PHASE-5

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

Project Documentation & Submission

INTRODUCTION:

We will thoroughly record the Flood Monitoring and Early Warning project throughout this phase and get it ready for submission. The project goals, platform development, IoT sensor deployment, and code implementation will all be covered in this guide. It will also include schematics, pictures, and diagrams of the early warning platform and IoT sensors. Finally, we will discuss how public safety and emergency response coordination are improved by the real-time flood monitoring and early warning system.

# DOCUMENTATION

## PROJECT OBJECTIVES:

The Flood Monitoring and Early Warning project's main goals are to:

1,Give locations close to water bodies and flood-prone zones real-time flood monitoring.

2,Early flood alerts should be distributed via a public forum.

3,Increase public safety by issuing prompt warnings.

4,Boost emergency response teams' cooperation.  
  
IoT Sensor Deployment:

Water level sensors for the project's IoT sensor rollout have been thoughtfully positioned in flood-prone zones. Periodically, these sensors take measurements of the water level and transmit the information to the early warning platform for evaluation.

Platform Development:

JavaScript, HTML, and CSS were used in the development of the web-based early warning platform. It shows data on water levels in real time and, if needed, sends out flood alerts. IoT sensors provide the platform with the data on water level, which is then shown on an intuitive and educational interface.

## 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.

**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.

# 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.

# 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.

# 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.

# . 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

Code Implementation:

The Python script created for the Internet of Things sensors makes it easier to gather and send water level data to the early warning system. Accurate reporting and smooth data transfer are guaranteed by the script.

