International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Low-Cost Disaster Detection and Early Warning System

Using ESP32 and Internet of Things (IoT)

Rohit Pawar, Omkar Mali, Rahul Pawar, Mr. V. B. Takale

Rohit Pawar E&TC & JSPM'S BSP
Omkar Mali E&TC & JSPM'S BSP
Rahul Pawar E&TC & JSPM'S BSP
Mr. V. B. Takale E&TC & JSPM'S

Abstract

Natural hazards and local emergencies can become dangerous when changes in rainfall, water level or environmental conditions go unnoticed. This paper describes a proposed low-cost monitoring and early-warning system built around an ESP32 microcontroller and IoT connectivity. Sensors collect selected environmental readings, and the controller checks those readings against set limits. If a reading indicates a possible risk, an LED and buzzer can alert people nearby. When Wi-Fi is available, the readings and alert status can also be sent to a remote dashboard. The design can be adapted for water-level monitoring, rainfall indication and other basic environmental measurements by choosing suitable sensors. It is intended as an educational prototype, not a certified public warning system. Its performance would depend on sensor choice, calibration, installation, power supply and network availability.

Index Terms—Disaster monitoring, early warning system, ESP32, Internet of Things, environmental sensors, water-level monitoring, alert system.

1. INTRODUCTION

Natural disasters and local emergencies do not always give people much time to respond. Heavy rain, for example, can cause water levels to rise, while unusual environmental readings may indicate that conditions are changing. If these changes are noticed late, people have less time to take precautions.

The ESP32 offers a compact and affordable way to collect sensor readings and process them locally. It also has Wi-Fi capability, so the device can send updates to an online dashboard when a network is available. A buzzer or LED can provide a local alert without waiting for a message from the internet.

A sensor reading by itself is not enough to make a useful warning system. The reading must be checked, the warning rule must be clear, and repeated false alarms should be reduced where possible. This paper outlines a modular monitoring design and its practical limits. The proposed system is not intended to predict earthquakes or detect every type of disaster.

2. BODY OF PAPER

The design has three main parts: reading the sensors, deciding whether a reading needs attention, and communicating the result. The exact sensors should be chosen for the hazard being monitored. Their datasheets should be checked for voltage, measurement range, accuracy and calibration needs before installation.

Development can start with one sensor and a local LED or buzzer. A display, Wi-Fi dashboard and data logging can be added later. The basic alert should continue to work as far as possible if the internet connection is lost. Before the prototype is used, the selected thresholds should be checked through controlled tests.

2.1 PROPOSED SYSTEM

The ESP32 is the main controller. Depending on the final version, it may receive data from a water-level sensor, rain sensor, temperature-and-humidity sensor, or optional soil-moisture and vibration sensors. The controller checks whether readings are valid and compares them with the limits set in the software.

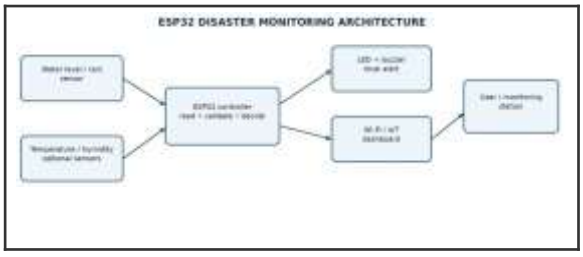


Fig. 1. Proposed ESP32-based disaster monitoring architecture.

If a valid reading passes a set limit, the controller changes the warning state and activates the selected local alert. When Wi-Fi is working, it can also send the reading and alert status to an IoT dashboard. The final report should name the sensors and online service actually used, along with the threshold values selected for the prototype.

2.2 SYSTEM ARCHITECTURE / BLOCK DIAGRAM

The proposed setup includes a regulated power supply, an ESP32 board, the selected sensors, local alert devices and, if needed, a Wi-Fi connection to an IoT service. Sensors provide the inputs; the ESP32 checks the measurements and applies the warning rules; LEDs or a buzzer show the local status; and the communication section sends updates when a network is available.

A water-level sensor can be used to observe a rising level in a tank, drain or test channel. A rain sensor gives a basic indication of wet conditions, while a temperature-and-humidity sensor records the local environment. Soil-moisture or vibration modules may be added for demonstrations, but their readings alone cannot reliably predict a landslide or earthquake.

For flood-oriented monitoring, the water-level input is the most direct indicator in this design. The sensor should be mounted at a stable reference point, away from avoidable turbulence where possible, and its reading should be compared with a known reference during setup. A single reading can be misleading because waves, splashes, debris or sensor alignment may cause short-lived changes. Sampling several readings and using a short moving average can make the displayed trend easier to interpret. The chosen sensor type and its safe operating range must be recorded in the final implementation.

