**PHASE 3: DEVELOPMENT PART 1**

**FLOOD MONITORING AND EARLY WARNING**

**SYSTEM**

To create a flood monitoring and early warning system using an ESP32 microcontroller, float level sensor, a buzzer, a 16x2 LCD display, and a GSM module.

**Setup of Hardware:**

1.Connect the ESP32 to the float level sensor.

2.To receive audio alerts, connect the buzzer to the ESP32.

3.For visual notifications, connect the 16x2 LCD display to the ESP32.

4.To send SMS notifications, connect the GSM module.

**WORKING FUCTION:**

**1.FLOAT LEVEL SENSOR:**

The water level is monitored using the float level sensor. When a predetermined threshold is reached, the float, which rises with the water, sends out an alert.

**2.BUZZER:**

When a flood is discovered by the float sensor, the buzzer is employed to provide auditory alerts.

**3.LCD DISPLAY:**

The 16x2 LCD display shows the water level and flood status visually.

**4.GSM MODULE:**

When a flood is detected, the GSM module is utilized to send SMS alerts. Use AT commands to communicate with the module.

PYTHON SCRIPT:

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME280.h>

#include <TinyGSM.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME280.h>

// Initialize the GSM module (use appropriate pin numbers)

TinyGSM modem(Serial2);

// Initialize the BME280 sensor for temperature, humidity, and pressure (use appropriate I2C pins)

Adafruit\_BME280 bme;

// Initialize the LCD display (use appropriate I2C address)

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// Define pins for the float level sensor and buzzer

const int floatSensorPin = 2;

const int buzzerPin = 5;

void setup() {

Serial.begin(115200);

Serial2.begin(115200, SERIAL\_8N1, 12, 14);

// Initialize the LCD

lcd.init();

lcd.backlight();

// Check if the GSM module is responding

if (!modem.init()) {

lcd.print("GSM init failed");

while (true);

}

// Initialize the BME280 sensor

if (!bme.begin(0x76)) {

lcd.print("BME280 init failed");

while (true);

}

// Set the float sensor pin as an input

pinMode(floatSensorPin, INPUT);

// Set the buzzer pin as an output

pinMode(buzzerPin, OUTPUT);

noTone(buzzerPin);

}

void loop() {

// Read the water level sensor

int waterLevel = digitalRead(floatSensorPin);

// Read temperature, humidity, and pressure

float temperature = bme.readTemperature();

float humidity = bme.readHumidity();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temp: ");

lcd.print(temperature);

lcd.print(" C");

lcd.setCursor(0, 1);

lcd.print("Humidity: ");

lcd.print(humidity);

lcd.print(" %");

if (waterLevel == HIGH) {

lcd.setCursor(0, 1);

lcd.print("Flood Alert!");

tone(buzzerPin, 1000);

sendSMS("Emergency: Flood detected!");

} else {

noTone(buzzerPin);

}

delay(5000); // Delay for 5 seconds between readings

}

void sendSMS(const char\* message) {

if (modem.isNetworkConnected()) {

if (modem.sendSMS("+1234567890", message)) {

Serial.println("SMS sent successfully");

} else {

Serial.println("SMS sending failed");

}

} else {

Serial.println("Not connected to the network");

}

}

**Explanation:**

1.Network Configuration: The script first sets up your Wi-Fi network on the ESP32 so that it may connect to it.

2.LCD Initialization: The 16x2 LCD display's I2C connectivity is set up so that you can display the water level.

3.Analog-to-Digital Converter (ADC) Initialization: The ADC is configured to read data from the float level sensor. You can change the attenuation value to suit the needs of your sensor.

4.Initialization of the GSM Module: To transmit SMS alerts, the GSM module is initialized using UART pins.

5.Water Level Reading: Using the float sensor, the read\_water\_level function reads the water level.

6.Water level is displayed on the LCD through the display\_on\_LCD function.

7.SMS Alert: If the water level rises beyond the set threshold, the script activates a siren (through GPIO pin 5) and uses the GSM module to send an SMS alert to a designated emergency contact.

8.Main Loop: The main loop transmits alerts, refreshes the display, continuously monitors the water level, and checks for flood situations. Between readings, it sleeps for 60 seconds (modify as necessary).

In conclusion, the system for monitoring floods and issuing early warnings is intended to deliver timely warnings and details on the rising water levels in urban areas. The system's functionality is provided by an ESP32 microcontroller, a float level sensor, a buzzer, a 16x2 LCD display, and a GSM module. The ESP32 microcontroller handles the data from the float level sensor, which continuously measures the water level. The 16x2 LCD display gives real-time feedback, enabling nearby water level monitoring. The system delivers SMS warnings to certain contacts using the GSM module and an auditory alert through the buzzer when the water level exceeds a predetermined threshold. The system is adaptable and can be used in a places that are susceptible to flooding. It functions as an early warning system, reducing potential flood-related dangers and securing the neighborhood.