Phase 3:Building the IoT Flood Monitoring and Early Warning System.

Project Description:

Floods are natural disasters that can cause significant damage to property and pose serious risks to lives. To mitigate these risks, we propose the development of an IoT Flood Monitoring and Early Warning System. This system will utilize sensors, data analysis, and real-time communication to provide early flood warnings to authorities and residents, allowing for proactive responses to mitigate flood-related damage.

Gathering Components:

- Ultrasonic sensor
- Buzzer
- dht22
- Raspberry pi pico
- LED light
- Register

Deploy IoT Sensors:

Sensor Selection: Choose suitable water level sensors that are appropriate forthe environment and expected water levels. Sensors can be ultrasonic, pressure- based, or float-type, depending on the specific requirements.

Sensor Installation: Securely install the sensors in the selected locations. The installation process may vary depending on the sensor type. Ensure they are well-protected against damage or vandalism.

Power Supply: Ensure a stable and reliable power supply for the sensors.Depending on the location, you might use solar panels, battery packs, or a combination of power sources to keep the sensors operational.

Connectivity: Set up the communication module on each sensor. This could beWi-Fi, LoRa, or cellular connectivity, depending on the available infrastructure inthe deployment area.

Data Transmission: Configure the sensors to transmit water level data to acentralized server or cloud platform. Implement data encryption and security measures to protect the data during transmission.

Real-time Monitoring: Implement real-time monitoring to confirm thatsensors are transmitting data correctly. You can use diagnostic tools and monitoring software to track the data.

Remote Access: Ensure that you have remote access to the sensors and that youcan reconfigure or troubleshoot them as needed.

Security Measures: Implement security measures to protect the sensors fromphysical and cyber threats. This may include enclosures, tamper detection, and authentication mechanisms.

Data Validation: Regularly validate the accuracy of the sensor data andrecalibrate the sensors if necessary.

Maintenance Plan: Develop a maintenance plan to address routine sensorchecks, cleanings, and repairs. This is crucial for long-term system reliability.

