Phase 5: Project Documentation & Submission

Problem Statement:

The problem is creating an effective and integrated flood monitoring and earlywarning system for flood-prone regions. This system must incorporate IoT sensor technology for real-time data collection, employ accurate data analysis algorithms to predict potential floods and implement a responsive notification system to swiftly alert both the public and relevant authorities. The ultimate objective is to mitigate flood-related risks, reduce property damage, and save lives by significantly enhancing public safety and emergency response coordination in areas susceptible to flooding.

PROCEDURE:

Step 1: Define Requirements

Clarify project objectives and key features.

Step 2: Hardware Setup

Select IoT sensors.

Install sensors and ensure connectivity.

Step 3: Data Transmission and Collection

Define data transmission protocol.

Set up data processing on a server or cloud service.

Step 4: Data Storage and Management

Create a database for data storage.

Implement data cleansing and transformation.

Step 5: Real-time Data Analysis

Develop algorithms for real-time data analysis.

Step 6: API Development

Build APIs for mobile app data access.

Step 7: Mobile App Development

Design a user-friendly mobile app.

Integrate with IoT data.

Step 8: User Testing and Feedback

Thoroughly test the system.

Gather user feedback for improvements.

Step 9: Deployment and Maintenance

Deploy the system.

Implement maintenance and updates.

Step 10: Scalability and Optimization

Plan for system scalability.

Continuously optimize the system for better performance.

IOT REQUIRMENTS:

1. **IoT Sensors:**
    - Water Level Sensors: Deploy sensors that measure water levels in rivers, streams, and flood-prone areas.
    - Weather Sensors: Use sensors to collect data on rainfall, temperature, humidity, wind speed, and atmospheric pressure.
    - GPS Sensors: Implement GPS for accurate geographical location tracking of each sensor.
2. **Data Transmission:**
    - Wireless Communication: Utilize reliable and secure wireless communication protocols (e.g., Wi-Fi, LoRa, NB-IoT, or cellular) for transmitting data from sensors to the central platform.
    - Real-Time Data: Ensure real-time or near-real-time data transmission to enable timely flood monitoring and alerts.
3. **Power Supply:**

- Battery Backup: Equip sensors with battery backup systems to ensure continuous operation during power outages.
- Solar Panels: Incorporate solar panels to recharge batteries and extend sensor lifespan.

4. **Data Quality and Accuracy:**
    - Data Validation: Implement data validation and quality control mechanisms to filter out erroneous data.
    - Calibration: Regularly calibrate sensors to maintain data accuracy.
5. **Scalability:**
    - Design the system to be scalable, allowing for the addition of more sensors as the network expands.
6. **Security:**
    - Secure Data Transmission: Encrypt data in transit to protect it from interception.
    - Authentication: Implement authentication mechanisms to ensure that only authorized devices can connect to the network.
7. **Data Storage:**
    - Choose a robust and scalable database system (e.g., MySQL or NoSQL) for storing the **collected sensor data.**
8. **Data Analysis:**
    - Develop data analysis algorithms that can process incoming data to detect patterns and predict potential floods.
9. **Early Warning System:**
    - Implement a notification system that generates alerts for the public and relevant authorities when flood risks are detected.
    - Utilize multiple communication channels (e.g., SMS, email, mobile apps) to ensure alerts reach a wide audience.
10. **User Interface:**
    - Create a user-friendly web-based interface for users to access sensor data, analysis results, and early warnings.
11. **Redundancy and Reliability:**
    - Include redundancy in the system's components and communication channels to ensure reliability.
    - Implement failover mechanisms to maintain operation during system failures.

12.Regulatory Compliance:

- Ensure compliance with local, state, and national regulations and standards related to data privacy, safety, and emergency notification.

13.Maintenance and Support:

- Plan for regular maintenance and support to keep the IoT sensors and the entire system in good working order.

