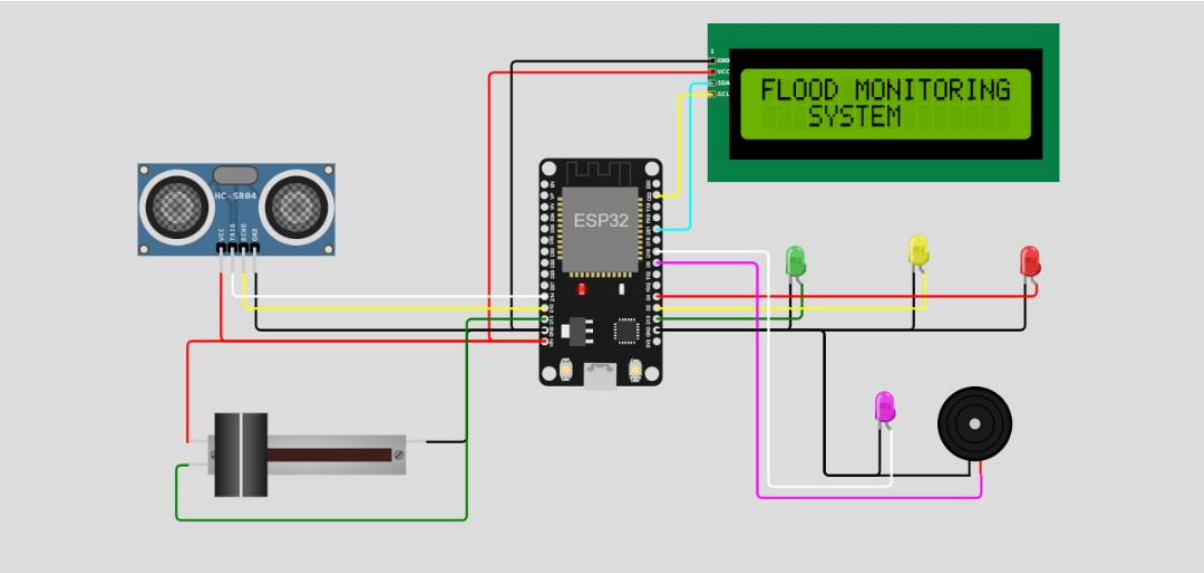
FLOOD MONITORING AND EARLY WARNING SYSTEM

CIRCUIT DIAGRAM:



CODING:

Here we have used C++ code for an ESP 32 microcontroller which easily controls all the required sensors and integrates with the IoT buzzer to alert them with its loud sound.

```
#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;
```

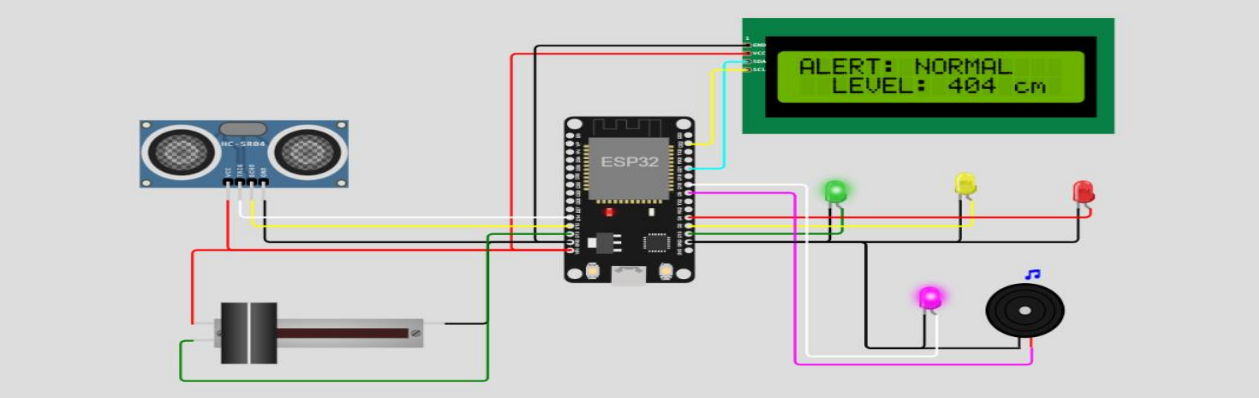
```
const int floatSwitch = 13;    // Red LED pin

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);
}
```

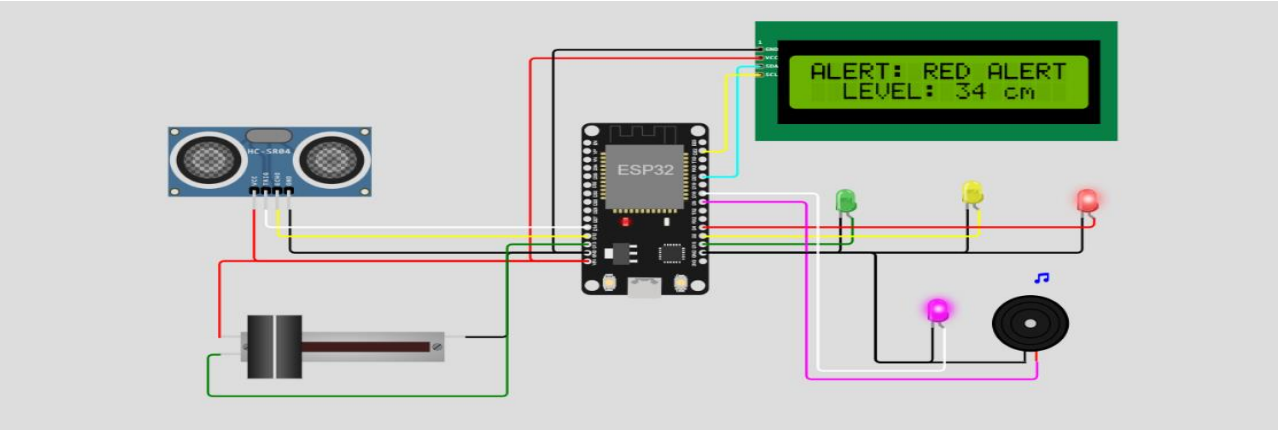
```
 else {
  waterLevel = "NORMAL";
  digitalWrite(greenLED, HIGH);
  digitalWrite(yellowLED, LOW);
  digitalWrite(redLED, LOW);
  digitalWrite(buzzerPin, LOW);
  delay(6000);

}

// 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
}
```

OUTPUT :



NORMAL ALERT



RED ALERT