FLOOD WATER MANAGEMENT AND EARLY WARNING SYSTEM

internet of things - group1 - phase-3

college name : ***Madha Institute of Engineering and Technology***

college code : ***2112***

student name : ***Elden Nicholas .N***

N.M id : ***au211221205004***

1. Hardware Components:

- **Water Level Sensors:** Deploy various water level sensors at strategic locations, such as rivers, dams, and flood-prone areas.
- **Microcontrollers:** Use microcontrollers like Arduino or Raspberry Pi to collect and process data from sensors.
- **Communication Modules:** Incorporate wireless communication modules (e.g., Wi-Fi, LoRa, or cellular) to transmit data to a central server.
- **Power Supply:** Design efficient power sources, such as solar panels and rechargeable batteries, for continuous operation.

2. Data Collection and Sensors:

- Implement water level sensors to measure river or stream levels.
- Use additional sensors for rainfall, temperature, and humidity to enhance data accuracy.
- Collect data at regular intervals and ensure real-time transmission to the central system.

3. Data Processing:

- Process sensor data to calculate water level variations, rainfall intensity, and other relevant parameters.
- Implement data fusion algorithms to provide more accurate early warnings.

4. Centralized Server:

- Set up a central server to receive, store, and process data from multiple IoT devices.
- Implement a database for historical data storage and analysis

5. Early Warning System:

- Develop algorithms that trigger early warnings based on predefined thresholds (e.g., rising water levels or heavy rainfall).
- Utilize machine learning and predictive modeling to forecast flood events.

6. User Interface:

- Create a user-friendly web-based or mobile app interface for end-users to access real-time data and receive alerts.
- Include visual representations like maps, graphs, and alerts.

7. Alerts and Notifications:

- Implement an alerting system to notify local authorities, emergency services, and the public in real-time.
- Notifications can be sent via SMS, email, or push notifications.

8. Integration with Local Authorities:

- Establish protocols for sharing data and alerts with local government agencies and disaster management teams.

9. Disaster Response Coordination:

- Develop features to facilitate coordination and response efforts, such as mapping the location of shelters and evacuation routes.

10. Scalability and Redundancy:

- Design the system to be scalable, allowing the addition of more IoT devices as needed.
- Implement redundancy to ensure system reliability during power outages or device failures.

11. Testing and Validation:

- Conduct thorough testing and validation of the system, including field testing under various weather conditions.

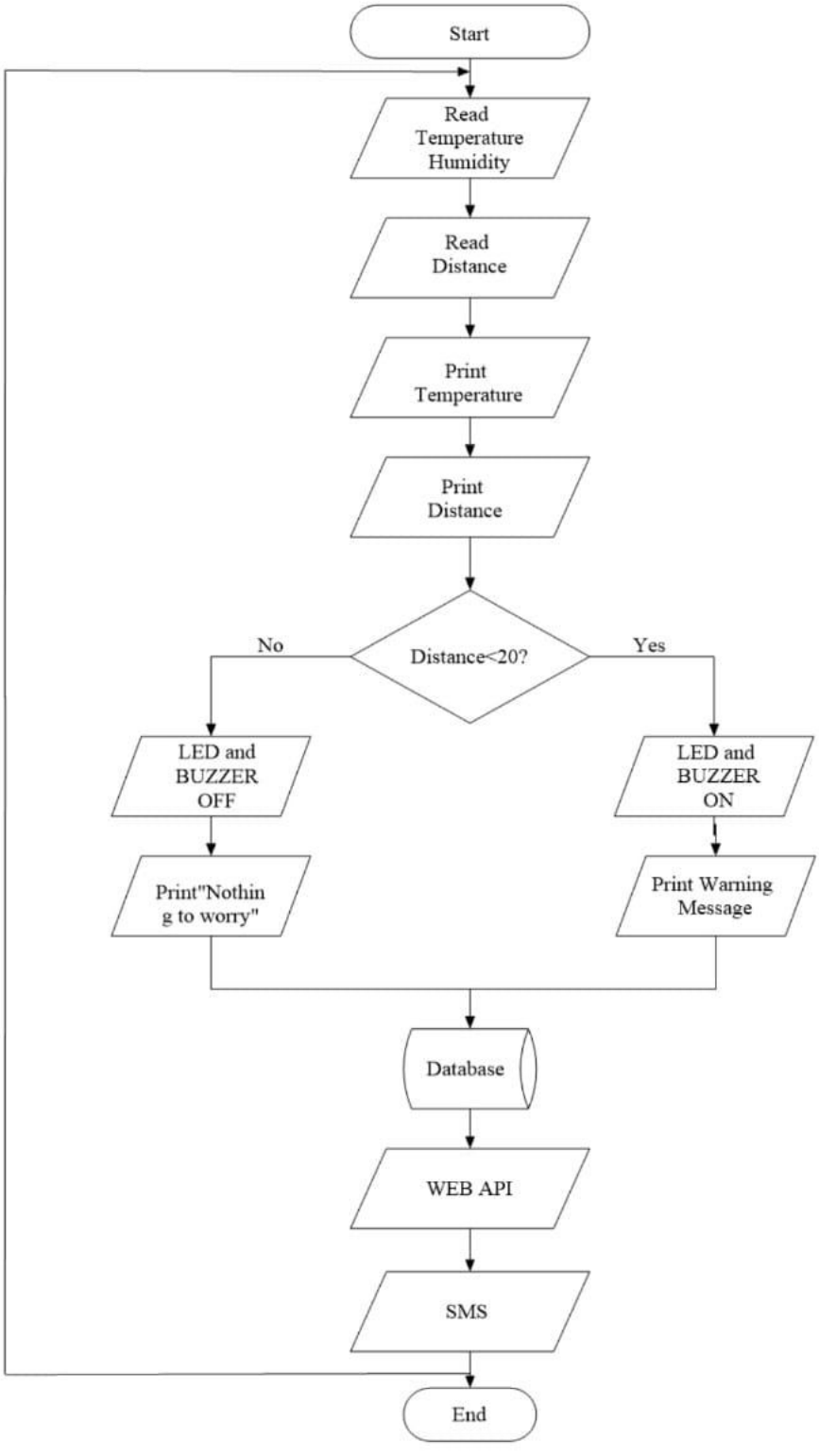
12. Maintenance and Updates:

- Plan for regular maintenance of IoT devices and software updates to ensure continued reliability.

13. Education and Public Awareness:

- Include features to educate the public about flood risks and safety measures through the user interface

Flowchart


Start
Read Temperature Humidity
Read Distance
Print Temperature
Print Distance
Distance<20?
No
Yes
LED and BUZZER OFF
LED and BUZZER ON
Print"Nothin g to worry"
Print Warning Message
Database
WEB API
SMS
End

a python source code for the above technology :

```
import random
import time

# Simulate data collection (e.g., water level in a river)
def collect_data():
    return random.uniform(1.0, 10.0)  # Replace with real data source

# Flood prediction algorithm (simple threshold-based)
def predict_flood(data):
    if data > 7.0:
        return True
    return False

# Alerting function
def send_alert(message):
    # Replace this with your alerting mechanism (email, SMS, etc.)
    print(f"ALERT: {message}")

# Main loop
while True:
    current_data = collect_data()
    is_flood_predicted = predict_flood(current_data)

    if is_flood_predicted:
        send_alert("Flood warning! Take immediate action.")

    # Adjust the sleep time according to your data update frequency
    time.sleep(60)  # Sleep for 1 minute (simulating real-time data)
```

Group Members
1.211221205001
2.211221205002
3.211221205003
4.211221205004
5.211221205005
6.211221205006