**PHASE 3**

**PROJECT TITLE**

**FLOOD MONITORING AND EARLY WARNING SYSTEM**

**Development Part 1**

# Summary of Phase 1 and 2:

In order to improve our Flood Monitoring and Early Warning System, we examined novel approaches like predictive modeling and historical data integration in Phases 1 and 2, as well as established the project objectives and constructed the IoT sensor network. We begin constructing the system in Phase.

# Purpose of Phase 3:

Phase 3's main goal is to start our Flood Monitoring and Early Warning System's physical implementation. We will **install IoT sensors in flood-prone locations** during this phase, and we will create the Python script needed to gather and send data on water levels to the early warning system.

Deploying IoT Sensors to Use:

# Hardware Conditions

The following gear is required in order to deploy IoT sensors:

* ESP32
* HC-SR04 Ultrasonic Sensor
* 16x2 Character LCD
* LEDs (for indicators that are green, orange, and red)
* The alert's buzzer

# Location of Sensors

* When placing sensors, pick high-risk sites away from water bodies and areas that are prone to flooding. Make sure the sensors are adequately shielded from the elements and submerged in water as required.

# Setting up

* Set up the Internet of Things sensors so that they reliably and precisely detect the water level. In order to transmit data, power sources must be installed and sensors must be linked to the network.

Working on the Python Script:

# Data Gathering

* To gather data on water level using IoT sensors, write a Python script. Make that the script is capable of accurately interpreting the data from the sensors.

# Transmission of Data

* Provide the Python script with data communication methods so that the early warning platform receives the gathered water level data. Select a dependable communication protocol (such as HTTP or MQTT) to transfer data.

# Error Reduction

* Include error handling in the script to handle problems with connectivity, malfunctioning sensors, or failed data transmissions. Make sure the script has a smooth way of recovering from these problems.

# Combining Early Warning Platform Integration

* Make arrangements for the Python script to communicate with the early warning system. Make that the sensors' data is properly received and processed on the platform side.

Wokwi Software Coding:

# Configuring the Scene

* To represent the ESP32 and IoT sensors, build a new simulation environment in the Wokwi software and choose the required virtual components.

# Creating the Storyline

* Create the Python script and upload it to the ESP32 virtual machine. Make sure the script reflects the features needed for IoT sensors in the real world.

