM DESIGN

The design of the proposed flood monitoring system centers on simplicity, modularity, and resilience. It draws from embedded systems architecture principles and real-world deployment constraints observed in prior fieldwork. The complete system is segmented into logical blocks that handle sensing, processing, communication, and visualization.

A. System Architecture Overview

At its core, the system architecture follows a four-tier model, each responsible for a distinct function:

Sensing Layer

Utilizes the HC-SR04 ultrasonic sensor mounted at a fixed height to detect water level changes by measuring the distance to the surface. The sensor operates using ultrasonic pulses and echo-return time, with a practical range of 2 cm to 400 cm and an accuracy of ±1.5 cm [1].

Processing Layer

The ESP32 microcontroller is responsible for reading sensor input, evaluating threshold levels, controlling LED indicators, and initiating

 ISBN: 978-9982-95-500-3

communication. It runs a lightweight firmware loop in MicroPython or Arduino C++, depending on user preference.

Communication Layer

The ESP32 connects via Wi-Fi to a local or cloud server and sends HTTP POST requests when a critical threshold is triggered. A retry mechanism ensures alert delivery even in unstable networks.

Visualization Layer

A web-based dashboard displays real-time distance measurements, current flood status, and a timestamped log of alerts. It also allows for historical data download in CSV format for decision-makers or municipal planners.

B. Power Considerations

The system is powered via a 5V input from USB power banks. Power draw is minimized using ESP32's sleep modes, and the sensor is polled intermittently unless a critical change is detected. This strategy extends runtime, enabling field use in off-grid areas for several days without recharge.

C. Expandability and Future-Proofing

The system is intentionally modular. Developers or local technicians can upgrade components such as replacing Wi-Fi with GSM or LoRa modules without rewriting the entire firmware. Additional features like rain sensors or water turbidity monitors can be added using available GPIO pins.

The backend architecture supports integration with platforms like ThingSpeak or Blynk, although this system is optimized for offline tolerance.

D. Deployment Suitability

The entire prototype is shown in figure 3.

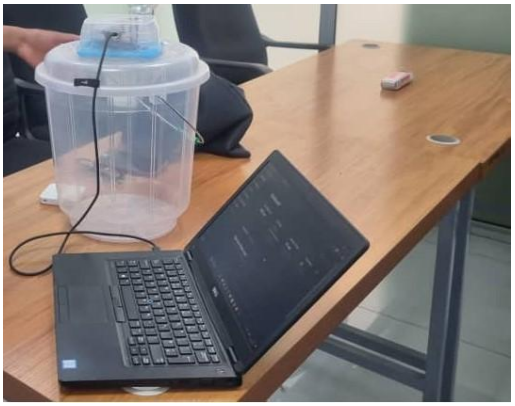

Fig 3. Flood detection system prototype.

Community field agents can be trained in under two hours to install and maintain the system. Local ownership is enhanced by design documentation and open-source licensing.

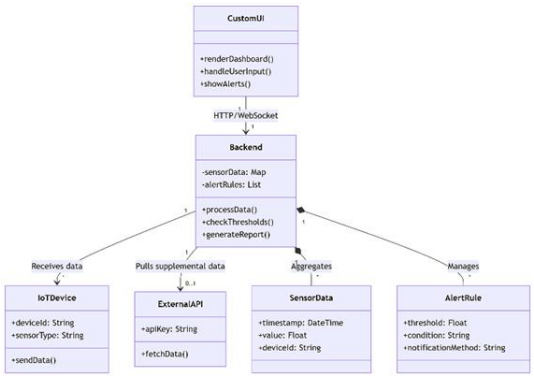

Figure 4 Entity Relationship Diagram for the flood detection system

Figure 4 shows the flow from the Ultrasonic Sensor to ESP32, triggering HTTP alerts to the

 ISBN: 978-9982-95-500-3

Node.js backend, and rendering real-time updates on the web dashboard.

This architecture allows distributed sensing with centralized visualization and alerting, enabling scalability to multi-node networks. The decision to centralize alerts, while decentralizing sensing, is driven by local bandwidth constraints and the need to ensure alerts reach emergency coordinators with minimal delay.