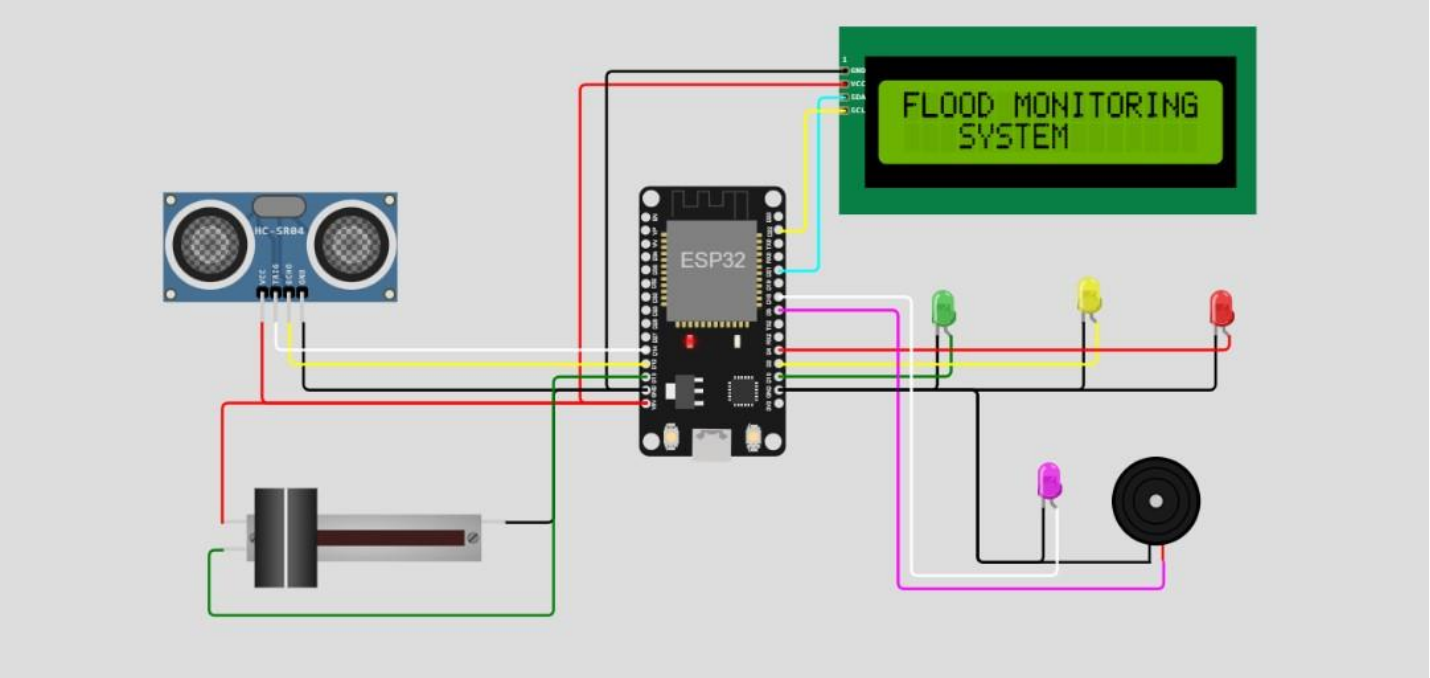
CODE:

```
#define BLYNK_TEMPLATE_ID "TMPL3ycX9ZWry"
#define BLYNK_TEMPLATE_NAME "FLOOD LEVEL MONITOR"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address for 16x2 LCD
const int triggerPin = 14; // Ultrasonic sensor trigger pin
const int echoPin = 12; // Ultrasonic sensor echo pin
const int buzzerPin = 5; // Buzzer pin
const int greenLED = 15; // Green LED pin
const int yellowLED = 2; // Yellow LED pin
const int redLED = 4;
const int LED = 18;
const int floatSwitch = 13; // Red LED pin
const int maxDistance = 400; // Maximum range of the ultrasonic sensor in cm
const int numParts = 3; // Number of parts to divide the range into
const int partDistance = maxDistance / numParts; // Distance for each part
int moodScore = 0;
BlynkTimer timer;
// Enter your Auth token
char auth[] = "N4fbX3x6_eTd04YjIyPR5Lvi4XYr5ugj";
char ssid[] = "Wokwi-GUEST";
char pass[] = "";
void setup() {
Serial.begin(6000);
Blynk.begin(auth, ssid, pass);
Wire.begin(21, 22);
lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print("FLOOD MONITORING");
lcd.setCursor(2, 1);
lcd.print(" SYSTEM");
delay(3000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("PEC(ECE)");
lcd.setCursor(2, 1);
lcd.print("(2021-2025)");
delay(3000);
lcd.clear();
```

```
pinMode(triggerPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(buzzerPin, OUTPUT);
pinMode(greenLED, OUTPUT);
pinMode(yellowLED, OUTPUT);
pinMode(redLED, OUTPUT);
pinMode(LED, OUTPUT);
}
void ultrasonic() {
long duration, distance;
// Trigger the ultrasonic sensor
digitalWrite(triggerPin, LOW);
delayMicroseconds(2);
digitalWrite(triggerPin, HIGH);
delayMicroseconds(10);
digitalWrite(triggerPin, LOW);
// Measure the time it takes for the pulse to return
duration = pulseIn(echoPin, HIGH);
// Calculate the distance in centimeters
distance = (duration / 2) / 29.1;
int blynkDistance = ((distance - maxDistance) * -1) / 3;
if (distance <= maxDistance) {
