Flood monitoring and early warning system

Creating a flood monitoring and early warning project using IoT and Arduino is a practical and cost-effective approach. Arduino boards are commonly used for IoT projects and can interface with various sensors and communication modules. Here's a simplified guide to building a basic flood monitoring system using Arduino:

Required components:

1. Arduino Board (e.g., Arduino Uno or Arduino MKR): The brain of your system.

2. Flood Sensors: These could include water level sensors and rain sensors.

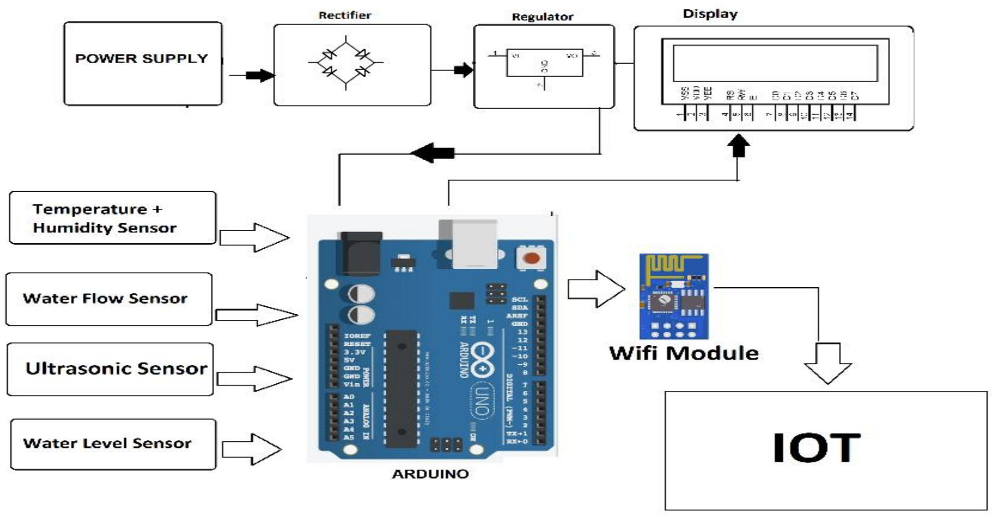
3. GSM/GPRS Module: To send SMS alerts.

4. Wi-Fi Module (optional): For Internet connectivity and remote monitoring.

5. Power Source: A reliable power supply, such as a battery or solar panel.

6. Enclosure: To protect the components from environmental factors.

7. PC or Raspberry Pi: For data processing and analysis.



Steps to Create a Flood Monitoring and Early Warning System:

1. Sensor Connection:

- Connect the flood sensors to the Arduino. Water level sensors are typically analog sensors, while rain sensors can be analog or digital. Follow the datasheets and instructions for the specific sensors you are using.

2. Arduino Programming:

- Write Arduino code to read data from the sensors and process it. For example, you can measure water level and monitor rainfall. You may want to calibrate the sensors and set threshold levels for flood alerts.

3. Communication Module:

- If using a GSM/GPRS module, write code to send SMS alerts when the flood conditions exceed the predefined thresholds. If using Wi-Fi, you can send data to a cloud platform for remote monitoring.

4. Data Processing:

- On a PC or Raspberry Pi, write a program to receive and process the data from the Arduino. This program can analyze the data and trigger alerts when flooding is detected.

5. Alerts and Notifications:

- Develop a notification system that can send alerts via SMS, email, or other communication channels. You can integrate services like Twilio for SMS alerts.

6. Power Management:

- Implement a power management system to ensure that the Arduino can run on battery or solar power. Low-power modes should be used to conserve energy.

7. Enclosure and Weatherproofing:

- Place the components in an enclosure to protect them from environmental factors. Ensure that the sensors are positioned correctly and that the enclosure is weatherproof.

8. Testing and Calibration:

- Test the system in a controlled environment and calibrate the sensors to ensure accuracy.

9. Community Engagement:

- Educate the local community about the flood monitoring system. Provide instructions on how to access alerts and what actions to take during flood events.

