PHASE : 3 DEVELOPMENT PART -1

PROJECT TITLE	9238 - FLOOD MONITORING AND EARLY WARNING
NAME	R.PRIYADHARSNI
TEAM ID	5246
TEAM NAME	PROJ_204183_TEAM_1
COLLEGE CODE - NAME	9238 - MANGAYARKARASI COLLEGE OF ENGINEERING PARAVAI MADURAI.
GROUP	5
GITHUB REPOSITORY LINK	https://github.com/priyadharsni/IBM-Naanmudhalvan-iot.git

IOT-FLOOD MONITORING AND EARLY WARNING

1.ABSTRACT:

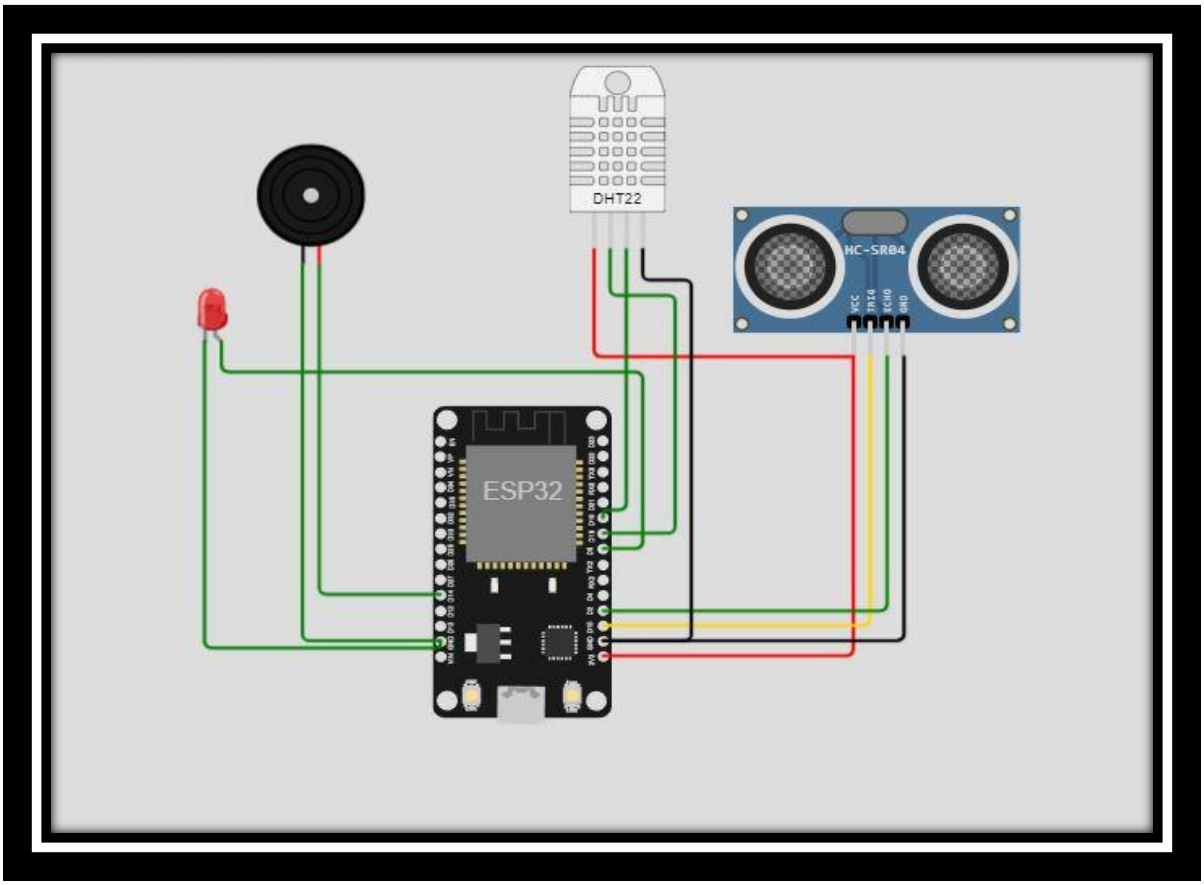
The Flood Monitoring and Early Warning Project is a comprehensive initiative designed to mitigate the devastating impacts of flooding events in vulnerable regions. Flooding is a recurrent natural disaster that leads to loss of lives, destruction of property, and disruption of livelihoods. This project aims to leverage advanced technology, data analytics, and community engagement to enhance flood monitoring, prediction, and timely dissemination of early warnings.