Blynk.virtualWrite(V0, blynkDistance);
} else {
Blynk.virtualWrite(V0, 0);
}
// Determine the water level description
String waterLevel = "Unknown";
if (distance < partDistance) {
waterLevel = "RED ALERT";
digitalWrite(greenLED, LOW);
digitalWrite(yellowLED, LOW);
digitalWrite(redLED, HIGH);
digitalWrite(buzzerPin, HIGH);
delay(2000);
digitalWrite(buzzerPin, LOW);
delay(2000);
digitalWrite(buzzerPin, HIGH);
delay(2000);
digitalWrite(buzzerPin, LOW);
delay(2000);
} else if (distance < partDistance * 2) {
waterLevel = "DANGER ";
digitalWrite(greenLED, LOW);
digitalWrite(yellowLED, HIGH);
digitalWrite(redLED, LOW);
digitalWrite(buzzerPin, LOW);
delay(3000);
digitalWrite(buzzerPin, LOW);
delay(3000);
}
else {
waterLevel = "NORMAL";
digitalWrite(greenLED, HIGH);
```

```
digitalWrite(yellowLED, LOW);
digitalWrite(redLED, LOW);
digitalWrite(buzzerPin, LOW);
delay(6000);
}
// Display the water level and distance on the LCD with proper formatting
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("ALERT: ");
lcd.print(waterLevel);
lcd.setCursor(2, 1);
lcd.print("LEVEL: ");
lcd.print(distance);
lcd.print(" cm");
int potValue = analogRead(floatSwitch);
// Map the potentiometer value (0-1023) to the moodScore range (0-10)
moodScore = map(potValue, 0, 1023, 0, 10);
// Print the current moodScore to the serial monitor
Serial.print("Mood Score: ");
Serial.println(moodScore);
// Check if moodScore is below 5
if (moodScore < 5) {
// Turn on the LED
digitalWrite(LED, HIGH);
// Activate the buzzer
tone(buzzerPin, 1000); // You can change the frequency as needed
// Display "Diffuser On" on LCD
} else {
// Turn off the LED
digitalWrite(LED, LOW);
// Deactivate the buzzer
noTone(buzzerPin);
// Display "Diffuser Off" on LCD
}
// Add your additional logic here based on moodScore
// For example, you can send commands to other actuators
}
void loop() {
ultrasonic();
Blynk.run(); // Run the Blynk library
}
```

