PROJECT REPORT

Disaster Mitigation

Project Report



IoT-Based Flood Early Warning System

STUDENT NAME	**Asem Ameen Qaid**
NIM	**24524038**
GROUP	**IP**

Undergraduate Program of Electrical Engineering
Faculty of Industrial Technology
Islamic University of Indonesia

2026

ABSTRACT

Floods are among the most common natural disasters and cause significant damage to infrastructure, the environment and human life. Heavy rainfall and rising water levels can develop quickly into dangerous situations if they are not detected early. This report presents a low-cost IoT-based flood early warning system that monitors rainfall and water level continuously and provides both local and remote alerts.

An ESP32 microcontroller reads a rain sensor and a water level sensor and classifies the environmental condition into three levels: SAFE, WARNING and DANGER. Each level drives a local alert through LEDs and a buzzer. At the same time the ESP32 publishes the readings over the MQTT protocol to a Mosquitto broker, and a Node-RED dashboard subscribes to the data and displays the current flood status and the water-level history in real time.

The system was assembled as a breadboard prototype and demonstrated in all three alert states, with simultaneous local warning and remote monitoring.

Keywords: flood early warning, Internet of Things, ESP32, MQTT, Mosquitto, Node-RED, disaster mitigation

CONTENTS

1 Introduction

2 System Design

3 Implementation

4 Results

5 Impact of the Early Warning System

6 Conclusion and Future Work

1 Introduction

1.1 Background

Floods are among the most common natural disasters that cause significant damage to infrastructure, the environment and human life. Heavy rainfall and rising water levels can quickly develop into dangerous situations if they are not detected at an early stage. Traditional monitoring that relies on manual observation leads to delayed emergency response and leaves communities vulnerable. An effective early warning system is therefore essential to reduce disaster risk and improve community preparedness.

Internet of Things (IoT) technology makes continuous environmental monitoring more efficient and reliable. IoT devices can collect sensor data in real time and transmit it wirelessly. In this project, an ESP32 microcontroller is combined with a rain sensor, a water level sensor, LEDs, a buzzer, MQTT communication, a Mosquitto broker and a Node-RED dashboard to create a low-cost flood early warning system with both local and remote alerts.

1.2 Problem Formulation

1. How can rainfall and water level be monitored in real time?
2. How can an IoT system provide an early warning before flooding occurs?
3. How can sensor data be transmitted efficiently using MQTT?
4. How can users monitor flood conditions remotely through a dashboard?

1.3 Objectives

- To design an IoT-based flood early warning system.
- To monitor rainfall and water level continuously.
- To classify environmental conditions into SAFE, WARNING and DANGER.
- To provide visual and audible alerts using LEDs and a buzzer.
- To transmit sensor data using MQTT.
- To display real-time monitoring information on a Node-RED dashboard.

2 System Design

2.1 Block Diagram

The system consists of two sensors connected to an ESP32 microcontroller (Figure 1). The rain sensor detects rainfall, while the water level sensor measures the water height. The ESP32 continuously reads both sensors and processes the data according to predefined thresholds. It

drives the local alerts directly and publishes the sensor data over Wi-Fi using the MQTT protocol. The Mosquitto broker relays the messages to Node-RED, which displays them on a dashboard. This enables local warning and remote monitoring at the same time.

