



Karkhe Dam Simulation

A Real-Time Monitoring and Control System
using Wokwi, Node-RED, and React.js

Mohammadhosein Abdollahi (6019532)
Anita Ghezek Ayagh Tehrani (5843905)



Motivation & Problem Statement

- Why Dam Monitoring?

Prevent floods, and manage water resources efficiently

Traditional methods lack real-time data & automation

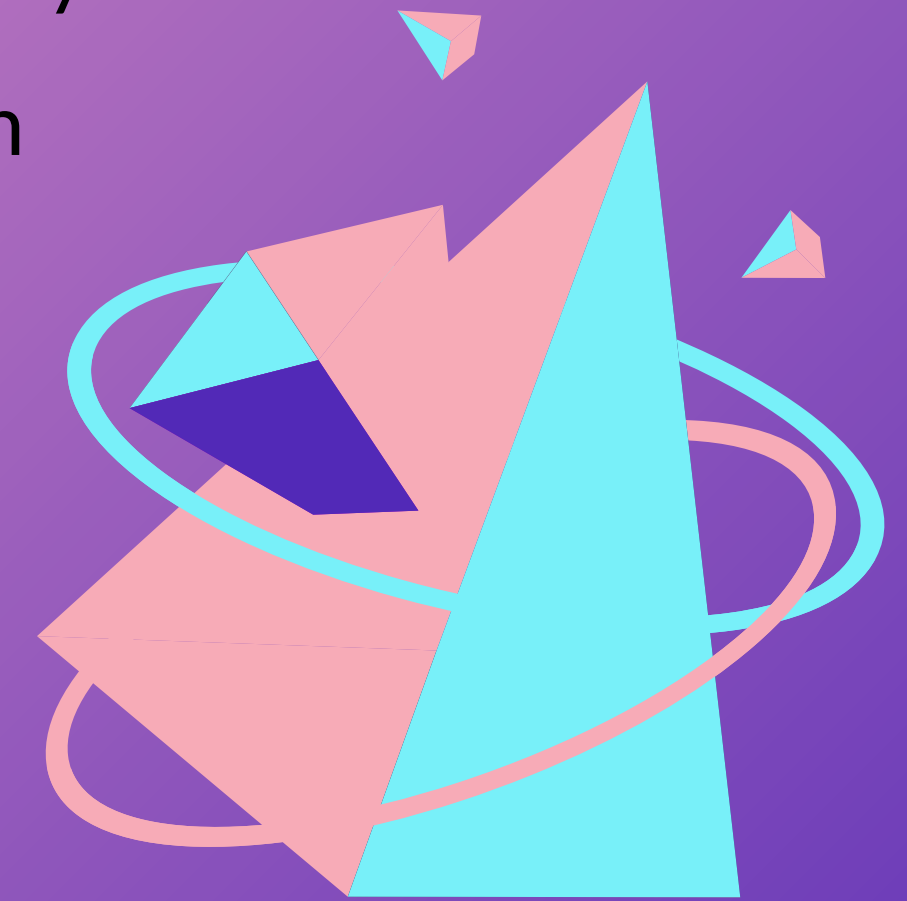
Need for smart water management solutions

