**FLOOD MONITORING AND EARLY WARNING SYSTEM**

|  |
| --- |
| **Phase 5 - Documentation** |

|  |  |
| --- | --- |
| **Date** | **01.11.2023** |
| **Team ID** | **Proj\_223985\_team** |
| **Project Name** | **Flood monitoring and Early warning** |
| **Members** | **Rasswanth G**  **Vishwa S**  **Saravanan S**  **Pragadheesh M**  **Yogesh R** |

**Introduction:**

Floods pose a significant threat to human life and infrastructure, necessitating effective flood monitoring and early warning systems. Technological advancements, such as advanced sensors, Arduino microcontrollers, ESP32 modules, and cloud computing, can enhance monitoring of water levels, rainfall, and weather patterns, enabling proactive measures to save lives and minimize damage.

**1. Sensor Integration:**

Integrate the sensors with the hardware components as follows:

**Water Level Sensor:**

* Connect the water level sensor to the Arduino, utilizing appropriate analog or digital pins. The water level sensor is crucial for monitoring water levels in the specified area.

**Rainfall Sensor:**

* Wire the rainfall sensor to the Arduino, enabling it to measure the amount of rainfall accurately.

**Temperature and Humidity Sensor:**

* Establish a connection between the temperature and humidity sensor and the Arduino. This sensor provides essential data about the ambient temperature and humidity.

**2. Arduino and ESP32 Communication:**

* Write code in the Arduino IDE to read data from the sensors accurately. Employ suitable communication protocols like I2C or UART to facilitate seamless data transfer between the Arduino and ESP32.

**3. ESP32 Wi-Fi Setup:**

* Configure the ESP32 to connect to an available Wi-Fi network using the appropriate libraries and credentials (SSID and password). Establish a stable connection to enable the transmission of data.

**4. Cloud Integration:**

* Create and configure a cloud server (e.g., AWS, Google Cloud, Azure) to receive and process data transmitted by the ESP32. Utilize protocols such as MQTT or HTTP for secure and reliable data transmission from the ESP32 to the cloud server.

**5. Data Processing and Alerting:**

* Implement algorithms on the ESP32 to process the incoming sensor data. Set predefined thresholds or patterns to identify potential flood conditions based on the sensor readings. For example, a rapid rise in water levels could trigger a flood warning.
* Define alert mechanisms to notify relevant parties, such as sending SMS notifications via a GSM/GPRS module, when critical flood conditions are detected.

**6. Real-Time Visualization:**

* Develop a web application using standard web technologies like HTML, CSS, and JavaScript. Utilize a back-end framework (e.g., Node.js) to handle data and facilitate real-time visualization. Incorporate libraries such as Chart.js to create graphs and charts that display the sensor data for easy interpretation.

**7. Testing and Calibration:**

* Thoroughly test the entire system under various environmental conditions, simulating different weather scenarios and flood situations. Calibrate the sensors to ensure precise readings and fine-tune the algorithms based on the results of the testing phase.

**8. Deployment:**

* Deploy the fully functional system in flood-prone areas, ensuring an uninterrupted power supply through appropriate means such as solar power or battery backup. Utilize weatherproof enclosures to protect the hardware components from adverse outdoor conditions.
* Regularly monitor the system's performance, conduct maintenance checks, and make necessary adjustments to ensure it continues to provide accurate and timely flood warnings.

**Additional Considerations:**

* Implement a data logging mechanism to record historical sensor data for analysis and future reference.
* Include a feature for remote configuration and firmware updates to enhance the system's adaptability and longevity.
* Collaborate with local authorities and communities to ensure they receive timely alerts and can take appropriate actions in response to flood warnings.

**Project Overview:**

The flood monitoring and early warning system is designed to detect and alert users to potential flood conditions. It incorporates various sensors and an ESP32 microcontroller to measure temperature, humidity, and water levels, providing real-time data to a cloud-based platform for monitoring.

**Purpose and Scope:**

The primary purpose of this system is to monitor environmental conditions and, more specifically, water levels in flood-prone areas. By doing so, it serves as an early warning system, helping to prevent damage and improve safety in flood-prone regions.

**Objectives:**

* Develop a system for real-time monitoring of temperature, humidity, and water levels.
* Interface with ThingSpeak, a cloud-based platform, to store and display data.
* Implement a warning mechanism through a buzzer when water levels reach a critical threshold.

**Hardware Components:**

* ESP32 microcontroller
* DHT22 sensor used for temperature and humidity
* Ultrasonic sensor used for water level measurement
* Buzzer used for early warning

**Components Descriptions:**

* ESP32 Microcontroller: The ESP32 is a versatile microcontroller that provides Wi-Fi connectivity, making it suitable for IoT applications. In this project, it acts as the central processing unit.
* DHT11 Sensor: The DHT11 sensor is utilized for temperature and humidity measurements. It provides accurate environmental data.
* Ultrasonic Sensor: The ultrasonic sensor is responsible for measuring water levels in rivers, lakes, or other bodies of water.
* Buzzer: The buzzer serves as an audible warning device. When water levels reach a critical threshold, it alerts users with a loud sound.

