Flood Monitoring and Early Warning System

Phase-5 project submission

Team Members:

Dharshini L -921021104008
Roobini G -921021104042
Mahaswetha M -921021104024
Nathiya P -921021104032
Nazreen M

INTRODUCTION

Floods are natural disasters that can cause extensive damage to property, infrastructure, and loss of lives. Developing an effective Flood Monitoring and Early Warning System (FMEWS) is essential to minimize the impact of floods on communities and the environment.

PROJECT OBJECTIVES

The objectives of the real-time flood monitoring and early warning system project are to:

- Develop a system that can collect and analyse real-time data from IoT sensors to monitor flood levels and predict flood events.
- Provide early warning to the public and emergency responders in flood-prone areas.
- Enhance public safety and emergency response coordination.

IOT SENSOR DEPLOYMENT

The IoT sensors will be deployed in strategic locations in flood-prone areas, such as near rivers, lakes, and reservoirs. The sensors will measure water levels, rainfall, and other relevant environmental data. The data will be transmitted to the cloud platform in real time using a variety of communication technologies, such as cellular, satellite, or radio.
Some hardware requirements are,

1. Wi-fi module
2. Arduino uno
3. Breadboard- 400 tie points
4. LED:(Green, Red, Orange) and Buzzer
5. 16×2 LCD Display
6. Temperature Sensor
7. Ultrasonic Sensor

8. Some Jumper Wires

HARDWARE DESCRIPTION:

1.WI-FI MODULE

The Arduino Uno Wi-Fi is an Arduino Uno with an integrated Wi-Fi module. The board is based on the ATmega328P with an ESP8266WiFi Module integrated. The ESP8266WiFi Module is a self -contained SoC with integrated TCP/IP protocol stack that can give access to your Wi-Fi network (or the device can act as an access point).

2.ARDUINO UNO

Here the Arduino uno is connected to water flow sensors to analyze the water level and temperature sensor to predict the humidity level, Further, these analyzed values will be passed to the Arduino which is been developed with Java, C++. The Arduino would pass the alert message to the IoT module.

3. ULTRASONIC SENSOR

The ultrasonic sensors measure the distance of the water level, and the Arduino micro-controller processes the signals from the sensors.

4.LED

When water level crosses the Intermediate level, the green led will glow and it will also show green alert in Lcd display with water level.