Python Script:

```
import time
import machine
import dht

# Define GPIO pins
TRIG_PIN = machine.Pin(2, machine.Pin.OUT)
ECHO_PIN = machine.Pin(3, machine.Pin.IN)
BUZZER_PIN = machine.Pin(4, machine.Pin.OUT)
DHT_PIN = machine.Pin(5)
LED_PIN = machine.Pin(6, machine.Pin.OUT)

def distance_measurement():
    # 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

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

buzz_start_time = None  # To track when the buzzer started

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

    # Check if the distance is less than a threshold (e.g., 50 cm)
    if dist < 50:
        # Turn on the buzzer and LED
        BUZZER_PIN.on()
        LED_PIN.on()
        status = "Flooding Detected"
        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 LED after 1 minute
        BUZZER_PIN.off()
        LED_PIN.off()
        status = "No Flooding Detected"
    else:
        status = "No Flooding Detected"

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

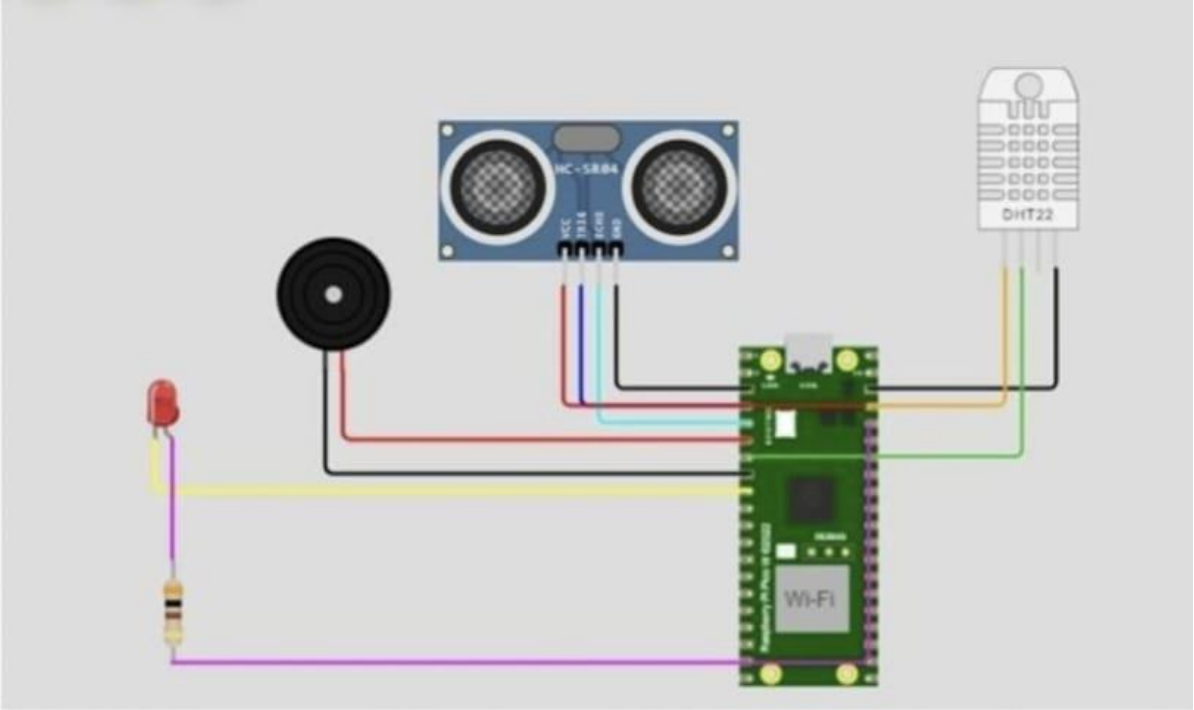
```
    time.sleep(2)
```

Diagram.json:

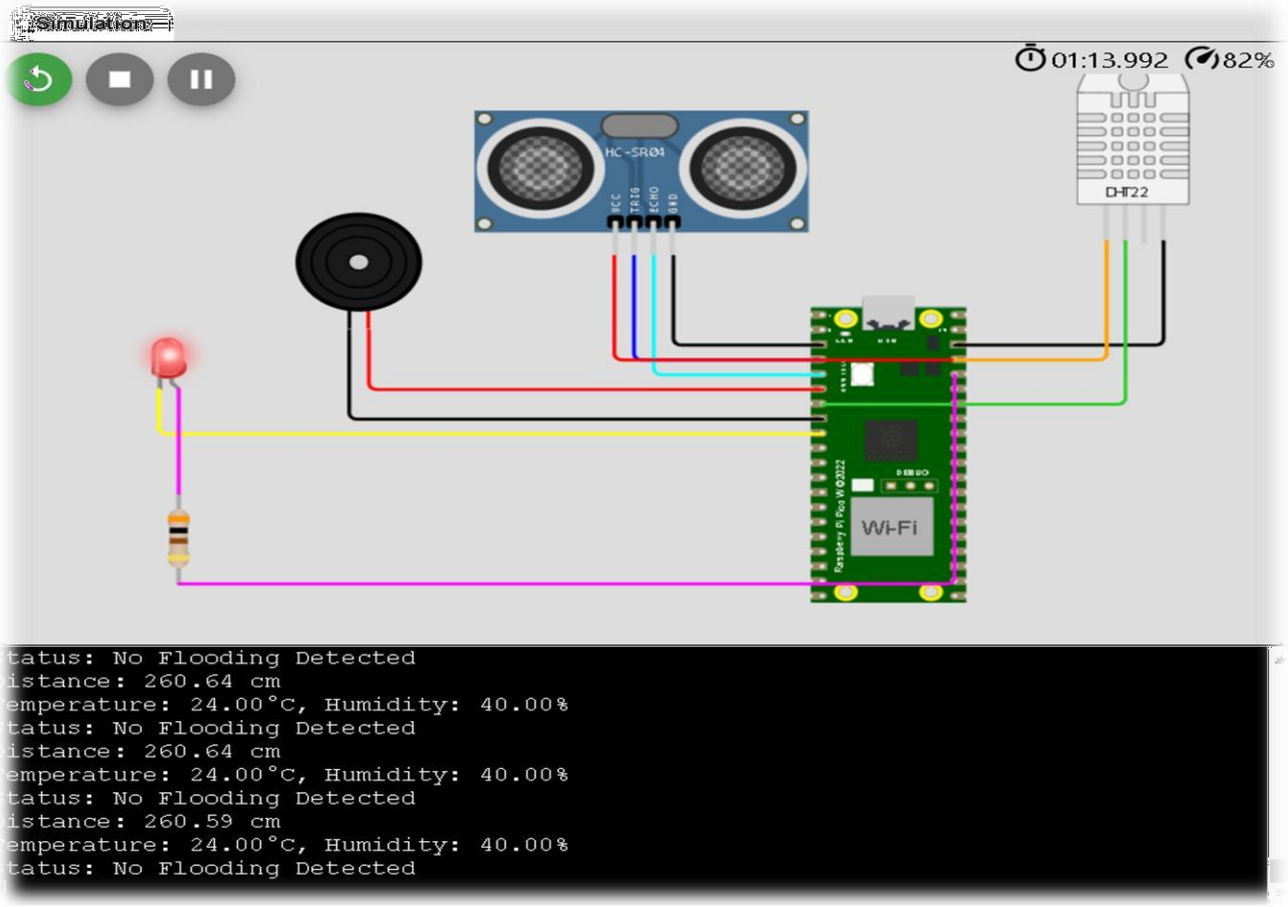
```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    {
      "type": "board-pi-pico-w",
      "id": "pico",
      "top": -118.45,
      "left": 32.35,
      "attrs": { "env": "micropython-20231005-v1.21.0" }
    },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic1",
      "top": -238.5,
      "left": -138.5,
      "attrs": { "distance": "257" }
    },
    {
      "type": "wokwi-buzzer",
      "id": "bz1",
      "top": -180,
      "left": -228.6,
      "attrs": { "volume": "0.1" }
    },
    { "type": "wokwi-dht22", "id": "dht1", "top": -268.5, "left": 167.4, "attrs":
{} },
    { "type": "wokwi-led", "id": "led1", "top": -99.6, "left": -313, "attrs": {
"color": "red" } },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 33.6,
      "left": -317.35,
      "rotate": 90,
      "attrs": { "value": "300" }
    }
  ],
  "connections": [
    [ "ultrasonic1:TRIG", "pico:GP2", "blue", [ "v0" ] ],
```