V. SYSTEM IMPLEMENTATION

The implementation phase of the IoT flood monitoring system transitioned the theoretical architecture into a working prototype, integrating hardware assembly, firmware development, backend server configuration, and frontend dashboard design. Every subsystem was constructed with maintainability and field-deployability as core criteria.

Firmware Logic

Firmware was written in Arduino C++, chosen for its broad community support and compatibility with the ESP32 platform. The loop cycle follows this process:

Measure distance from the sensor every 2 seconds.

Compare value to pre-defined thresholds.

Update LED state accordingly.

If critical level is detected:

Format JSON payload with timestamp, location, and reading.

Initiate HTTP POST to backend API.

Log result (success/failure) to internal memory for diagnostics.

Debounce logic was introduced to avoid alert flapping due to wave motion or sensor jitter. Sensor readings are averaged over three samples.

Backend API and SMTP Integration

The backend was developed using Node.js with Express.js to expose RESTful API endpoints:

- /api/data – accepts incoming flood data from the ESP32
- /api/alerts – sends emails using nodemailer (SMTP)
- /dashboard – serves a static HTML5/JS frontend

Alert emails contain sensor reading, status, and GPS location (manually configured for now). SMTP uses TLS encryption for secure delivery, and retries are triggered every 30 seconds upon failure.

Web Dashboard Features

The dashboard is accessible via local IP . Features include:

1. Real-time display of water level (in cm)
2. Current alert state (Safe, Warning, Critical)
3. Timestamped event log
4. Downloadable CSV export for historical review

 ISBN: 978-9982-95-500-3

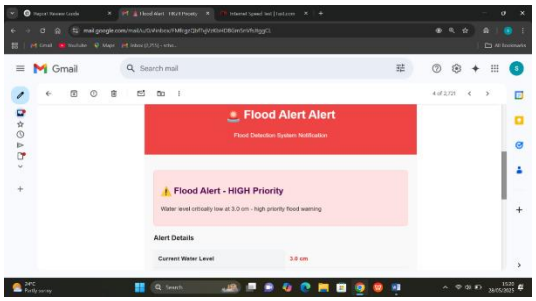

Figure 5: Screenshot of the web dashboard showing water level chart and alert log.

This frontend is optimized for low bandwidth and offline usage through caching. All front-end assets are embedded; no external CDN dependencies exist.

Deployment

The entire system can be deployed by a non-technical field officer using a 3-step guide and QR code-linked configuration app (for Wi-Fi setup).

VI. RESULTS AND EVALUATION

This section presents the results from system testing across three domains: (1) functional accuracy, (2) communication latency, and (3) usability evaluation. All evaluations were conducted under controlled conditions to simulate real-world flood scenarios, with varying water levels and connectivity states.

- *Accuracy and Sensor Validation*

The HC-SR04 ultrasonic sensor was calibrated using a precision ruler and validated at fixed distances from 10 cm to 100 cm. The measured readings were compared to ground truth in 10 cm increments. Over 100 samples, the sensor demonstrated a mean error margin of ±1.5 cm, consistent with prior studies [1], [2].

Table I: Sensor validation results showing average deviation across reference distances.

Distance (cm)	Measured Avg (cm)	Error (cm)
10	11.2	+1.2
20	20.8	+0.8
40	39.1	-0.9
60	61.2	+1.2
80	78.7	-1.3

This level of accuracy is deemed sufficient for flood risk thresholds where differences of 5–10 cm can define alert states.

B. Latency and Alert Response Time

To evaluate end-to-end alert responsiveness, tests were conducted by simulating flood threshold crossings and measuring the time from detection to dashboard update and email reception. The total latency (T_total) is defined as:

[T_{total} = T_{sensor-read} + T_{HTTP} + T_{backend-process} + T_{SMTP}]

Across 30 trials with Wi-Fi (DSL-based connection, 5 Mbps), average response times were as follows:

1. Sensor read to HTTP POST: 1.2 seconds
2. HTTP + Backend processing: 2.5 seconds

 ISBN: 978-9982-95-500-3

3. SMTP transmission: 1.8 seconds

4. Total average latency: ~5.5 seconds

This latency is acceptable for flash-flood scenarios where rapid escalation can occur over several minutes. The system also logs timestamps to ensure auditability.