**Software Components:**

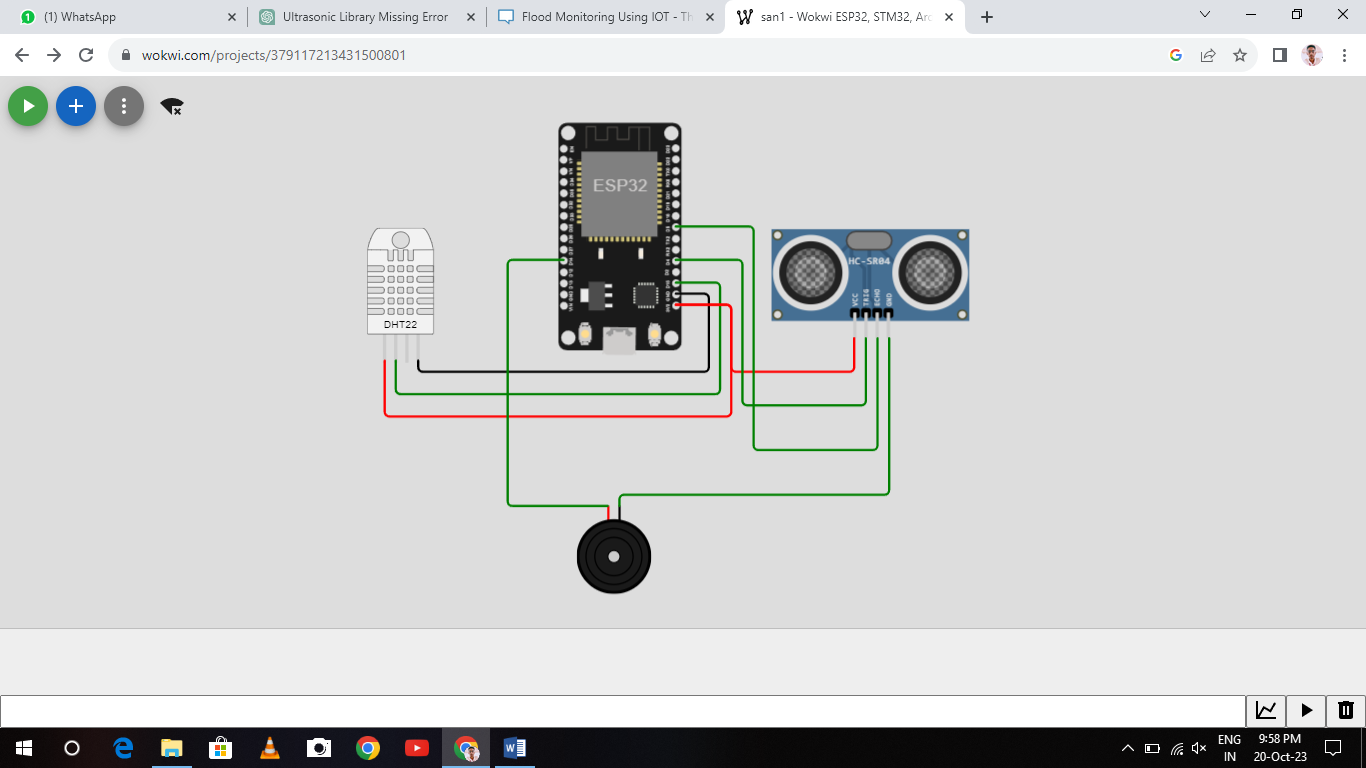
* DHTesp library for DHT22 sensor
* Ultrasonic library for ultrasonic sensor
* ThingSpeak library for cloud integration

**Software Descriptions:**

* DHTesp Library: This library enables communication with the DHT22 sensor, making it possible to read temperature and humidity values.
* Ultrasonic Library: The Ultrasonic library provides functions for interfacing with the ultrasonic sensor, allowing accurate water level measurements.
* ThingSpeak Library: The ThingSpeak library is used for sending data to the ThingSpeak platform, where it can be visualized and monitored.

**System Architecture:**

The system architecture involves the ESP32 as the central controller, which interfaces with the DHT22 and ultrasonic sensors. The ESP32 collects data from these sensors, processes it, and sends the data to the ThingSpeak platform over Wi-Fi. ThingSpeak then displays the data, and when water levels surpass a predefined threshold, the ESP32 triggers the buzzer to issue a warning.

****

**Source Code:**

#include <WiFi.h>

#include <DHTesp.h>

#include <Ultrasonic.h>

#include <ThingSpeak.h>

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

const unsigned long channelID = 2314097;

const char\* writeAPIKey = "CYPMNXUMXZ4YM8U6";

#define DHT\_PIN 15

DHTesp dht;

#define TRIGGER\_PIN 4

#define ECHO\_PIN 5

Ultrasonic ultrasonic(TRIGGER\_PIN, ECHO\_PIN);

#define BUZZER\_PIN 14

const int WaterLevelThreshold = 50;

WiFiClient client;

void setup() {

**Serial**.begin(115200);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

**Serial**.println("Connecting to WiFi...");

}

**Serial**.println("Connected to WiFi");

ThingSpeak.begin(client);

dht.setup(DHT\_PIN, DHTesp::DHT22);

pinMode(BUZZER\_PIN, OUTPUT);

}

void loop() {

TempAndHumidity data = dht.getTempAndHumidity();

long distance = ultrasonic.read();

if (isnan(data.temperature) || isnan(data.humidity) || distance == 0) {

**Serial**.println("Failed to read from sensors.");

return;

