PHASE-4

PROJECT TITLE  
**FLOOD MONITORING AND EARLY WARNING SYSTEM**

Development Part 2 - Building the Early Warning Platform

Table of Contents

# 1. Introduction

- Overview of Project Phases

- Purpose of Phase 4

# 2. Continuing the Project Development

- Deploying IoT Sensors

- Developing the Early Warning Platform

- Technologies Used

- Platform Features

# 3. User Interface Design

- Design Principles

- Real-time Data Display

- Flood Warning Notifications

# 4. Data Integration

- Connecting IoT Sensors

- Data Reception and Processing

- Alert Generation

# 5. Testing and Validation

- Unit Testing

- Integration Testing

- User Acceptance Testing

# 6. Challenges and Solutions

- Data Accuracy and Reliability

- Performance and Scalability

- User-Friendly Interface

- Security and Privacy

# 7. Expected Outcomes

- A Functional Early Warning Platform

- Seamless Data Integration with IoT Sensors

- Improved Public Safety

# 8. Conclusion

- Summary of Phase 4

- Transition to Phase 5

# 1. Introduction:

## Overview of Project Phases:

The fourth phase of our Flood Monitoring System project is now complete. The project's goals were established, the IoT sensor network was created, the early warning platform was created, and they were all integrated utilizing Python and IoT technology in the earlier phases. We will concentrate on creating the early warning platform throughout this phase.

## Purpose of Phase 4:

Building a strong and user-friendly early warning platform is Phase 4's main objective. Real-time water level information will be presented on this platform, and flood alerts will be sent to the general public and emergency response teams through it.

# 2. Continuing the Project Development:

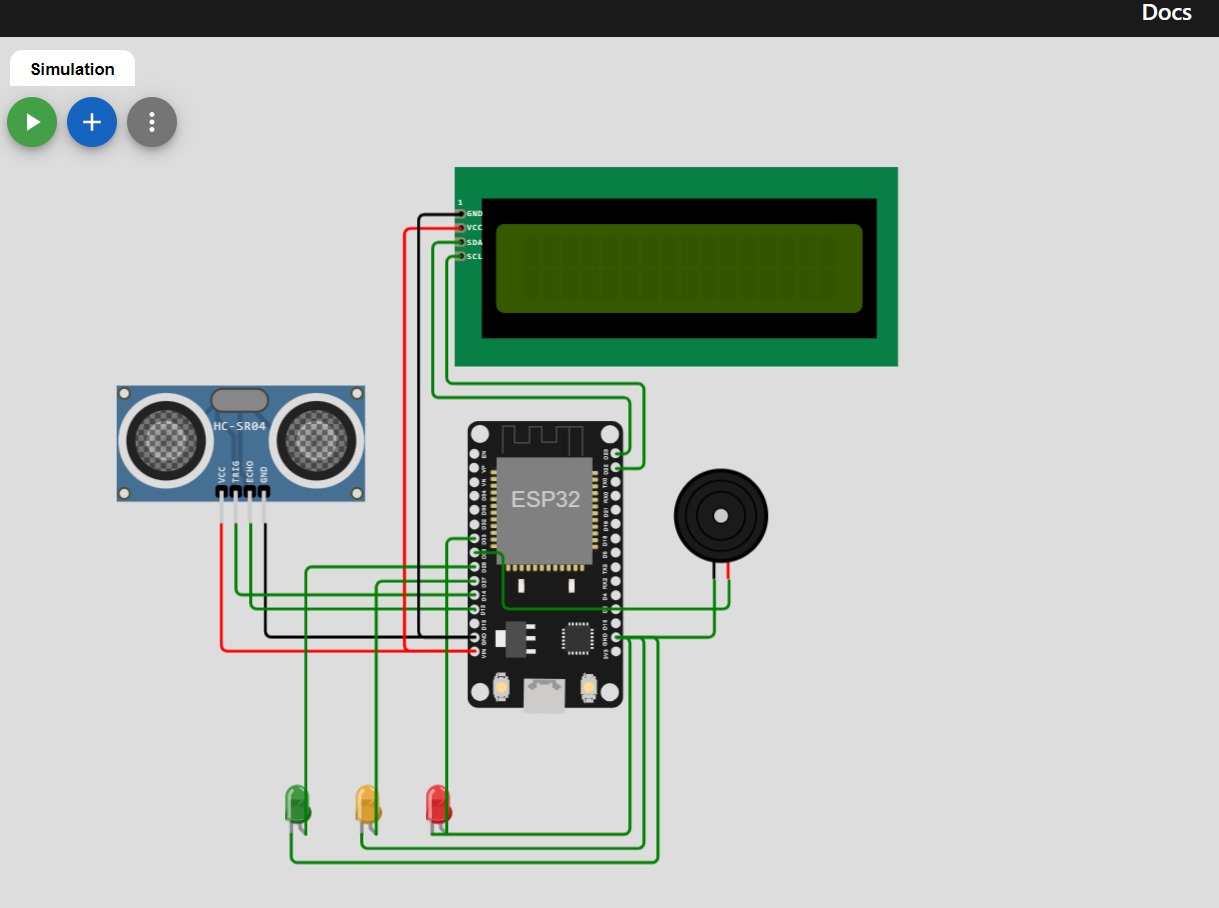
## Utilizing IoT Sensors:

IoT sensors were placed in flood-prone regions in Phase 3 to gauge water levels. Phase 4 is predicated on the presence and functionality of these sensors.

**Wiring Connection Steps:**

1. **HC-SR04 Ultrasonic Sensor:**
   * Connect the VCC pin of the HC-SR04 to the 5V power supply.
   * Connect the GND pin of the HC-SR04 to the ground (GND) of the ESP32.
   * Connect the TRIGGER pin of the HC-SR04 to GPIO pin 14 on the ESP32.
   * Connect the ECHO pin of the HC-SR04 to GPIO pin 12 on the ESP32.
2. **16x2 Character LCD Display:**
   * Connect the VCC (power) pin of the LCD to the 5V power supply.
   * Connect the GND (ground) pin of the LCD to the ground (GND) of the ESP32.
   * Connect the SDA (Serial Data) pin of the LCD to GPIO pin 21 on the ESP32.
   * Connect the SCL (Serial Clock) pin of the LCD to GPIO pin 22 on the ESP32.
3. **LEDs (Green, Orange, Red):**
   * Connect the anode (longer lead) of the green LED to GPIO pin 26 on the ESP32.
   * Connect the anode of the orange LED to GPIO pin 27 on the ESP32.
   * Connect the anode of the red LED to GPIO pin 33 on the ESP32.
   * Connect the cathodes (shorter leads) of all three LEDs to a common ground (GND) on the ESP32.

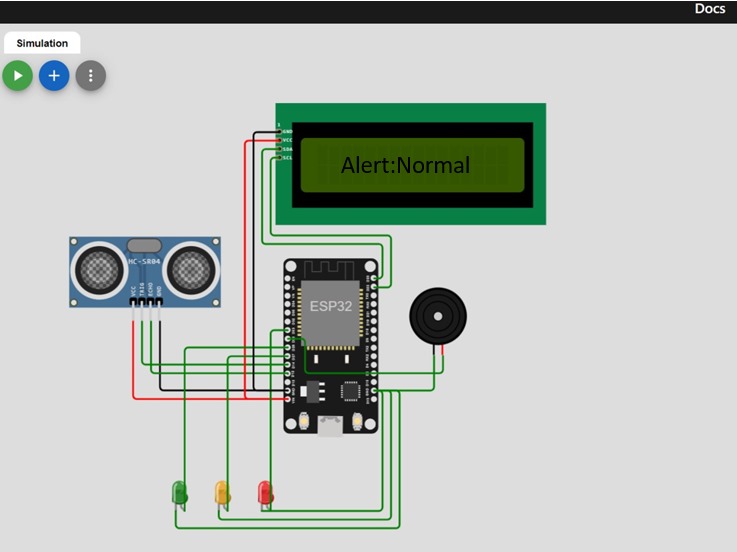
**CIRCUIT:**



**OUTPUTS BASED ON DIFFERENT WATER LEVELS:**

1. **Normal Water Level (Green Alert):**
   * The LCD will display: "Alert: Normal"
   * The green LED will be turned on.
   * The orange and red LEDs will be turned off.
   * The buzzer will be turned off.
2. **Rising Water Level (Orange Alert):**
   * The LCD will display: "Caution: Rising"
   * The green LED will be turned off.
   * The orange LED will be turned on.
   * The red LED and buzzer will be turned off.
3. **Flooding Detected (Red Alert):**
   * The LCD will display: "Alert: Flooding"
   * The green and orange LEDs will be turned off.
   * The red LED and buzzer will be turned on.

