FLOOD MONITORING AND EARLY WARNING SYSTEM

Project Objectives:

Develop a real-time flood monitoring and early warning system to enhance public safety and emergency response coordination in flood-prone areas.

Deploy IoT sensors to collect data on water levels, weather conditions, and rainfall in real time.

Create a central platform for data aggregation, analysis, and dissemination of early warnings.

Implement an alert system to notify relevant authorities and the public about potential flood events.

IoT Sensor Deployment:

Identify flood-prone areas and strategically place IoT sensors, including water level sensors, weather stations, and rain gauges.

Connect sensors to a network infrastructure for real-time data transmission.

Ensure the sensors are equipped with the necessary power source (e.g., solar panels, batteries) and communication capabilities (e.g., Wi-Fi, cellular).

Platform Development:

Data Aggregation: Collect data from IoT sensors, including water level, weather, and rainfall data.

Data Storage: Store data in a secure and scalable database for historical analysis.

Real-Time Processing: Implement real-time data processing to detect anomalies or trends that may indicate potential flooding.

Visualization: Create user-friendly dashboards for monitoring data in real time.

Early Warning System: Develop an algorithm to trigger early warnings based on predefined criteria.

Alert Mechanisms: Establish communication channels for sending alerts to relevant authorities and the public.

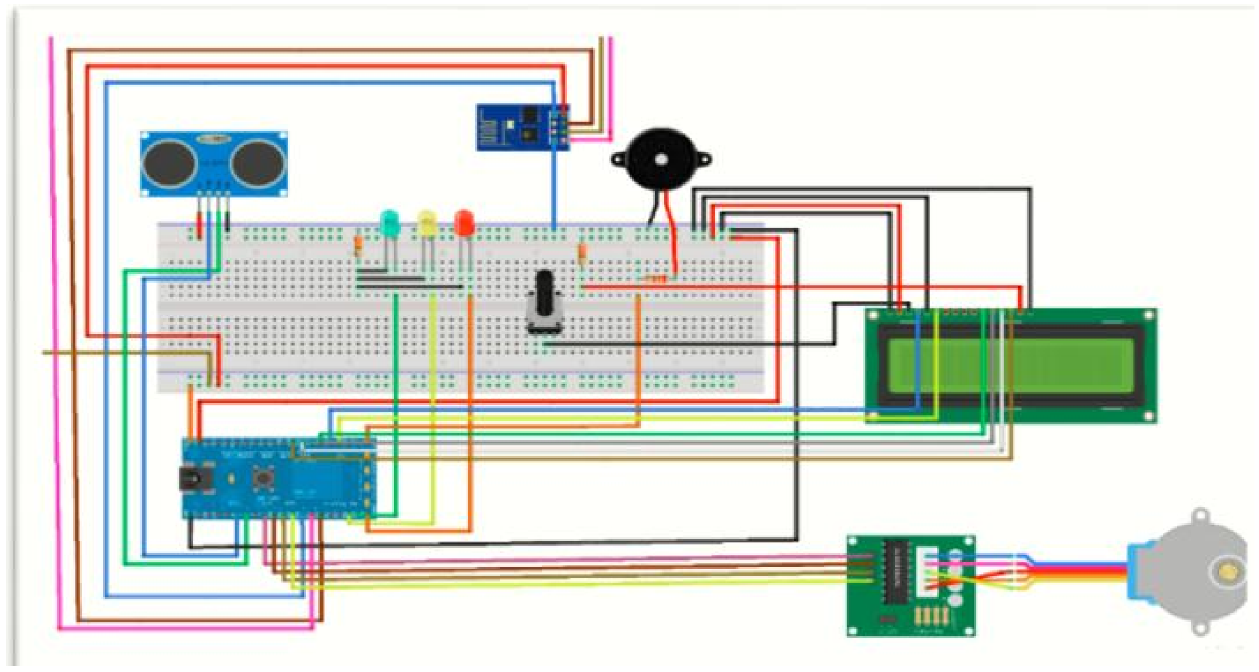
Code Implementation:

Utilize programming languages and frameworks suitable for data collection, storage, and real-time processing. Common choices include Python, Node.js, and databases like PostgreSQL.

Develop the IoT sensor code for data transmission, including error handling and data integrity checks.

Create web-based dashboards and user interfaces for data visualization using technologies such as HTML, CSS, and JavaScript.

Implement alert mechanisms through email, SMS, or other communication channels, and integrate them into the system.