# 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>

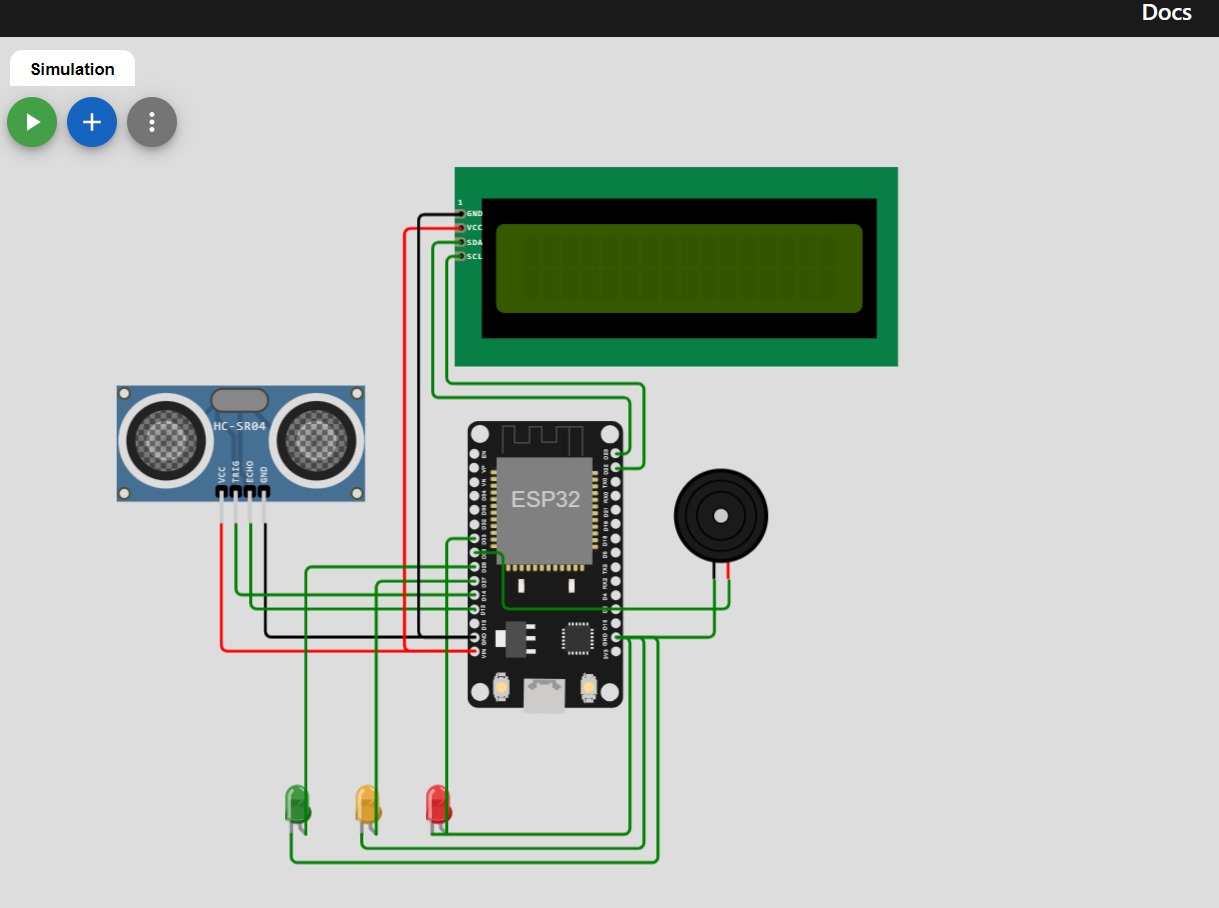
</html>

Diagrams, Schematics, and Screenshots:

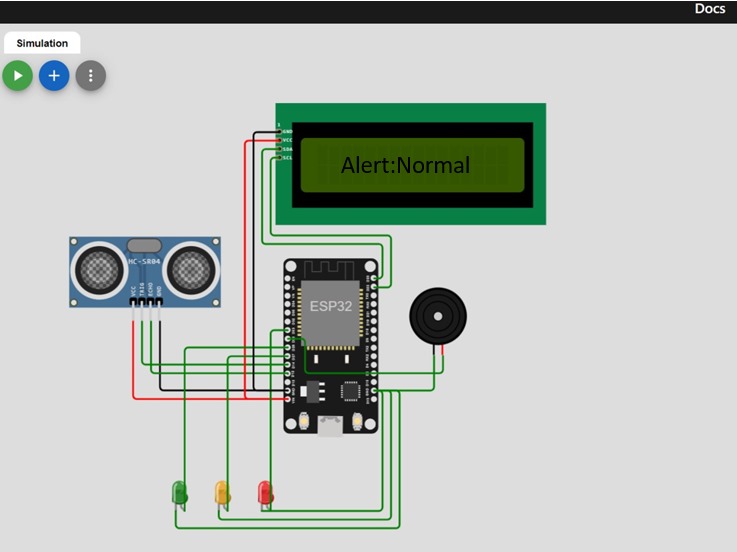
To provide a visual representation of the project, the following diagrams, schematics, and screenshots are included in the documentation:

1. Schematic layout of IoT sensors' deployment in flood-prone areas.
2. Diagram illustrating the data flow from IoT sensors to the early warning platform.
3. Screenshots of the early warning platform's user interface, displaying real-time water level data and issuing warnings.

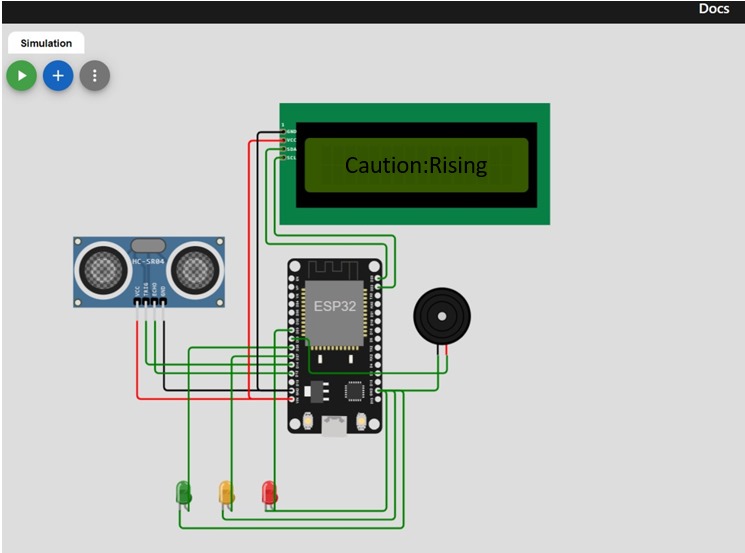
SIMULATION DIAGRAM:



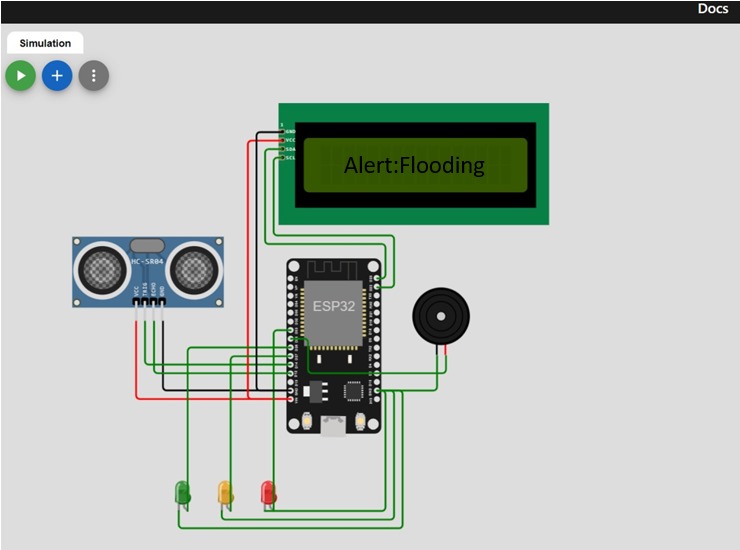
OUTPUT:1,NORMAL WATER LEVEL(GREEN ALERT)



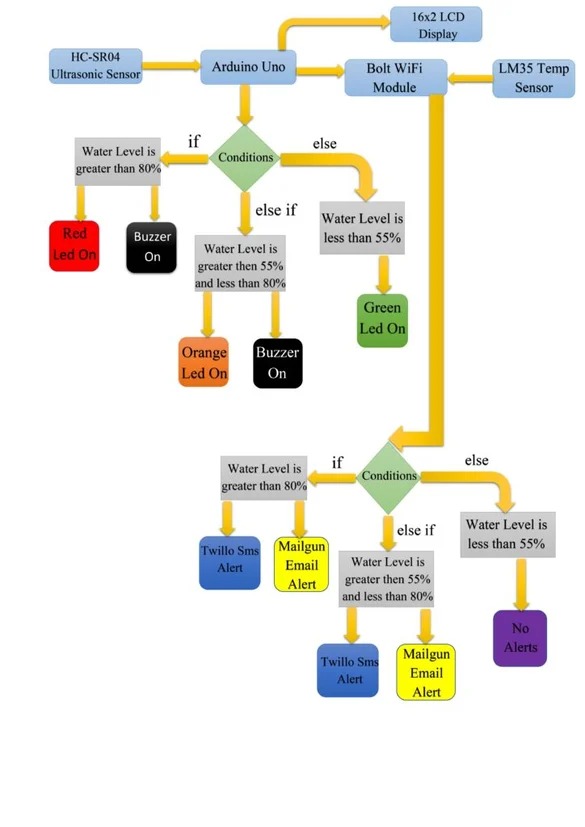
2,RISING WATER LEVEL(ORANGE ALERT)