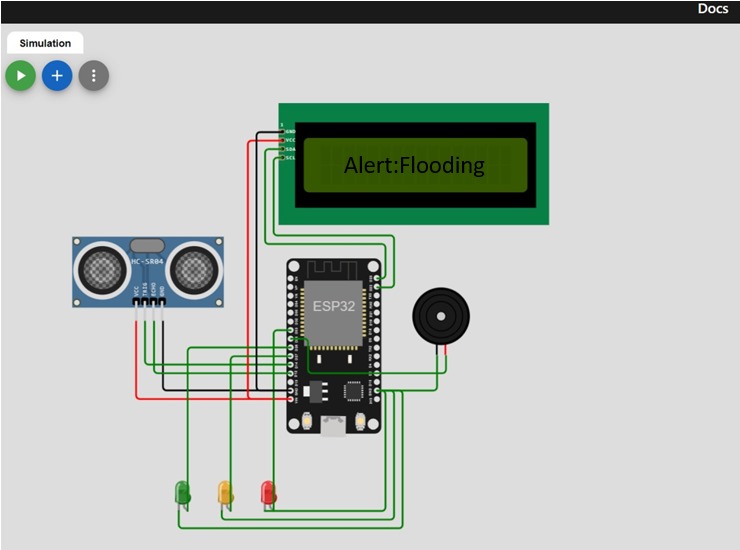
**OUTPUT: 1,Normal Water Level(Green Alert)**



## **2,Rising Water Level (Orange Alert)**

## 

## **3,Flooding Detected(Red alert)**



## Making progress on the Early Warning Platform:

Building the early warning platform, which will process data from IoT sensors and send out flood warnings, will be the main objective of this phase.

## Technologies Used:

Web technologies including HTML, CSS, and JavaScript will be used in the platform's development for the user interface. Data processing and alert generating may both be accomplished using server-side technology.

## Platform Features:

The early warning technology will be built to deliver water level data in real-time and send out flood alerts when water levels cross crucial thresholds. Users can use a web browser to access the platform.

# PYTHON SCRIPT:

import machine

import time

import ubinascii

from umqtt.simple import MQTTClient

# DefinesGPIO pins for components

TRIGGER\_PIN = 14

ECHO\_PIN = 12

LCD\_SCL\_PIN = 22

LCD\_SDA\_PIN = 21

GREEN\_LED\_PIN = 26

ORANGE\_LED\_PIN = 27

RED\_LED\_PIN = 33

BUZZER\_PIN = 25

# Defines MQTT settings

MQTT\_BROKER = "your\_mqtt\_broker\_address"

MQTT\_PORT = 1883

MQTT\_TOPIC = "flood\_alerts"

# Initialized components

trigger = machine.Pin(TRIGGER\_PIN, machine.Pin.OUT)

echo = machine.Pin(ECHO\_PIN, machine.Pin.IN)

lcd = machine.I2c(scl=machine.Pin(LCD\_SCL\_PIN), sda=machine.Pin(LCD\_SDA\_PIN))

green\_led = machine.Pin(GREEN\_LED\_PIN, machine.Pin.OUT)

orange\_led = machine.Pin(ORANGE\_LED\_PIN, machine.Pin.OUT)

red\_led = machine.Pin(RED\_LED\_PIN, machine.Pin.OUT)

buzzer = machine.Pin(BUZZER\_PIN, machine.Pin.OUT)

# Function to measure distance using HC-SR04 sensor

def measure\_distance():

trigger.value(1)

time.sleep\_us(10)

trigger.value(0)

while echo.value() == 0:

pulse\_start = time.ticks\_us()

while echo.value() == 1:

pulse\_end = time.ticks\_us()

duration = time.ticks\_diff(pulse\_end, pulse\_start)

distance = (duration / 2) / 29.1 # Speed of sound: 343 m/s

return distance

# Create a unique client ID for MQTT

client\_id = ubinascii.hexlify(machine.unique\_id())