- Solution: IoT-Based System

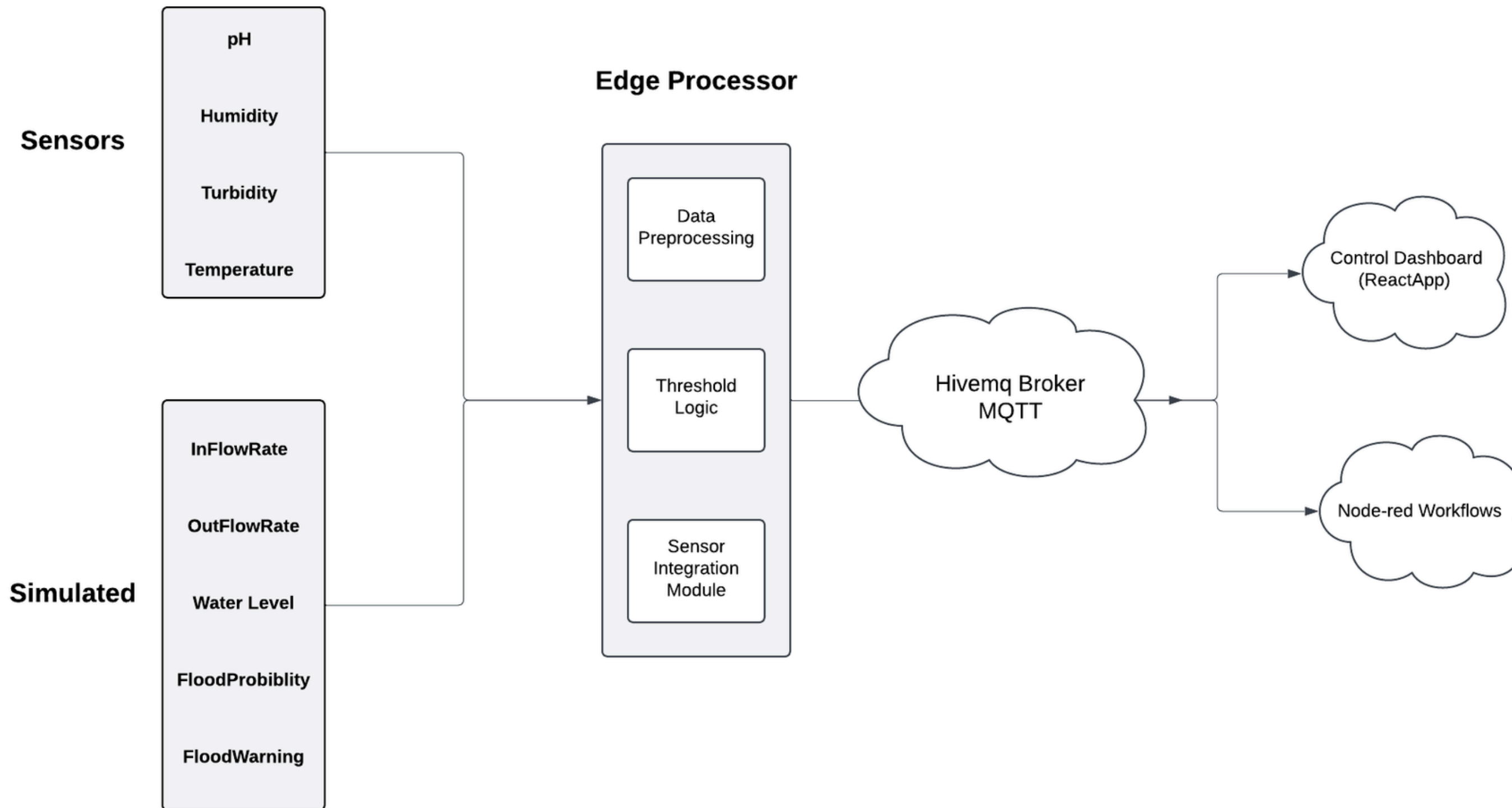
Uses ESP32 + Sensors for real-time data collection

Publishes data to MQTT Cloud (HiveMQ)

Dashboards (Node-RED & React.js) for monitoring & remote control



System Architecture



Sensors



Sensors	Type	Measurement
DHT22	Digital	Temperature & Humidity
pH Sensor	Analog	Water pH Level
Turbidity Sensor	Analog	Water Clarity