}

**Serial**.print("Temperature (C): ");

**Serial**.println(data.temperature);

**Serial**.print("Humidity (%): ");

**Serial**.println(data.humidity);

**Serial**.print("Water Level (cm): ");

**Serial**.println(distance);

ThingSpeak.setField(1, data.temperature);

ThingSpeak.setField(2, data.humidity);

ThingSpeak.setField(3, distance);

int status = ThingSpeak.writeFields(channelID, writeAPIKey);

if (status == 200) {

**Serial**.println("Data sent to ThingSpeak successfully");

} else {

**Serial**.print("Data send failed, status code: ");

**Serial**.println(status);

}

if (distance < WaterLevelThreshold) {

digitalWrite(BUZZER\_PIN, HIGH);

} else {

digitalWrite(BUZZER\_PIN, LOW);

}

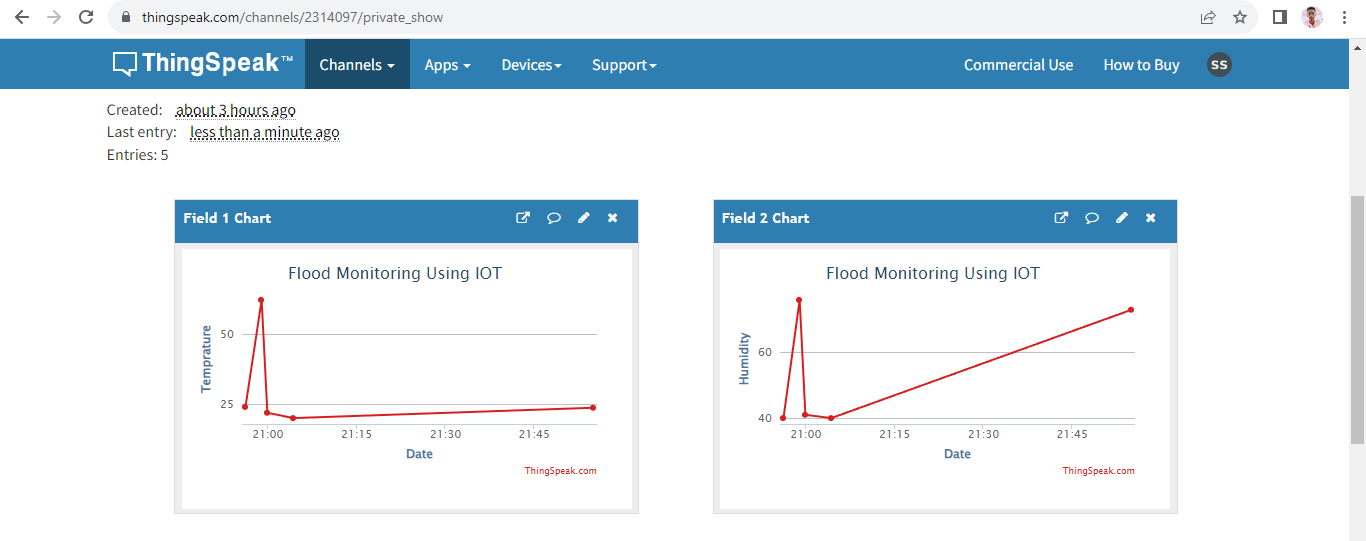
delay(900000); }

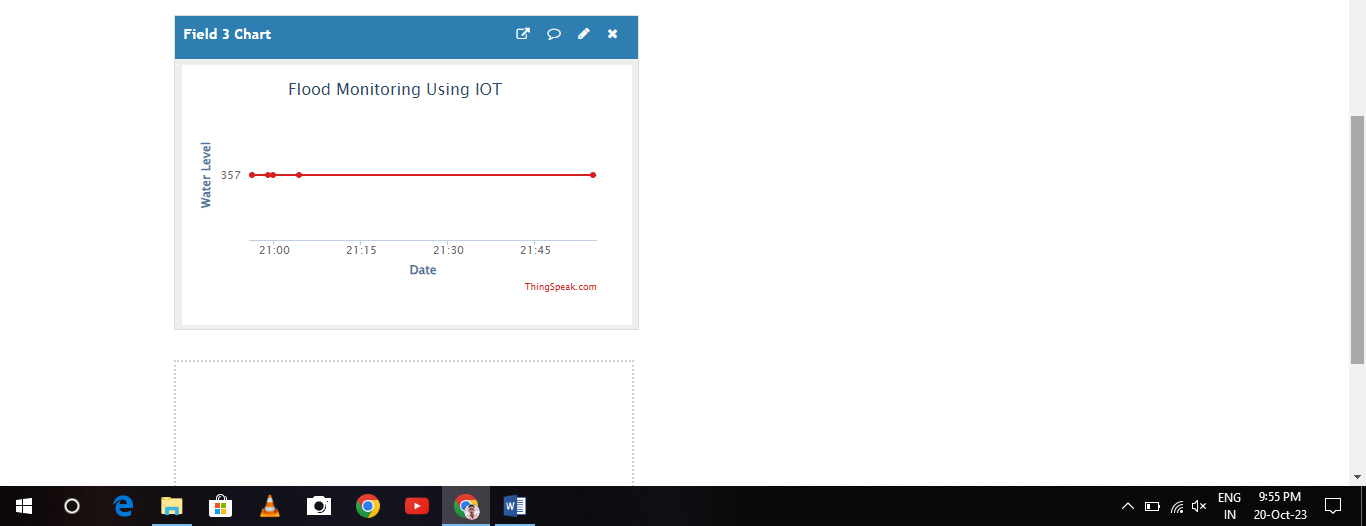
**Operation:**

The system operates by continuously reading temperature, humidity, and water level data from the DHT22 and ultrasonic sensors. It sends this data to ThingSpeak via Wi-Fi for remote monitoring. If the water level surpasses a predefined threshold, the buzzer is triggered to issue an early warning.

