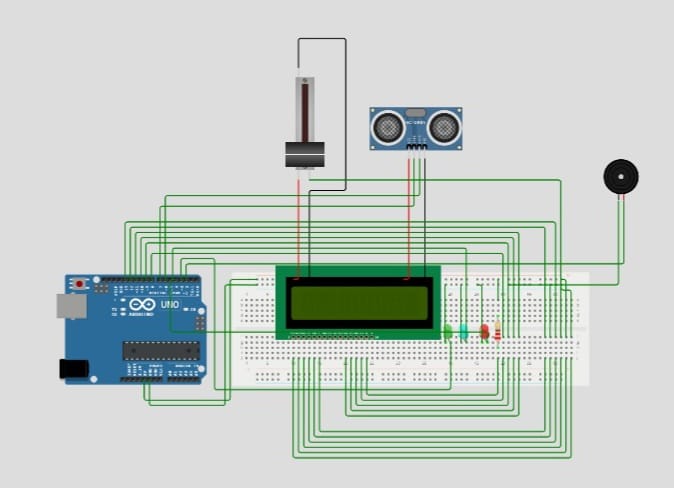
Flood Monitoring & Early Warning

Phase-5

Include diagrams, schematics, and screenshots of the

IoT sensors and early warning platform



C++ code for flood monitoring & Early warning

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10,  9, 8);

int trig = 7;

int echo = 6;

int red = 5;

int blue = 4;

int green = 3;

int buz = 2;

int distance;

long duration;

void setup()

{

 pinMode(red,OUTPUT);

 pinMode(blue,OUTPUT);

 pinMode(green,OUTPUT);

 pinMode(buz,OUTPUT);

 pinMode(trig,OUTPUT);

 pinMode(echo,INPUT);

**Serial**.begin(9600);

 lcd.begin(16,2);

 lcd.setCursor(1,0);

 lcd.print("Distance:");

}

void loop()

{

 digitalWrite(trig,LOW);

 digitalWrite(trig,HIGH);

 delayMicroseconds(10);

 digitalWrite(trig,LOW);

 float duration = pulseIn(echo,HIGH);

 // Distance = speed \* Time

 float distance = (0.0343 \* duration)/2 ;

 lcd.setCursor(10,0);

 lcd.print(distance);

  if(distance>250)

  {

   lcd.setCursor(3,1);

   lcd.print("Safe Zone  ");

   digitalWrite(red,LOW);

   digitalWrite(green,HIGH);

   digitalWrite(blue,LOW);

   digitalWrite(buz,LOW);

   delay(1000);

  }

   else if(distance>200&&distance>100)

  {

   lcd.setCursor(3,1);

   lcd.print("Medium Zone");

   digitalWrite(red,LOW);

   digitalWrite(green,LOW);

   digitalWrite(blue,HIGH);

   digitalWrite(buz,LOW);

   delay(1000);

  }

  else if(distance<100)

  {

   lcd.setCursor(3,1);

   lcd.print("Danger Zone");

   digitalWrite(red,HIGH);

   digitalWrite(green,LOW);

   digitalWrite(blue,LOW);

   digitalWrite(buz,HIGH);

  }

}

Python code for flood monitoring & Early warning

# Flood Detection System

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

lcd\_rs = 13

lcd\_en = 12

lcd\_d4 = 11

lcd\_d5 = 10

lcd\_d6 = 9

lcd\_d7 = 8

trig = 7

echo = 6

red = 5

blue = 4

green = 3

buz = 2

GPIO.setup(red, GPIO.OUT)

GPIO.setup(blue, GPIO.OUT)

GPIO.setup(green, GPIO.OUT)

GPIO.setup(buz, GPIO.OUT)

GPIO.setup(trig, GPIO.OUT)

GPIO.setup(echo, GPIO.IN)

def setup():

GPIO.output(red, GPIO.LOW)

GPIO.output(blue, GPIO.LOW)

GPIO.output(green, GPIO.LOW)

GPIO.output(buz, GPIO.LOW)

GPIO.output(trig, GPIO.LOW)

GPIO.output(trig, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(trig, GPIO.LOW)

lcd\_init()

print\_distance()

def lcd\_init():

lcd = LiquidCrystal(lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7)

lcd.begin(16, 2)

lcd.setCursor(1, 0)

lcd.print("Distance:")

def print\_distance():

while True:

GPIO.output(trig, GPIO.LOW)

GPIO.output(trig, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(trig, GPIO.LOW)

duration = pulseIn(echo, GPIO.HIGH)

distance = (0.0343 \* duration) / 2

lcd.setCursor(10, 0)

lcd.print(distance)

if distance > 250:

lcd.setCursor(3, 1)

lcd.print("Safe Zone ")

GPIO.output(red, GPIO.LOW)

GPIO.output(green, GPIO.HIGH)

GPIO.output(blue, GPIO.LOW)

GPIO.output(buz, GPIO.LOW)

time.sleep(1)

elif distance > 200 and distance > 100:

lcd.