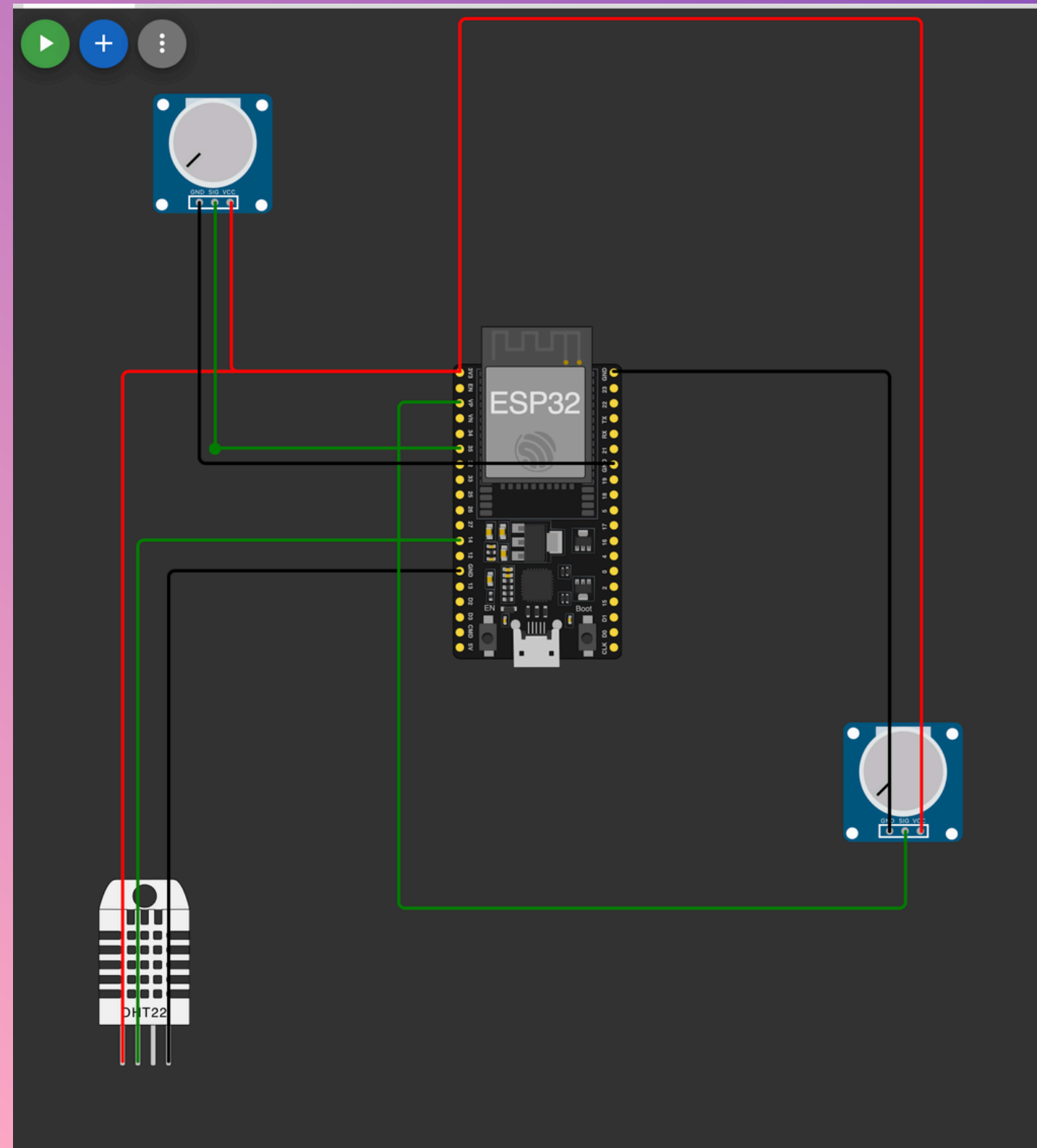


```
// Define sensor pins
#define DHTPIN 14           // DHT22 Sensor Pin
#define DHTTYPE DHT22      // DHT22 Sensor Type
#define TURBIDITY_PIN 35    // Analog for turbidity
#define PH_PIN 36           // Analog pH sensor

// Read sensor data
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
float pH = analogRead(PH_PIN) > 0 ? analogRead(PH_PIN) / 4095.0 * 0.7 + 7.5 : random(75, 82) / 10.0;
int turbidityRaw = analogRead(TURBIDITY_PIN);
```