2. PROPOSAL OF THE PROJECT:

The development of a flood monitoring system that can accurately predict and detect flood events, including ultrasonic sensor, float sensor, water temperature sensor, rain gauge, weather

station, flow meter, camera and image sensor, LORA sensor. The objective of this proposal is to create an alert message to all the mobile phones .so our main aim is to alert the people before the flood.

3.HARDWARE SETUP:



HARDWARE COMPONENTS:-

1.ESP32:

The ESP32 serves as the main controller and runs the python program.it provides the necessary GPIO pins for connecting sensors.

2.ULTRASONIC SENSOR:

Ultrasonic sensor are used to measure the distance of the water level.

3.DHT22:

The DHT22 is a popular sensor used for measuring temperature and humidity in various applications, including flood monitoring.

4.BUZZER:

A buzzer can be a valuable component in flood monitoring systems to provide audible alerts and warnings to people in the affected area.

5.LCD:

LEDs incorporated into public displays or signs can serve as a means of conveying flood alerts to the public. Flashing or colored LEDs can quickly attract attention and communicate urgency.

6.WIFI MODULE:

While not explicitly mentioned in the code, built-in Wi-Fi module is used to connect the ESP32 to the Wi-Fi network.

Hardware Connections:

Here's how the hardware components should be connected based on your code:

Ultrasonic Sensors (HC-SR04):

- **Each ultrasonic sensor (e.g., HC-SR04) requires four connections:**
- **VCC (Voltage): Connect to a 5V pin on the ESP32 for power**
- **GND (Ground): Connect to a ground (GND) pin on the ESP32.**
- **Trig (Trigger): Connect to the GPIO pins defined in trig_pins (pins 2, 18, and 22 in your code).**
- **Echo: Connect to the GPIO pins defined in echo_pins (pins 4, 19, and 23 in your code).**

Wi-Fi Module:

- **You should ensure that your ESP32 is connected to a Wi-Fi network, either using built-in Wi-Fi or an external Wi-Fi module. The program relies on this network connection to send data to Firebase.**

Power Supply:

- **The ESP32 and sensors should be powered appropriately. The ESP32 can be powered through a USB power supply, and sensors may need a separate 5V supply. Ensure that all components share a common ground.**