# Connect to the MQTT broker

client = MQTTClient(client\_id, MQTT\_BROKER, MQTT\_PORT)

client.connect()

# Define water level thresholds for different alerts

GREEN\_THRESHOLD = 20 # Normal water level

ORANGE\_THRESHOLD = 10 # Caution: Rising water level

RED\_THRESHOLD = 5 # Alert: Flooding

while True:

distance = measure\_distance()

# Determine the alert level based on distance

if distance < RED\_THRESHOLD:

alert\_level = "red"

elif distance < ORANGE\_THRESHOLD:

alert\_level = "orange"

else:

alert\_level = "green"

# Set LED indicators and buzzer

if alert\_level == "green":

green\_led.on()

orange\_led.off()

red\_led.off()

buzzer.off()

elif alert\_level == "orange":

green\_led.off()

orange\_led.on()

red\_led.off()

buzzer.off()

else:

green\_led.off()

orange\_led.off()

red\_led.on()

buzzer.on()

# Publish the alert to the MQTT broker

client.publish(MQTT\_TOPIC, alert\_level)

time.sleep(2)

CODING FOR WEB APPLICATION:

<!DOCTYPE html>

<html>

<head>

<title>Flood Monitoring Alerts</title>

</head>

<body>

<h1>Flood Monitoring Alerts</h1>

<div id="alertMessage"></div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js"></script>

<script>

// MQTT Broker settings

var broker = "your\_mqtt\_broker\_address"; // Replace with your MQTT broker address

var port = 1883;

var topic = "flood\_alerts"; // MQTT topic for flood alerts

// Create an MQTT client

var client = new Paho.MQTT.Client(broker, port, "web\_client");

// Callback for when the client connects

client.onConnectionLost = function(responseObject) {

if (responseObject.errorCode !== 0) {

console.log("Connection lost: " + responseObject.errorMessage);

}

};

// Callback for when the client receives a message

client.onMessageArrived = function(message) {

var alertMessage = document.getElementById("alertMessage");

alertMessage.innerHTML = "Alert: " + message.payloadString;

};

// Connect to the MQTT broker

client.connect({

onSuccess: function() {

client.subscribe(topic);

},

useSSL: true, // Enable SSL if your broker supports it

});