**C++ Code Explaination:**

**1. #include <LiquidCrystal.h>**

Include the LiquidCrystal library for controlling the LCD display.

**2. LiquidCrystal lcd(13, 12, 11, 10, 9, 8);**

Initialize an instance of the LiquidCrystal class to control the LCD display, specifying the pins connected to the LCD module.

**3. int trig = 7; int echo = 6; int red = 5; int blue = 4; int green = 3; int buz = 2;**

Define variables to store pin numbers for the ultrasonic sensor's trigger (trig), echo (echo), and various LEDs and a buzzer.

**4. int distance; long duration;**

Declare two variables to store the distance measured and the duration of the ultrasonic pulse.

**5.** **void setup()**

**{**

**pinMode(red, OUTPUT);**

**pinMode(blue, OUTPUT);**

**pinMode(green, OUTPUT);**

**pinMode(buz, OUTPUT);**

**pinMode(trig, OUTPUT);**

**pinMode(echo, INPUT);**

**Serial.begin(9600);**

**lcd.begin(16, 2);**

**lcd.setCursor(1, 0);**

**lcd.print("Distance:");**

**}**

In the **setup()** function:

* Set the pins for the LEDs (red, blue, green) and buzzer as OUTPUT.
* Set the trigger pin for the ultrasonic sensor as OUTPUT and the echo pin as INPUT.
* Initialize the Serial communication at a baud rate of 9600 for debugging purposes.
* Initialize the LCD with 16 columns and 2 rows, and display the initial message on the LCD.

**6.** **void loop()**

**{**

**digitalWrite(trig, LOW);**

**digitalWrite(trig, HIGH);**

**delayMicroseconds(10);**

**digitalWrite(trig, LOW);**

**float duration = pulseIn(echo, HIGH);**

**float distance = (0.0343 \* duration) / 2;**

**lcd.setCursor(10, 0);**

**lcd.print(distance);**

In the **loop()** function:

* Trigger the ultrasonic sensor to measure distance by sending a pulse and then listening for the echo.
* Calculate the distance using the formula: **Distance = speed \* Time**.
* Display the measured distance on the LCD.

**7. if (distance > 250)**

**{**

**lcd.setCursor(3, 1);**

**lcd.print("Safe Zone ");**

**digitalWrite(red, LOW);**

**digitalWrite(green, HIGH);**

**digitalWrite(blue, LOW);**

**digitalWrite(buz, LOW);**

**delay(1000);**

**}**

**else if (distance > 200 && distance > 100)**

**{**

**lcd.setCursor(3, 1);**

**lcd.print("Medium Zone");**

**digitalWrite(red, LOW);**

**digitalWrite(green, LOW);**

**digitalWrite(blue, HIGH);**

**digitalWrite(buz, LOW);**

**delay(1000);**

**}**

**else if (distance < 100)**

**{**

**lcd.setCursor(3, 1);**

**lcd.print("Danger Zone");**

**digitalWrite(red, HIGH);**

**digitalWrite(green, LOW);**

**digitalWrite(blue, LOW);**

**digitalWrite(buz, HIGH);**

**}**

**}**

Based on the measured distance, the code changes the message on the LCD and the LED color:

* If the distance is greater than 250 units, it's considered a "Safe Zone," and the LED is set to green.
* If the distance is between 200 and 100 units, it's a "Medium Zone," and the LED is set to blue.
* If the distance is less than 100 units, it's a "Danger Zone," and the LED is set to red, and the buzzer is turned on.

**Finally:**

This code essentially creates a distance measuring system using an ultrasonic sensor and provides visual and auditory feedback based on the measured distance. The LCD displays the current distance, and the LED color and buzzer indicate the safety zone.

**To Explain how the real-time flood monitoring and early warning system can enhance public safety and emergency Response coordination**

A real-time flood monitoring and early warning system can significantly enhance public safety and emergency response coordination by providing timely and accurate information to both the public and authorities. Here's how such a system can make a difference:

**Early Detection and Warning:**

Real-time flood monitoring systems use various sensors and data sources to detect changes in water levels, rainfall, and other relevant factors. When critical thresholds are reached, the system can issue automated warnings. Early detection allows authorities to issue warnings to the public, giving them more time to prepare and evacuate if necessary, reducing the risk of fatalities and property damage.