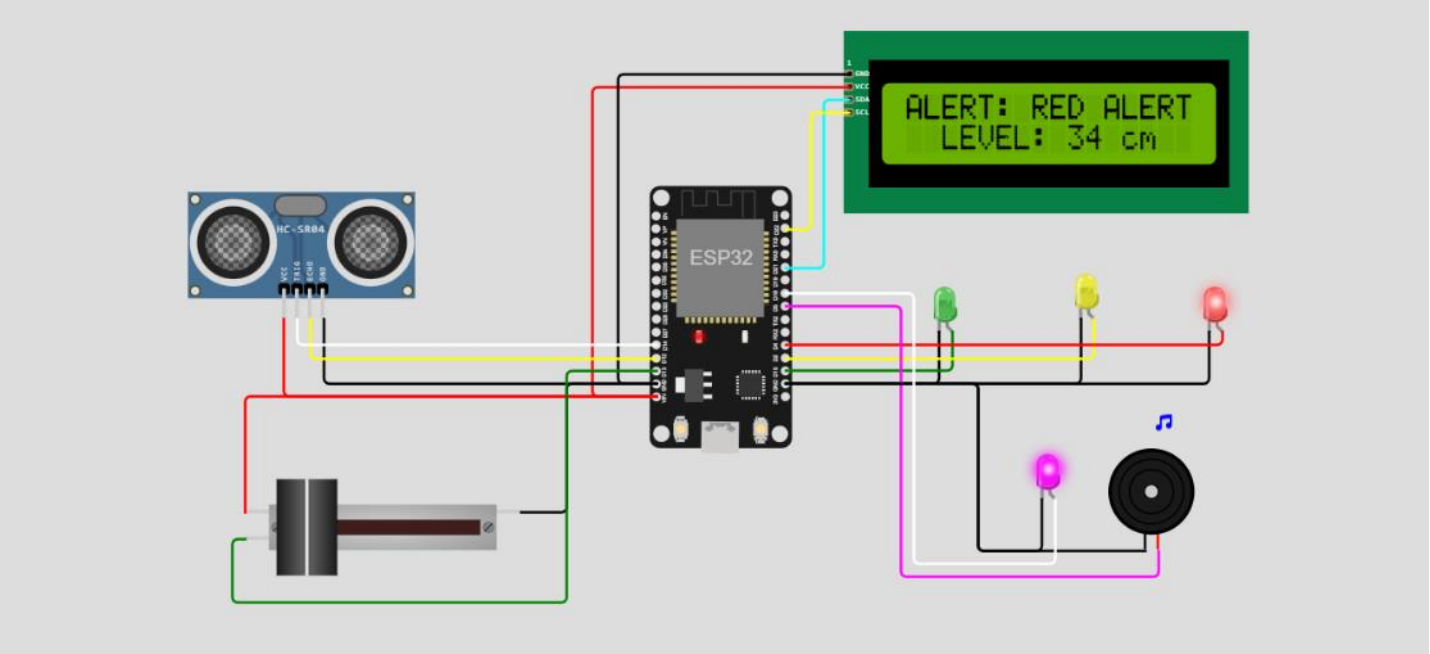
PROJECT SCREENSHOTS:



Picture of flood monitoring and early warning device

FLOOD MONITORING DEVICE

Output of our project :



CONCLUSION:

In conclusion, Flood Monitoring and Early Warning systems are indispensable tools in mitigating the devastating impacts of floods. These innovative solutions leverage technology, data, and community engagement to enhance public safety, minimize property damage, and foster long-term resilience in the face of climate-related disasters. By providing timely alerts and promoting preparedness, these systems not only save lives but also serve as cornerstones for sustainable development. As the world faces increasing climate uncertainties, the continued advancement and implementation of these systems are imperative for the well-being and security of communities worldwide.