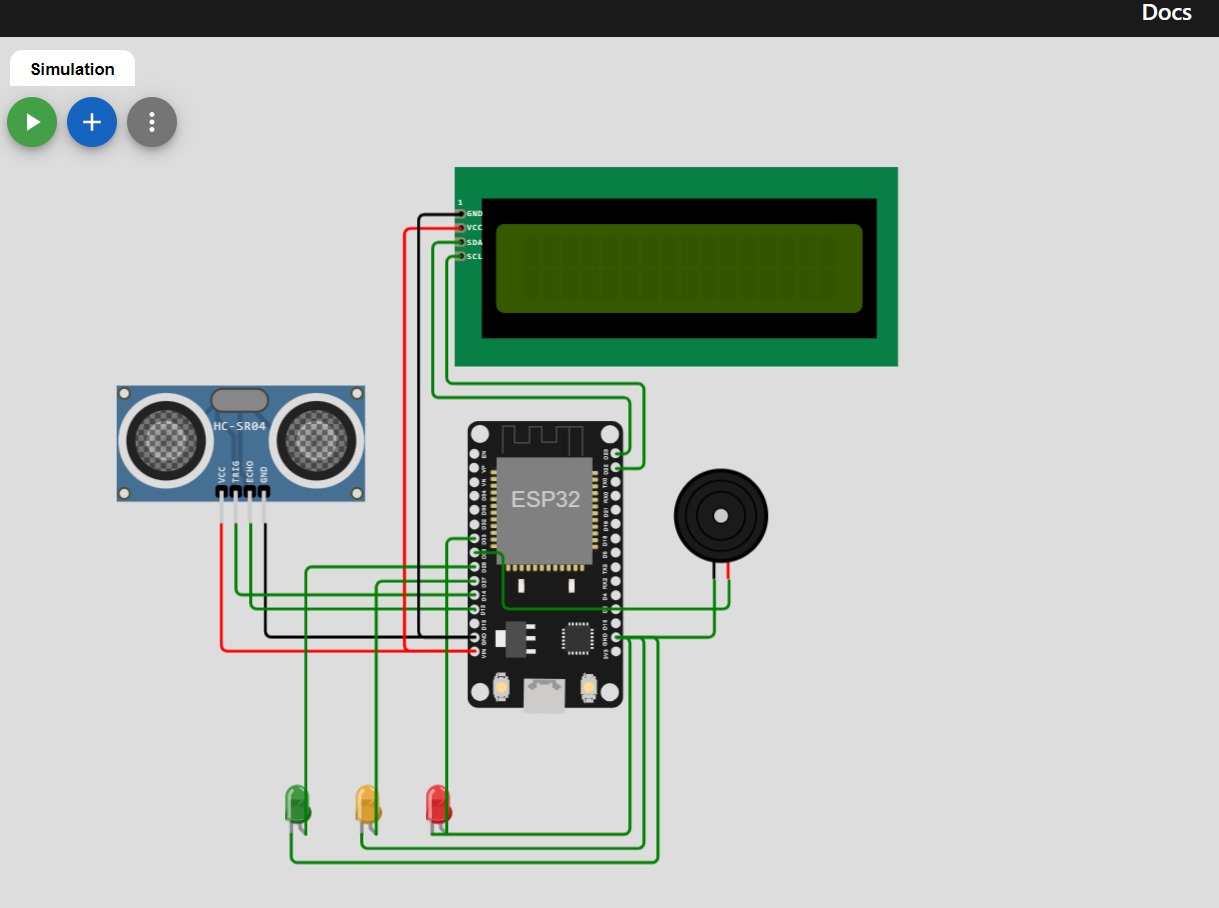
# Evaluation in Wokwi

* Start the Wokwi simulation to evaluate the IoT sensors and simulated ESP32's capabilities. To verify the procedures for data collection and transmission, simulate various water level scenarios.

# Problem-solving

* Determine and fix any problems that might come up while testing. To make sure everything runs smoothly, debug the simulation and script.