The significance of the water level threshold lies in defining the point at which a flood warning is triggered.

**Digital Outputs:**

****

****

**Development Part 2: Early Warning Platform**

**Phase Description:**

Phase 4 represents the second part of the development stage in the project. This phase is dedicated to the creation of the early warning platform, a critical component of the Flood Monitoring and Early Warning System. The platform will be designed using web development technologies such as HTML, CSS, and JavaScript. It will serve as the interface for displaying real-time water level data and issuing flood warnings when necessary. The primary goal of this phase is to ensure that the platform effectively and accurately conveys important information to both the public and relevant authorities

This phase builds upon the work initiated in the previous phases of the project, particularly the deployment of IoT sensors and the establishment of data collection mechanisms. The early warning platform represents a key milestone in enhancing the project's capabilities and providing a user-friendly, accessible system for flood monitoring and response.

**Objective:**

The primary objective of this phase is to develop a functional and user-friendly early warning platform that effectively conveys real-time water level data and flood warnings to the public and relevant authorities. The platform will play a crucial role in enhancing public safety and facilitating coordinated responses to flood emergencies.

**Data Flow and Interactions:**

1. The early warning platform serves as the user interface, displaying real-time data and warnings to the users.
2. The Data Fetching Module continuously requests data from the IoT sensor network, specifically focusing on water level, temperature, and humidity data.
3. The fetched data is processed and displayed on the platform, allowing users to monitor current environmental conditions.
4. The Warning System monitors the water level data, applying predefined criteria and thresholds to determine when to issue flood warnings.
5. If a flood warning is triggered, the Warning System displays the warning message on the platform.
6. Interactive Elements enable users to interact with the platform, acknowledge warnings, or dismiss them.
7. Optional Map Integration (if applicable) provides visual context regarding the geographical data related to flood-prone areas and sensor locations.

**Warning System Implementation**

The Warning System Implementation involves several key components and processes:

**Data Monitoring:** The system continuously monitors the water level data received from IoT sensors. This monitoring is essential to detect any significant changes in water levels.

**Thresholds and Criteria:** The Warning System defines specific criteria and thresholds for issuing flood warnings. These criteria are typically based on the rate of change in water levels and the desired level of accuracy. For example, a warning may be triggered if water levels rise rapidly or reach a certain height.

**Real-time Data Analysis:** The system performs real-time data analysis to evaluate whether the predefined criteria and thresholds are met. This analysis involves comparing the current water level with the established thresholds.

**Warning Triggers:** When the analysis determines that the water level has exceeded the defined thresholds, the Warning System triggers the issuance of a flood warning. The warning can vary in severity based on the specific conditions, such as a flood advisory, flood watch, or flood warning.

**Alert Mechanisms:** The Warning System includes mechanisms to alert users and relevant authorities about the impending flood risk. These mechanisms can include visual alerts on the early warning platform, audible alerts (e.g., sirens or alarms), email notifications, SMS messages, or mobile app notifications.

**User Acknowledgment:** The system may provide users with the ability to acknowledge warnings, confirming that they have received the alert. This acknowledgment feature can be valuable for tracking user responses and ensuring that warnings are not missed.

**User Interaction:** In addition to acknowledging warnings, the Warning System may allow users to dismiss warnings if they believe that the situation does not pose an immediate threat. User interaction ensures that individuals can make informed decisions based on their assessment of the situation.

**Emergency Response Coordination:** The issuance of flood warnings also serves to facilitate coordination and communication among emergency response teams and relevant authorities. This coordination is vital for swift and organized responses to flood emergencies.

**Visual Alerts on the Early Warning Platform:**

1. **Platform Display:** The early warning platform, which is typically a web-based interface, continuously monitors real-time data, including water levels. When the system detects that predefined criteria and thresholds are met (e.g., rising water levels), it triggers the issuance of a visual alert.
2. **Alert Messages:** These visual alerts are often displayed as alert messages or banners on the platform. They are designed to catch the user's attention and convey essential information about the flood risk.
3. **Severity Differentiation:** Visual alerts may vary in severity based on the specific conditions. For example, a flood advisory might indicate a potential risk, while a flood warning signifies an imminent threat. Users can differentiate the severity of the situation based on the type of alert displayed.
4. **User Interaction:** Users who access the platform can view these alerts, acknowledge them, and take appropriate actions based on the information provided. This interaction ensures that individuals are aware of the situation and can make informed decisions.
5. **Real-time Updates:** The platform continuously updates these alerts as the situation evolves. Users can rely on the platform for up-to-date information about flood risks.