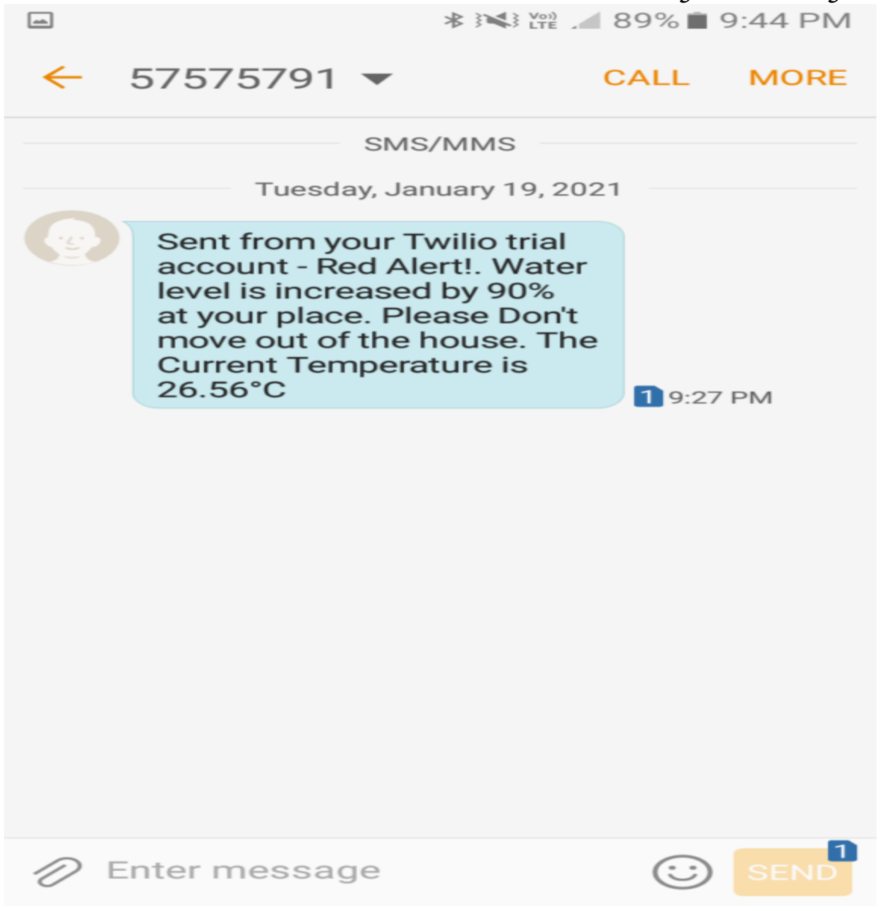
WORKING FLOW

For doing the practical demonstration. First connect the USB cable type-B to the Laptop's USB slot for power supply. Also simultaneously run the python program. Firstly, the ultrasonic sensor will sense the water level in distance and then the Arduino program will help to convert it into percentage. Also, the sensed water level will be displayed on Lcd display (In Percentage) along with zone/area the water level is present.

When water level is at Min/Normal level. That resembles 'Green Alert'. This means that water is at normal position and no sign about flood condition. Also, green led will glow and it will also show green alert in Lcd display with water level.

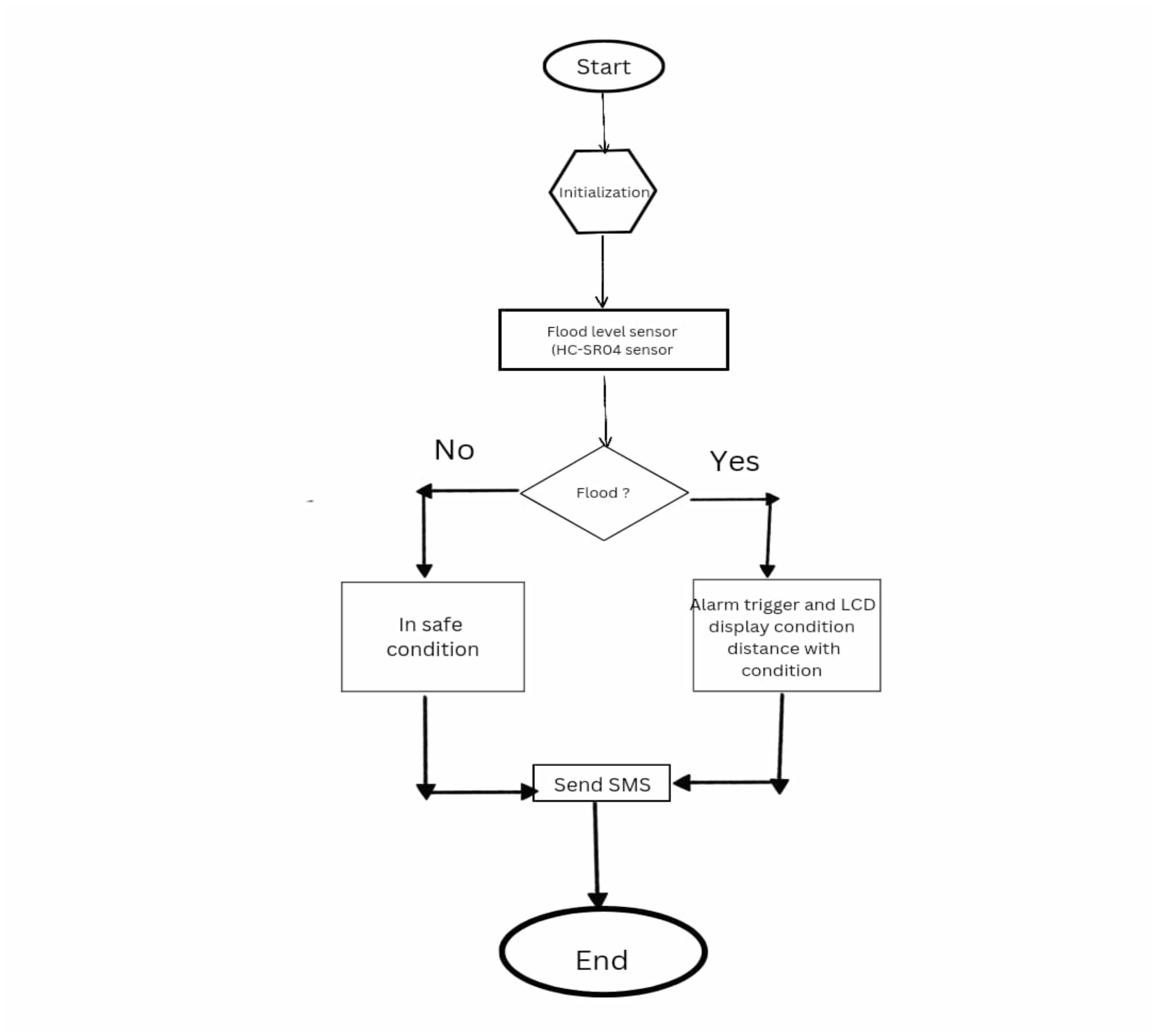
When water level crosses the Intermediate level. That resembles 'Orange Alert'. This means that water has crossed the 55% mark and there can be chances of flood condition at that place. Also, orange led will glow and buzzer will buzz. It will also show orange alert in Lcd display