**SIMULATION DIAGRAM:**



Conclusion:

# An overview of Phase 3

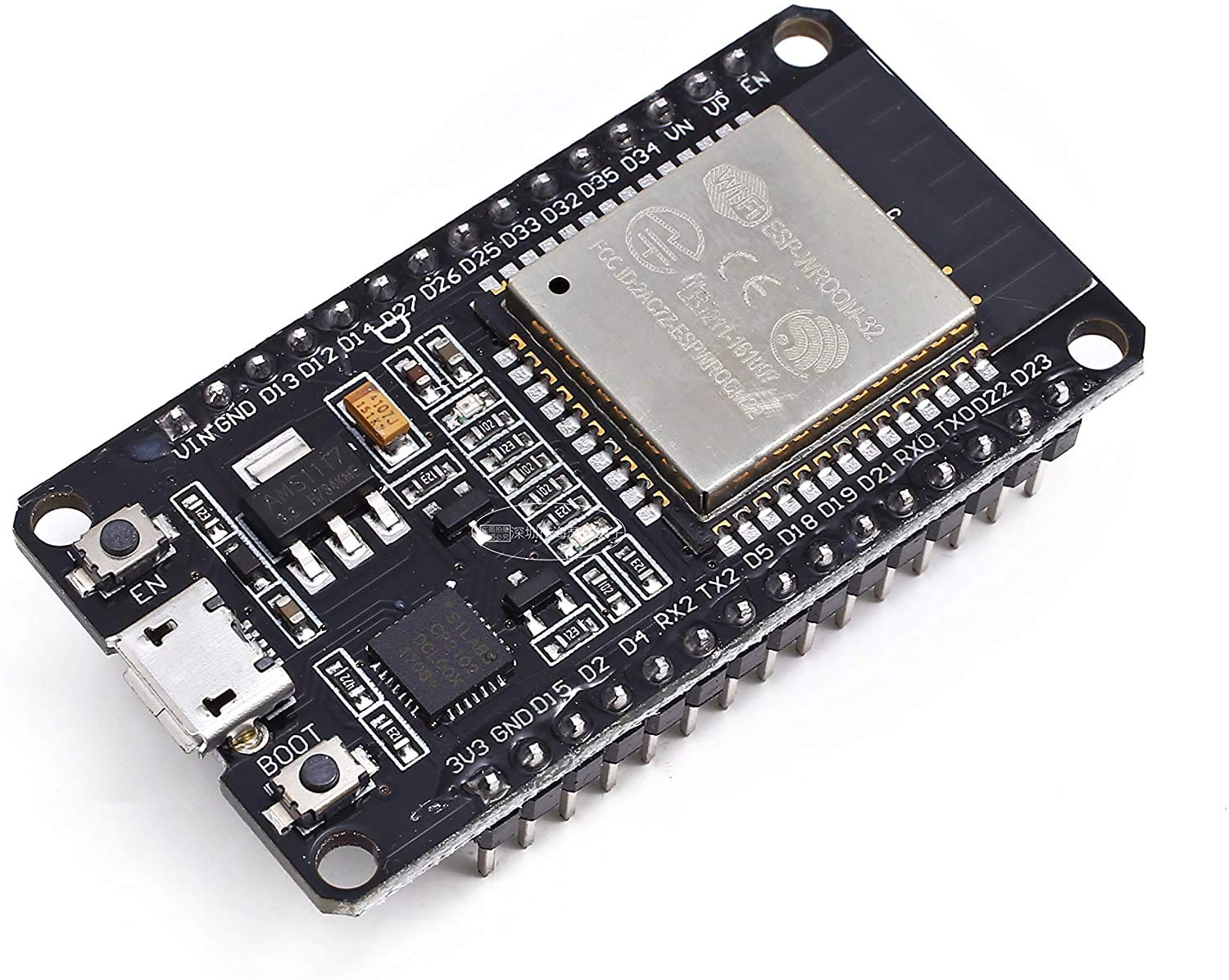
* The actual implementation of our Flood Monitoring and Early Warning System has started with Phase 3. We set up Internet of Things sensors and created a Python script to gather and send data. This stage creates the framework for implementing the system in the actual world.