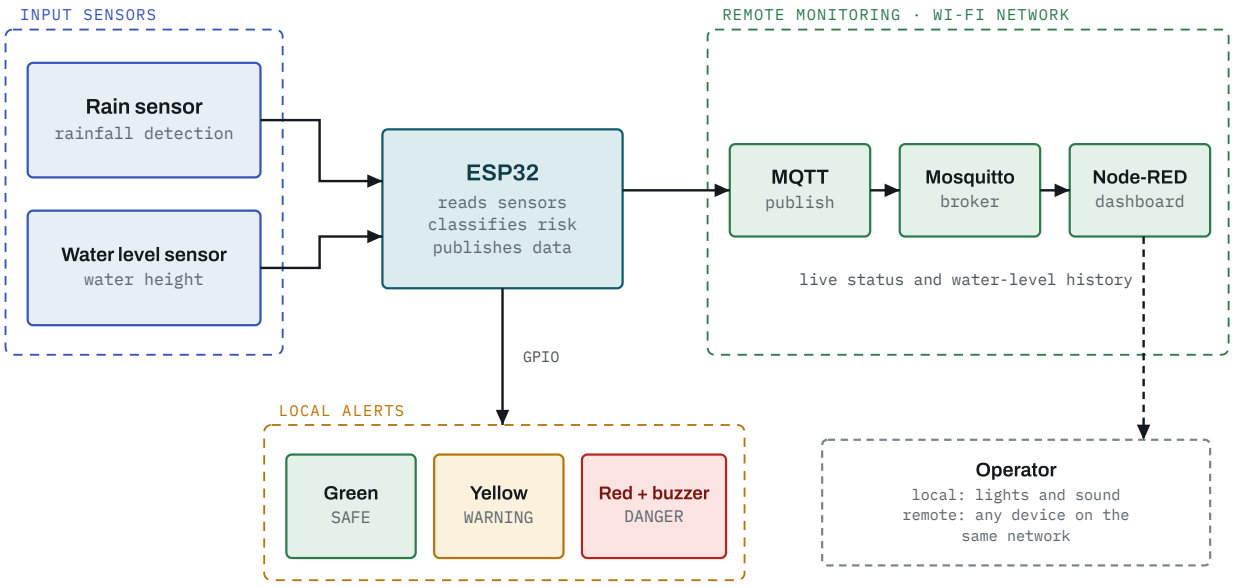


Figure 1. Block diagram of the IoT-based flood early warning system.

2.2 Alert Levels

Based on the sensor values, the system classifies the environmental condition into three levels, each with its own local alert (Figure 2, Table 1).

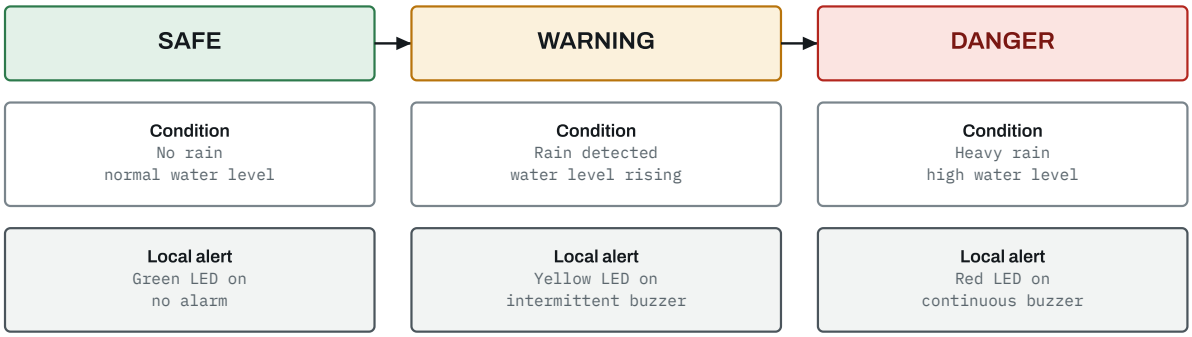


Figure 2. Flood risk levels, the conditions that define them and the corresponding local alerts.

Table 1. Status levels and local alerts

Level	Condition	LED	Buzzer
SAFE	No rain, normal water level	Green	Off
WARNING	Rain detected, water level rising	Yellow	Intermittent
DANGER	Heavy rain, high water level	Red	Continuous

2.3 Program Flow

Figure 3 shows the operating cycle of the ESP32. After initialising its outputs and connecting to the Wi-Fi network and MQTT broker, the controller repeatedly reads both sensors, compares the values with the thresholds, sets the local alert for the resulting level and publishes the readings and status to the broker.