**Audible Alerts (Sirens or Alarms):**

1. **Detection Mechanism:** In some cases, flood monitoring systems may be equipped with audible alert mechanisms, such as sirens or alarms. These mechanisms are triggered by the Warning System when water levels reach critical thresholds.
2. **Audible Warning:** When the system determines that a significant flood risk exists, it activates the sirens or alarms. These audible alerts are designed to be loud and attention-grabbing, ensuring that people in the affected area can hear them.
3. **Community-wide Notification**: Audible alerts are particularly valuable for notifying an entire community or area about an imminent flood risk. They can be heard in public spaces and residential areas.
4. **Response Actions:** When individuals hear audible flood warnings, they should take immediate action based on local emergency response plans. This might involve evacuating flood-prone areas, seeking higher ground, or following specific safety instructions.

**HTML CODE**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Flood monitoring</title>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<div class="data-head data-color">Flood Monitoring and Alerting Dashboard</div>

<div class="data-container data-color1">

<div class="data-item">

<h1>Temperature</h1>

<p class="value" id="temperature"></p>

</div>

</div>

<div class="data-container data-color2">

<div class="data-item">

<h1>Humidity</h1>

<p class="value" id="humidity"></p>

</div>

</div>

<div class="data-container data-color3">

<div class="data-item">

<h1>Waterlevel</h1>

<p class="value" id="Waterlevel"></p>

</div>

</div>

<div class="warning-message" id="waterLevelWarning" style="display: none;">Water level is rising!</div>

<div class="map-container" id="map"></div>

<div class="popup" id="popup">

<div class="popup-content">

<span class="close" id="closePopup">&times;</span>

<h2>Water Level Alert</h2>

<p>The water level is rising. Take necessary precautions.</p>

</div>

</div>

<script src="script.js"></script>

<footer>

<p>&copy; 2023 AS</p>

</footer>

</body>

</html>

**CSS CODE**

body{

background-color: lightblue;

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

.data-container{

display: flex;

justify-content: space-between;

align-items: center;

width: 50%;

height: 100px;

margin: 50px auto;

padding: 30px;

background-color: AliceBlue;

box-shadow: 0 2px 20px rgba(0, 0, 0, 0.1);

border-radius: 5px;

}

.data-color{

background-color: Tomato;

}

.data-color1{

background-color: orange;

box-shadow: 0 5px 20px rgba(0, 0, 0, 0.1);

border-radius: 20px;

}

.data-color2{

background-color: white;

box-shadow: 0 5px 20px rgba(0, 0, 0, 0.1);

border-radius: 20px;

}

.data-color3{

background-color: green;

box-shadow: 0 5px 20px rgba(0, 0, 0, 0.1);

border-radius: 20px;

}

.data-item{

text-align: center;

flex: 1;

}

.data-item h1{

font-size: 42px;

font-weight: bold;

margin-bottom: 10px;

}

.data-item p{

font-size: 48px;

font-weight: bold;

color: #6EB7FF;

}

.data-head{

margin: auto;

width: 50%;

text-align: center;

font-size: 45px;

font-weight: bold;

margin: 50px auto;

padding: 20px;

box-shadow: 0 5px 20px rgba(0, 0, 0, 0.1);

border-radius: 20px;

}

.warning-message {

color: red;

text-align: center;

font-weight: bold;

font-size: 130px;

width: 100%;

}

.map-container {

width: 100%;

height: 100vh;

}

nav ul {

list-style-type: none;

padding: 0;

}

nav li {

display: inline;

margin-right: 10px;

}

main {

margin-top: 20px;

}

aside {

float: right;

width: 30%;

margin-top: 20px;

}

footer {

clear: both;

background-color:white;

text-align: center;

margin-top: px;

width: 100%;

height: 20px;

}

.popup {

display:;

color: red;

font-weight: bold;

font-size: 40px;

position: fixed;

text-align: center;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.7);

z-index: 1;

}

**JAVASCRIPT CODE :**

var channelURL = https://thingspeak.com/channels/2314097/api\_keys=NYDK9XEUPYYONZQC

var temperatureField = document.getElementById('temperature');

var humidityField = document.getElementById('humidity');

var waterLevelField = document.getElementById('Waterlevel');

var waterLevelWarning = document.getElementById('waterLevelWarning');

fetch(channelURL)

.then(response => response.json())

.then(data => {

var latestEntry = data.feeds[data.feeds.length - 1];

var temperatureData = latestEntry.field1;

var humidityData = latestEntry.field2;

var waterLevelData = latestEntry.field3;

temperatureField.innerHTML = temperatureData + "&#8451;";

humidityField.innerHTML = humidityData + "%";

waterLevelField.innerHTML = waterLevelData;

function showWaterLevelPopup() {

if (waterLevelData > 250){

document.getElementById("popup").style.display = "block";

} else {

document.getElementById("popup").style.display = "none";

}

}

document.getElementById("closePopup").addEventListener("click", function() {

document.getElementById("popup").style.display = "none";

});

setInterval(showWaterLevelPopup, 5000);

if (waterLevelData > 250) {

waterLevelWarning.style.display = 'block';

} else {

waterLevelWarning.style.display = 'none';

}

})

.catch(error => {

console.error('Error fetching data:', error);