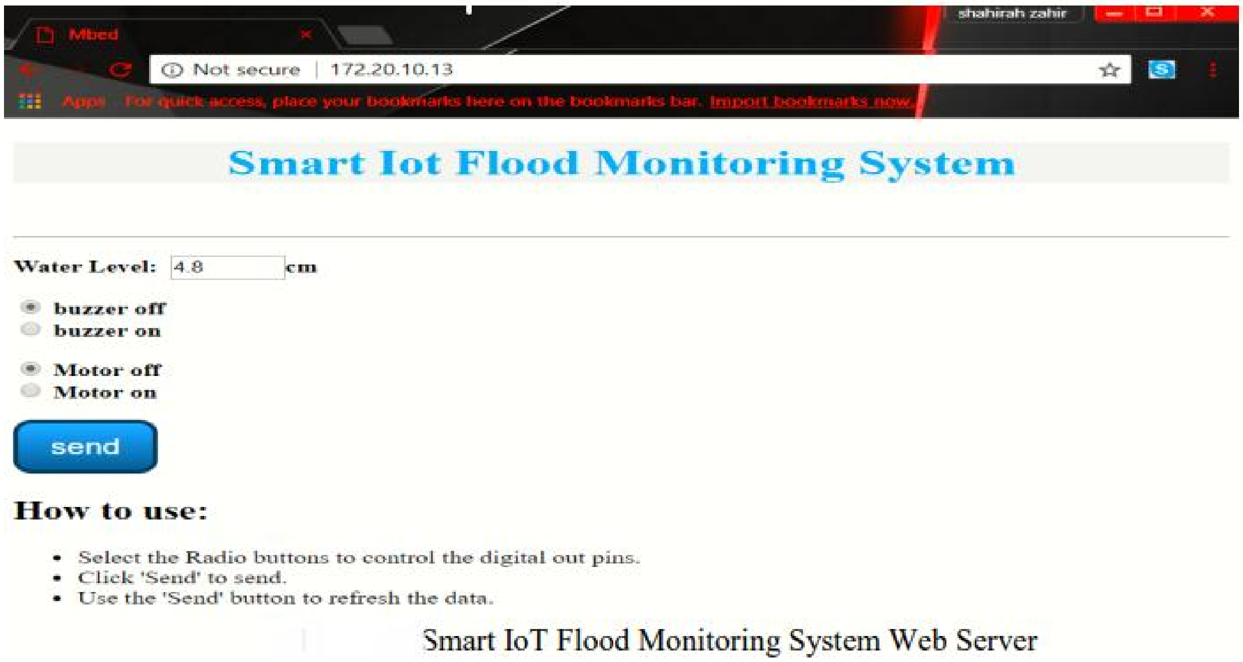
Audino code :

Certainly! Below is a simplified example of Arduino code for a basic IoT sensor that measures water levels. Please note that this is a simplified example, and a real-world implementation would be more complex and might use additional sensors and communication methods:

```
#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_ADS1015.h>

#include <ESP8266WiFi.h>
```

```
#include <ESP8266HTTPClient.h>


// WiFi credentials
const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";


// Adafruit ADS1115 Analog-to-Digital Converter
Adafruit_ADS1115 ads;


// Server URL to send data
const String serverUrl = "http://yourserver.com/endpoint";


void setup() {
  Serial.begin(115200);


  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");


  // Initialize the ADC

  ads.begin();


  // You can adjust the gain for your specific application

  ads.setGain(GAIN_ONE);


  // Other setup code for your sensor

}


void loop() {

  // Read water level from the sensor

  int16_t adc0 = ads.readADC_SingleEnded(0);


  // Convert the ADC reading to the actual water level

  // You need to calibrate this based on your sensor and setup


  // Send data to the server
  sendData(adc0);

  // Delay before taking the next reading
  delay(10000); // Adjust as needed
}

void sendData(int waterLevel) {
  // Create an HTTP client
  HTTPClient http;

  // Build the URL with the data to send
  String url = serverUrl + "?water_level=" + String(waterLevel);

  // Make a GET request to the server
  http.begin(url);

  int httpCode = http.GET();

  if (httpCode > 0) {
    // Successfully sent data
```

```
      Serial.println("Data sent successfully");
    } else {
      // Failed to send data
      Serial.println("Data sending failed");
    }

    // Close the connection
    http.end();
}
```


Mbed
Not secure | 172.20.10.13
shahirah zahir
Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now.

Smart Iot Flood Monitoring System

Water Level: 4.8 cm

● buzzer off
○ buzzer on

● Motor off
○ Motor on

send

How to use:

- Select the Radio buttons to control the digital out pins.
- Click 'Send' to send.
- Use the 'Send' button to refresh the data.