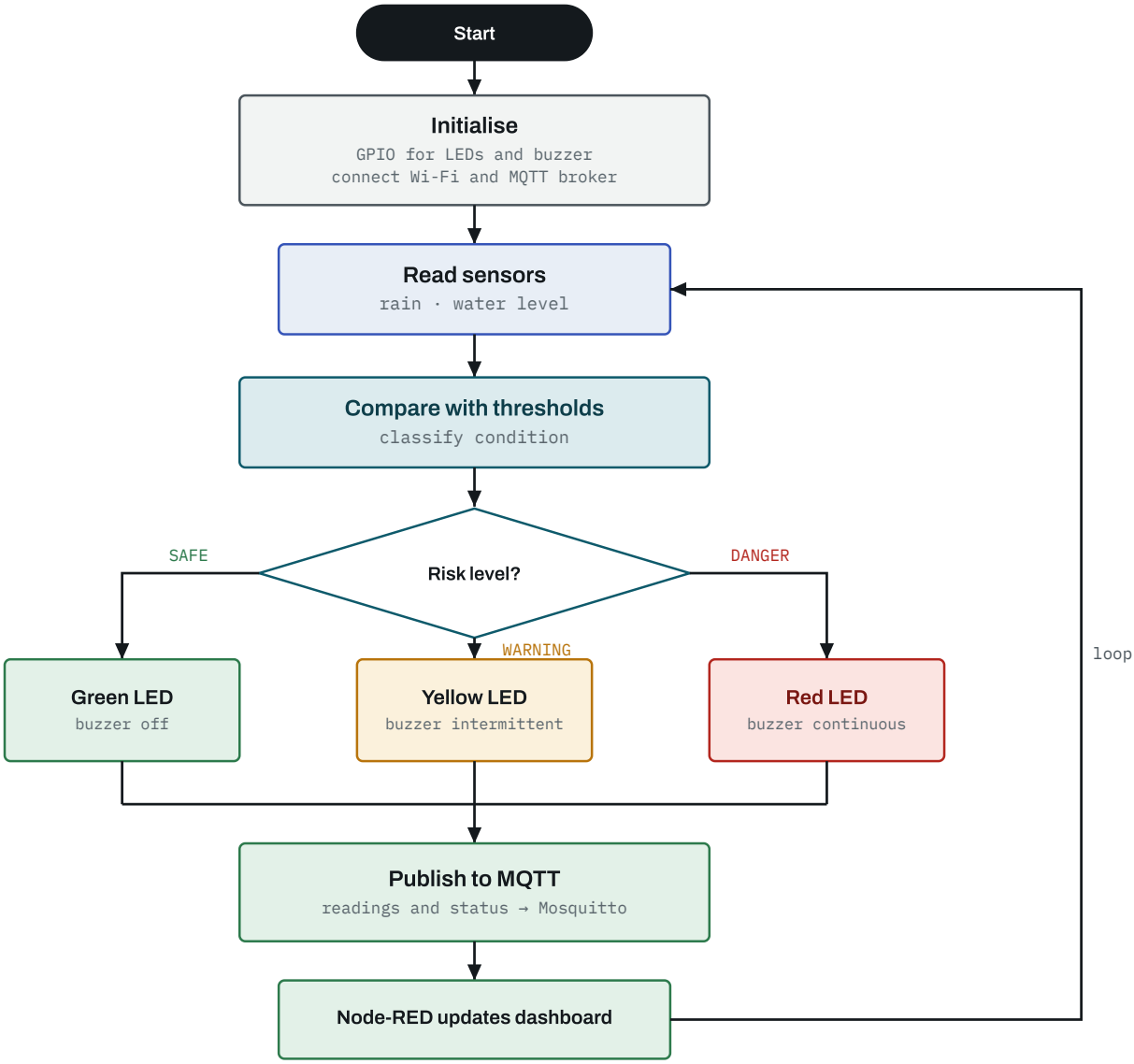


Figure 3. Operating flowchart of the ESP32 firmware.

2.4 Communication

MQTT is a lightweight publish–subscribe protocol suited to small IoT devices. The ESP32 acts as a publisher and sends its readings to a topic on the Mosquitto broker. Node-RED subscribes to the same topic and receives every new message as soon as it is published, so the dashboard always shows the latest readings and status without polling the device.

3 Implementation

3.1 Development Environment

The software tools were prepared first (Figure 4): the Arduino IDE for programming the ESP32, Python for back-end scripting, and Mosquitto and Node-RED for message brokering and the user interface. The hardware was simulated on the Wokwi platform before physical assembly.