});

function initMap() {

var myLatLng = { lat: 12.9716, lng: 77.5946 };

var map = new google.maps.Map(document.getElementById('map'), {

zoom: 15, // Adjust the zoom level

center: myLatLng

});

var marker = new google.maps.Marker({

position: myLatLng,

map: map,

title: 'My Location'

});

}

var script = document.createElement('script');

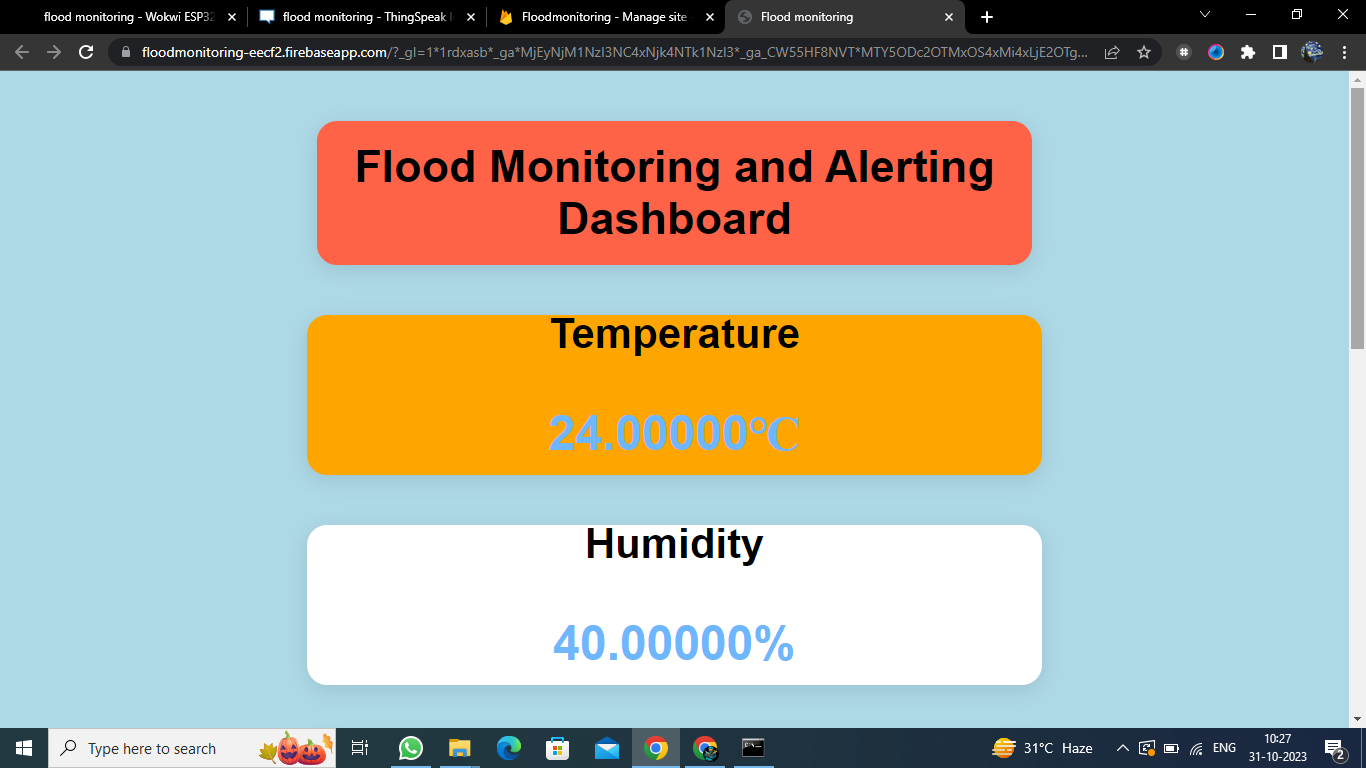
script.src = `https://maps.googleapis.com/maps/api/js?key=AIzaSyAZY0CFqdt41GgSFOl7fvtR-cyw-HY4gCU&callback=initMap`;

