

Flood Monitoring and Early Warning system using IoT

Team Member:

- Gowsalya N – 921021104012
- Epshiba R - 921021104010
- Deepika E – 921021104005
- Meenakshi R - 921021104027

Phase-4 Document Submission

Flood Monitoring and Early Warning:

- Floods are a major natural disaster that can cause widespread damage and loss of life. When water levels rise suddenly in dams, riverbeds, and other bodies of water, it can lead to flooding in surrounding areas. This can cause significant damage to property and infrastructure, as well as loss of life.
- It is important to have early warning systems in place to alert people of rising water levels so that they can take necessary precautions. One way to do this is to use water level sensors to monitor water levels in real time. If the water level exceeds a certain threshold, the sensor can generate an alert that can be sent to people via a variety of methods, such as LED signals, buzzers, SMS messages, or emails.
- This project aims to develop a water level monitoring system that can alert people of rising water levels in riverbeds. The system will use water level sensors to measure the water level in real time. If the water level exceeds a certain threshold, the system will generate an alert that will be sent to people via LED signals, a buzzer, SMS messages, and emails.
- This system can help to protect people and property from the devastating effects of floods by providing early warning of rising water levels.

Working with Wokwi Simulator:

About Sketch.ino:

- The code uses an ultrasonic sensor (HC-SR04) to measure the distance to the water level. The distance value is then mapped to a percentage value and displayed on an LCD screen.
- The system also has three alert levels: red, orange, and green.
- If the distance to the water level is less than 3 meters, the system will trigger a red alert, which will turn on a red LED and sound a buzzer.
- If the distance to the water level is between 3 and 10 meters, the system will trigger an orange alert, which will turn on an orange LED.
- If the distance to the water level is greater than 10 meters, the system will trigger a green alert, which will turn on a green LED.
- The system also sends the water level data to a serial port, which can be used to monitor the system remotely.

Sketch.ino:

```
//Early Flood Detection Using IOT

//<LiquidCrystal.h> is the library for using the LCD 16x2

#include <LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7); // Create an instance of the LiquidCrystal library

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;

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);

// 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();

}

void loop()

{

// 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 water level data to serial

Serial.print("Water Level:");

Serial.println(String(per));

lcd.setCursor(0, 0);

lcd.print("Water Level:");

lcd.print(String(per));

lcd.print("% ");

// Check water level and set alert levels
```

```
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);

}

}
```

Here is a brief explanation of the code:

- The first few lines of code include the necessary libraries and define some constants, such as the pin numbers for the various components.
- The `setup()` function is used to initialize the system. It starts serial communication, initializes the LCD screen, and sets the pin modes for the various components.
- The `loop()` function is executed repeatedly. It reads the distance to the water level, maps it to a percentage value, and displays it on the LCD screen. It also checks the water level and sets the appropriate alert level.

About diagram.json:

- The version field specifies the version of the Wokwi schematic format.
- The author field specifies the author of the schematic.
- The editor field specifies the editor used to create the schematic.

- The parts field is an array of objects that represent the components in the schematic. Each object in the array has a type field that specifies the type of component and an id field that uniquely identifies the component.
- The connections field is an array of objects that represent the connections between the components in the schematic. Each object in the array has a source field and a destination field that specify the two components that are connected.
- The dependencies field is an object that specifies any dependencies that the schematic has.

