Flood Monitoring and Early Warning system using IoT

Team Member:

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

Phase-5 Final 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.

Objective:

The main objective of the Flood Monitoring and Early Warning System using IoT is to develop a system that provides real-time data collection, predictive analysis, and early warning alerts, aimed at enhancing disaster preparedness and minimizing the impact of flooding in vulnerable areas.

Components Required:

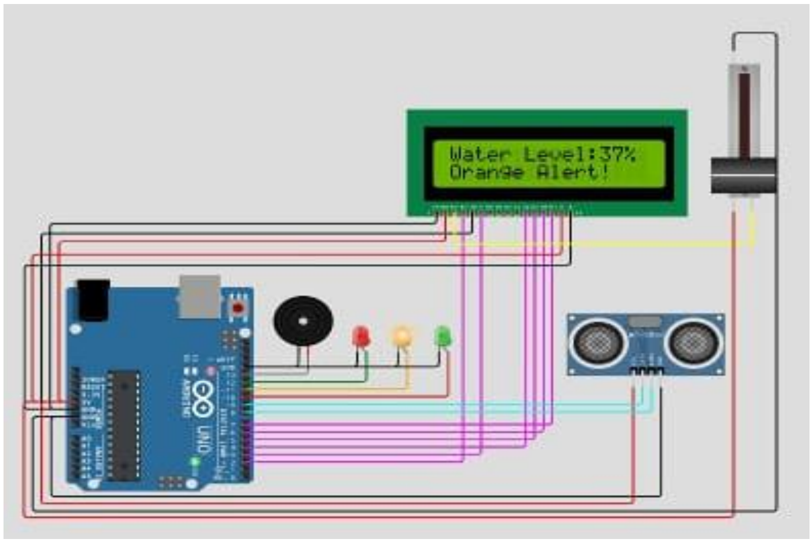
These are the approximately estimated components that are needed for the Flood monitoring and early warning system:

<u>Hardware components:</u>

- ESP32
- Arduino UNO
- Breadboard
- 5mm LED and Buzzer
- 16×2 LCD Display
- LM35 Temperature Sensor
- HC-SR04 Ultrasonic Sensor
- Some Jumper Wires
- Male to Female Jumper Wires
- Male to Male Jumper Wires
- Female to Female Jumper Wires
- 9v Battery and Snap Connector