Wokwi Diagram



Wokwi Setup



```
void connectToWiFi() {
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }
  Serial.println(" Connected to Wokwi-
  GUEST!");
}
```

```
void connectToMQTT() {
  while (!client.connected()) {
    Serial.print("Connecting to MQTT
    Broker at ");
    Serial.print(mqttServer);
    Serial.print(":");
    Serial.println(mqttPort);

    if (client.connect("ESP32Client",
    mqttusername, mqttpassword)) {
      Serial.println(" Connected to MQTT
      Broker!");
    }

    client.subscribe("KarkheDam_Simulation/o
    penDam");
  } else {
    Serial.print(" Failed to connect,
    state: ");
    Serial.println(client.state());
    delay(2000);
  }
}
```

```
void publishData(const char* topic, float value) {
  char payload[50];
  sprintf(payload, "%.2f", value);
  if (client.publish(topic, payload)) {
    Serial.printf("Published %s: %.2f\n", topic,
    value);
  } else {
    Serial.printf("Failed to publish %s\n", topic);
  }
}
```

```
void mqttCallback(char* topic, byte* payload,
  unsigned int length) {
  String message;
  for (unsigned int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  if (String(topic) ==
  "KarkheDam_Simulation/openDam") {
    openDam = (message == "true");
    Serial.printf("Dam control received: %s\n",
    openDam ? "Open" : "Closed");
  }
}
```



