Department of Electronics & Telecommunication Engineering University of Moratuwa

EN3251 Internet of Things

IoT-Based Flood Monitoring and Early Warning System

Project Proposal

De Zoysa A.K.N.	210108C
Dimagi D.H.P	210131N
Dissanayaka D.M.P.C	210140P

Date - 2024.10.24

Contents

1 Introduction **2**

2 System Description **3**
2.1 End-Devices 3
2.2 Management Interface 4
2.3 User Interactions 4

3 Functional Description and System Design **5**

4 Architecture **6**

5 Codes **7**

6 References **10**

1 Introduction

The increasing threat of floods due to climate change and unpredictable weather patterns has raised the demand for advanced monitoring and early-warning systems. Floods can cause widespread damage, affecting human lives, infrastructure, and the environment. To mitigate these risks, timely detection and warnings are essential.

This project proposes an IoT-based Flood Monitoring System, which integrates ultrasonic sensors and DHT22 sensors to monitor water levels, temperature, and humidity in real-time. By leveraging the power of MQTT and ThingSpeak, the system enables continuous data collection, cloud storage, and remote access for users, along with early notifications through SMS alerts. The use of Node-RED for real-time visualization offers an intuitive interface, making flood monitoring more accessible and effective.

Our design also includes a rechargeable battery-powered system to ensure uninterrupted operation during power outages, enhancing the system's reliability. This solution not only provides an automated, scalable method for flood detection but also allows proactive measures by sending alerts and visualizing critical data on a web dashboard.

2 System Description

2.1 End-Devices

The flood monitoring system consists of two ultrasonic sensors to measure water levels at different locations and a DHT22 sensor to capture temperature and humidity data. An ESP8266 (NodeMCU) is used to collect and transmit the data via MQTT to the ThingSpeak cloud, where it is stored and analyzed. The system is managed via a Node-RED Dashboard for real-time visualization, and SMS notifications are sent through Twilio to alert users when flood risks are detected. A rechargeable battery ensures continuous operation during power outages.

- **Ultrasonic Sensors:** These sensors are placed in two locations to measure the water level. They work by sending sound waves and measuring how long they take to bounce back from the water surface. This gives the distance to the water, indicating its level.

Figure 1: Ultrasonic Sensor

- **DHT22 Sensor:** This sensor measures both temperature and humidity in the environment. It's useful for understanding weather conditions that might contribute to floods.

Figure 2: DHT22 Sensor

- **ESP8266 (NodeMCU):** This is the "brain" of the system. It collects data from the sensors and sends it to the cloud using the MQTT communication protocol. It connects to the internet via Wi-Fi.

Figure 3: Node MCU

- **Power Supply with Rechargeable Battery:** The system is designed to keep running even during power cuts. It automatically switches to battery power when there's no AC power, ensuring continuous operation.

2.2 Management Interface

The Management Interface in an IoT system provides users with a platform to monitor, control, and interact with the end-devices in real-time. In our project, the management interface includes a Node-RED Dashboard for visualizing sensor data like water levels, temperature, and humidity. It offers a web-based interface that users can access on computers or mobile devices. The system also integrates with ThingSpeak for data storage and analysis, and triggers alerts or notifications when certain conditions are met, enhancing user interaction and decision-making during flood risks.

- **Node-RED Dashboard:** A web-based platform where users can see real-time data. The data from the sensors is displayed visually so that users can monitor flood conditions remotely through their computer or smartphone.
- **ThingSpeak Cloud:** This is where the data from the sensors is stored and analyzed. ThingSpeak uses MQTT to receive the data and triggers alerts based on predefined conditions (e.g., when water levels reach a certain threshold).
- **SMS Notifications via Twilio:** The system sends SMS alerts to people in the area when the water levels exceed a certain threshold. This ensures people are warned of potential floods in real-time.

2.3 User Interactions

User Interactions in our project refer to how users can engage with and monitor the system. Users interact with the flood monitoring system through the Node-RED Dashboard, where they can view real-time data on water levels, temperature, and humidity. Additionally, they receive automatic SMS notifications when flood thresholds are exceeded, ensuring timely alerts. The interface allows users to remotely track the status of end-devices and receive updates on potential flood risks, helping them take action when necessary.

- **Node-RED Dashboard:** Users can access this dashboard through a browser on their computer or phone to monitor the system and check flood risks in real-time.
- **ThingSpeak API:** The cloud provides the capability for users to access and analyze sensor data. Based on the data, automated actions (like sending notifications) can be triggered.

3 Functional Description and System Design

Step	Description
1. Data Collection	• Ultrasonic sensors measure water levels. • DHT22 sensor records temperature and humidity.
2. Data Transmission	• ESP8266 (NodeMCU) collects sensor data. • Publishes data to MQTT broker. • Data sent to ThingSpeak for storage and processing. • Node-RED Dashboard displays the data.
3. Data Processing and Display	• Node-RED processes incoming data for real-time monitoring. • Sends SMS notification if water levels exceed the threshold.
4. Power Management	• Rechargeable battery for continuous operation. • Automatic switch from AC to battery power during outages.