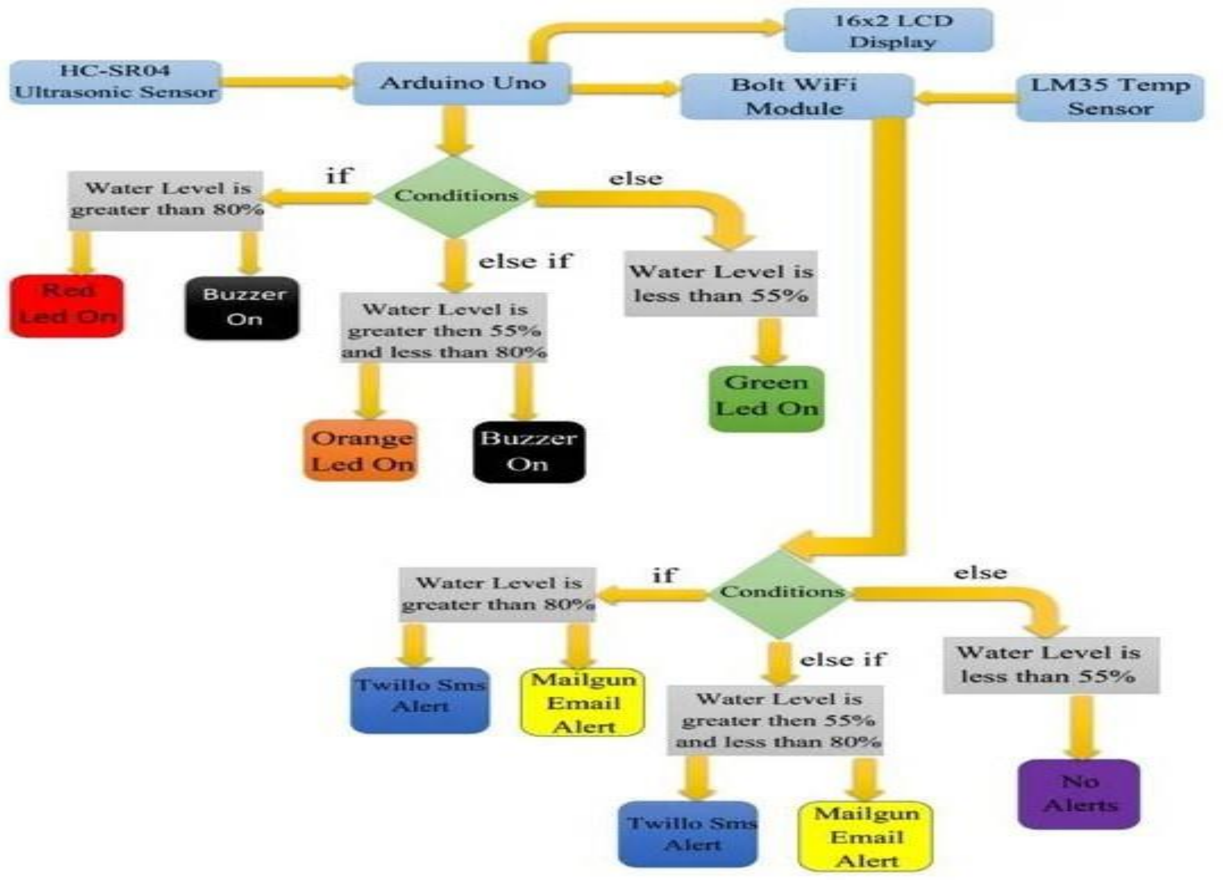
<u>Software components:</u>

- Arduino IDE
- Python 3.7 IDLE
- IBM IoT Cloud
- Twillo SMS Messaging API
- Mailgun EMAIL Messaging API Software components



Working of the System:

- This IoT-based flood monitoring system uses an ESP32 microcontroller to collect data from an ultrasonic sensor and send it to ThingSpeak for graphical monitoring and critical alerts. The Arduino UNO also controls a red LED and buzzer, which are used to alert people in the event of a flood. A green LED is used to indicate normal conditions.
- Arduino Uno reads the data from the ultrasonic sensor and determines if the water level has exceeded the threshold. If it has, the Arduino Uno sends an alert to ThingSpeak, which will then send an alert to people via SMS, email, or another method. The Arduino Uno also turns on the red LED and buzzer to alert people locally.
- The green LED is used to let people know that the water level is normal and there is no need for concern.
- This system is a valuable tool for flood monitoring and early warning. By providing early notice of rising water levels, it can help people to evacuate safely and protect their property.
- The system will generate an alert that will be sent to people via LED signals, a buzzer, SMS messages, and emails.

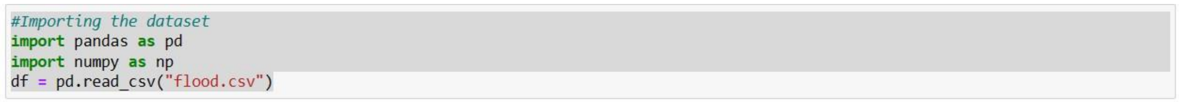


16x2 LCD Display
HC-SR04 Ultrasonic Sensor
Arduino Uno
Bolt WiFi Module
LM35 Temp Sensor
if
Conditions
else
Water Level is greater than 80%
else if
Water Level is less than 55%
Red Led On
Buzzer On
Water Level is greater then 55% and less than 80%
Green Led On
Orange Led On
Buzzer On
Water Level is greater than 80%
if
Conditions
else
else if
Water Level is less than 55%
Twillo Sms Alert
Mailgun Email Alert
Water Level is greater then 55% and less than 80%
No Alerts
Twillo Sms Alert
Mailgun Email Alert

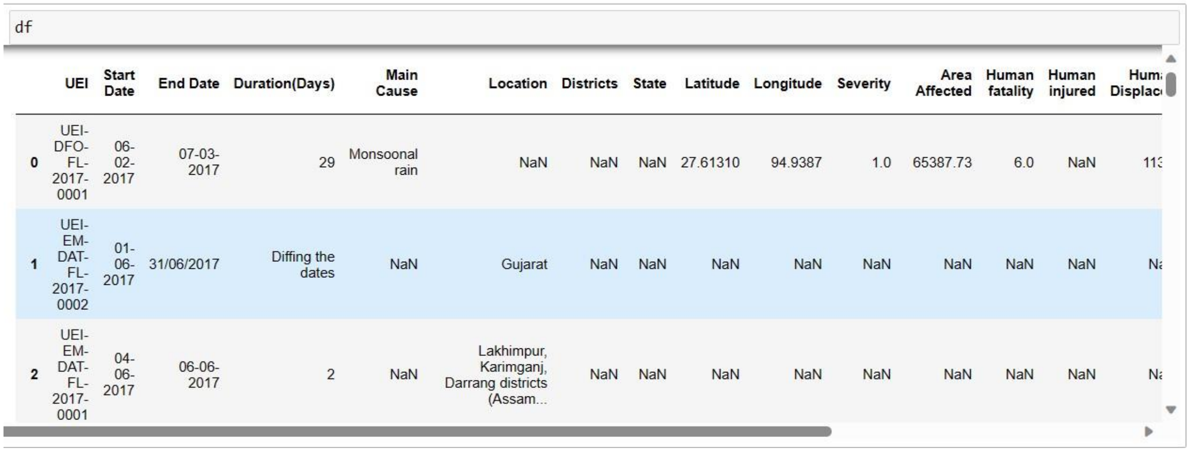
Working with flood dataset for analysing the water levels and predicting the flood:

Cleaning the Dataset:

- First, we download the dataset from the link: https://www.kaggle.com/datasets/aditya2803/india-floods-inventory
- Then we import the dataset into the program and clean the dataset.
- Data cleaning is the process of identifying and correcting errors and inconsistencies in a dataset.

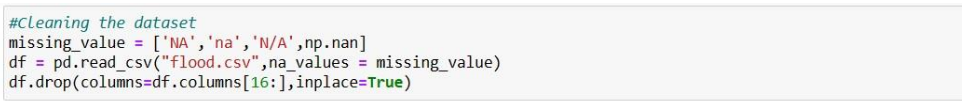


```
#Importing the dataset
import pandas as pd
import numpy as np
df = pd.read_csv("flood.csv")
```

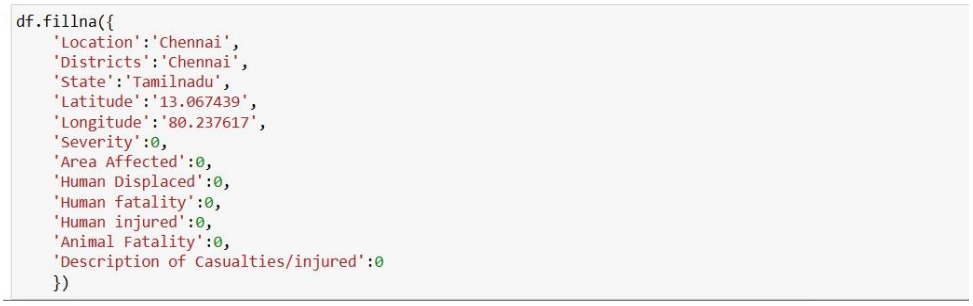


```
df
```

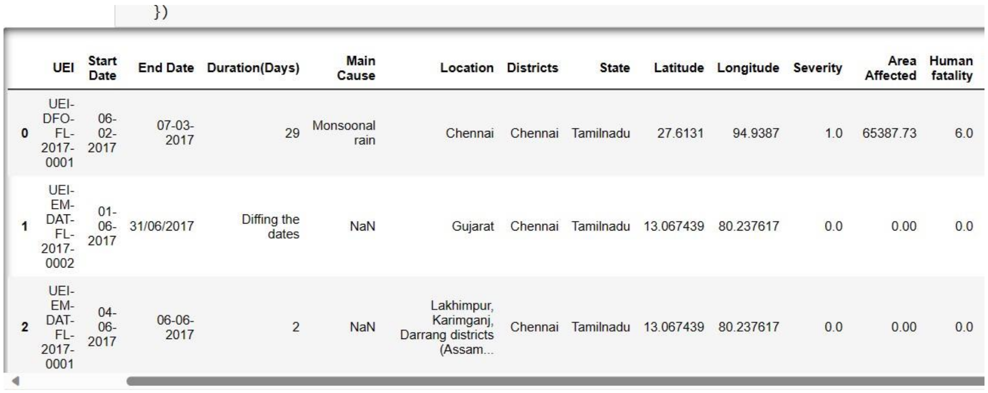
	UEI	Start Date	End Date	Duration(Days)	Main Cause	Location	Districts	State	Latitude	Longitude	Severity	Area Affected	Human fatality	Human injured	Hum Displac
0	UEI-DFO-FL-2017-0001	06-02-2017	07-03-2017	29	Monsoonal rain	NaN	NaN	NaN	27.61310	94.9387	1.0	65387.73	6.0	NaN	113
1	UEI-EM-DAT-FL-2017-0002	01-06-2017	31/06/2017	Diffing the dates	NaN	Gujarat	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	UEI-EM-DAT-FL-2017-0001	04-06-2017	06-06-2017	2	NaN	Lakhimpur, Karimganj, Darrang districts (Assam...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na



```
#Cleaning the dataset
missing_value = ['NA','na','N/A',np.nan]
df = pd.read_csv("flood.csv",na_values = missing_value)
df.drop(columns=df.columns[16:],inplace=True)
```



```
df.fillna({
    'Location':'Chennai',
    'Districts':'Chennai',
    'State':'Tamilnadu',
    'Latitude':'13.067439',
    'Longitude':'80.237617',
    'Severity':0,
    'Area Affected':0,
    'Human Displaced':0,
    'Human fatality':0,
    'Human injured':0,
    'Animal Fatality':0,
    'Description of Casualties/injured':0
    })
```



```
    })
```