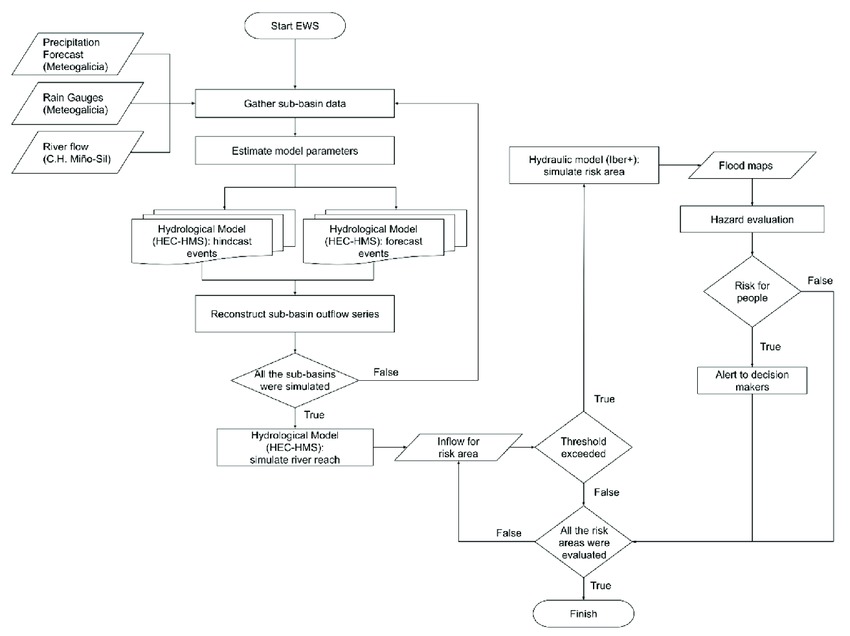
3,FLOODING DETECTED (REDALERT)



CODE FLOWCHART:



OVERALL PROJECT FLOWCHART:



Enhancing Public Safety and Emergency Response Coordination:

The early warning system and real-time flood monitoring greatly improve public safety and emergency response coordination by:

Timely Warnings: Residents are able to take the appropriate safety precautions and, if required, evacuate when the system promptly broadcasts flood warnings.

Data-Driven Decision-Making: By using up-to-date water level information, emergency response teams may make well-informed judgments that lead to more efficient responses.

Effective Resource Allocation: By providing early warning of approaching floods, resources, including manpower and equipment, may be distributed more effectively and response times can be shortened.

SUBMISSION:

To access the full project details, code, diagrams, schematics, and screenshots, please visit the GitHub repository: https://github.com/ARUNTHATHI940/IOT\_PHASE-1.git

Replication and Deployment Instructions:

The steps below can be used to reproduce this project, install IoT sensors, create the early warning system, and combine them using Python:

Get the required gear, such as an ESP32, an internet-connected PC, and IoT sensors (such as water level sensors).

Configure your web development tools (HTML, CSS, and JavaScript) and ESP32 development environment.

Install IoT sensors next to water bodies in flood-prone areas, making sure the power supply and connection are correct.

Write a Python script to collect water level data from IoT sensors and send it to the early warning platform.

Utilizing web development tools, create the early warning platform. Create it such that it can take in water level data from Internet of Things sensors, display it, and, if needed, send out flood alerts.

Check that the system is transmitting data and issuing warnings.

Make sure the public may access the platform for in-the-moment observation.

Example Outputs:

Screenshots and examples of the following will be available in this document itself and also this document uploaded in the repository:

An image of the early warning platform showing the current water level data in real time.