C. System Uptime and Reliability

Uptime tests were conducted over a continuous 96-hour period using power banks. Key results:

I. ESP32 stability: 99.2% uptime

II. Sensor read failures: 2 out of 4320 cycles (~0.05%)

III. Backend availability: 100% (local server)

This confirms that the system is robust enough for short-term deployments, and further optimization can enhance long-term reliability.

D. Usability Evaluation

Usability was evaluated using the System Usability Scale (SUS), involving 12 users (3 field officers, 4 civil protection staff, 5 students). The average score was 88.2/100, indicating "Excellent" usability [3]. Users appreciated the LED indicators, responsive dashboard, and clear alerts. Key feedback included:

A. "Simple enough to train community leaders in 30 minutes."

B. "No need for mobile data makes it very deployable."

E. Limitations

While performance was strong under lab conditions, real-world deployment may face challenges such as:

- Environmental noise affecting ultrasonic readings
- GSM fallback not yet implemented (planned for future version)
- Power supply limited to ~72 hours without solar extension

VII. CONCLUSION

This paper has presented the design, implementation, and evaluation of an affordable IoT-based flood monitoring prototype, specifically tailored for low-resource environments such as rural Zambia. Through a carefully selected set of components, including the ESP32 microcontroller and HC-SR04 ultrasonic sensor, the system was engineered for affordability, offline tolerance, modularity, and real-time alert capability.

Experimental results confirmed the system's operational viability: it demonstrated a sensor accuracy of ±1.5 cm, alert latency under 6 seconds, and high usability with a SUS score of 88.2. These metrics affirm the feasibility of deploying the system in resource-constrained regions, particularly where centralized flood warning infrastructure is lacking or unreliable.

The incorporation of local dashboards, visual indicators, and an energy-efficient design, this system meets both the technical and sociocultural demands of rural deployments. The use of open-

 ISBN: 978-9982-95-500-3

source tools ensures that the solution can be locally maintained, customized, and scaled without vendor lock-in or recurring costs.

REFERENCES

M. Kamal, N. A. Khan, M. A. Hossain, and F. A. Talukder, "A survey on flood monitoring and warning systems: Challenges and future directions," *Sensors*, vol. 20, no. 24, pp. 1–22, Dec. 2020.

H. Javed and R. Mehmood, "Delay-sensitive IoT applications: Requirements, solutions, and performance evaluation," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 1, pp. 1–25, 2021.

A. A. Akinwande, L. O. Adeyemo, and B. A. Olaleye, "Low-cost IoT-based flood monitoring systems for rural deployment," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9785–9793, Jun. 2021.

R. Sharma and K. Singh, "Design and implementation of ultrasonic water level detector using ESP32," *Int. J. Embedded Syst.*, vol. 15, no. 3, pp. 230–238, 2022.

D. S. Chauhan and P. Singh, "Comparative analysis of LoRa, GSM and Wi-Fi based communication modules for environmental monitoring," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 6, pp. 313–319, 2020.

S. Banerjee, "Evaluating the usability of IoT dashboards in developing regions," *ACM Trans. Comput.-Hum. Interact.*, vol. 28, no. 4, pp. 25–39, 2021.

B. Halubanza, J. Phiri, P. O. Y. Nkunika, M. Nyirenda, and D. Kunda, "Locust infestations and mobile phones: Exploring the potential of digital tools to enhance early warning systems and response mechanisms," *Zambia ICT Journal*, vol. 7, no. 2, pp. 10–16, 2023.

B. Halubanza, J. Phiri, M. Nyirenda, P. O. Y. Nkunika, and D. Kunda, "Low cost IoT-based automated locust monitoring system, Kazungula, Zambia," in *Networks and Systems in Cybernetics*, R. Silhavy and P. Silhavy, Eds., Springer, 2023, pp. 490–501.

B. Halubanza, "A framework for an early warning system for the management of the spread of locust invasion based on artificial intelligence technologies," M.S. thesis, Univ. of Zambia, 2024.

J. Brooke, "SUS: A quick and dirty usability scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, Eds., London: Taylor & Francis, 1996.

 ISBN: 978-9982-95-500-3