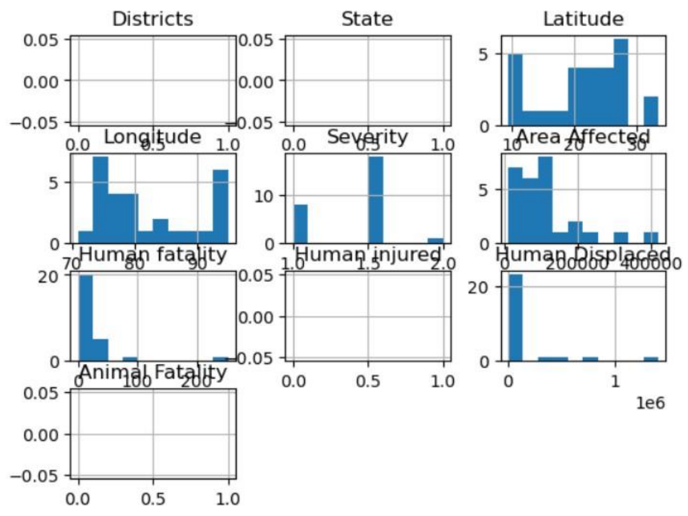
	UEI	Start Date	End Date	Duration(Days)	Main Cause	Location	Districts	State	Latitude	Longitude	Severity	Area Affected	Human fatality
0	UEI-DFO-FL-2017-0001	06-02-2017	07-03-2017	29	Monsoonal rain	Chennai	Chennai	Tamilnadu	27.6131	94.9387	1.0	65387.73	6.0
1	UEI-EM-DAT-FL-2017-0002	01-06-2017	31/06/2017	Diffing the dates	NaN	Gujarat	Chennai	Tamilnadu	13.067439	80.237617	0.0	0.00	0.0
2	UEI-EM-DAT-FL-2017-0001	04-06-2017	06-06-2017	2	NaN	Lakhimpur, Karimganj, Darrang districts (Assam...	Chennai	Tamilnadu	13.067439	80.237617	0.0	0.00	0.0

Analysing the Dataset:

- Data analysis is the process of collecting, cleaning, and examining data to extract meaningful insights.
- Exploratory data analysis (EDA) is a process of visually exploring the data to identify patterns and anomalies. This can be done using a variety of charts and graphs.

```
#Analysing the dataset
```

```
import matplotlib.pyplot as plt
df.hist()
plt.show()
```



IoT Sensor Deployment:

ESP32 microcontroller:

- ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and Bluetooth. It is a popular choice for Internet of Things (IoT) applications, as it is easy to use and program, and has a wide range of features.
- The ESP32 includes a dual-core Tensilica Xtensa LX6 microprocessor, up to 320 KiB of SRAM, and 4 MB of flash memory. It also has a wide range of peripherals, including Wi-Fi, Bluetooth, UART, SPI, I2C, ADC, DAC, and GPIO.
- The ESP32 can be programmed using a variety of languages, including C/C++, Arduino IDE, and MicroPython. It is also supported by a wide range of development boards and modules, making it easy to get started with IoT development.

Ultrasonic Sensor:

- An ultrasonic sensor is an electronic device that measures the distance to a target object by emitting ultrasonic sound waves and converts the reflected sound into an electrical signal.
- Ultrasonic sensors work by sending out a short pulse of ultrasonic sound and then measuring the time it takes for the echo to return. The speed of sound in air is known, so the distance to the target object can be calculated by multiplying the time delay by the speed of sound.

Temperature Sensor:

- A temperature sensor is a device that measures the temperature of an object or environment.
- It converts the temperature into an electrical signal that can be read by a computer or other electronic device.

