Team: Vishwaa, Akil & Renukanthan

IoT-Based Flood Early Warning System

Overview

The IoT-based flood early warning system aims to monitor environmental factors such as humidity, temperature, soil moisture, and water level to predict and provide early warnings about potential flooding events. The system uses an ESP32 microcontroller along with a DHT11 sensor, a soil moisture sensor, and a water level sensor. Data from these sensors is sent to a mobile application created using Blynk to provide real-time monitoring and alerts to users.

Components Used:

- **ESP32 Microcontroller:** The ESP32 serves as the main control unit for the project. It is chosen due to its WiFi capabilities, which are crucial for sending sensor data to the cloud and interacting with the Blynk app.
- **DHT11 Sensor:** This sensor is used to measure the temperature and humidity of the surrounding environment. Changes in these factors can indicate weather conditions that may contribute to flooding.
- **Water Level Sensor:** This sensor helps to determine the water level in a specific area. It is an essential component for assessing the risk of flooding.
- **Soil Moisture Sensor:** The soil moisture sensor is used to measure the moisture level in the soil, helping determine how saturated the ground is. High soil saturation can indicate increased chances of flooding, especially during heavy rains.
- **Blynk App:** The Blynk app is used to provide a user-friendly interface to monitor sensor data. Users receive real-time updates and alerts on their mobile devices, allowing them to respond promptly to changing environmental conditions.

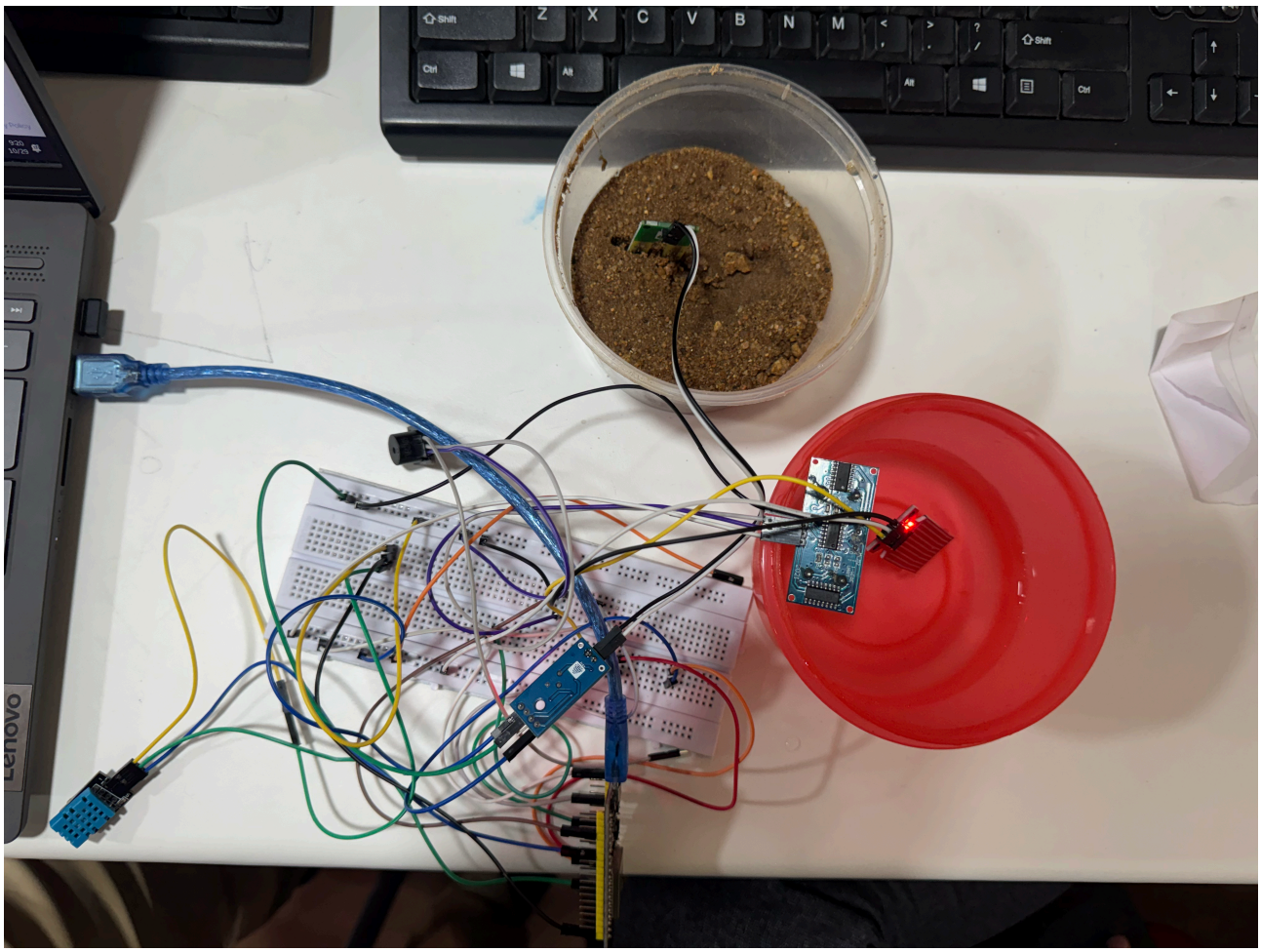
Features and Functionality:

1. **Real-time Monitoring:** Sensor data, including temperature, humidity, soil moisture, and water level, are continuously monitored and sent to the ESP32. The ESP32, in turn, uploads the data to the Blynk platform.