When the water lever increases than the intermediate level then it alerts through the message like:



Sent from your Twilio trial account - Red Alert!. Water level is increased by 90% at your place. Please Don't move out of the house. The Current Temperature is 26.56°C

FLOWCHART


Start
Initialization
Flood level sensor (HC-SR04 sensor
No
Flood ?
Yes
In safe condition
Alarm trigger and LCD display condition distance with condition
Send SMS
End

PLATFORM DEVELOPMENT

CODING PART

The coding used for implementing flood monitoring and early warning system such as connection between the components and execution are,

Python file :

```
import conf
from boltiot import Sms, Email, Bolt
import json, time
```

```
intermediate_value = 55
max_value = 80


mybolt = Bolt(conf.API_KEY, conf.DEVICE_ID)
sms = Sms(conf.SID, conf.AUTH_TOKEN, conf.TO_NUMBER, conf.FROM_NUMBER)
mailer = Email(conf.MAILGUN_API_KEY, conf.SANDBOX_URL, conf.SENDER_EMAIL,
conf.RECIPIENT_EMAIL)


def twillo_message(message):
try:
print("Making request to Twilio to send a SMS")
response = sms.send_sms(message)
print("Response received from Twilio is: " + str(response))
print("Status of SMS at Twilio is :" + str(response.status))
except Exception as e:
print("Below are the details")
print(e)
def twillo_message(message):
  try:
     print("Making request to Twilio to send a SMS")
     response = sms.send_sms(message)
     print("Response received from Twilio is: " + str(response))
     print("Status of SMS at Twilio is :" + str(response.status))
  except Exception as e:
     print("Below are the details")
     print(e)

def mailgun_message(head,message_1):
  try:
     print("Making request to Mailgun to send an email")
     response = mailer.send_email(head,message_1)
     print("Response received from Mailgun is: " + response.text)
  except Exception as e:
     print("Below are the details")
     print(e)

while True:
    print ("Reading Water-Level Value")
    response_1 = mybolt.serialRead('10')
    response = mybolt.analogRead('A0')
    data_1 = json.loads(response_1)
    data = json.loads(response)
    Water_level = data_1['value'].rstrip()
    print("Water Level value is: " + str(Water_level) + "%")
    sensor_value = int(data['value'])
    temp = (100*sensor_value)/1024
    temp_value = round(temp,2)
    print("Temperature is: " + str(temp_value) + "°C")
   try:
      if int(Water_level) >= intermediate_value:
           message ="Orange Alert!. Water level is increased by " +str(Water_level) + "% at
your place. Please be Safe. The current Temperature is " + str(temp_value) + "°C."
           head="Orange Alert"
           message_1="Water level is increased by " + str(Water_level) + "% at your place.
Please be Safe. The current Temperature is " + str(temp_value) + "°C."
           twillo_message(message)
           mailgun_message(head,message_1)

        if int(Water_level) >= max_value:
           message ="Red Alert!. Water level is increased by " + str(Water_level) + "% at your
place. Please Don't move out of the house. The Current Temperature is " + str(temp_value)
+ "°C"
           head="Red Alert!"
           message_1="Water level is increased by " + str(Water_level) + "% at your place.
Please Don't move out of the house. The Current Temperature is " + str(temp_value) + "°C."
           twillo_message(message)
           mailgun_message(head,message_1)

    except Exception as e:
        print ("Error occured: Below are the details")
        print (e)
    time.sleep(15)
```

