FLOOD MONITORING AND EARLY WARNING

REG NO: 610821106095

NAME : SATHISH S

Hardware Components:

Arduino Board (e.g., Arduino Uno or Arduino MKR series)

IoT Module (e.g., ESP8266, ESP32, or Arduino MKR GSM/NB 1500 for connectivity)

Water Level Sensors (ultrasonic or capacitive)

Rainfall Sensors (if monitoring local weather)

GPS Module (for location tracking)

GSM/GPRS Module (for SMS alerts)

LED Displays or Alarms (for local alerts)

Power Supply (solar panel with battery backup for remote areas)

Enclosure and Weatherproofing (to protect components from the elements)

Data Logging Components (SD card or external storage, if needed)

Software Program (Example):

ARDUNIO WITH IOT PROGRAM

#include <Arduino.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME280.h>

#include <SoftwareSerial.h>

#include <TinyGPS++.h>

// Define pins for water level sensor, rainfall sensor, and LED indicators

const int waterLevelPin = A0;

const int rainfallPin = A1;

const int floodWarningPin = 2;

const int rainWarningPin = 3;

// Define variables for sensor data

float waterLevel = 0;

float rainfall = 0;

float alarmThreshold = 75.0; // Set your own threshold for water level

// GPS setup

SoftwareSerial gpsSerial(4, 5); // GPS module communication pins

TinyGPSPlus gps;

// GSM module setup

SoftwareSerial gsmSerial(6, 7); // GSM module communication pins

void setup() {

// Initialize serial communication for debugging

Serial.begin(9600);

// Initialize GPS and GSM modules

gpsSerial.begin(9600);

gsmSerial.begin(9600);

// Initialize LED pins

pinMode(floodWarningPin, OUTPUT);

pinMode(rainWarningPin, OUTPUT);

}

void loop() {

// Read sensor data

waterLevel = analogRead(waterLevelPin);

rainfall = analogRead(rainfallPin);

// Check water level and issue a flood warning

if (waterLevel >= alarmThreshold) {

digitalWrite(floodWarningPin, HIGH);

sendFloodAlert();

} else {

digitalWrite(floodWarningPin, LOW);

}

// Check rainfall and issue a rain warning

if (rainfall > 800) { // Adjust threshold based on sensor

digitalWrite(rainWarningPin, HIGH);

sendRainAlert();

} else {

digitalWrite(rainWarningPin, LOW);

}

// GPS data

while (gpsSerial.available() > 0) {

if (gps.encode(gpsSerial.read())) {

// Read GPS data (latitude, longitude, etc.) from the GPS module

float latitude = gps.location.lat();

float longitude = gps.location.lng();

// Send location data to a central server or store it locally

sendLocationData(latitude, longitude);

}

}

// Delay to control data transmission frequency

delay(60000); // 1 minute (adjust as needed)

}

void sendFloodAlert() {

// Send a flood alert via SMS using GSM module

gsmSerial.println("AT+CMGF=1"); // Set SMS mode to text

delay(100);

gsmSerial.print("AT+CMGS=\"+1234567890\""); // Replace with recipient's phone number

delay(100);

gsmSerial.print("Flood alert! Water level is critical.");

delay(100);

gsmSerial.write(26); // Send CTRL+Z

delay(1000);

}

void sendRainAlert() {

// Send a rain alert via SMS using GSM module

gsmSerial.println("AT+CMGF=1"); // Set SMS mode to text

delay(100);

gsmSerial.print("AT+CMGS=\"+1234567890\""); // Replace with recipient's phone number

delay(100);

gsmSerial.print("Heavy rainfall detected.");

delay(100);

gsmSerial.write(26); // Send CTRL+Z

delay(1000);

}

void sendLocationData(float latitude, float longitude) {

// Send location data to a central server or store it locally

// You can use an HTTP POST request to send data to a server

// or save it to an SD card for later retrieval.

}

PYTHON PROGRAM

Import paho.mqtt.client as mqtt

Import json

Import random

Import time

# Simulating water level data for testing

Def get\_water\_level():

Return random.uniform(0, 100)

# MQTT settings

Broker\_address = “mqtt.eclipse.org”

Port = 1883

Topic = “water\_level\_data”

# Callback when connection is established

Def on\_connect(client, userdata, flags, rc):

Print(“Connected with result code “+str(rc))

Client.subscribe(topic)

# Callback when a message is received from the server

Def on\_message(client, userdata, msg):

Print(f”Received message: {msg.payload}”)

# Create MQTT client

Client = mqtt.Client()

Client.on\_connect = on\_connect

Client.on\_message = on\_message

# Connect to the broker

Client.connect(broker\_address, port, 60)

# Loop to continuously send water level data

Try:

While True:

Water\_level = get\_water\_level()

Payload = {“water\_level”: water\_level}

# Convert dictionary to JSON

Payload\_json = json.dumps(payload)

# Publish data to the topic

Client.publish(topic, payload\_json)

Print(f”Sent water level data: {water\_level}”)

Time.sleep(10) # Adjust the interval based on your requirements

Except KeyboardInterrupt:

Print(“Script terminated by user”)

Client.disconnect()

```