SMART WATER SYSTEM

PROBLEM DEFINITION:

Definition:

➔ Smart water system can make water management more efficient and safer for consumers and workers through real-time data collection, alerting and actions to prevent issues from occurring.

➔ These smart devices and sensors can help with real-time data collection and alerts to prevent issues from occurring and reduce the workload on the workers responsible for physically checking every inch of infrastructure.

Design thinking:

Project Objectives:

● Smart water solutions are a data-driven approach to handling the modern challenges of water distribution. By analyzing data, water utilities can get a better understanding of the daily cycles of water demand.

● Smart water management systems can provide a more resilient and efficient water supply system, reducing costs and improving sustainability.

IoT Sensor Design:

Flow Sensors: Flow sensors measure the rate of water flow in pipes. They can be installed in water supply lines, irrigation systems, and even at the point of use (e.g., faucets and showers). Flow sensors help in detecting leaks, monitoring water consumption, and optimizing irrigation schedules.

Water Quality Sensors: Water quality sensors measure parameters like pH, turbidity, conductivity, temperature, and the presence of contaminants such as heavy metals and chemicals. Monitoring water quality is essential for ensuring the safety of drinking water and maintaining the health of ecosystems.

Water Pressure Sensors: Water pressure sensors monitor the pressure in water distribution systems. They can detect abnormal pressure drops, which may indicate leaks or other issues in the system

Water Leak Sensors: These sensors are typically placed in areas prone to leaks, such as basements or under sinks. They can detect the presence of water and send alerts to prevent water damage.

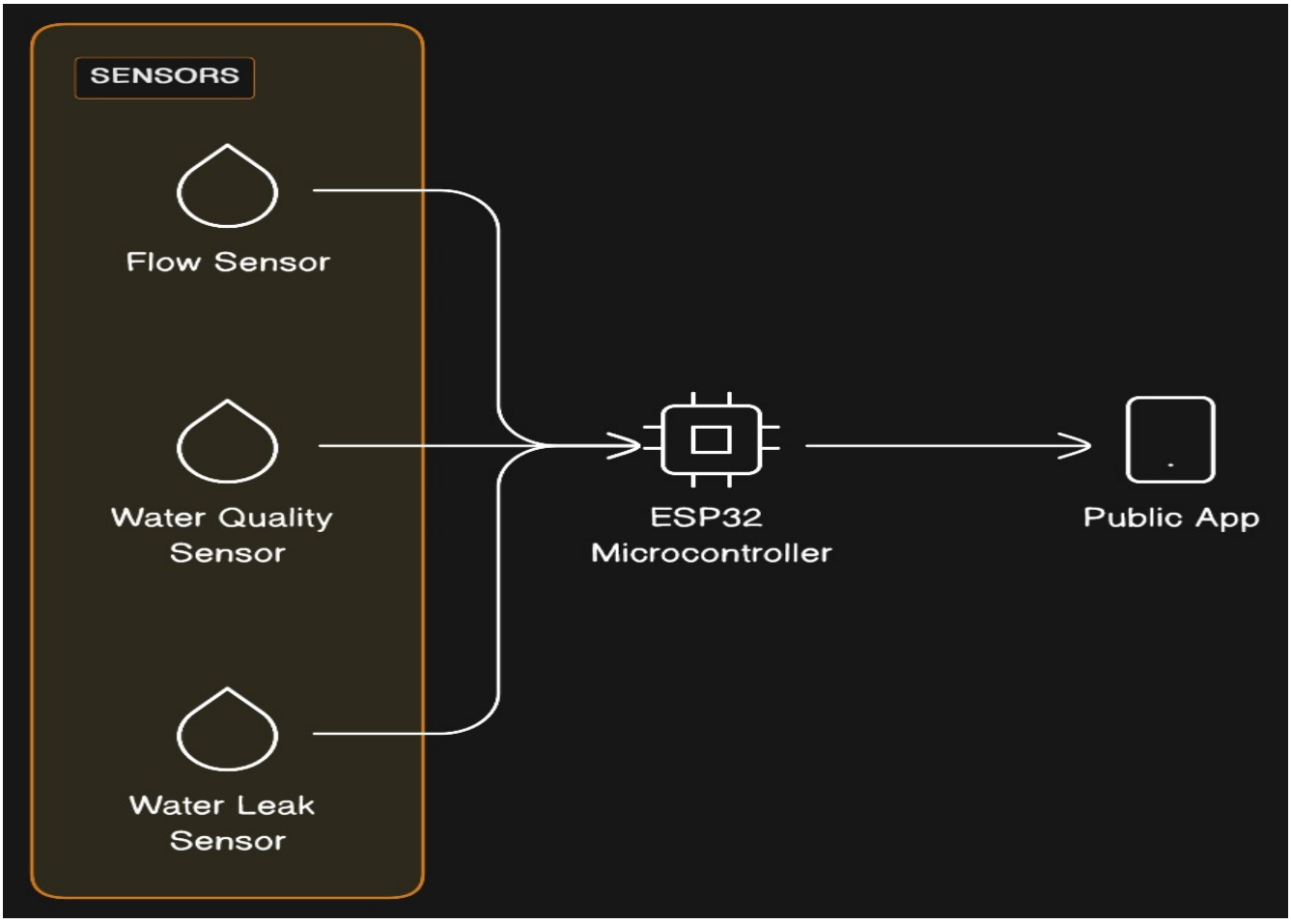
Real-time transit information platform:

ESP32 Firmware: Write firmware for the ESP32 using the Arduino IDE or PlatformIO. You'll need libraries for each sensor to interface with them. Read sensor data at regular intervals.
Wi-Fi Connectivity: Configure the ESP32 to connect to your local Wi-Fi network. You'll need to provide your Wi-Fi credentials in the code.
Data Collection: Collect data from each sensor, such as water flow rate, leak status, water pressure, and water quality parameters. Store this data in variables.
Data Transmission: Transmit the collected data to a cloud server or a local server using HTTP, MQTT, or other suitable protocols. You can use libraries like the ArduinoJson library for JSON data serialization and the PubSubClient library for MQTT communication.
App Development: Develop a mobile or web app to visualize the data. You can use frameworks like React Native for mobile apps or React/Vue.js for web apps.
User Authentication: Implement user authentication to secure the app, ensuring only authorized users can access the data.
Real-Time Updates: Implement real-time data updates in your app to provide users with live information about water consumption and quality.
Data Analysis: Optionally, you can integrate data analysis and visualization tools to provide insights into water consumption patterns and water quality trends.
Notifications: Set up notifications to alert users in case of water leaks, abnormal pressure, or poor water quality.
Historical Data: Store historical data for later analysis and trend analysis.
User Interface: Design a user-friendly interface for the app that displays the sensor data in a clear and understandable way.

Integration Approach:

IoT sensors can send data to a data sharing platform through various communication protocols and technologies. Here are few methods
1. **Wi-Fi:** IoT sensors equipped with Wi-Fi capabilities can directly connect to a Wi-Fi network and transmit the collected data to the data sharing platform through the internet. The data is typically sent in packets using HTTP, MQTT, or other appropriate protocols.
2. **Cellular Networks:** IoT sensors can utilize cellular network connectivity, such as 3G, 4G, or 5G, to transmit data to the data sharing platform. These sensors have built-in cellular modules that establish a connection with the internet and transfer data using protocols like MQTT or CoAP.
3. **LoRaWAN:** LoRaWAN (Long Range Wide Area Network) is a low-power, long-range wireless communication technology ideal for IoT devices. IoT sensors built with LoRaWAN capability can transmit data to gateways, which

forward it to the data sharing platform via backhaul connections like Ethernet or cellular networks.
4. **Zigbee/Z-Wave:** Zigbee and Z-Wave are wireless communication protocols commonly used for home automation systems. IoT sensors with Zigbee/Z-Wave capabilities can transmit data to a central hub or gateway, which then sends it to the data sharing platform over Wi-Fi, Ethernet, or other internet connectivity options.
5. **Bluetooth:** IoT sensors equipped with Bluetooth can establish a connection with a smartphone, gateway, or other Bluetooth-enabled devices acting as a bridge to the data sharing platform. The connected device then forwards the data to the platform via Wi-Fi or cellular networks.
6. **Edge Computing:** In edge computing scenarios, IoT sensors process and analyze the collected data locally before sending only relevant or summarized information to the data sharing platform. This reduces the amount of data transmitted over the network and optimizes bandwidth usage.



Programming Code:

- To build the project by developing the data-sharing platform.
- Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time water consumption data.
- Design the platform to receive and display water consumption data from IoT sensors and promote water conservation efforts.