Data Preprocessing and Threshold Logic

```
bool openDam = false; // Variable to store dam status
float waterLevel = 250;

// Adjust turbidity based on other parameters
if (temperature > 30) turbidity += 20;
if (pH < 7.0 || pH > 8.0) turbidity += 15;
if (humidity > 80) turbidity -= 10;
if (waterLevel > 260) turbidity += 25;
turbidity = fmax(fmin(turbidity, 400), 50); // Clamp turbidity between 50 and 400 NTU

// Rainfall logic: dynamically adjust attributes if rainfall occurs
if (temperature < 18) {
    float rainfallIncrease = random(5, 15) * 0.1f; // Increase by a small, random value
    waterLevel = fmin(waterLevel + rainfallIncrease, 280.0f); // Clamp water level to a max of 280 meters
    turbidity -= turbidity * 0.1; // Reduce turbidity by 10%
    pH += (7.5 - pH) * 0.1; // Adjust pH towards neutral (7.5)
    humidity += (80 - humidity) * 0.1; // Gradually increase humidity towards 80%
}

// Calculate inflow and outflow rates dynamically
float inflowRate = (humidity > 60) ? random(300, 400) : random(100, 250);
float outflowRate = openDam ? random(400, 500) : random(100, 250); // Higher outflow if dam is open

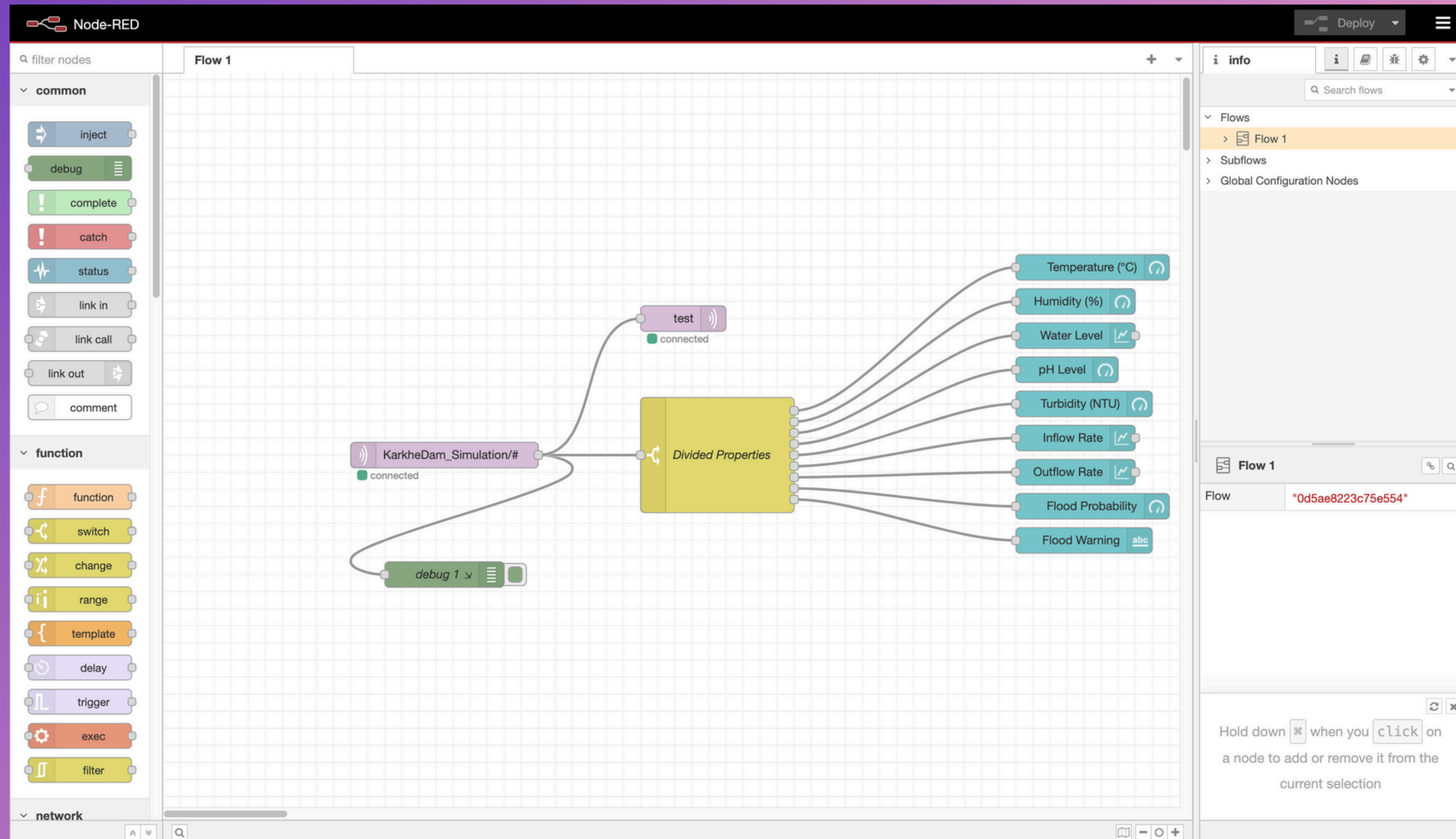
// Calculate actual water level
waterLevel += (inflowRate - outflowRate) * 0.05; // Adjust scale for simulation realism
waterLevel = fmax(fmin(waterLevel, 280.0f), 160.0f); // Clamp water level between 160 and 280 meters

// Close dam if water level is below 160 meters
if (waterLevel <= 160) {
    openDam = false;
    outflowRate = random(100, 250); // Reset to default outflow rate
}

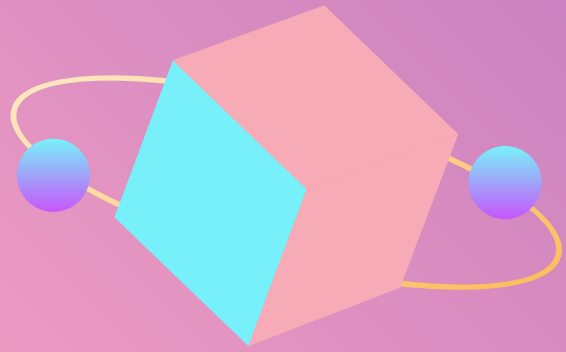
// Calculate flood probability
float floodProbability = 0;
if (waterLevel > 260) floodProbability += 50;
if (inflowRate > 300) floodProbability += 30;
if (turbidity > 300) floodProbability += 20;
floodProbability = std::min(floodProbability, 100.0f);
bool floodWarning = floodProbability > 60;
```