The location and protection of each sensor affect how well it works. Sensors and wiring should be protected from moisture, dust and

corrosion where necessary. Since ESP32 GPIO pins use 3.3 V logic, the pins must not receive a voltage above their rated limit. A suitable driver circuit may be needed for a buzzer, relay or other load.

2.3 SOFTWARE ARCHITECTURE AND INPUT/OUTPUTS

Possible inputs include a water-level sensor, rain sensor, temperature/humidity sensor and optional soil-moisture or vibration module. A reset or test button may also be included. Outputs can include a status LED, buzzer, optional OLED/LCD and Wi-Fi data transmission. The final pin connections should be chosen only after the exact board and modules have been selected.

Each reading should be displayed with a clear meaning and unit. For example, water level could be shown in centimetres from a fixed reference, temperature in degrees and humidity as a percentage. A rain module may only provide a wet/dry indication unless it has been properly calibrated for intensity measurement. The software should check each sensor against its own documented limit instead of combining unrelated readings into an unexplained score. For a flood demonstration, a sustained rise in water level, especially alongside rain, could be used as a warning rule.

Representative hardware modules (illustrative images):



Fig. 2(a). ESP32 controller

Main processing unit: reads sensor values, checks thresholds, drives local alerts and can transmit data over Wi-Fi.



Fig. 2(b). DHT22 temperature/humidity sensor

Measures ambient temperature and relative humidity; useful for environmental observation alongside hazard-specific sensors.

The DHT22 measures temperature and relative humidity and is often used in environmental monitoring projects. It sends digital readings to the controller, so the software must follow the module's timing requirements and use a suitable library. It should be read at an appropriate interval rather than continuously at very high speed. The sensor can help record local conditions, but temperature or humidity alone does not confirm a flood, fire or other disaster. Its position, enclosure temperature and condensation can affect readings, so the result should be checked against a reference where possible.



Fig. 2(c). MQ-135 gas/air-quality sensor

Optional extension for observing certain air-quality changes; it is not a replacement for certified smoke or toxic-gas alarms.

The MQ-135 module is commonly used for learning and basic air-quality experiments because its sensing element responds to several gases. Its output can vary with warm-up time, surrounding conditions and differences between individual sensors. A raw analogue reading should not be presented as an accurate concentration in parts per million unless a suitable calibration procedure and reference gas data have been used. In this project, it should be treated as an optional indicator of change, not as proof that air is safe. It does not replace an approved gas or smoke detector.



Fig. 2(d). Flame sensor

Optional fire-related input for a demonstration; its readings require suitable placement and testing to limit false alarms.

A typical low-cost flame module responds to light within a particular infrared or optical range. Depending on the module, it may provide a digital trigger or an analogue reading. Sunlight, room lighting, reflections, distance and sensor direction can all affect its response. It should only be checked in a controlled and safe setting; an uncontrolled fire must never be used for a demonstration. This module is an optional classroom input, not a certified fire alarm.



Fig. 2(e). PIR motion sensor

Optional movement/occupancy input for monitoring an emergency area; it does not independently detect a natural disaster.

A passive infrared (PIR) sensor detects changes in infrared energy, often when a person moves across its field of view. It can be used for a simple movement or occupancy demonstration near a monitored area, but it does not identify a person and may not respond to someone who remains still. Its performance depends on placement, field of view, sensitivity and surrounding temperature. It is an additional input only; it does not directly measure rainfall or water level.

Note: The component photographs are illustrative and do not confirm that every module is installed in the final prototype. The main concept is to monitor water level and rainfall, with other environmental sensors added if they are part of the selected build. The ESP32 checks valid readings against configured limits and activates a local alert. Remote updates are possible when Wi-Fi is available. Image source: the supplied IJSREM reference paper; attribution and permission should be checked before submission.

3. WORKING OF THE SYSTEM

The warning logic should be easy to understand and adjust for the test location. The simplest version can use one threshold, while a more developed version can use separate warning and danger levels. The controller should check that readings are valid before raising an alert. A short persistence check or a gap between the trigger and reset levels can help stop the alarm from switching repeatedly when a reading sits near the limit.

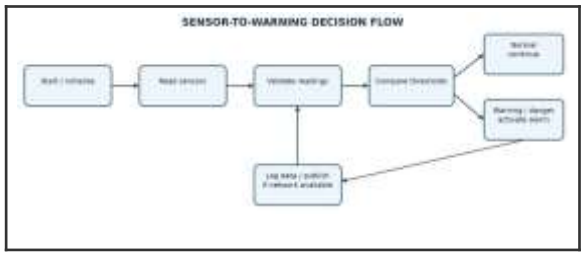


Fig. 3. Example sensor-to-warning decision flow.