Aurduino code

```
//IOT Based Flood Monitoring And Alerting System.
#include<LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

const int in = 8;
const int out = 9;
const int green = 10;
const int orange = 11;
const int red = 12;
const int buzz = 13;
void setup() {
Serial.begin(9600);
lcd.begin(16, 2);
pinMode( in , INPUT);
pinMode(out, OUTPUT);
pinMode(green, OUTPUT);
pinMode(orange, OUTPUT);
pinMode(red, OUTPUT);
pinMode(buzz, OUTPUT);
digitalWrite(green, LOW);
digitalWrite(orange, LOW);
digitalWrite(red, LOW);
digitalWrite(buzz, LOW);
lcd.setCursor(0, 0);
lcd.print("Flood Monitoring");
lcd.setCursor(0, 1);
lcd.print("Alerting System");
delay(5000);
lcd.clear();
}
void loop() {
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;
per = map(dist, 10.5, 2, 0, 100);
#map
function is used to convert the distance into percentage.
if(per < 0) {
per = 0;
}
if (per > 100) {
per = 100;
}
Serial.println(String(per));
lcd.setCursor(0, 0);
lcd.print("Water Level:");
lcd.print(String(per));
lcd.print("% ");
if (per >= 80) #MAX Level of Water--Red Alert!{
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 (per >= 55) #Intermedite Level of Water--Orange Alert!!

```
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 #MIN / NORMAL level of Water--Green Alert!!
lcd.setCursor(0, 1);
lcd.print("Green Alert! ");
digitalWrite(green, HIGH);
digitalWrite(orange, LOW);
digitalWrite(red, LOW);
digitalWrite(buzz, LOW);
}

delay(15000);
}
```

Sketch.ino:

A sketch can be written in any text editor, but the Arduino IDE is a popular choice because it provides a number of features that are specific to Arduino programming, such as syntax highlighting, code completion, and a built-in serial monitor.

A sketch typically consists of two functions:

1. setup (): This function is called once, when the Arduino board first starts up. It is used to

initialize the board and its peripherals.

2. loop (): This function is called repeatedly, over and over again. It is used to implement the main logic of the sketch.

In addition to these two functions, a sketch can also contain other functions, such as functions to handle interrupts or to perform specific tasks.

To upload a sketch to an Arduino board, you can use the Arduino IDE. Simply connect the board to your computer using a USB cable, select the board and port in the Arduino IDE, and then click on the Upload button.

Once the sketch has been uploaded, it will start running immediately. You can use the serial monitor to view the output of the sketch and to interact with it.

Here is an example of a simple sketch.ino code:

```
void setup () {
  // set pin 13 as an output pin
  pinMode(13, OUTPUT);
}

void loop() {
  // turn on the LED
  digitalWrite(13, HIGH);
  // wait for one second
  delay(1000);
  // turn off the LED
  digitalWrite(13, LOW);
  // wait for one second
  delay(1000);
}
```

Sketch.ino:

```
//<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;
```

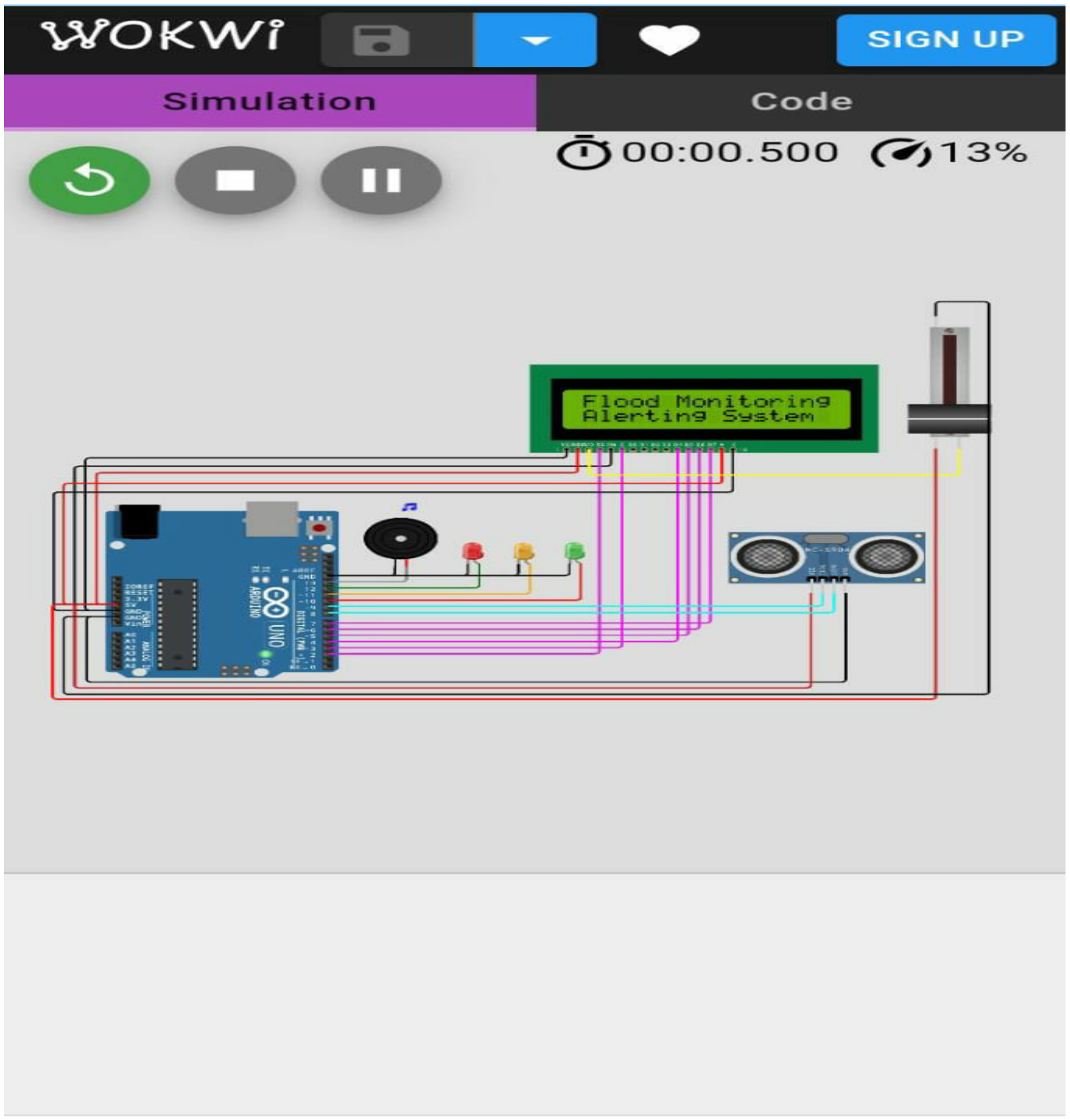
```
}
// Print water level data to serial
Serial.print("Water Level:");
Serial.println(String(per));
lcd.setCursor(0, 0);
lcd.print("Water Level:");
lcd.print(String(per));
lcd.print("%  ");
// Check water level and set alert levels
if (dist <= 3)
{
  lcd.setCursor(0, 1);
  lcd.print("Red Alert!   ");
  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);
  }
}
```

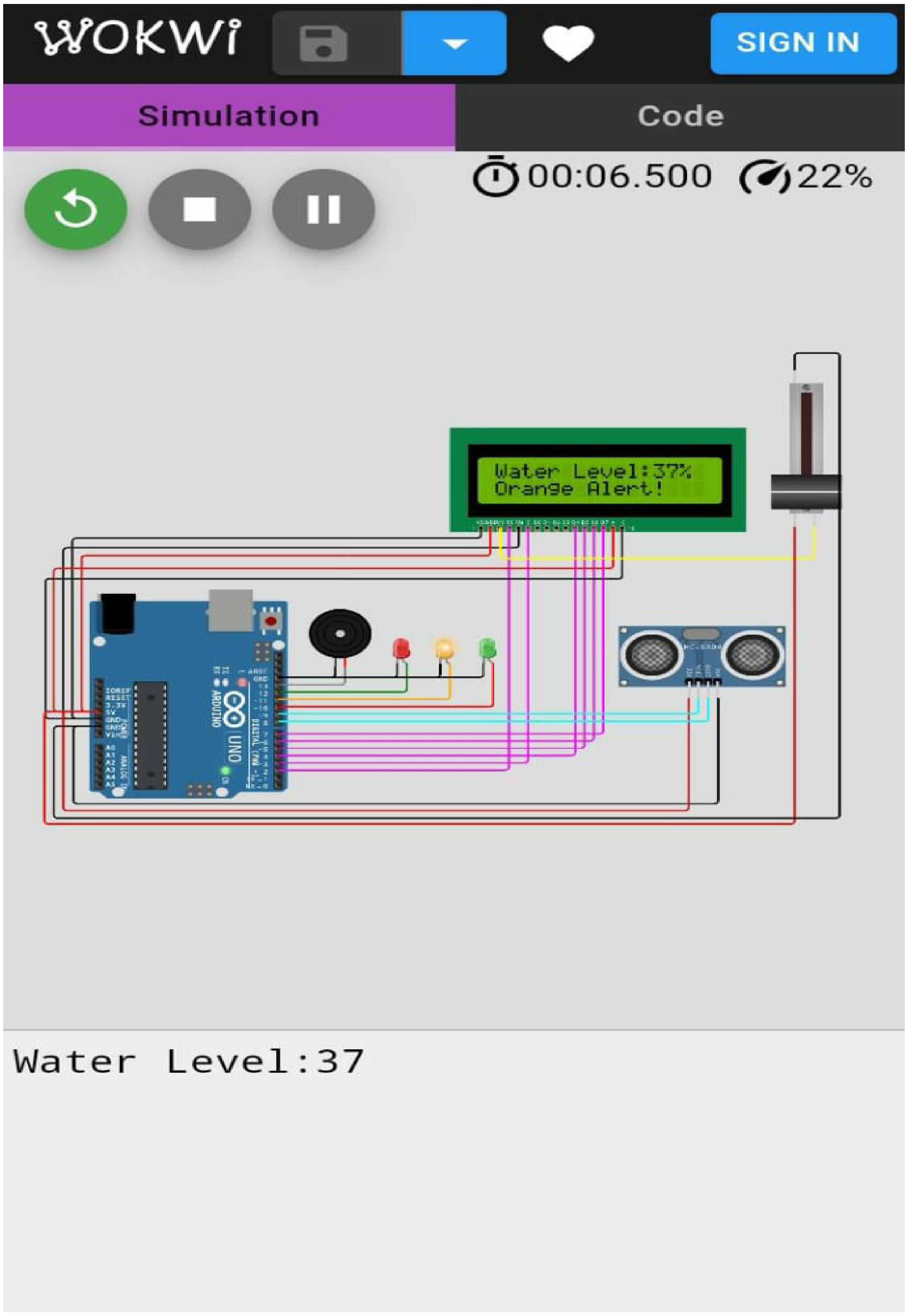
Diagram.json:

```
{
  "version": 1,
  "author": "Warship Battle",
  "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": {}
}
```

Output :

Simulating with code in the Wokwi Simulator:


WOKWI
SIGN UP
Simulation
Code
00:00.500
13%
Flood Monitoring
Alerting System



```
Water Level:37
```

CONCLUSION :

In conclusion, flood monitoring and early warning systems play a critical role in mitigating the devastating impact of floods. By utilizing advanced technologies and real-time data, these systems can provide timely alerts and valuable information to both authorities and the public, allowing for better preparedness and response. However, their effectiveness relies on continuous improvements, community education, and government support to ensure that they are well-maintained and accessible to all vulnerable populations. It's imperative to recognize the importance of these systems in saving lives, protecting property, and reducing the overall socio-economic consequences of flooding.