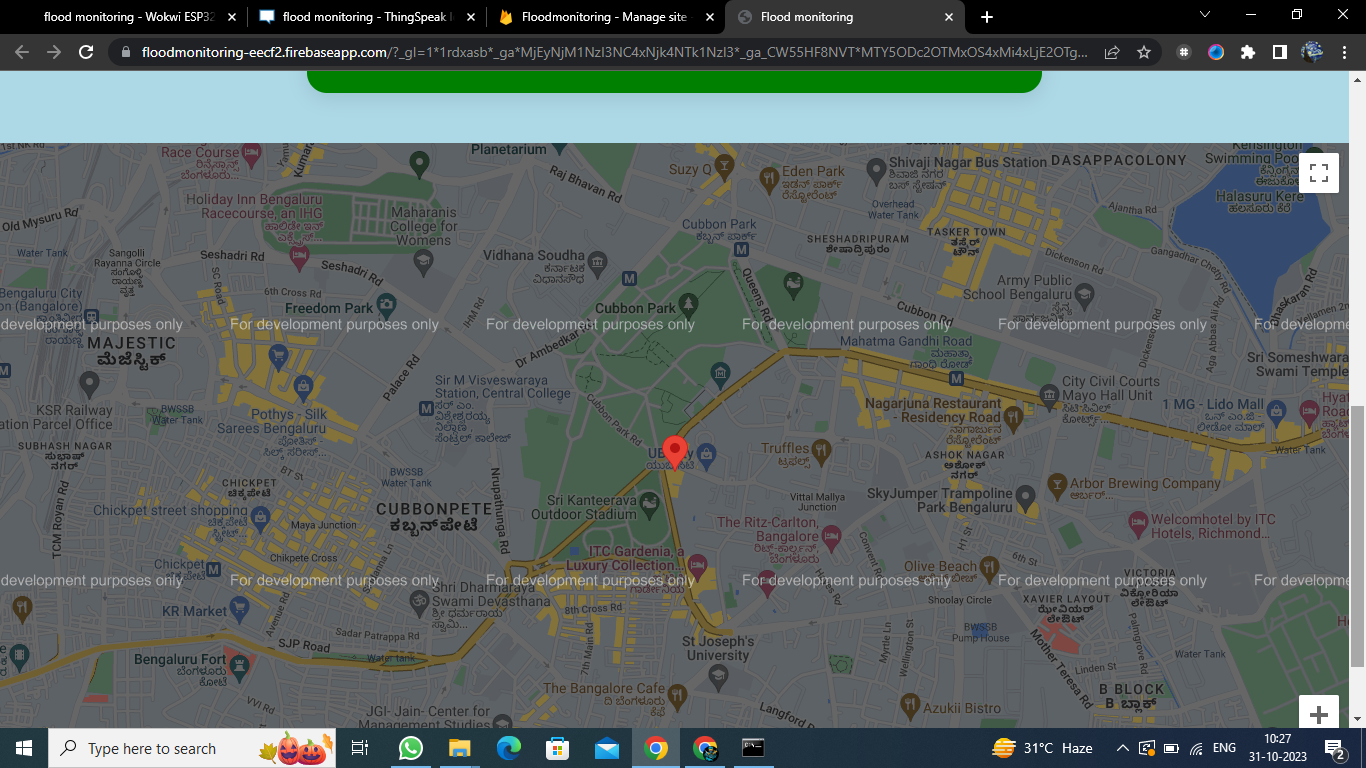
script.defer = true;

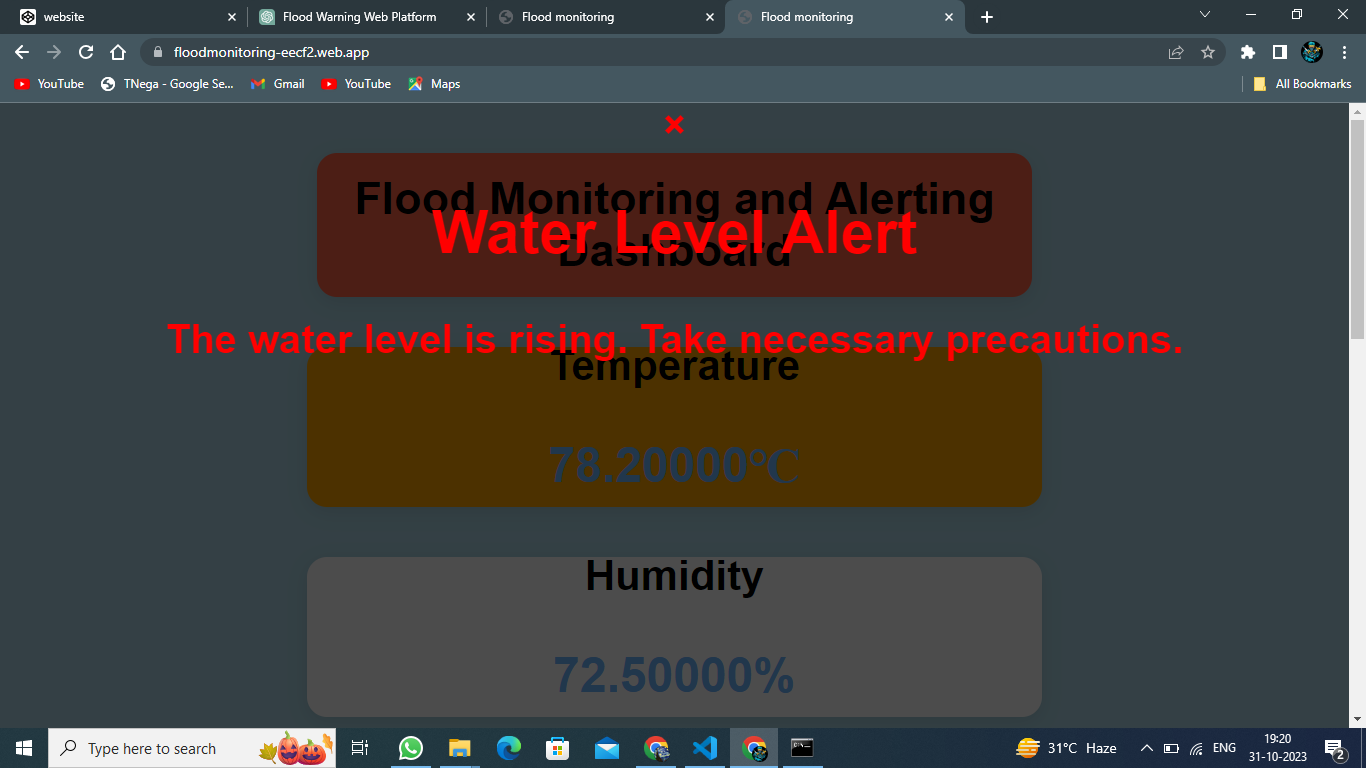
script.async = true;