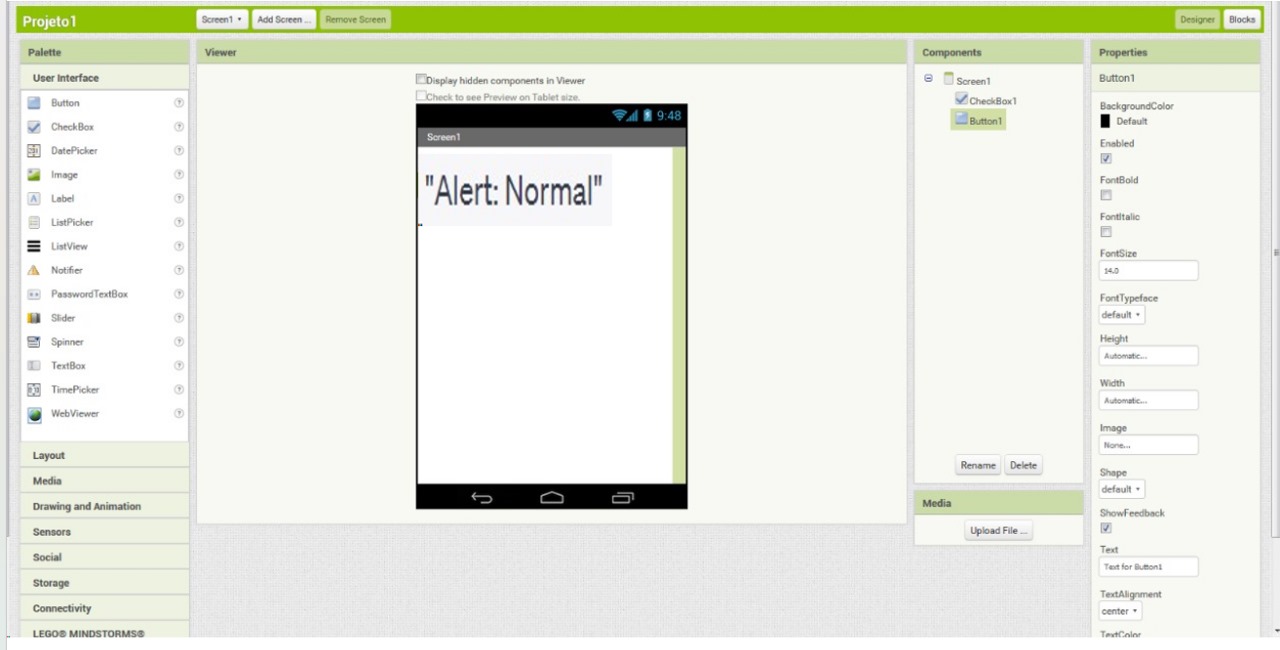
An image of the flood warning screen from the early warning platform.

Sample results of data transmitted to the platform from IoT sensors.