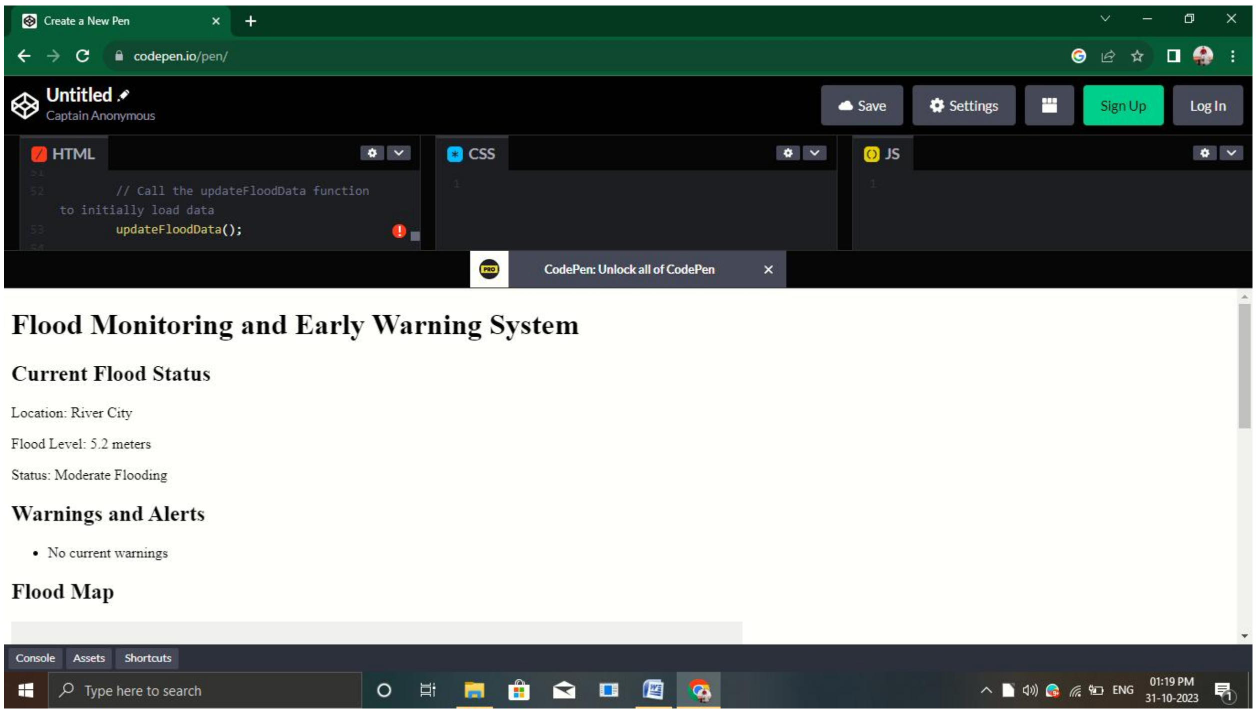
Smart IoT Flood Monitoring System Web Server

Html code for Flood monitoring and early warning system.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Flood Monitoring and Early Warning System</title>
  <style>
    /* Add your CSS styles here */
  </style>
</head>
<body>
  <header>
    <h1>Flood Monitoring and Early Warning System</h1>
  </header>

  <section id="status">
    <h2>Current Flood Status</h2>
    <p>Location: <span id="location">Unknown</span></p>
    <p>Flood Level: <span id="flood-level">N/A</span></p>
    <p>Status: <span id="status-text">No data
available</span></p>
```

```
    </section>


    <section id="warnings">
        <h2>Warnings and Alerts</h2>
        <ul>
            <li>No current warnings</li>
        </ul>
    </section>


    <section id="map">
        <h2>Flood Map</h2>
        <iframe src="https://www.example.com/flood-map"
width="800" height="600" frameborder="0"></iframe>
    </section>


    <footer>
        <p>&copy; 2023 Your Organization</p>
    </footer>
    <!-- Add your JavaScript code here to update the data dynamically -->
    <script>
        // Example JavaScript code to fetch and update flood data
        function updateFloodData() {
            // Simulated data (replace with real data fetching)
            const location = "River City";
            const floodLevel = 5.2;
            const status = "Moderate Flooding";

            document.getElementById("location").textContent = location;
            document.getElementById("flood-level").textContent = floodLevel + " meters";
            document.getElementById("status-text").textContent = status;
        }

        // Call the updateFloodData function to initially load data
        updateFloodData();
        // You should implement functions to fetch real data
periodically and update the page.
    </script>
</body>
</html>
```

Output:



```
        // Call the updateFloodData function
  to initially load data
        updateFloodData();
```

Flood Monitoring and Early Warning System

Current Flood Status

Location: River City

Flood Level: 5.2 meters

Status: Moderate Flooding

Warnings and Alerts

- No current warnings

Flood Map

Enhancing Public Safety and Emergency Response:

Explain how real-time data from IoT sensors can provide early warnings to authorities and the public.

Describe how flood predictions and early alerts can help in evacuations and resource allocation.

Discuss how the system can enhance coordination between emergency services and local communities during flood events.

This is a high-level outline of the project documentation. In practice, such documentation would include detailed technical specifications, code snippets, risk assessments, and more extensive details on the IoT sensor deployment, platform development, and its benefits to public safety and emergency response coordination. It is recommended to work with a team of experts in IoT, software development, and emergency management to create a comprehensive and effective system.