Micropython code to get the data from ESP32 and its IOT sensors:

```
from machine import Pin
import time
import network
import urequests
WIFI_SSID = 'Wowki-GUEST'
WIFI_PASSWORD = ''
SERVER_IP = 'http://localhost:8080'
SERVER_PORT = '8080'
# Connect to WiFi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PASSWORD)
# Wait for the connection to be established
while not wifi.isconnected():
time.sleep(1)
# Define the pins for your sensors
FLOW_SENSOR_PIN = 14
PRESSURE_SENSOR_PIN = 27
LEAK_SENSOR_PIN = 32
# Set up the pins
flow_sensor = Pin(FLOW_SENSOR_PIN, Pin.IN)
pressure_sensor = Pin(PRESSURE_SENSOR_PIN, Pin.IN)
leak_sensor = Pin(LEAK_SENSOR_PIN, Pin.IN)
def read_sensors():
flow_rate = flow_sensor.value()
pressure = pressure_sensor.value()
leak_detected = leak_sensor.value()
# Send the sensor values to the server
url = 'http://{}:{}/update'.format(SERVER_IP, SERVER_PORT)
```

```
headers = {'content-type': 'application/json'}
data = {'flow_rate': flow_rate, 'pressure': pressure, 'leak_detected': leak_detected}
response = urequests.post(url, json=data, headers=headers)
while True:
read_sensors()
time.sleep(1)
```

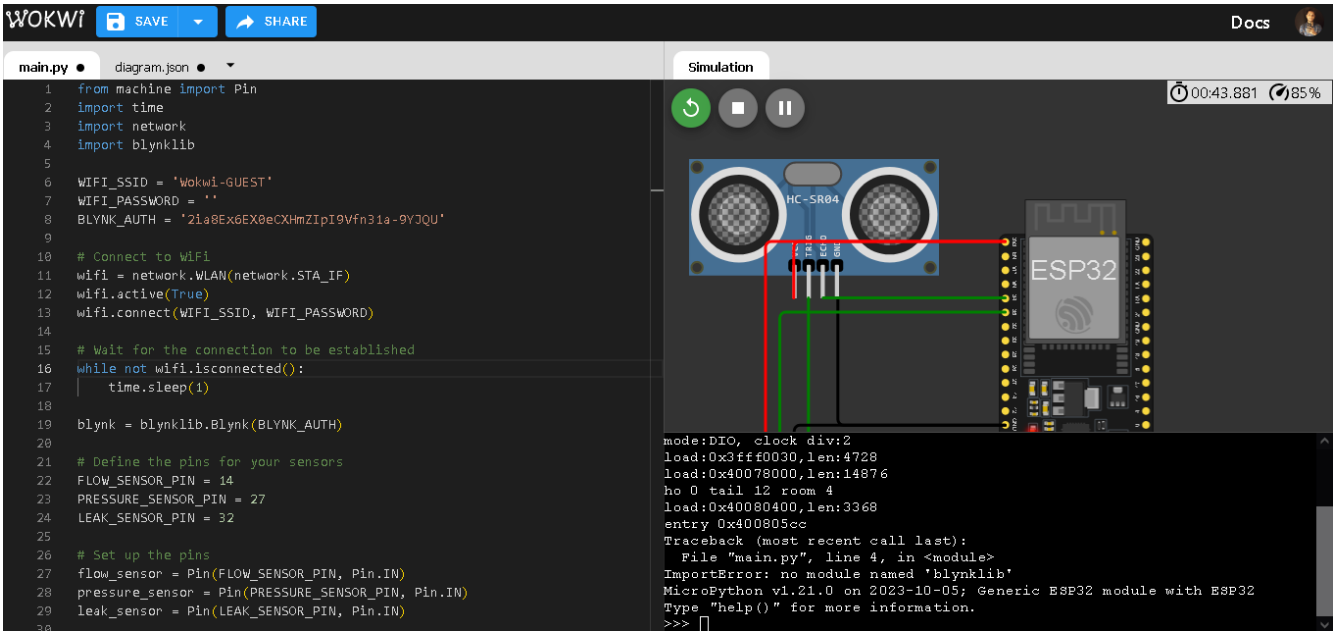
To create a HTTP server using Node.js

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
fs.readFile('yourfile.html', function(err, data) {
res.writeHead(200, {'Content-Type': 'text/html'});
res.write(data);
return res.end();
});
}).listen(8080);
```

HTML Code to view real time data sharing:

```
<!DOCTYPE html>
<html>
<head>
<title>Real Time Data</title>
<script>
var websocket;
window.addEventListener('load', onLoad);
function onLoad(event) {
websocket = new WebSocket('ws://your-esp32-ip-address');
websocket.onopen = onOpen;
websocket.onclose = onClose;
websocket.onmessage = onMessage;
}
function onOpen(event) {
console.log('Connection opened');
}
function onClose(event) {
console.log('Connection closed');
setTimeout(() => onLoad(null), 5000);
}
```

```
function onMessage(event) {
var data = event.data;
document.getElementById('data').innerHTML = data;
}
</script>
</head>
<body>
<h1>Real Time Data from ESP32</h1>
<p id="data"></p>
</body>
</html>
```



Conclusion:

Using micropython code in wokwi to get the realtime data from ESP32 and IOT sensors and by creating HTTP server using Node.js to write a HTML code to visualize and store the data we got from the IOT sensors has been performed above.