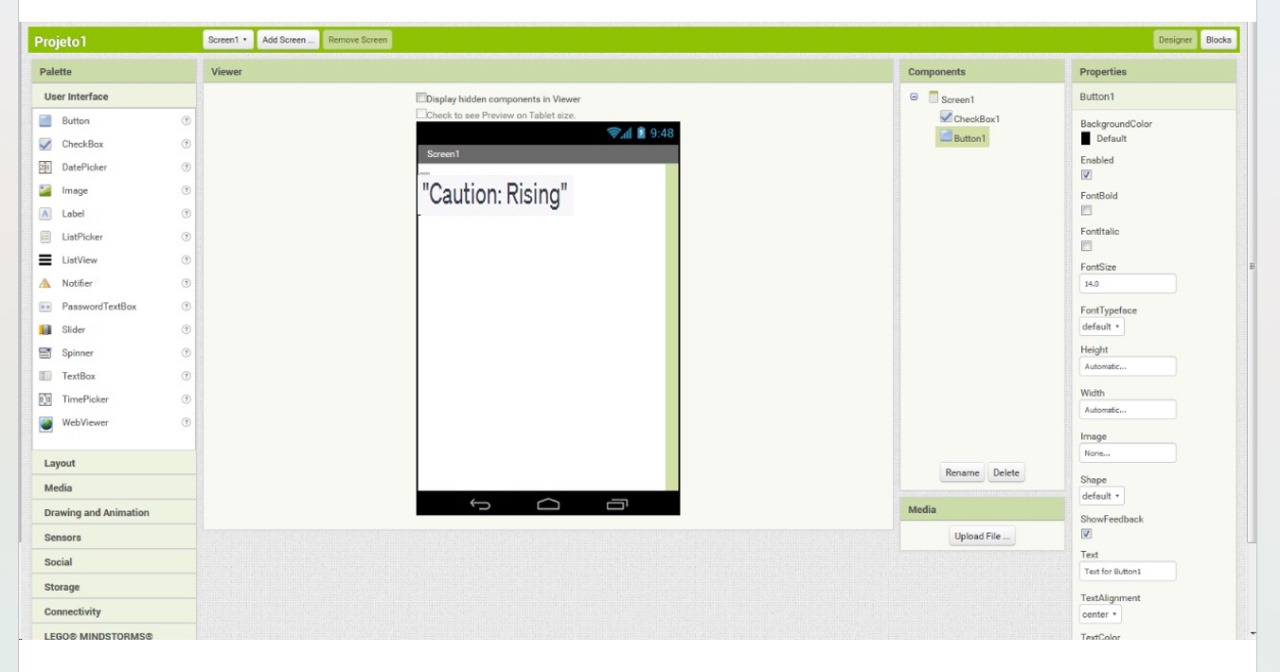
This literature offers a thorough description of the Flood Monitoring and Early Warning project and shows how it may improve emergency response coordination and public safety.

ALERTS SHOWN IN WEB APPLICATION:

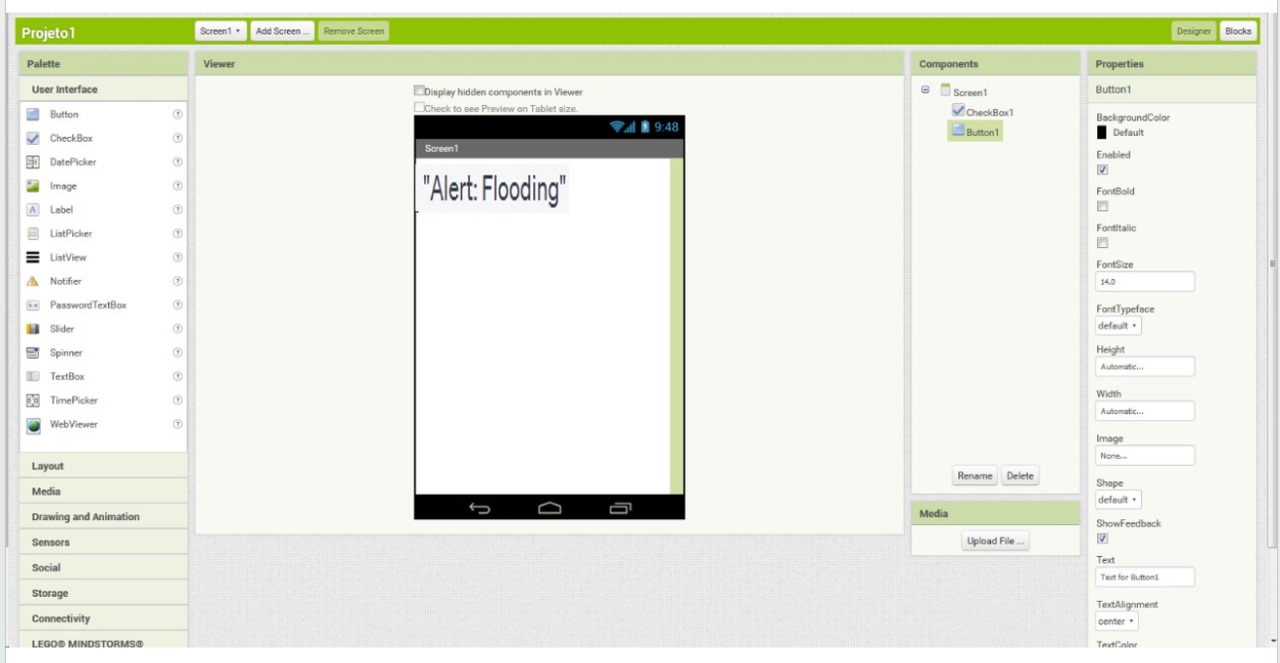
1, NORMAL WATER LEVEL(GREEN ALERT)