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.
- The file Sketch.ino is in the link:

https://github.com/gowsalyanagaraj/IoT_Phases/tree/295cb746cd2d66b7f2b8f57f2c6b084b69e6425a/wokwi-codes

```
//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 file diagram.json is in the link:
  https://github.com/gowsalyanagaraj/IoT_Phases/tree/295cb746cd2d66b7f2b8f57f2c6b084b69e6425a/wokwi-codes

About libraries.txt:

The following libraries are included in the project:

- LiquidCrystal: This library is used to control a 16x2 character LCD display.

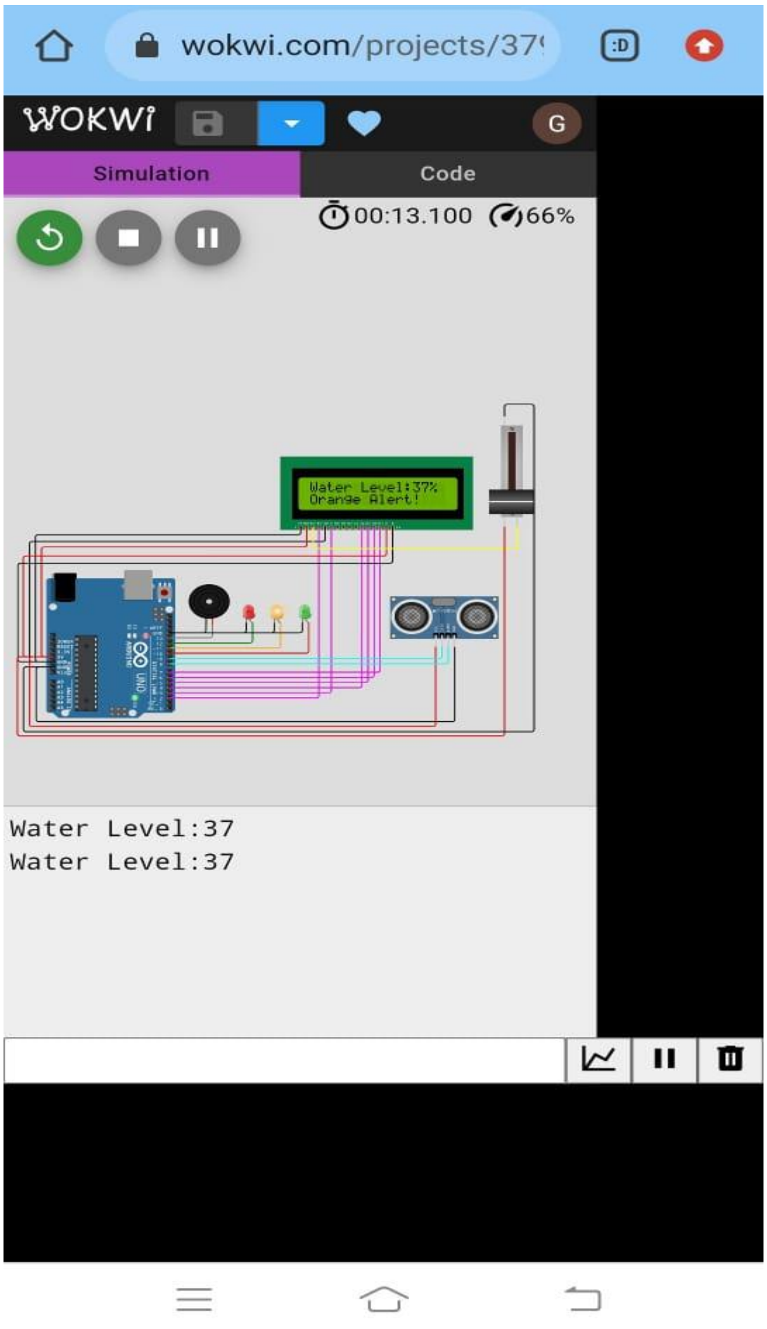
- DHT22: This library is used to read data from a DHT22 temperature and humidity sensor.
- DHT sensor library: This library is a generic library that can be used to read data from any DHT sensor.

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.

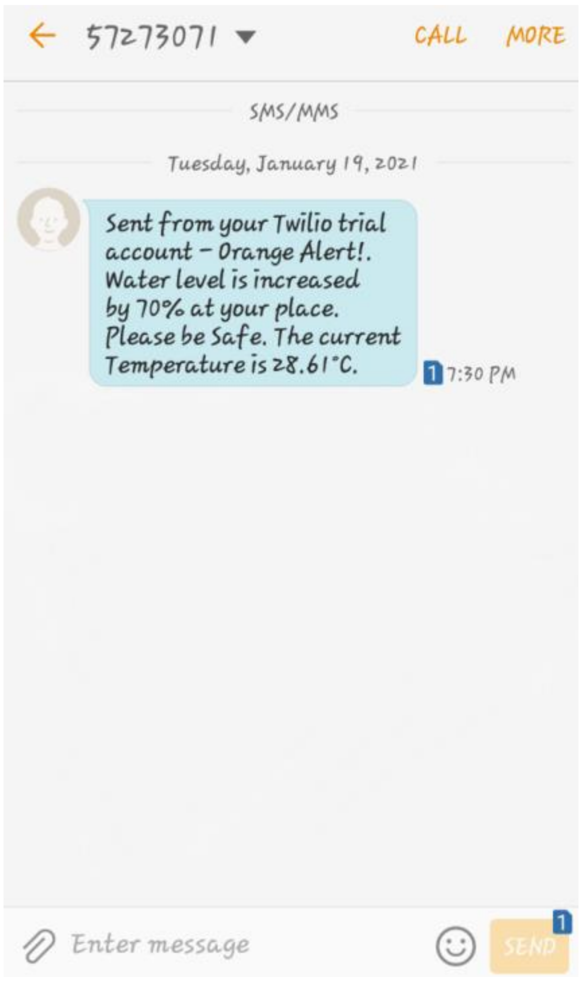