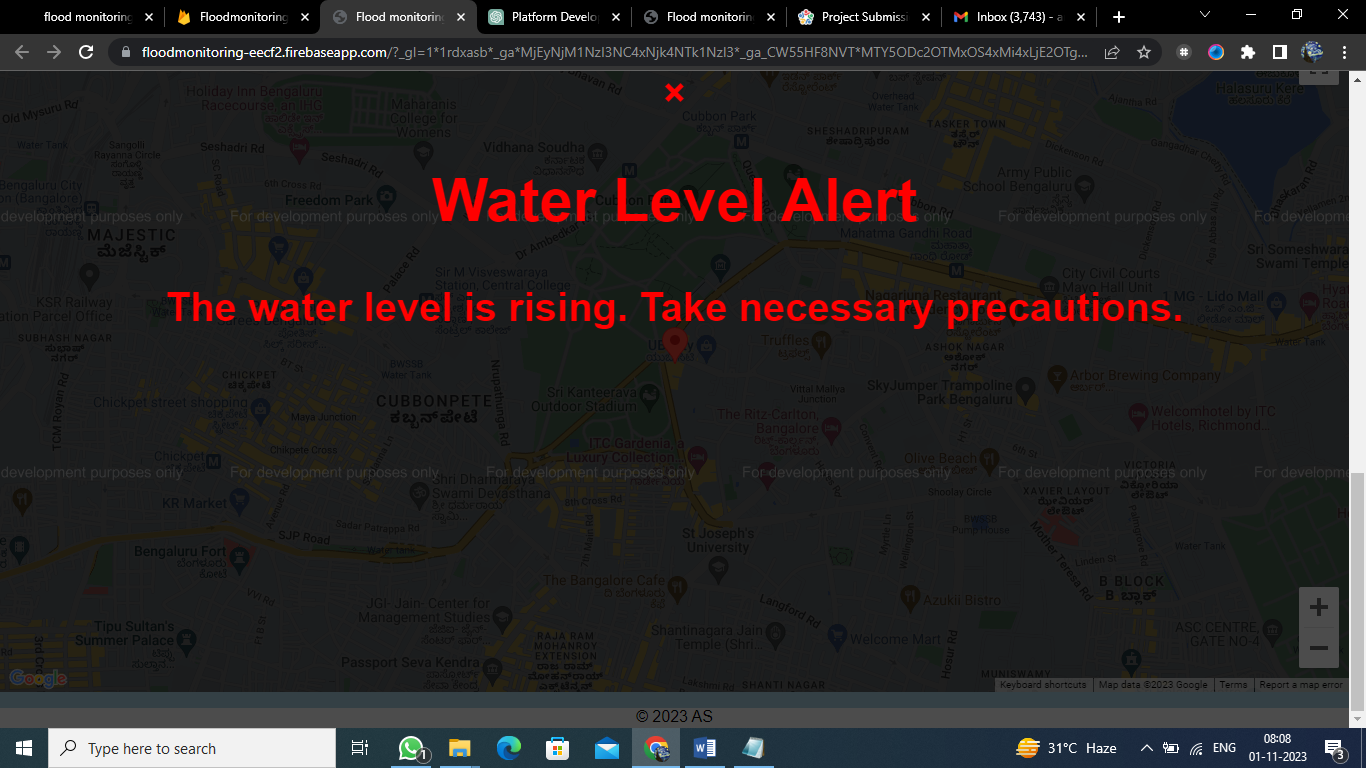
document.head.appendChild(script);

**OUTPUT IMAGES**

**:**

****

****

****

**Conclusion:**

* The development of the Flood Monitoring and Early Warning System has reached a critical milestone in Phase 4, where the focus has been on creating the early warning platform and implementing the warning system. This phase marks a significant step towards achieving the project's primary objectives, including real-time flood monitoring, early warning issuance, and public safety enhancement.
* The Early Warning Platform, designed using web development technologies such as HTML, CSS, and JavaScript, provides a user-friendly interface for accessing and visualizing real-time environmental data. Users can monitor critical information, including water levels, temperature, and humidity, and receive timely flood warnings.
* The Warning System Implementation plays a pivotal role in the system's functionality. It continuously monitors water levels and evaluates them against predefined criteria and thresholds. When flood risks are identified, the system issues warnings through a combination of visual alerts on the platform and, in some cases, audible alerts using sirens or alarms. Users can interact with the platform, acknowledging or dismissing warnings, and make informed decisions based on the information provided.
* The integration of these components and the early warning system represents a significant advancement in flood prevention and response. It empowers individuals and relevant authorities with accurate, up-to-date information, enabling them to take precautionary measures and coordinate emergency responses effectively.
* As the project moves forward, ongoing efforts will focus on testing, optimizing, and ensuring the system's reliability. Continuous monitoring, maintenance, and user training will be crucial to the long-term success of the Flood Monitoring and Early Warning System.

The successful development of the early warning platform and warning system underscores the potential of IoT solutions in environmental monitoring and disaster mitigation. The system serves as a valuable tool in enhancing public safety and supporting timely responses to flood emergencies.