PROGRAM:-

MICRO PYTHON CODING

```
import machine
import time
import urequests
import ujson
import network
import dht

# Define your Wi-Fi credentials
wifi_ssid = 'Wokwi-GUEST'
wifi_password = ''  # Replace with the actual Wi-Fi password

# Connect to Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

# Wait for Wi-Fi connection
while not wifi.isconnected():
    pass

# Define GPIO pins
TRIG_PIN = machine.Pin(15)
ECHO_PIN = machine.Pin(2)
alarm_pin = machine.Pin(5, machine.Pin.OUT)
buzzpin = machine.Pin(14, machine.Pin.OUT)
DHT_PIN = machine.Pin(18)

# Set trig pin as an output
TRIG_PIN.init(machine.Pin.OUT)
```

```
# Set echo pin as an input
ECHO_PIN.init(machine.Pin.IN)

# Set the alarm pin initially to OFF
alarm_pin.off()

# Maximum water level for alarm (in cm)
alarm_threshold = 100  # Adjust this value as needed

# Maximum parking distance
max_distance = 50

def read_dht_sensor():
    d = dht.DHT22(DHT_PIN)
    d.measure()
    return d.temperature(), d.humidity()

def measure_distance():
    # Trigger ultrasonic sensor
    TRIG_PIN.on()
    time.sleep_us(10)
    TRIG_PIN.off()

    # Wait for echo to be HIGH (start time)
    while not ECHO_PIN.value():
        pass
    pulse_start = time.ticks_us()
```

```
    # Wait for echo to be LOW (end time)
    while ECHO_PIN.value():
        pass
    pulse_end = time.ticks_us()

    # Calculate distance
    pulse_duration = time.ticks_diff(pulse_end, pulse_start)
    distance = pulse_duration / 58  # Speed of sound (343 m/s) divided by 2

    return distance

buzz_start_time = None

# Firebase Realtime Database URL and secret
firebase_url = 'https://flood-monitoring-511bd-default-rtdb.asia-southeast1.firebasedatabase.app/'  # Replace with your Firebase URL
firebase_secret = 'jnMJGSFxTcAMawQ6b0yUZp4SitouPTTy5p0Ql4aG'  # Replace with your Firebase secret

# Function to send data to Firebase
def send_data_to_firebase(distance, temperature, humidity, status):
    data = {
        "Distance": distance,
        "Temperature": temperature,
        "Humidity": humidity,
        "Status": status
    }
    url = f'{firebase_url}/sensor_data.json?auth={firebase_secret}'
```

```
    try:
        response = urequests.patch(url, json=data)  # Use 'patch' instead of 'put'
        if response.status_code == 200:
            print("Data sent to Firebase")
        else:
            print(f"Failed to send data to Firebase. Status code: {response.status_code}")
    except Exception as e:
        print(f"Error sending data to Firebase: {str(e)}")


while True:
    dist = measure_distance()
    temp, humidity = read_dht_sensor()


    # Check if the distance is less than a threshold (e.g., 50 cm)
    if dist < max_distance:
        # Turn on the buzzer and alarm
        buzzpin.on()
        alarm_pin.on()
        status = "Flood Alert"
        buzz_start_time = time.ticks_ms()
    elif buzz_start_time is not None and time.ticks_diff(time.ticks_ms(), buzz_start_time) >= 60000:  # 1
minute
        # Turn off the buzzer and alarm after 1 minute
        buzzpin.off()
        alarm_pin.off()
        status = "No Flood Alert"
    else:
        status = "No Flood Alert"
```

```
    print(f"Distance: {dist:.2f} cm")
    print(f"Temperature: {temp:.2f}°C, Humidity: {humidity:.2f}%")
    print("Status:", status)

    # Send data to Firebase
    send_data_to_firebase(dist, temp, humidity, status)

    time.sleep(2)  # Adjust the sleep duration as needed
```

This is our Python program which is used to design for a flood monitoring and early warning using ultrasonic sensor to monitor the water level and send this information to a Firebase Real-time Database

1. Importing Libraries:

We start the program that begins by importing necessary libraries, including machine, time, urequests, and network.

2. Wi-Fi Setup:

- **It defines Wi-Fi credentials (wifi_ssid and wifi_password) for connecting to a Wi-Fi network.**
- **It creates a Wi-Fi interface and connects to the specified Wi-Fi network.**
- **The program waits for the Wi-Fi connection to establish before proceeding.**

3. Sensor Setup:

The program sets up the GPIO pins for ultrasonic sensors (trig_pins for trigger pins and echo_pins for echo pins)

4. Distance Measurement:

- **There's a function measure_distance(trig_pin, echo_pin) for measuring distances using ultrasonic sensors. It calculates distances based on the time it takes for an ultrasonic pulse to return.**
- **The speed of sound is used to calculate the distance, and the results are in centimeters.**