A. Normal Monitoring

In normal operation, the ESP32 reads each enabled sensor at a set interval, checks whether the values are plausible and updates the local indicator or dashboard. The interval should suit the sensor and the response needed, while also considering power use.

B. Warning and Danger States

When a valid reading crosses the warning limit, the device records the warning state and activates the chosen indicator. A higher danger limit can trigger a louder or more noticeable alert and, where configured, a remote message. These limits are settings for the chosen site and sensor; they are not universal disaster thresholds.

C. Fault Handling and Network Loss

If a sensor is unplugged, stops updating or produces an implausible value, the system should show a fault rather than report that conditions are safe. If Wi-Fi is lost, local monitoring and alerts should continue where possible, and the communication problem should be made visible.

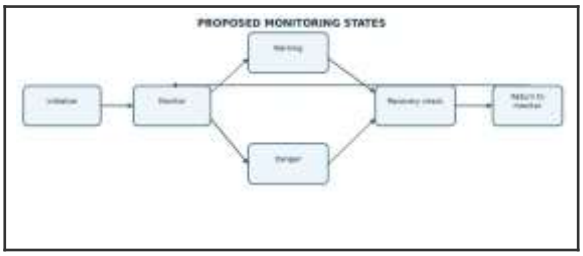


Fig. 4. Suggested monitoring states for the system.

3.1 CONTROL FLOW AND SOFTWARE DESIGN

The firmware can run as a repeating state machine. In each cycle, it reads the sensors, checks the measurements, evaluates the warning rules, updates the outputs and sends data if the network is available. Possible states are Initialize, Monitor, Warning, Danger and Sensor/Network Fault. The conditions for moving between states should be written down and tested.

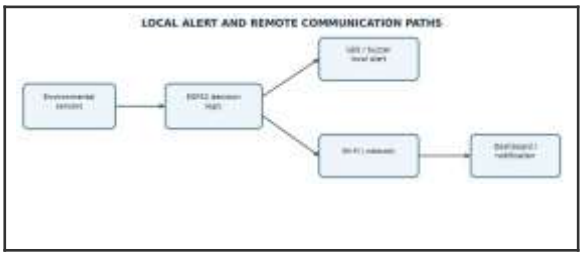


Fig. 5. Local alert and IoT communication paths.

Startup checks, sensor timeouts and a clear recovery process can help the software handle common faults. A watchdog and communication routines that do not block sensor checks may improve reliability. Relevant events should be logged. Any performance claim, including response time or accuracy, should be based on measurements taken after the prototype has been built and tested.

4. IMPLEMENTATION CONSIDERATIONS

A. Power Supply

Use a regulated supply appropriate for the ESP32 board and sensor modules. For battery operation, estimate current consumption and expected runtime, and use suitable charging and protection circuitry. Power interruption should be considered in the fault-handling design.

B. Enclosure and Sensor Installation

The controller should be installed in an enclosure suitable for the intended environment. Cable entries, sensor openings and connectors need protection against moisture, dust, corrosion and mechanical damage. Outdoor use requires environmental testing and maintenance planning.

C. Wi-Fi and IoT Dashboard

The ESP32 can send measurements to a configured dashboard or server using an appropriate network protocol. Remote records should include device identifier, timestamp, units and warning status. Network credentials must be stored securely and must not be exposed in a public paper.

D. Calibration and Validation

For validation, record the test date, sensor name, reference condition, measured output, threshold and expected alert. The test plan should include normal conditions, gradual and rapid changes, a disconnected sensor and Wi-Fi loss. These checks show how the prototype behaves and where it has limitations; they do not establish certified prediction accuracy. Thresholds should be selected for the test site and reviewed whenever the hardware or sensor position changes.

Compare each sensor with a known or independently measured condition where possible. Test normal, warning and danger conditions, as well as a disconnected sensor and network failure. Record actual readings, response times and false alarms only after carrying out the tests; do not estimate or invent results.

5. LIMITATIONS AND SAFETY

This project is an educational monitoring prototype. It is not a certified disaster-warning or life-safety system and should not replace official alerts, emergency services, professional instruments or local disaster-management procedures. A real installation would need site-specific review and appropriate approval.

Low-cost sensors may give inaccurate readings, drift over time or stop working in harsh conditions. Other possible problems include false alarms, missed events, power cuts, network outages and limited sensing range. Soil-moisture readings alone cannot predict a landslide, and a basic vibration module cannot provide certified earthquake detection or prediction. Outdoor and wet installations also need proper electrical protection.

The software can reduce avoidable alarms by checking whether a condition remains present for several readings before changing the warning state. It should still respond quickly enough for the demonstration. A clear rule for returning to normal can also stop the buzzer from switching repeatedly around one threshold. These steps may make the system easier to use, but they cannot remove sensor error. If an important reading is missing, the device should show an unknown or fault state instead of silently indicating normal conditions.