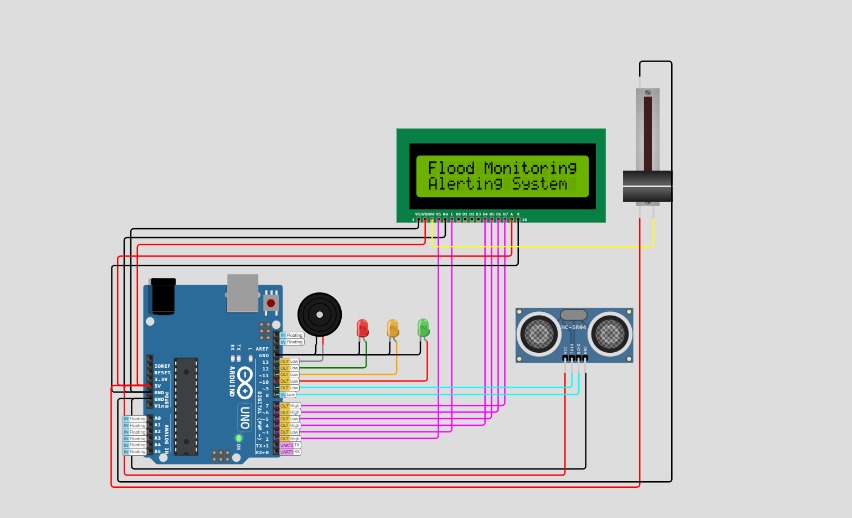
2,RISING WATER LEVEL(ORANGE ALERT)



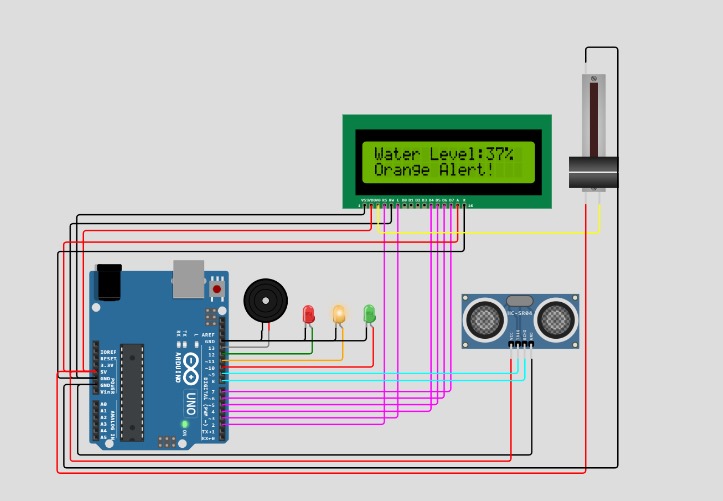
3, FLOODING DETECTED(RED ALERT)



**FLOOD MONITORING AND EARLY WARNING SYSTEM:**



**ALERTING:**



# CONCLUSION:

To sum up, the Flood Monitoring and Early Warning initiative has made significant progress in enhancing emergency response and public safety during flood disasters. Through the integration of Internet of Things sensors, predictive modeling, and historical data, we have developed a system that can provide prompt and precise flood alerts. The project is available for wider use and modification due to its open-source nature, which is outlined on GitHub. This research is an important step toward a safer, more resilient future and demonstrates the potential of technology in protecting communities from natural disasters.