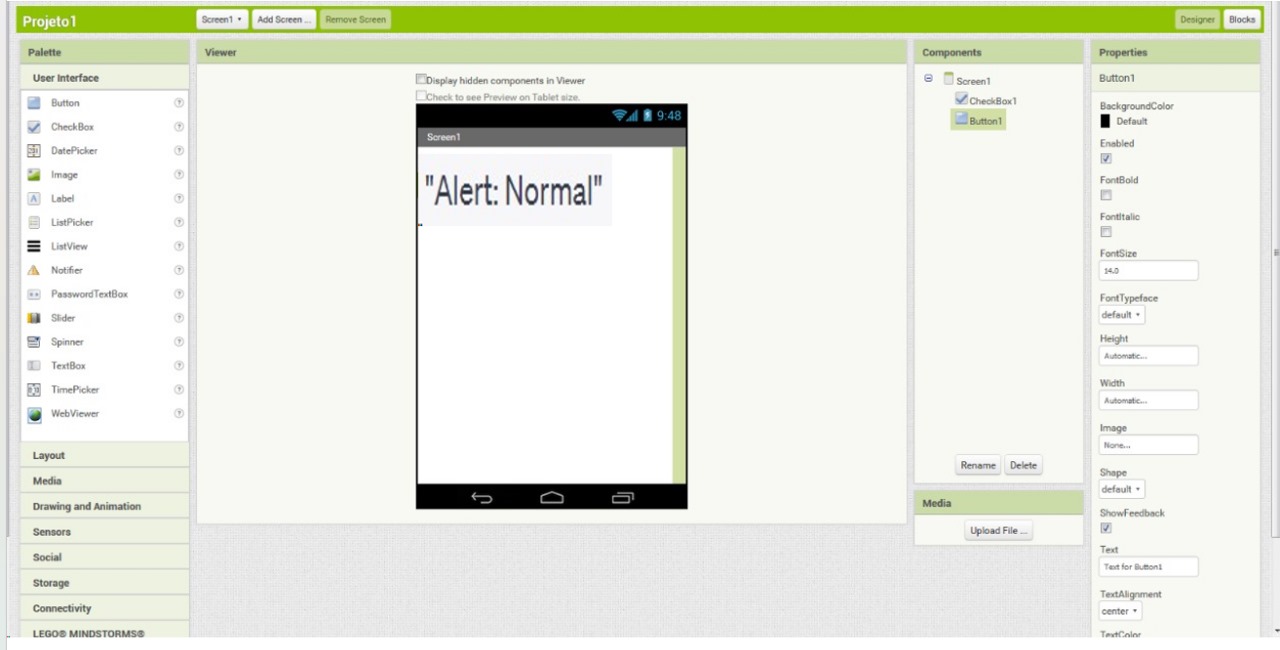
</script>

</body>

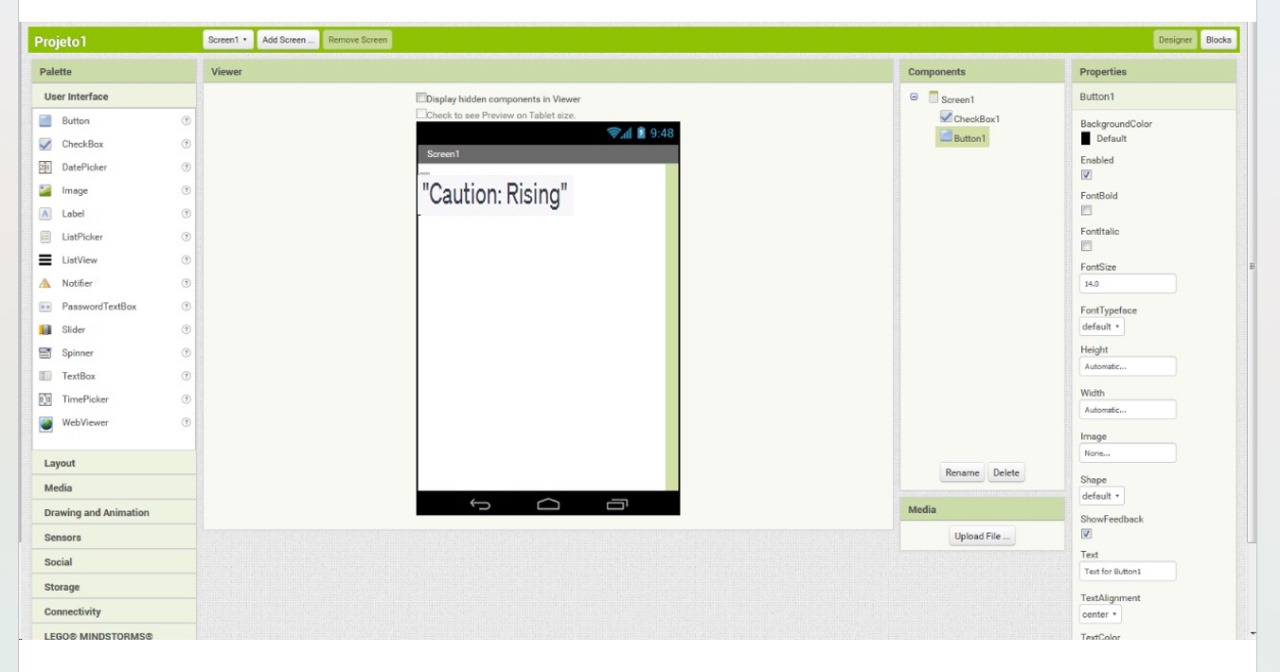
</html>

ALERTS SHOWN IN WEB APPLICATION:

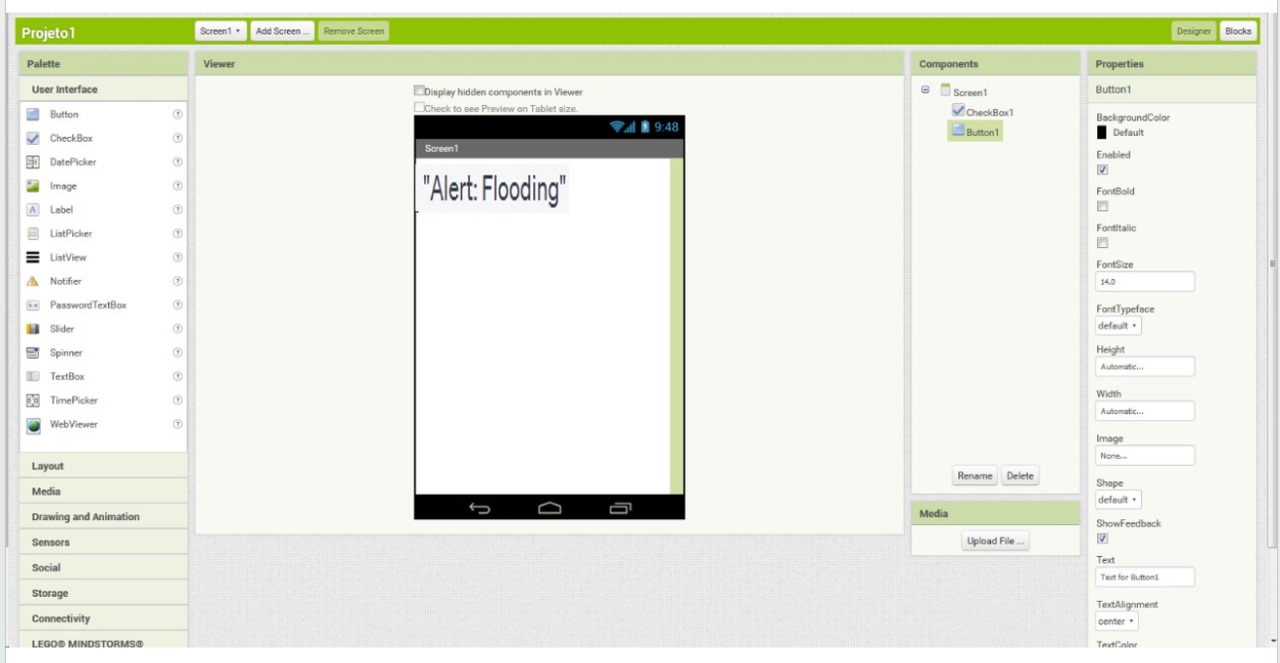
1, NORMAL WATER LEVEL(GREEN ALERT)



2, RISING WATER LEVEL(ORANGE ALERT)



3, FLOODING DETECTED(RED ALERT)



# 3. User Interface Design:

## Design Principles :

User-friendliness and clarity will be prioritized in the platform's design. It should be simple to use and give the general public and emergency response teams precise information.

## Display of Real-time Data:

Data on water levels will be updated in real-time on the platform. This comprises both graphic and numerical values.

## Flood Warning Notices in :

The platform will produce flood warnings if water levels rise above set thresholds. These alerts could come in the form of text messages and visual alerts.

# 4. Data Integration:

## Attaching IoT Sensors :

The early warning platform will be linked to the IoT sensors that were deployed in Phase 3. Security safeguards and data transmission mechanisms will be in place.

## Processing and Receiving of Data:

The platform will gather data from the Internet of Things sensors, process it, and present it in an intuitive manner. We'll preserve the reliability and correctness of the data.

## Alert Generation :

Based on preset parameters, the platform will be configured to generate flood warnings. Alerts will be sent out as soon as necessary.

# 5. Testing and Validation:

## Unit Testing:

The early warning platform's components will each be evaluated separately to make sure they work as planned.

## Integration Testing:

To guarantee the platform's seamless integration with IoT sensors, extensive testing will be performed.

## Testing of user acceptance:

Users will test the platform to make sure it satisfies their needs and expectations, including the general public and emergency response teams.

# 6. Challenges and Solutions:

## Data reliability and accuracy:

Data validation and quality checks will be used to maintain data accuracy.