# Change to Phase 4

* Phase 4 will see the system's continued development with an emphasis on enhancing the IoT sensors' capabilities and integrating data into the early warning platform.

COMPONENTS:

**ESP32:**



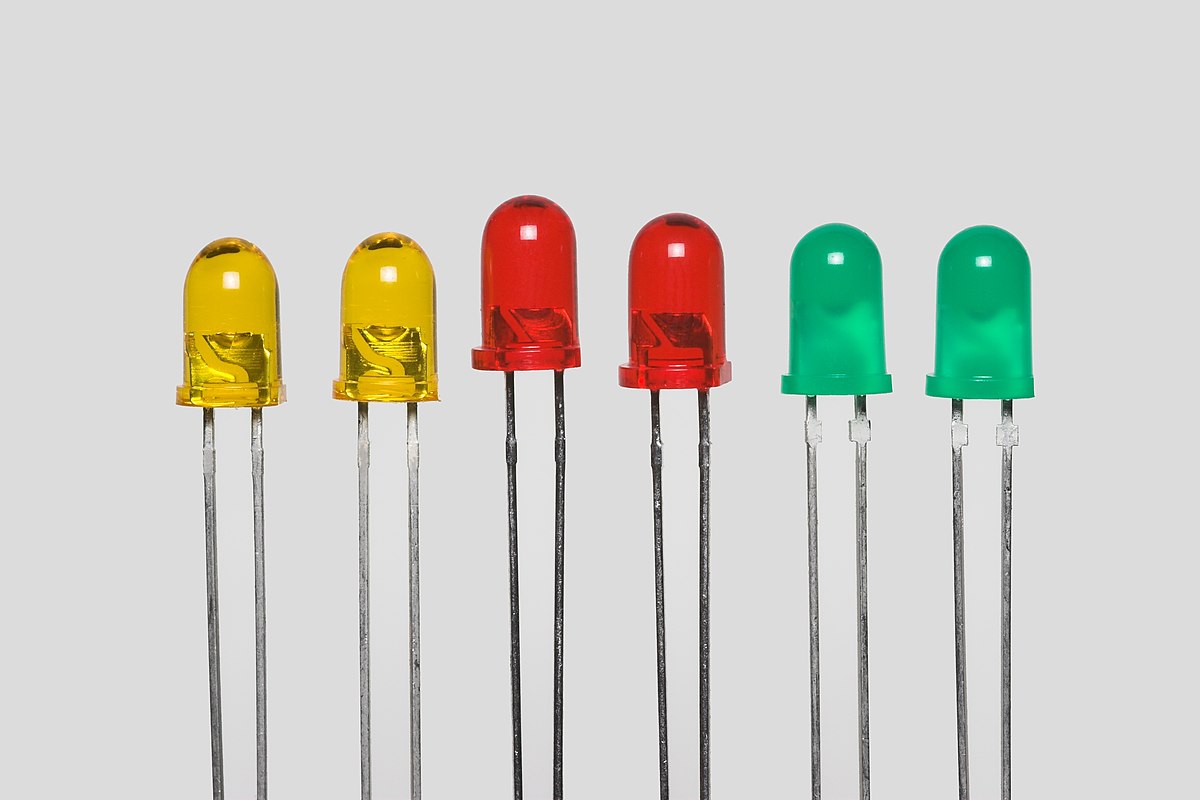
**HC-SR04 Ultrasonic Sensor:**



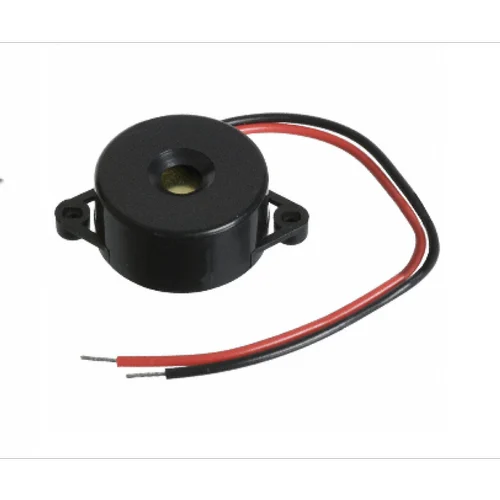
**16x2 Character LCD:**



**LEDs (for indicators that are green, orange, and red):**



**The alert's buzzer:**



# PYTHON SCRIPT:

import machine

import time

import urequests

# Define GPIO 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

# Initialize 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

# 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()

# Display distance on the LCD

lcd.writeto(0x27, "Water Level: {:.2f} cm".format(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()

# Display the appropriate alert message on the LCD

if alert\_level == "green":

lcd.writeto(0x27, "Alert: Normal")

elif alert\_level == "orange":

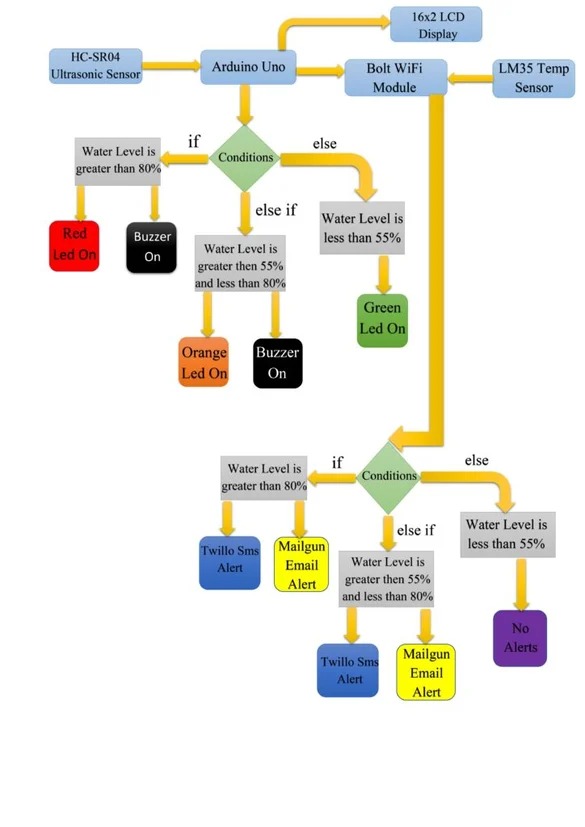
lcd.writeto(0x27, "Caution: Rising")

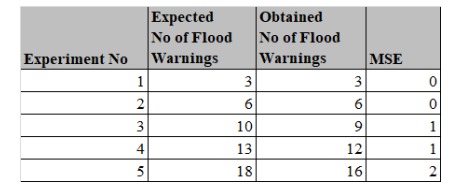
else:

lcd.writeto(0x27, "Alert: Flooding")

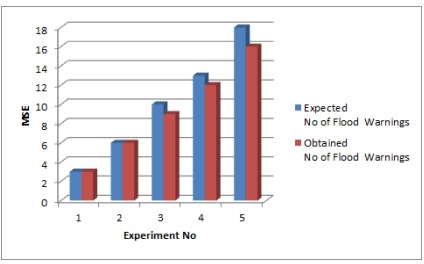
time.sleep(2)

**FLOWCHART:**





**DATASETS**



**MSE COMPARISONS**