**Public Awareness:**

These systems can disseminate alerts and warnings through various communication channels, including mobile apps, text messages, sirens, and social media. This ensures that the public is informed quickly.

Public awareness empowers individuals to take necessary precautions, such as moving to higher ground, securing their property, or seeking shelter, thus reducing their vulnerability.

**Coordination of Emergency Response:**

Flood monitoring systems provide real-time data to emergency responders, enabling them to plan and allocate resources effectively.

Emergency response agencies can use the data to assess the extent of flooding, identify vulnerable areas, and prioritize their efforts to save lives and protect property.

**Evacuation Planning:**

Advanced warning systems give communities more time to plan and execute evacuations. This is especially crucial for people with limited mobility or those living in flood-prone areas.

Local authorities can provide evacuation routes, organize shelters, and mobilize transportation services to assist residents in leaving high-risk zones.

**Resource Allocation:**

With real-time data on flood conditions, emergency services can allocate resources strategically. They can deploy first responders, search and rescue teams, and equipment to areas most in need.

This efficient resource allocation can help prevent bottlenecks and delays in responding to emergencies.

**Post-Event Assessment:**

After a flood event, the data collected by monitoring systems can help authorities assess the damage and prioritize recovery efforts.

It enables a faster and more targeted response to provide aid, restore infrastructure, and assist affected communities.

**Improved Decision-Making:**

Decision-makers can access real-time data to make informed choices regarding evacuations, road closures, and other emergency measures.

These systems provide decision-makers with the information they need to minimize loss of life and property.

**Continuous Monitoring and Adaptation:**

Real-time flood monitoring systems operate 24/7, allowing for continuous surveillance of changing conditions.

Authorities can adapt their response strategies as conditions evolve, ensuring that their actions remain effective and responsive.

In summary, a real-time flood monitoring and early warning system is a critical tool in enhancing public safety and emergency response coordination. It enables early detection, effective communication, informed decision-making, and efficient resource allocation, ultimately reducing the impact of floods on both lives and property. Public safety is greatly improved as individuals have the time and information they need to protect themselves and their communities.

**To Include example outputs of IoT sensor data transmission and platform UI for flood monitoring and early warning**

Creating a full-fledged IoT sensor data transmission and platform UI for flood monitoring and early warning is beyond the scope of a simple text response, but I can provide you with an example of what such a system might look like.

**IoT Sensor Data Transmission**

In a flood monitoring system, you would have various sensors deployed in flood-prone areas to collect data such as water levels, rainfall, and weather conditions. These sensors would transmit data to a central platform or server in real-time. Here's an example of data transmitted from these sensors:

**Water Level Sensor:**

Sensor ID: WL-001

Location: River XYZ

Timestamp: 2023-10-31 15:30:00

Water Level: 4.2 meters

**Rainfall Sensor:**

Sensor ID: RF-002

Location: Area ABC

Timestamp: 2023-10-31 15:30:00

Rainfall Intensity: 5.1 mm/hr

**Weather Sensor:**

Sensor ID: WS-003

Location: City DEF

Timestamp: 2023-10-31 15:30:00

Temperature: 20°C

Humidity: 80%

Atmospheric Pressure: 1010 hPa

**Platform UI for Flood Monitoring and Early Warning**

The platform would have a user interface that displays this data in real-time, allowing users to monitor the flood situation and receive early warnings. Below is an example of what the UI might look like:

**Dashboard:**

At the top of the dashboard, We have key metrics and alerts, such as the current water level in the most critical area, rainfall intensity, and the overall flood risk level.

A map displaying sensor locations and current data points, color-coded to indicate the severity of the situation.

**Sensor Data Panel:**

A section that lists all deployed sensors with their respective data, updated in real-time. For example, a table showing the latest data for each sensor.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sensor  ID | Location | Timestamp | Water Level (m) | Rainfall (mm/hr) | Temperature (°C) | Humidity (%) | Pressure (hPa) |
| WL-001 | River XYZ | 2023-10-31 15:30:00 | 4.2 | - | - | - | - |
| RF-002 | Area ABC | 2023-10-31 15:30:00 | - | 5.1 | - | - | - |
| WS-003 | City DEF | 2023-10-31 15:30:00 | - | - | 20 | 80 | 1010 |

**Alerts and Warnings:**

An alert section that triggers warnings when certain conditions are met, such as a sudden rise in water level or heavy rainfall.

Notifications sent to users' mobile devices or email in real-time.

**Historical Data and Trends:**

A section to review historical sensor data and analyze trends, allowing users to make informed decisions for flood management.

Remember that this is a simplified example. A real flood monitoring system would be more complex and incorporate advanced data analytics, predictive modeling, and communication protocols for emergency responses.