Figure 4: Functional Workflow

4 Architecture

The architecture of the flood monitoring system defines how data flows between the end-devices (sensors), microcontroller (ESP8266), cloud platform (ThingSpeak), and the user interface (Node-RED Dashboard). End-devices collect environmental data, which is transmitted by the ESP8266 using the MQTT protocol to the ThingSpeak cloud for storage and processing. The Node-RED Dashboard displays this data in real-time. Additionally, the architecture includes an SMS notification system that sends alerts when flood risk thresholds are exceeded, ensuring users are informed promptly.

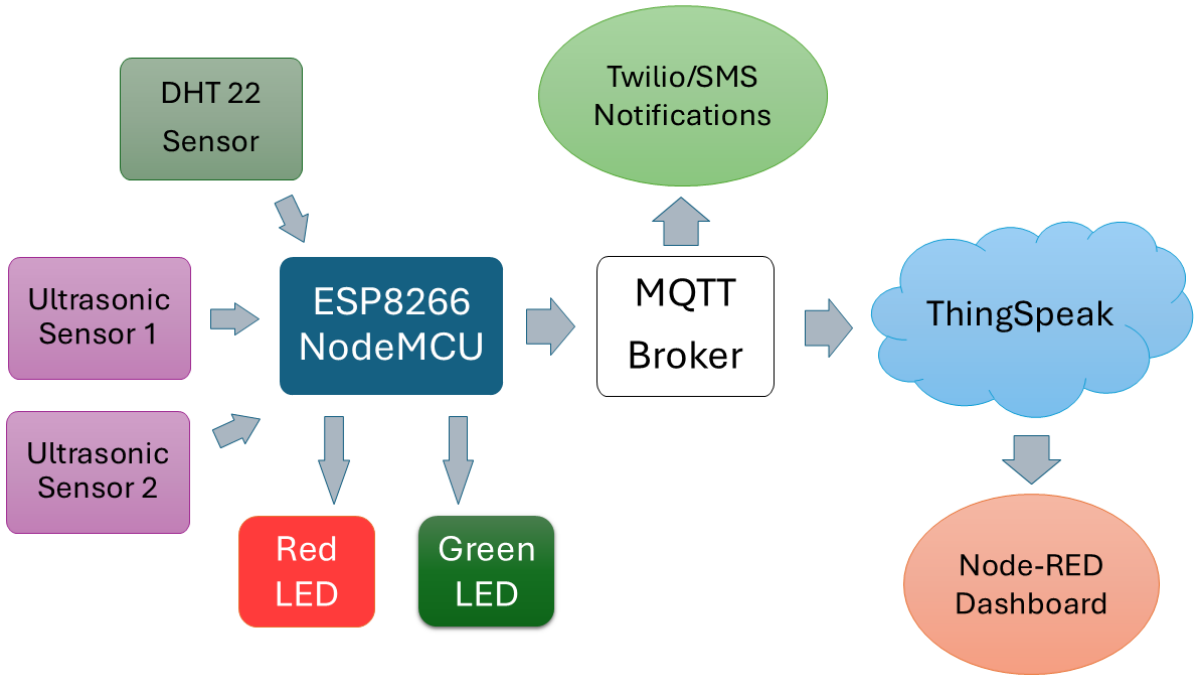

Figure 5: System Architecture

5 Codes

```
#include "DHT.h"      // download dhtlib from Manage Libraries and include in into this
#define DHTTYPE DHT11
#include <SPI.h>
#include <ThingSpeak.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <WiFiClientSecure.h>  // For HTTPS requests
#include <PubSubClient.h>      // Add MQTT library

#define dht_data_pin D2  // connect data pin of DHT to D2 pin
#define buzzer_pin D1    // connect input pin to D1 of nodemcu

const char* ssid = "Wifi_SSID";
const char* password = "Wifi_Password";

unsigned long channel_number = xxxxxxxxxxxxx;
const char* writeapikey = "xxxxxxxxxxxxxxxxxxx";

// Twilio credentials
const char* twilio_account_sid = "your_twilio_account_sid";  // Replace with your Twilio Account SID
const char* twilio_auth_token = "your_twilio_auth_token";    // Replace with your Twilio Auth Token
const char* twilio_phone_number = "xxxxxxxxxx";             // Replace with your Twilio phone number
const char* user_phone_number = "xxxxxxxxxx";               // Replace with the user's phone number

// MQTT credentials
const char* mqtt_server = "mqtt.example.com";  // Replace with your MQTT broker address
const int mqtt_port = 1883;
const char* mqtt_user = "your_mqtt_username";  // Replace with your MQTT username
const char* mqtt_password = "your_mqtt_password";  // Replace with your MQTT password

WiFiClient espClient;
WiFiClientSecure twilio_client;   // Secure client for Twilio API
PubSubClient mqtt_client(espClient);

DHT dht(dht_data_pin, DHTTYPE);

const int trigP = D5;
const int echoP = D6;

long duration;
int water_level;
int msgval;

// MQTT topic
const char* topic_humidity = "home/sensors/humidity";
const char* topic_temperature = "home/sensors/temperature";
const char* topic_water_level = "home/sensors/waterlevel";
const char* topic_alert = "home/alerts";

void setup() {
  // Initializing Ultrasonic sensor
  pinMode(trigP, OUTPUT);
  pinMode(echoP, INPUT);

  // Initializing DHT sensor
  dht.begin();
  delay(700);

  // Initializing Buzzer
  pinMode(buzzer_pin, OUTPUT);
  Serial.begin(115200);

  // Connect to WiFi and MQTT
  connect_to_wifi();
  mqtt_client.setServer(mqtt_server, mqtt_port);

  ThingSpeak.begin(espClient);

  // Set up Twilio client
  twilio_client.setInsecure();  // Skip certificate validation (ESP8266-specific)
}

void loop() {
  if (!mqtt_client.connected()) {
```

```
    reconnect_mqtt();  // Reconnect to MQTT broker if disconnected
  }
  mqtt_client.loop();

  // Start the DHT sensor
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  delay(1000);
  digitalWrite(trigP, LOW);
  delayMicroseconds(2);
  digitalWrite(trigP, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigP, LOW);

  duration = pulseIn(echoP, HIGH);
  water_level = duration * 0.034 / 2;

  Serial.println("Humidity is :");
  Serial.print(humidity);
  Serial.println("Temperature is :");