2. **Alerts and Notifications:** Thresholds are set for various parameters such as water level and soil moisture. When these thresholds are exceeded, the system sends alerts via the Blynk app, notifying users of potential flood risks.

3. **Remote Access:** The use of the Blynk app enables remote monitoring of the system. Users can track environmental data from anywhere, provided they have internet access.

Image:



Programming:

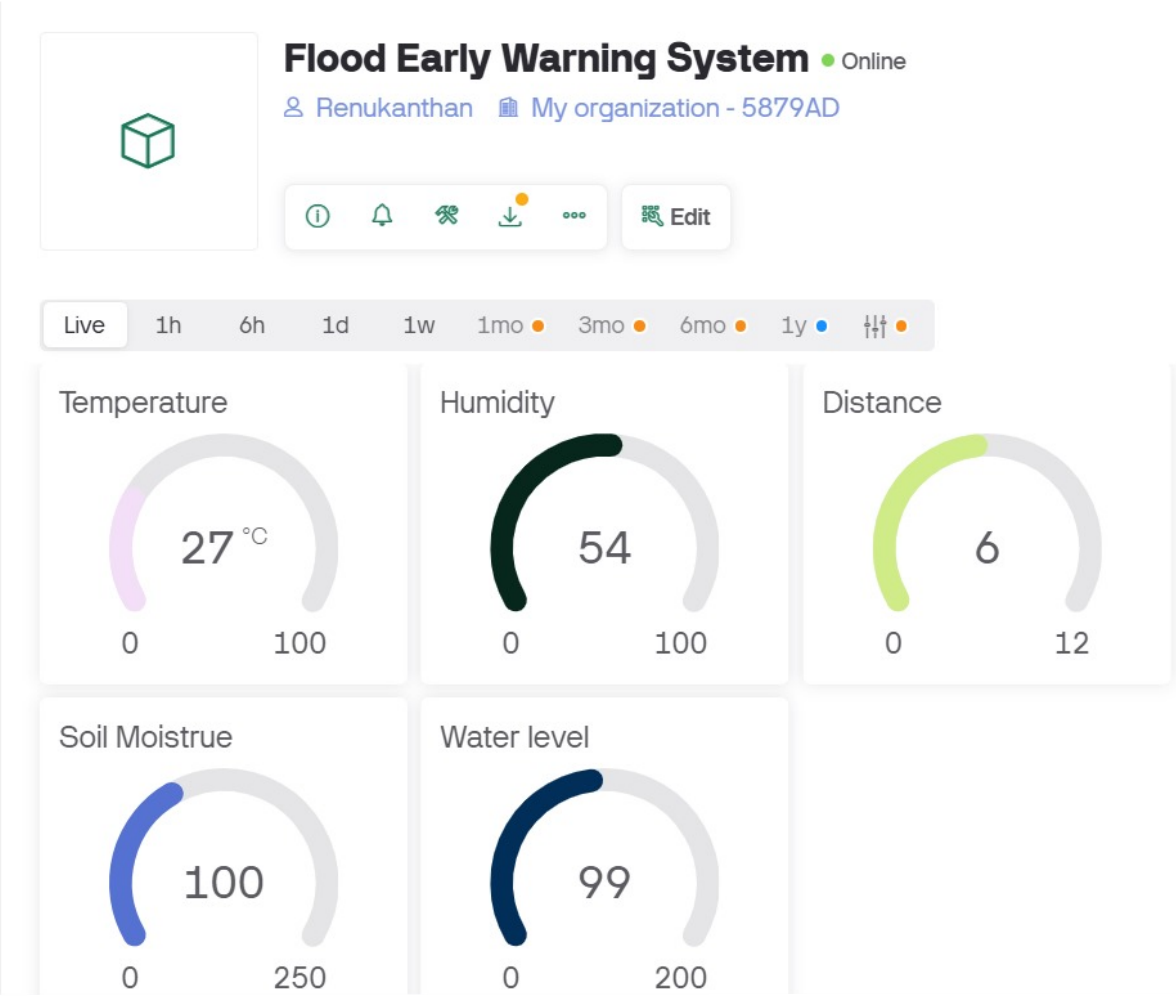
```
#define BLYNK_TEMPLATE_ID "TMPL3V_toA18k"

#define BLYNK_TEMPLATE_NAME "Flood Early Warning System"

#define BLYNK_PRINT Serial

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#include <DHT.h>

#include <NewPing.h>

// Blynk auth token and Wi-Fi credentials

char auth[] = "OPEHoEG9nCAekkyM8Vg6n-vtD8uy9e4L";

char ssid[] = "Airtel_Renukanthan";

char pass[] = "Renu@0987";

// Pin definitions

#define TRIG_PIN 15

#define ECHO_PIN 13

#define DHT_PIN 32

#define SOIL_PIN 34

#define WATER_PIN 35

#define BUZZER_PIN 27

// Ultrasonic and DHT setup

#define MAX_DISTANCE 200
```

```
NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);

#define DHTTYPE DHT11

DHT dht(DHT_PIN, DHTTYPE);

void setup() {

  Serial.begin(115200);

  // Start Blynk and DHT sensor

  Blynk.begin(auth, ssid, pass);

  dht.begin();

  // Set up buzzer as output

  pinMode(BUZZER_PIN, OUTPUT);

}

void loop() {

  Blynk.run(); // Run Blynk

  readAndSendSensorData(); // Collect and send sensor data

}

void readAndSendSensorData() {

  // Ultrasonic Sensor for water level

  int distance = sonar.ping_cm();

  Serial.print("Water Level (Ultrasonic): ");

  Serial.println(distance);

  Blynk.virtualWrite(V1, distance); // Send to Blynk
```

```
// Check water level threshold
if (distance > 0 && distance < 20) { // Ensure valid distance and check threshold
  tone(BUZZER_PIN, 1000); // Turn on buzzer
} else {
  noTone(BUZZER_PIN); // Turn off buzzer
}
// Soil Moisture Sensor
int soilMoisture = analogRead(SOIL_PIN);
soilMoisture = map(soilMoisture, 0, 4095, 0, 100); // Map to percentage
Serial.print("Soil Moisture: ");
Serial.println(soilMoisture);
Blynk.virtualWrite(V2, soilMoisture);
// Water Level Sensor
int waterLevel = analogRead(WATER_PIN);
waterLevel = map(waterLevel, 0, 4095, 0, 100); // Map to percentage
Serial.print("Water Sensor Level: ");
Serial.println(waterLevel);
Blynk.virtualWrite(V3, waterLevel);
// DHT22 Temperature & Humidity
float temp = dht.readTemperature();
float humidity = dht.readHumidity();
```

```
  // Check if DHT sensor values are valid
  if (!isnan(temp) && !isnan(humidity)) {
    Serial.print("Temperature: ");
    Serial.print(temp);
    Serial.print(" °C, Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
    Blynk.virtualWrite(V4, temp); // Temperature to Blynk
    Blynk.virtualWrite(V5, humidity); // Humidity to Blynk
  } else {
    Serial.println("Failed to read from DHT sensor!");
  }
  delay(2000); // Delay between readings
}
```

