**Phase 3 project**

**Project Title: FLOOD MONITORING AND EARLY WARNING**

**Project ID :** proj\_223735\_Team\_8

**College :** Gnanamani College of Technology

**College code :** 6208

**Branch :** B.Tech-Information Technology

**Year :** IIIrdyear

**Team Members:**

P.Sanjay (620821205304)

D.Pragadeeswaran(620821205302)

R.C.Diwyaprasad(620821205014)

To deploy IoT sensors in flood-prone areas and configure them to measure water levels, you will need hardware components such as water level sensors, microcontrollers (e.g., Raspberry Pi, Arduino), a network connection (Wi-Fi, cellular, or LoRa), and access to an early warning platform for data transmission. Here is a high-level guide on how to develop a Python script for the IoT sensors:

**1. \*\*Hardware Setup\*\*:**

- Choose suitable water level sensors, microcontrollers, and power supply methods based on your specific requirements and environmental conditions.

**2. \*\*Connect Water Level Sensors\*\*:**

- Connect the water level sensors to your microcontroller following the sensor's datasheet. Usually, water level sensors provide analog or digital output that you need to read using the microcontroller.

**3. \*\*Install Python\*\*:**

- Ensure that your microcontroller has Python installed. You might need to install Python and necessary libraries if they are not pre-installed

**4. \*\*Write Python Script\*\*:**

- Create a Python script to read data from the water level sensors and send it to the early warning platform. You can use libraries like `RPi.GPIO` (for Raspberry Pi) or `pySerial` for Arduino to read sensor data.

# Example script using Raspberry Pi GPIO

import RPi.GPIO as GPIO

import time

import requests

SENSOR\_PIN = 4 # GPIO pin where the sensor is connected

WARNING\_API\_URL = "https://earlywarningplatform.com/api/data"

GPIO.setmode(GPIO.BCM)

GPIO.setup(SENSOR\_PIN, GPIO.IN)

while True:

water\_level = GPIO.input(SENSOR\_PIN) # Read water level data

payload = {"water\_level": water\_level}

response = requests.post(WARNING\_API\_URL, json=payload)

if response.status\_code == 200:

print("Data sent successfully")

else:

print("Failed to send data")

time.sleep(60) # Read and send data every 60 seconds

In this example, we read the sensor data and send it as JSON to the early warning platform's API. You might need to adjust the script to match the data format and endpoint of your specific platform.

**5. \*\*Data Transmission\*\*:**

- Ensure that you have the necessary credentials and permissions to send data to the early warning platform. Use the platform's API or communication protocol to transmit the data securely.

**6. \*\*Power Management\*\*:**

- Consider how you will power the IoT sensors. This may involve using batteries, solar panels, or other power sources depending on the deployment location.

**7. \*\*Testing and Deployment\*\*:**

- Test the system in a controlled environment before deploying it in flood-prone areas. Ensure that the sensors are correctly measuring water levels and that data transmission is reliable.

**8. \*\*Monitoring and Maintenance\*\*:**

- Set up monitoring and maintenance procedures to ensure that the IoT sensors continue to function properly in the field.

Remember that this is a simplified example, and real-world deployments can be much more complex, involving data validation, security, and redundancy measures to ensure the early warning system's reliability and accuracy.