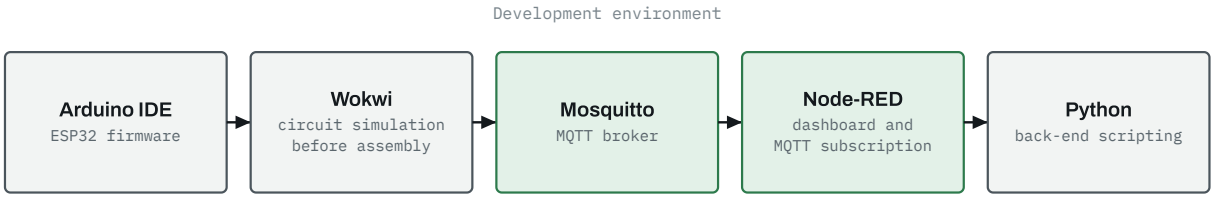


Figure 4. Software tools used to develop the system.

3.2 Hardware Prototype

The prototype was assembled on a breadboard (Figure 5). The rain sensor and water level sensor connect to the ESP32 development board, and three LEDs (green, yellow and red) and a buzzer provide the local alerts.

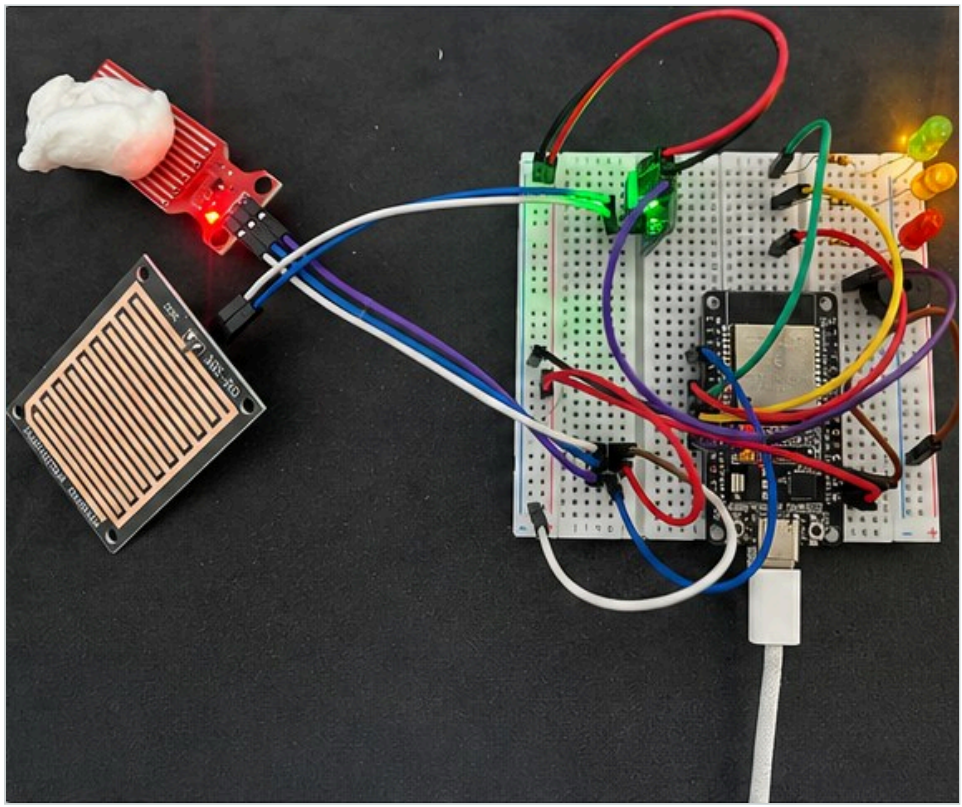


Figure 5. Breadboard prototype with the ESP32, rain sensor, water level sensor, status LEDs and buzzer.

3.3 Monitoring Dashboard

The Node-RED dashboard (Figure 6) shows the current flood status, the rain status, a history chart of the water-level readings and a gauge of the current water value. It can be opened from any device connected to the same network.