```
Water Level:37
Water Level:37
```

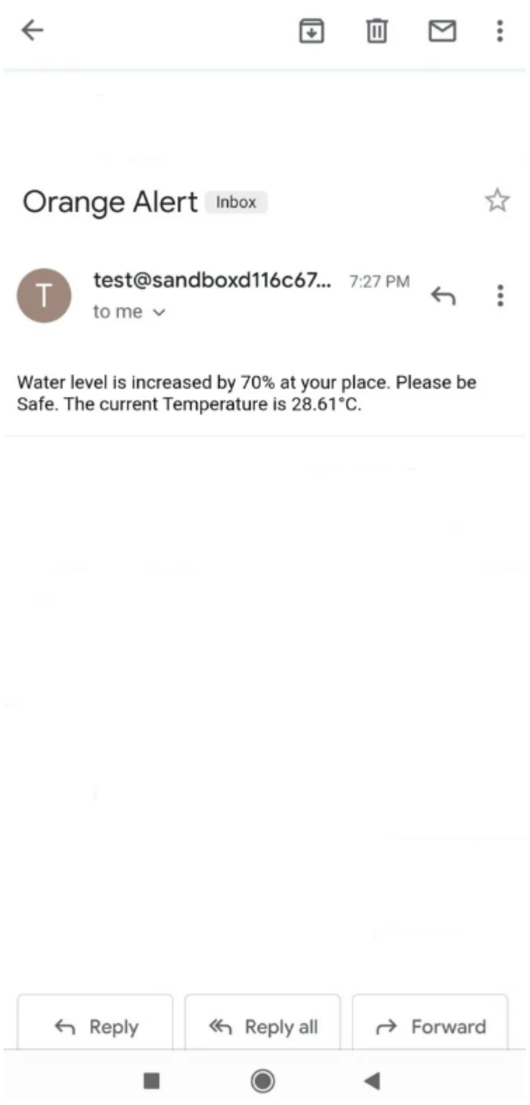
Email and SMS software:

- To connect the above project to Twilio and Mailgun APIs for sending SMS and email messages, we will need to create accounts with Twilio and Mailgun.
- Once we have created accounts, we will need to obtain our Twilio account SID, Twilio auth token, Mailgun domain name, and Mailgun API key.
- Once we have obtained our Twilio and Mailgun credentials, we can update the Arduino Uno program to send SMS and email messages.

- Write the following python script to connect the Twillo and Mailgun: https://github.com/gowsalyanagaraj/IoT_Phases/tree/5f4a8e897510bd01efd67cbbf89e826361ec6f0c/email_and_sms_software



Sent from your Twilio trial account - Orange Alert!. Water level is increased by 70% at your place. Please be Safe. The current Temperature is 28.61°C.


Orange Alert

test@sandboxd116c67... 7:27 PM
to me

Water level is increased by 70% at your place. Please be Safe. The current Temperature is 28.61°C.

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.