5. Firebase Integration:

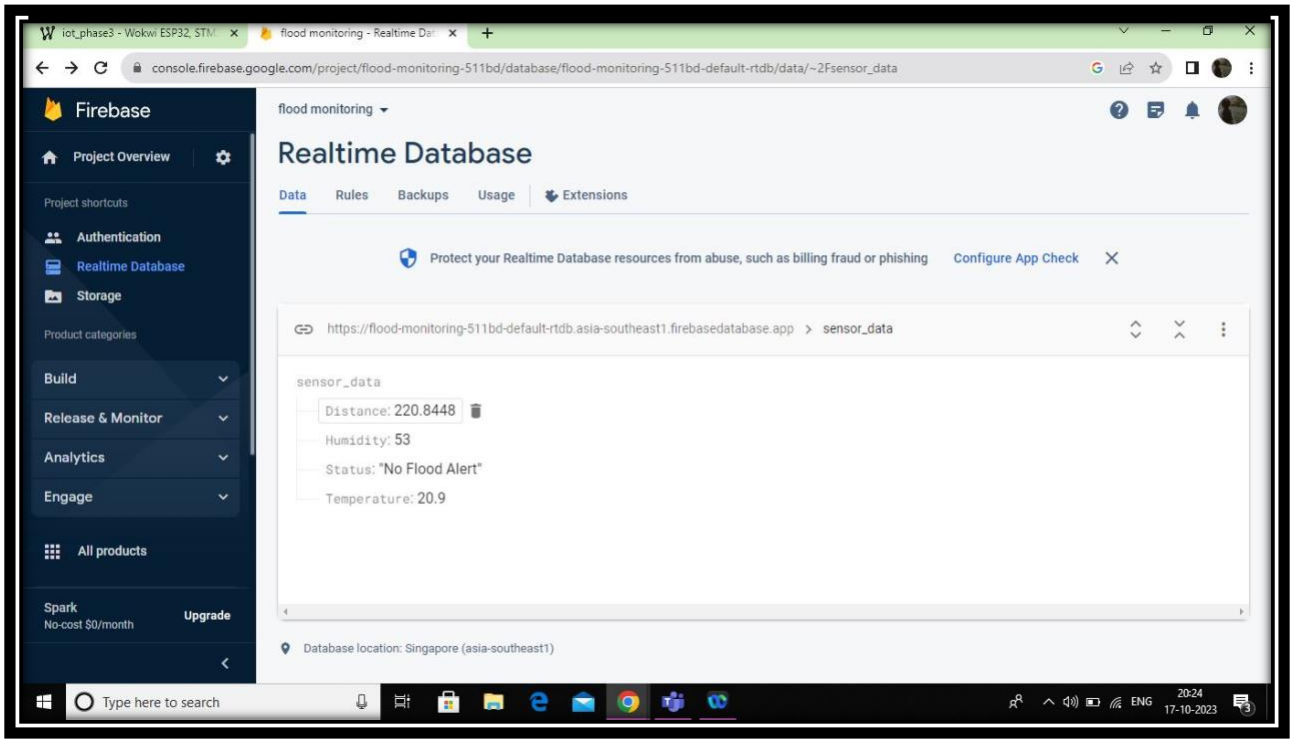
- **The program sends the parking availability data to a Firebase Realtime Database. It uses a Firebase URL (firebase_url) and a secret key (firebase_secret) for authentication.**

Overall, this program continuously monitoring the flood level in dam using ultrasonic updates the status of each water level and sends this information to a Firebase Realtime Database, making it suitable for flood monitoring and early warning with remote monitoring capabilities. Note that some parts of the program, such as the Wi-Fi credentials and specific sensor pin configurations, need to be customized to match your hardware setup.

Firebase Database:

Real time Database:

Firebase Realtime Database is a cloud-hosted NoSQL database provided by Firebase, a mobile and web application development platform that is now part of Google's cloud offerings. Firebase Realtime Database is designed for real-time data synchronization and is commonly used in applications where you need to store, retrieve, and synchronize data across various clients and platforms.



CONCLUSION:-

In conclusion flood monitoring and early warning systems are critical components of disaster risk reduction and management. They play a pivotal role in safeguarding lives, property and the environment in flood prone regions. Early warning systems can significantly reduce the loss of life during flood events.