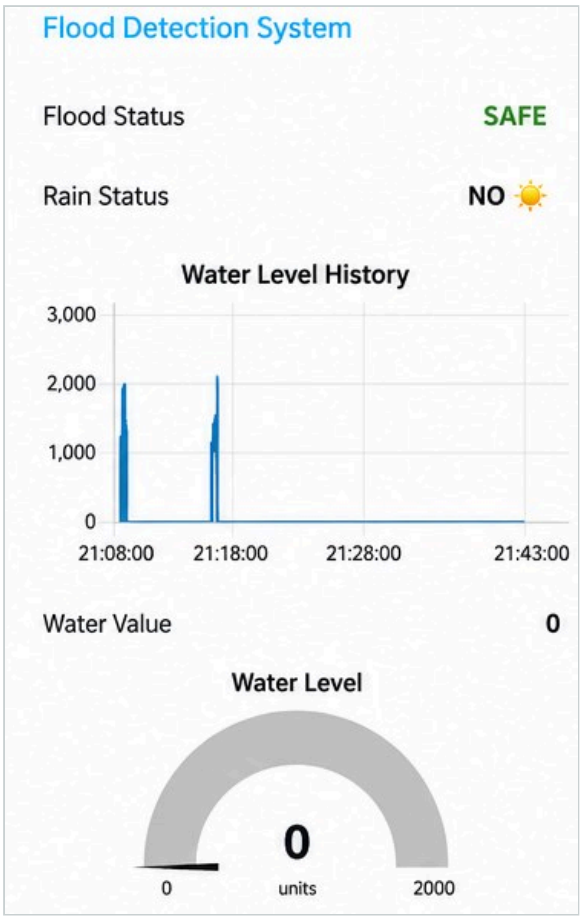


Figure 6. Node-RED monitoring dashboard showing flood status, rain status and water-level history.

4 Results

The prototype was operated under the three conditions it is designed to recognise. In each case the ESP32 classified the condition correctly and produced the corresponding local alert: the green LED with no alarm for SAFE, the yellow LED with an intermittent buzzer for WARNING, and the red LED with a continuous buzzer for DANGER. At the same time the readings and status were published over MQTT and appeared on the Node-RED dashboard, whose water-level history chart records the readings over time (Figure 6).

Table 2. Functional test results

Function	Observed behaviour	Result
Rain and water level sensing	Both sensors read continuously by the ESP32	Working
SAFE classification	Green LED on, buzzer off	Working

Function	Observed behaviour	Result
WARNING classification	Yellow LED on, intermittent buzzer	Working
DANGER classification	Red LED on, continuous buzzer	Working
MQTT transmission	Readings published to the Mosquitto broker	Working
Remote dashboard	Status and water-level history updated in Node-RED	Working

5 Impact of the Early Warning System

The system offers several benefits for disaster mitigation. First, it allows continuous monitoring of rainfall and water level, so abnormal conditions can be identified before flooding occurs. Second, the LEDs and buzzer provide immediate local warnings that help people react quickly during emergencies. Third, MQTT communication delivers the sensor data to the Node-RED dashboard in real time, enabling remote monitoring from any device connected to the same network.

Finally, the system is low-cost and easy to implement, and it can be expanded with cloud platforms, SMS notifications or mobile applications. This makes it suitable as a building block for smart city and community disaster management.

6 Conclusion and Future Work

An IoT-based flood early warning system was designed and implemented using an ESP32, a rain sensor, a water level sensor, LEDs, a buzzer, MQTT communication, a Mosquitto broker and Node-RED. The system monitors environmental conditions in real time, classifies flood risk as SAFE, WARNING or DANGER, and provides both local and remote alerts. The implementation shows that IoT technology can improve disaster preparedness through continuous monitoring and fast delivery of information.

The current prototype operates on a local network, and its thresholds were set for demonstration rather than calibrated against real flood data. Future improvements include:

- cloud data storage, so that readings can be accessed outside the local network;
- mobile and SMS notifications;
- GPS integration for multi-site deployments;
- integration of weather forecasting services;
- AI-based flood prediction from historical data.