Scalability and Performance:

As the project grows, steps will be made to make sure the platform can accommodate more users and data.

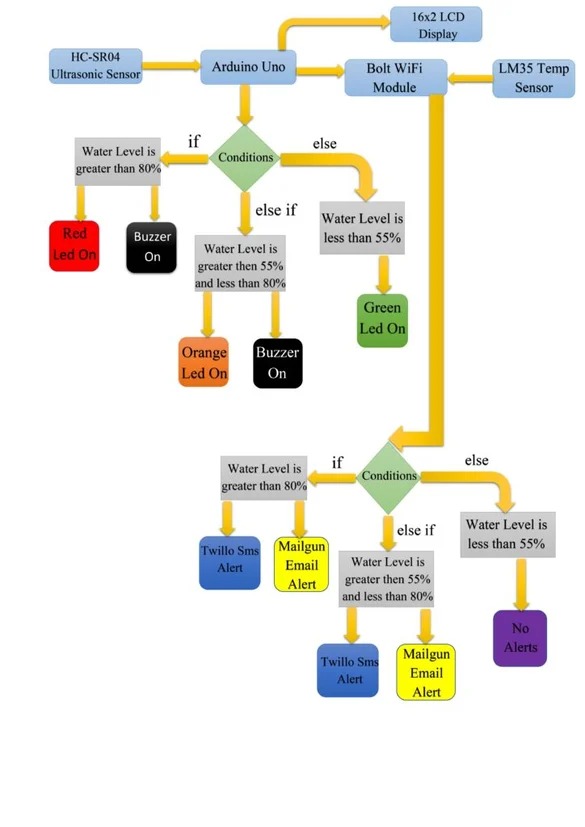
## Friendly User Interface:

The user interface will be enhanced for improved usability using user feedback from testing.

## Privacy and Security:

Sensitive data and user information will be protected with security measures.

FLOWCHART:



# 7. Expected Outcomes:

## An effective early warning system:

The platform will be fully functional and ready to show data in real-time and issue alerts as necessary.

## IoT Sensors and Seamless Data Integration:

IoT sensor integration will be effective, ensuring accurate and continuous data flow.

## Enhancing Public Safety:

Through the timely distribution of flood alerts, the platform will contribute to increased public safety.

# 8. Conclusion:

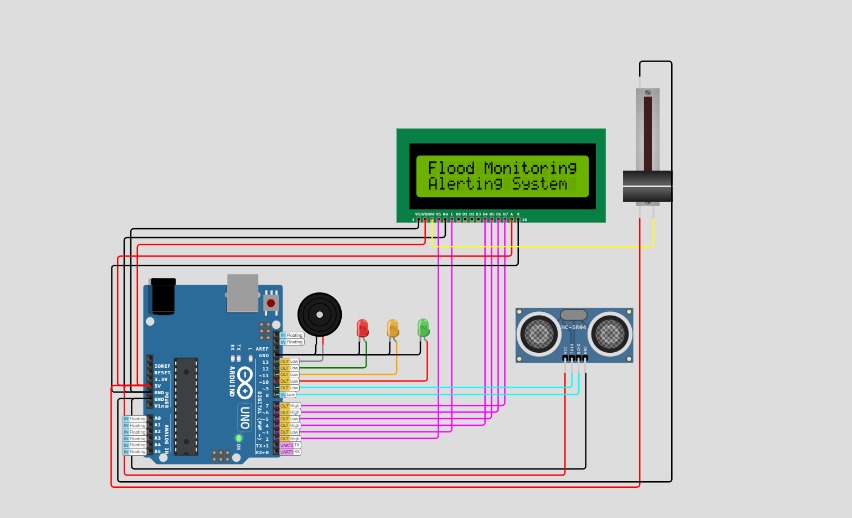
## Phase 4 Summary:

The fourth and last phase of our project is where we continue to construct the early warning platform. The platform serves as the user interface for real-time data display and flood warnings, making it a crucial part of our flood monitoring system.

## Moving onto Phase 5:

Phase 5 will see the project advance by testing the fully integrated system and getting ready for actual deployment. It is an important step in attaining the goals of our project.

**FLOOD MONITORING AND EARLY WARNING SYSTEM:**



**ALERTING:**