Output Dashboard:



Strengths of the Project:

- **Early Warning Capability:** By combining data from multiple sensors, the system provides a more comprehensive assessment of potential flood risks.
- **User-friendly Interface:** The Blynk app makes it easy for users to monitor sensor readings and receive alerts, improving overall usability.
- **Scalability:** The system can be scaled up to include additional sensors for better accuracy or adapted for use in larger areas.
- **Low Cost:** The components used are relatively inexpensive, making the project a cost-effective solution for early flood detection.

Areas for Improvement:

- **Power Supply Considerations:** The ESP32 and sensors require a stable power source. Adding a battery backup system could enhance reliability, especially during power outages.
- **Accuracy of Sensors:** The DHT11 sensor has a lower accuracy compared to more advanced sensors like the DHT22. Upgrading the sensor could improve temperature and humidity readings.
- **Communication Robustness:** WiFi connectivity can be disrupted during extreme weather events. Implementing a secondary communication method, such as LoRaWAN, could increase system reliability.

Future Enhancements:

- **Integration with Weather APIs:** Adding data from online weather APIs could improve the accuracy of flood predictions.
- **Advanced Analytics:** Use of machine learning models to analyze data trends from sensors for better prediction of potential flood events.
- **SMS Alerts:** In addition to the Blynk app, implementing SMS notifications would make the system more robust and ensure alerts reach users even if the app is inaccessible.

Conclusion

The IoT-based flood early warning system effectively combines multiple sensors to monitor environmental conditions and provide timely alerts to users. By leveraging the ESP32 and the Blynk app, the system offers a practical solution for flood preparedness. While there is room for enhancement in terms of sensor accuracy, communication robustness, and additional alert mechanisms, the project provides a solid foundation for further development and real-world implementation.