6. APPLICATIONS AND ADVANTAGES

Possible uses in a classroom or pilot setting include observing water-level changes in a test channel, recording rainfall and local weather conditions, monitoring temperature in a facility and demonstrating sensor-based alerts in a laboratory or campus project.

The design is modular, relatively inexpensive and useful for learning embedded systems and IoT. A local LED or buzzer can work without a cloud connection, while a dashboard can help review readings or monitor more than one node when internet service is available. Before public use, the system would need thorough testing, backup arrangements and approval from the relevant authorities.

7. FUTURE SCOPE

Possible future additions include backup sensors, solar power with battery monitoring, cellular communication, GPS location, data logging and a secure dashboard. Several monitoring nodes could also be connected to provide readings from different locations.

Trend analysis or anomaly detection could be explored in a later version, but any model would need suitable data and careful evaluation. Before the device is used outside a demonstration, it should be checked for sensor faults, power loss, network failure, severe weather and maintenance requirements.

8. CONCLUSION

This paper outlines a proposed low-cost monitoring and early-warning prototype based on an ESP32, selected environmental

sensors, local alerts and optional IoT connectivity. The design separates sensor input, decision-making, local warning and remote reporting, so individual parts can be changed or expanded as needed.

The prototype can demonstrate how embedded electronics and IoT are used to monitor environmental conditions. A dependable disaster-warning service requires more than a controller and sensors: it also needs suitable equipment, site-specific thresholds, calibration, redundancy and careful validation. For this reason, the proposed system should be treated as an educational aid, not a replacement for official or certified warning systems.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our SPI Coordinator, Mr. V. B. Takale, for his valuable guidance, encouragement, and support throughout the preparation of this review paper on "LOW-COST DISASTER DETECTION AND EARLY WARNING SYSTEM". We would also like to express our sincere gratitude to our Project Coordinator, Mr. M. N. Waghmode, for his valuable guidance, continuous encouragement, and technical support throughout the preparation of this review paper. We are also thankful to Mr. N. E. Bhutekar, Head of the Electronics and Telecommunication (E&TC) Department, for his valuable suggestions, academic guidance, and support. We extend our sincere thanks to our respected Principal, Dr. P. T. Kale, Vice Principal, Dr. M. S. Yadav, and Academic Dean, Dr. V. S. Dabhade, for their encouragement and academic support. We are grateful to all faculty members of the E&TC Department for their cooperation, technical assistance, and constructive suggestions. Finally, we thank JSPM Bhivrabai Sawant Polytechnic, Wagholi, Pune, for providing the necessary facilities and a supportive learning environment to complete this review paper successfully.

REFERENCES

[1] Espressif Systems, "ESP32 Series Datasheet," official technical documentation.

[2] Espressif Systems, "ESP-IDF Programming Guide," official documentation for firmware development and networking.

[3] World Meteorological Organization, "Multi-hazard Early Warning Systems," guidance on people-centred early warning systems.

[4] National Disaster Management Authority, Government of India, disaster-management guidelines and public advisories relevant to the selected hazard.

[5] Manufacturer datasheet for the exact water-level sensor selected, including operating range and installation limitations.

[6] Manufacturer documentation for the selected rain sensor/module and its calibration requirements.

[7] Manufacturer datasheet for the selected temperature/humidity sensor and its accuracy and operating range.

[8] Espressif Systems, ESP32 hardware design guidelines, including supply and GPIO requirements.

[9] Arduino, "Arduino IDE Documentation," if the Arduino framework is used.

[10] ThingSpeak, official documentation for IoT data collection and visualization, if selected.

[11] Blynk, official documentation for IoT dashboards and notifications, if selected.

[12] International Telecommunication Union, publications on ICTs for disaster management and emergency communications.

[13] National Centre for Seismology, Government of India, earthquake information and educational resources.

[14] Central Water Commission, Government of India, flood forecasting and hydrological information resources.

[15] Document sensor calibration and testing procedures using the actual component datasheets and test equipment used.

[16] IJSREM reference manuscript supplied for layout guidance; illustrative component photographs require source attribution and permission review before reuse.



Rahul Pawar
E&TC Department
JSPM'S BSP, Wagholi, Pune



Rohit Navnath Pawar
E&TC Department
JSPM'S BSP, Wagholi, Pune
Email: rohitpawar5488@gmail.com



Omkar Popat Mali
E&TC Department, JSPM'S BSP Wagholi, Pune



Mr. Takale Vikas Bhiku
E&TC Department, JSPM'S BSP Wagholi, Pune
Email: vbtakale_entc@jspmbspoly.edu.in