The diagram.json file contains the following code:

```
{
  "version": 1,
  "author": "Gowsalya Nagaraj",
  "editor": "wokwi",
  "parts": [
    {
      "type": "wokwi-arduino-uno",
      "id": "uno",
      "top": -64.64,
      "left": 105.33,
      "rotate": 90,
      "attrs": {}
    },
    { "type": "wokwi-lcd1602", "id": "lcd1", "top": -312.02, "left": 508, "attrs": {} },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -45.08,
      "left": 439.42,
      "attrs": { "color": "red" }
    },
  ],
}
```

```
{  
  "type": "wokwi-hc-sr04",  
  "id": "ultrasonic1",  
  "top": -52.66,  
  "left": 680.31,  
  "attrs": { "distance": "7" }  
},  
{  
  "type": "wokwi-slide-potentiometer",  
  "id": "pot1",  
  "top": -339.66,  
  "left": 767.48,  
  "rotate": 270,  
  "attrs": { "travelLength": "30" }  
},  
{  
  "type": "wokwi-buzzer",  
  "id": "bz1",  
  "top": -83.14,  
  "left": 364.46,  
  "attrs": { "volume": "0.1" }  
},  
{  
  "type": "wokwi-led",  
  "id": "led2",  
  "top": -44.56,  
  "left": 483.32,  
  "attrs": { "color": "orange" }  
},  
{  
  "type": "wokwi-led",
```



```
"id": "led3",  
"top": -45.47,  
"left": 527.48,  
"attrs": { "color": "limegreen" }  
}  
,  
"connections": [  
  [ "lcd1:D4", "uno:4", "magenta", [ "v0" ] ],  
  [ "lcd1:D5", "uno:5", "magenta", [ "v0" ] ],  
  [ "lcd1:D6", "uno:6", "magenta", [ "v0" ] ],  
  [ "lcd1:D7", "uno:7", "magenta", [ "v0" ] ],  
  [ "led3:A", "uno:10", "red", [ "v0" ] ],  
  [ "led2:A", "uno:11", "orange", [ "v0" ] ],  
  [ "led1:A", "uno:12", "green", [ "v0" ] ],  
  [ "bz1:2", "uno:13", "gray", [ "v0" ] ],  
  [ "uno:GND.1", "led1:C", "black", [ "h0" ] ],  
  [ "uno:GND.1", "led2:C", "black", [ "h0" ] ],  
  [ "uno:GND.1", "led3:C", "black", [ "h0" ] ],  
  [ "uno:GND.1", "bz1:1", "black", [ "h0" ] ],  
  [ "uno:GND.2", "lcd1:VSS", "black", [ "h-27.73", "v-236.01", "h405.87" ] ],  
  [ "uno:GND.2", "lcd1:RW", "black", [ "h-37.21", "v-223.22", "h460.88" ] ],  
  [ "uno:5V", "lcd1:VDD", "red", [ "h-18.3", "v-203.63", "h13.46" ] ],  
  [ "lcd1:RS", "uno:2", "magenta", [ "v0" ] ],  
  [ "lcd1:E", "uno:3", "magenta", [ "v0" ] ],  
  [ "uno:5V", "lcd1:A", "red", [ "h-46.76", "v-186.9", "h566.43" ] ],  
  [ "uno:GND.2", "lcd1:K", "black", [ "h-55.24", "v-183", "h584.41" ] ],  
  [ "uno:GND.3", "ultrasonic1:GND", "black", [ "h-26.54", "v101.57", "h653.32" ] ],  
  [ "uno:5V", "ultrasonic1:VCC", "red", [ "h-37.14", "v129.59", "h633.92" ] ],  
  [ "ultrasonic1:TRIG", "uno:9", "cyan", [ "v0" ] ],  
  [ "ultrasonic1:ECHO", "uno:8", "cyan", [ "v0" ] ],  
  [ "lcd1:V0", "pot1:SIG", "yellow", [ "v39.74", "h14.45" ] ],
```

```
[ "uno:5V", "pot1:VCC", "red", [ "h-56.08", "v146.93", "h757.73" ] ],  
[ "uno:GND.3", "pot1:GND", "black", [ "h-46.52", "v120.3", "h800.57", "v-607.79", "h-52.4" ] ]  
],  
"dependencies": {}  
}
```

About libraries.txt:

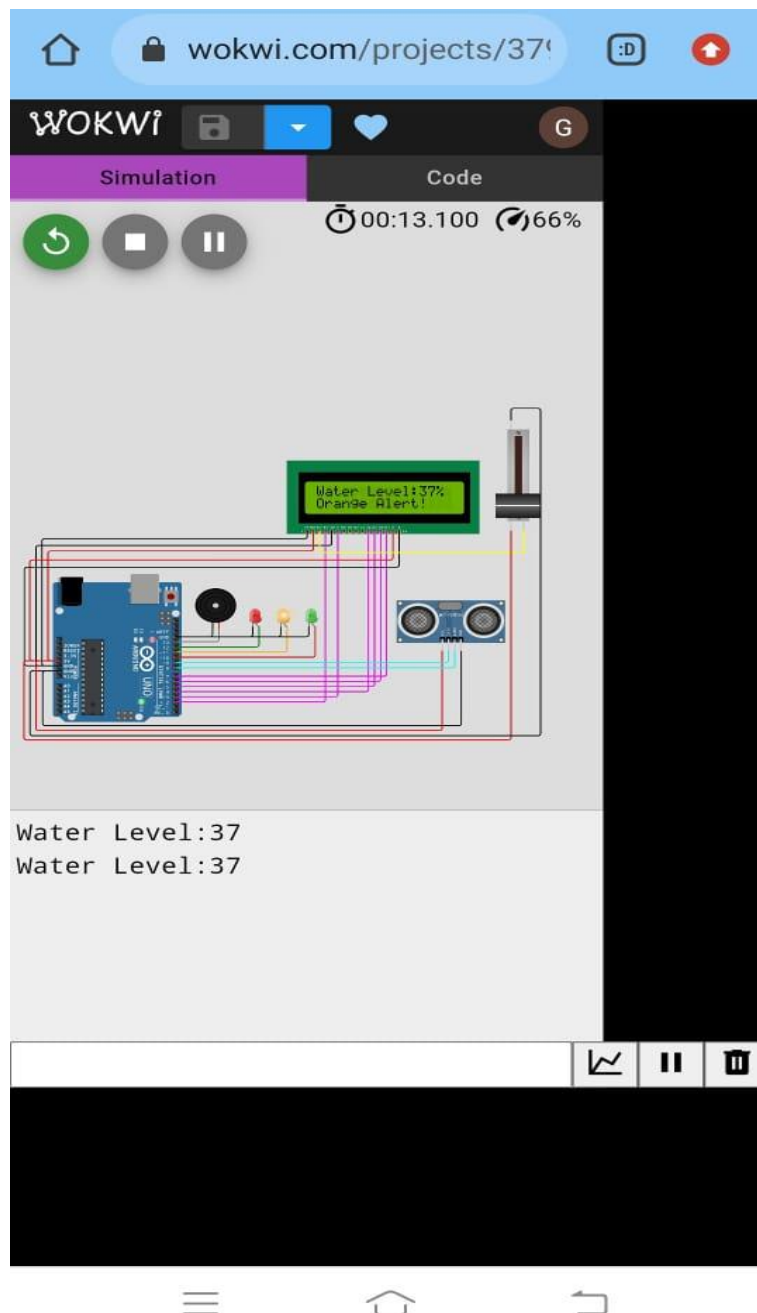
The following libraries are included in the project:

- LiquidCrystal: This library is used to control a 16x2 character LCD display.
- DHT22: This library is used to read data from a DHT22 temperature and humidity sensor.
- DHT sensor library: This library is a generic library that can be used to read data from any DHT sensor.

Simulating the code in the Wokwi Simulator:

- The Wokwi Simulator will load our project and start simulating it. We can use the controls on the left-hand side of the simulator to interact with the components in our project. For example, we can use the slider to adjust the potentiometer or the button to turn on the buzzer.
- To simulate the ultrasonic sensor, we can use the "Distance Sensor" component in the Wokwi Simulator. Simply drag the component onto our project and connect it to the TRIG and ECHO pins of your ultrasonic sensor. We can then use the "Distance" slider to adjust the distance between the ultrasonic sensor and the water surface.
- To simulate the DHT22 sensor, we can use the "DHT22 Sensor" component in the Wokwi Simulator. Simply drag the component onto our project and connect it to the data pin of our DHT22 sensor. We can then use the "Temperature" and "Humidity" sliders to adjust the temperature and humidity readings from the sensor.
- Once we have configured the simulator, we can start simulating the project. To do this, simply click the "Play" button. The simulator will start running our project code and the components in our project will start behaving accordingly.
- We can use the simulator to test the functionality of our project and to troubleshoot any problems that we encounter.

The link for the project: <https://wokwi.com/projects/379571837255046145>



Since we had trouble using Wokwi Simulator in the PC web browser, we have implemented the project in the mobile web browser.

Conclusion:

The early flood detection system using an ultrasonic sensor and a DHT22 sensor is a simple and effective way to detect rising water levels and prevent flood damage. The system can be used in a variety of settings, such as homes, businesses, and farms. It can be used to protect property from flood damage and to warn people of rising water levels so that they can evacuate to safety.