  Serial.print(temperature);
  Serial.println("Water level is :");
  Serial.print(water_level);

  if (water_level < 5 && humidity < 30 && temperature < 25) {
    digitalWrite(buzzer_pin, LOW);
    msgval = 2;
    Serial.println("Flood Alert");
    send_flood_alert_sms();  // Send SMS alert when flood conditions are met
  } else if (water_level < 5 && humidity > 30 && temperature > 25) {
    digitalWrite(buzzer_pin, LOW);  // Buzzer is on
    msgval = 4;
    Serial.println("Water is high but humidity and temperature are constant");
  } else if (water_level > 90 && humidity > 30 && temperature > 25) {
    digitalWrite(buzzer_pin, LOW);
    msgval = 3;
    Serial.println("Water level is very low");

  } else if (humidity < 30 && temperature < 25 && water_level > 90 && water_level < 5) {
    msgval = 5;
    Serial.println("Temperature and humidity are low but water level is constant");
  } else {
    digitalWrite(buzzer_pin, HIGH);  // Buzzer is off
    msgval = 1;
    Serial.println("Everything looks safe");
  }

  // Publish data to MQTT
  publish_mqtt(humidity, temperature, water_level, msgval);

  // Upload data to ThingSpeak
  upload_to_cloud(humidity, temperature, water_level, msgval);
}

void connect_to_wifi() {
  delay(2000);
  Serial.print("Connecting to ");
  Serial.print(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi Connected, IP Address: ");
  Serial.println(WiFi.localIP());
}

void reconnect_mqtt() {
  while (!mqtt_client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (mqtt_client.connect("ESP8266Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(mqtt_client.state());
      delay(5000);
```

```
    }
  }
}

void publish_mqtt(float h, float t, int w, int val) {
  mqtt_client.publish(topic_humidity, String(h).c_str());
  mqtt_client.publish(topic_temperature, String(t).c_str());
  mqtt_client.publish(topic_water_level, String(w).c_str());
  mqtt_client.publish(topic_alert, String(val).c_str());
}

void upload_to_cloud(float h, float t, int w, int val) {
  ThingSpeak.setField(1, h);
  ThingSpeak.setField(2, t);
  ThingSpeak.setField(3, w);
  ThingSpeak.setField(4, val);
  ThingSpeak.writeFields(channel_number, writeapikey);
}

// Send flood alert SMS via Twilio
void send_flood_alert_sms() {
  if (!twilio_client.connect("api.twilio.com", 443)) {
    Serial.println("Connection to Twilio failed");
    return;
  }

  String post_data = "To=" + String(user_phone_number) + "&From=" + String(twilio_phone_number) + "&Body=Flood Alert! Water
level is critical.";

  // Send the HTTP POST request to Twilio API
  twilio_client.println("POST /2010-04-01/Accounts/" + String(twilio_account_sid) + "/Messages.json HTTP/1.1");
  twilio_client.println("Host: api.twilio.com");
  twilio_client.println("Authorization: Basic " + String(twilio_account_sid) + ":" + String(twilio_auth_token));
  twilio_client.println("Content-Type: application/x-www-form-urlencoded");
  twilio_client.print("Content-Length: ");
  twilio_client.println(post_data.length());
  twilio_client.println();
  twilio_client.println(post_data);


  Serial.println("Flood alert SMS sent to user.");
}
```

6 References

1. IoT Design Pro. *IoT Based Flood Monitoring System*. Available at: https://iotdesignpro.com/projects/iot-based-flood-monitoring-system.

2. YouTube. *IoT Based Flood Detection System*. Available at: https://youtu.be/zBvYOq7z_Oc?si=dJkO1xXej7J1q9I1.

3. Github repository *Flood Detection System Code*. Available at: https://github.com/Asif-Ali1234/flood-detection/blob/master/README.md.