Node-Red



Flow Explanation



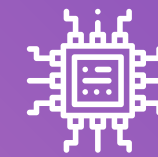
MQTT-IN

Receive the data from the broker



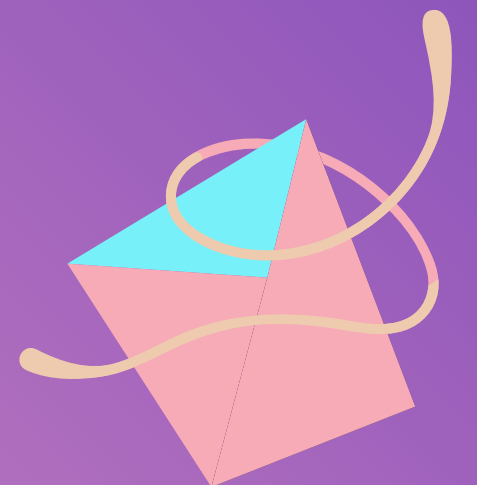
DIVIDED PROPERTIES

Jupiter is the biggest planet in the Solar System



Dashboard

Show the values in the proper way



Node-RED Dashboard UI Elements

Component	Node Type	Topic (MQTT)	Title on UI	Min Value	Max Value	Description
Temperature Gauge	ui_gauge	KarkheDam_Simulation/temperature	Temperature (°C)	0	50	Displays the current temperature from the DHT22 sensor
Humidity Gauge	ui_gauge	KarkheDam_Simulation/humidity	Humidity (%)	0	100	Displays the relative humidity measured by the DHT22 sensor
Water Level Chart	ui_chart	KarkheDam_Simulation/waterLevel	Water Level	150	350	Line graph showing water level changes over time
pH Level Gauge	ui_gauge	KarkheDam_Simulation/pH	pH Level	7	8.5	Indicates the water's acidity or alkalinity level

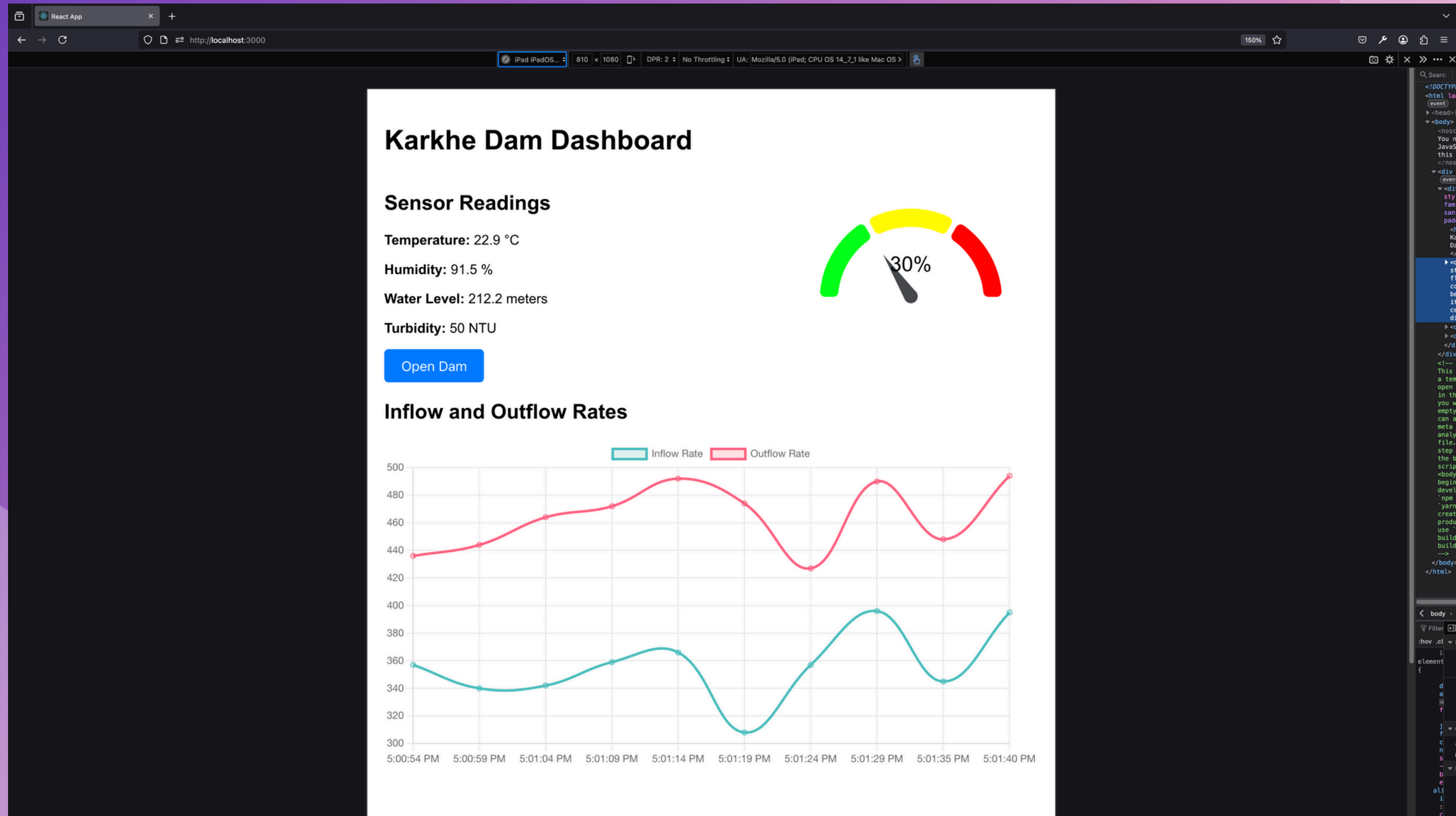
Component	Node Type	Topic (MQTT)	Title on UI	Min Value	Max Value	Description
Turbidity Gauge	ui_gauge	KarkheDam_Simulation/turbidity	Turbidity (NTU)	50	400	Displays water clarity (higher values mean murkier water)
Inflow Rate Chart	ui_chart	KarkheDam_Simulation/inflowRate	Inflow Rate	100	500	Line graph tracking the inflow rate of water into the dam
Outflow Rate Chart	ui_chart	KarkheDam_Simulation/outflowRate	Outflow Rate	50	300	Line graph showing the dam's outflow rate
Flood Probability Gauge	ui_gauge	KarkheDam_Simulation/floodProbability	Flood Probability	0	100	Displays the real-time flood probability percentage
Flood Warning Text	ui_text	KarkheDam_Simulation/floodWarning	Flood Warning	0(No)	1(Yes)	Displays "Flood Warning: Yes"when flood probability exceeds 60%