10. Regulatory Compliance:

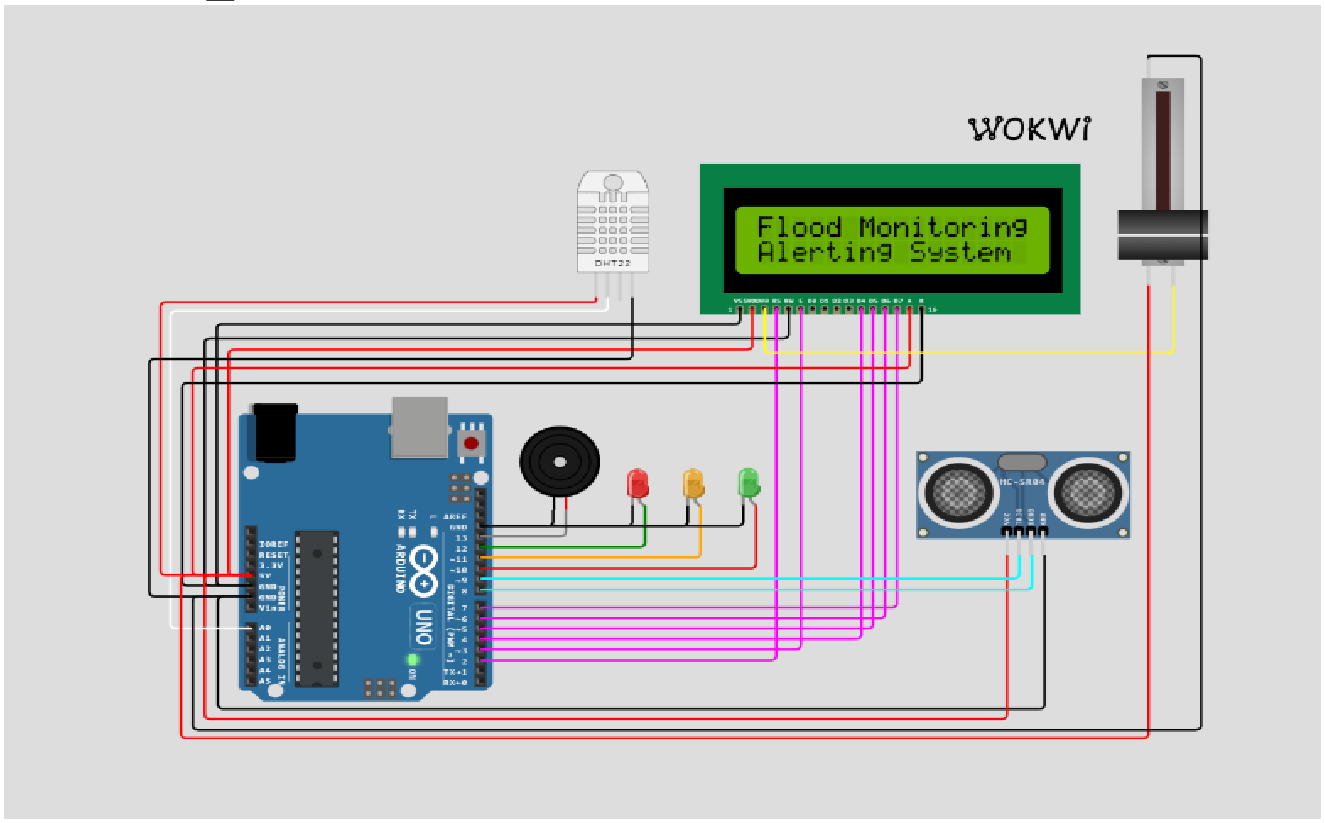
- Ensure your project adheres to local regulations and privacy laws, especially when handling user data and sending alerts.

11. Scalability:

- Design the system for scalability, so you can expand and cover more areas as needed.

Remember that this is a simplified overview of the project. The success of your flood monitoring and early warning system depends on the accuracy of your sensors, the reliability of your communication infrastructure, and the effectiveness of your data processing and alerting mechanisms. Collaborate with experts in meteorology, IoT, and data analysis for a more robust and reliable solution.

Circuit_diagram:



Sample code for Arduino

```
//Early Flood Detection Using IOT ~ A project by Sabyasachi Ghosh
//<LiquidCrystal.h> is the library for using the LCD 16x2
#include <LiquidCrystal.h>
//"DHT.h" is the library for using the Temperature sensor DHT22
#include "DHT.h"
#include <ThingSpeak.h>
#include <WiFi.h>
#define DHTPIN A0                        //here we are initialising a pin for
DHT22
#define DHTTYPE DHT22                    //We have to declare the type of DHT
sensor we are using for its correct functionality
LiquidCrystal lcd(2,3,4,5,6,7);          // Create an instance of the
LiquidCrystal library
DHT dht(DHTPIN, DHTTYPE);                // Create an instance of the DHT library
for the DHT22 sensor
const int in=8;                          //This is the ECHO pin of The Ultrasonic
sensor HC-SR04
const int out=9;                         //This is the TRIG pin of the ultrasonic
Sensor HC-SR04
// Define pin numbers for various components
const int green=10;
const int orange=11;
const int red=12;
const int buzz=13;
//sambungin ke wifi kita
const char *ssid =  "Nob!ta"; //your network SSID (name)
const char *pass =  "88888888"; //your network password
WiFiClient client;
//thingspeak settings
unsigned long channel = 2310946; //your channel ID number** dari channel
thingspeak yg telah kita buat
const char *apiKey = "991XP5UF0LFGRKNS"; //your channel write API Key


void setup()
{
  // Start serial communication with a baud rate of 9600
  Serial.begin(9600);
  // Initialize the LCD with 16 columns and 2 rows
  lcd.begin(16, 2);
  // Set pin modes for various components
  pinMode(in, INPUT);
  pinMode(out, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(orange, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(buzz, OUTPUT);
  // Initialize the DHT sensor
  dht.begin();
  // Set initial states for LEDs and buzzer to LOW (off)
  digitalWrite(green,LOW);
  digitalWrite(orange,LOW);
  digitalWrite(red,LOW);
  digitalWrite(buzz,LOW);
  // Display a startup message on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Flood Monitoring");
  lcd.setCursor(0,1);
  lcd.print("Alerting System");
  // Wait for 5 seconds and then clear the LCD
  delay(5000);
  lcd.clear();
  //connect to WiFi
    Serial.print("Connecting to: "); Serial.println(ssid);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("\nWiFi connected\n");

    ThingSpeak.begin(client); //initialize ThingSpeak
}

void loop()
{
  // Read temperature and humidity from the DHT22 sensor
  float T = dht.readTemperature();
  float H = dht.readHumidity();
  // Check if the sensor data is valid
  if (isnan(H) && isnan(T)) {
    lcd.print("ERROR");
    return;
  }
  float f = dht.readTemperature(true);
  // Read distance from the ultrasonic sensor (HC-SR04)
  long dur;
  long dist;
  long per;
  digitalWrite(out,LOW);
  delayMicroseconds(2);
  digitalWrite(out,HIGH);
  delayMicroseconds(10);
  digitalWrite(out,LOW);
  dur=pulseIn(in,HIGH);
  dist=(dur*0.034)/2;
  // Map the distance value to a percentage value
  per=map(dist,10.5,2,0,100);
  // Ensure that the percentage value is within bounds
  if(per<0)
  {
    per=0;
  }
  if(per>100)
  {
    per=100;
  }
  // Print sensor data and percentage value to serial
  Serial.print(("Humidity: "));
  Serial.print(H);
  Serial.print(("%  Temperature: "));
  Serial.print(T);
  Serial.print("%   Water Level:");
  Serial.println(String(per));
  lcd.setCursor(0,0);
  lcd.print("Temperature:");
  lcd.setCursor(0,1);
  lcd.print("Humidity   :");
  lcd.setCursor(12,0);
  lcd.print(T);
  lcd.setCursor(12,1);
  lcd.print(H);
  delay(1000);
  lcd.clear();
  lcd.print("Water Level:");
  lcd.print(String(per));
  lcd.print("%  ");
  // Check water level and set alert levels
  int x = ThingSpeak.writeField(channel, 1, dist, apiKey);
  if(dist<=3)
  {
      lcd.setCursor(0,1);
      lcd.print("Red Alert!   ");
```

```
      digitalWrite(red,HIGH);
      digitalWrite(green,LOW);
      digitalWrite(orange,LOW);
      digitalWrite(buzz,HIGH);
      delay(2000);
      digitalWrite(buzz,LOW);
      delay(2000);
      digitalWrite(buzz,HIGH);
      delay(2000);
      digitalWrite(buzz,LOW);
      delay(2000);
  }
  else if(dist<=10)
  {
      lcd.setCursor(0,1);
    lcd.print("Orange Alert!  ");
    digitalWrite(orange,HIGH);
    digitalWrite(red,LOW);
    digitalWrite(green,LOW);
    digitalWrite(buzz,HIGH);
    delay(3000);
    digitalWrite(buzz,LOW);
    delay(3000);
  }else
  {
     lcd.setCursor(0,1);
    lcd.print("Green Alert!  ");
    digitalWrite(green,HIGH);
    digitalWrite(orange,LOW);
    digitalWrite(red,LOW);
    digitalWrite(buzz,LOW);
  }
}
```

Wokwi link : https://wokwi.com/projects/378925669888337921