```
    [ "ultrasonic1:ECHO", "pico:GP3", "cyan", [ "v0" ] ],
    [ "ultrasonic1:GND", "pico:GND.1", "black", [ "v0" ] ],
    [ "bz1:2", "pico:GP4", "red", [ "v0" ] ],
    [ "bz1:1", "pico:GND.2", "black", [ "v0" ] ],
    [ "dht1:GND", "pico:GND.8", "black", [ "v0" ] ],
    [ "dht1:SDA", "pico:GP5", "limegreen", [ "v0" ] ],
    [ "ultrasonic1:VCC", "pico:3V3_EN", "red", [ "v0" ] ],
    [ "dht1:VCC", "pico:3V3_EN", "orange", [ "v0" ] ],
    [ "led1:C", "pico:GP6", "yellow", [ "v0" ] ],
    [ "led1:A", "r1:1", "magenta", [ "v0" ] ],
    [ "r1:2", "pico:3V3", "magenta", [ "h0" ] ]
  ],
  "dependencies": {}
}
```

Food Monitoring design:



Output:



Data Transmission:

To send data to the early warning platform, you might use protocols like MQTT, HTTP, or other IoT-specific communication methods.Set up the requiredcredentials and endpoint for your platform.

Loop and Send Data:

Continuously collect data from the sensor and send it to your platform.

Error Handling and Logging:

Implement error handling to manage network issues or sensor failures.Consider adding logging to keep track of the sensor's performance.

Deployment and Integration:

Deploy your IoT device to the desired location.Integrate your IoT platformwith the early warning system.

Security Considerations:

Ensure that your IoT device and data transmission are secure. Implement authentication and encryption as needed.

Testing and Monitoring:

Test your system thoroughly and monitor its performance over time.

Conclusion:

To developing an IoT sensor system to send water level data to an early warning platform involves several key steps, including hardware setup, data collection, transmission, error handling, and security considerations. The actual implementation will depend on your specific project's requirements and the IoT hardware and platform you are using. It's essential to thoroughly test and monitor your system to ensure its reliability and performance. If you have specific questions or need further assistance with any aspect of this project, please feel freeto ask.