Node-Red Dashboard UI



Control Dashboard



Subscribing Data =

```
4 clientRef.current.on("message", (topic, message) :void => {
5   const value :number = parseFloat(message.toString());
6   const now :string = new Date().toLocaleTimeString();
7   console.log(`Received message on topic ${topic}: ${value}`);
8
9   switch (topic) {
10    case "KarkheDam_Simulation/temperature":
11      setTemperature(value);
12      break;
13    case "KarkheDam_Simulation/humidity":
14      setHumidity(value);
15      break;
16    case "KarkheDam_Simulation/waterLevel":
17      setWaterLevel(value);
18      break;
19    case "KarkheDam_Simulation/turbidity":
20      setTurbidity(value);
21      break;
22    case "KarkheDam_Simulation/floodProbability":
23      setFloodProbability(value);
24      break;
25    case "KarkheDam_Simulation/floodWarning":
26      setFloodWarning( value: value === 1);
27      if (value === 1 && !isDamOpen) {
28        sendNotification( title: "Flood Warning!", body: "Flood warning! Please open the dam!");
29      }
30      break;
31    case "KarkheDam_Simulation/inflowRate":
32      setInflowData( value: (prev :any[] ) => [...prev.slice(-9), value]);
33      setTimestamps( value: (prev :any[] ) => [...prev.slice(-9), now]);
34      break;
35    case "KarkheDam_Simulation/outflowRate":
36      setOutflowData( value: (prev :any[] ) => [...prev.slice(-9), value]);
37      if (value > 300) {
38        setIsDamOpen( value: true);
39        sendNotification( title: "Dam Opened", body: "The dam is open, releasing water.");
40      }
41      break;
42    default:
43      console.warn( message: `Unhandled topic: ${topic}`);
44  }
```

Links to Project

Wokwi

Github

Control Dashboard Demo

