Smart water system

Phase 5:

1. Project Objectives: Objective: Create a system that collects data from IoT sensors, processes it using a Raspberry Pi, and displays the information on a mobile app.

Goals: Monitor environmental conditions (temperature, humidity, etc.), track specific metrics, and provide real-time or historical data to users via a mobile app.

2. IoT Sensor Setup: Selection of Sensors: Choose appropriate sensors (like temperature, humidity, light, etc.) for data collection.

Connectivity: Interface sensors with a microcontroller or a small IoT device to collect and transmit data.

Data Transmission: Utilize Wi-Fi, Bluetooth, or other wireless protocols to send data to a central hub (e.g., Raspberry Pi).

3. Mobile App Development: Platform & Tools: Select a development platform (Android, iOS, or both) and appropriate tools (Android Studio, Xcode, etc.).

Features: Design the app UI/UX, implement features to display real-time sensor data, historical trends, alerts, and user interactions.

Connectivity: Integrate the app with the back-end system (Raspberry Pi) to fetch and display sensor data.

4. Raspberry Pi Integration: Data Processing: Use Python or other programming languages to receive, process, and store sensor data.

Integration: Interface Raspberry Pi with the IoT devices/sensors, gather incoming data, perform computations, and potentially store data in a database.

API Development: Create APIs or services to facilitate communication between the Raspberry Pi and the mobile app.

5. Code Implementation: IoT Sensor Code: Develop code to read data from sensors, format it, and transmit it to the Raspberry Pi.

Raspberry Pi Code: Write code to receive, process, and potentially store the incoming sensor data.

Mobile App Code: Develop code for the app's front-end and back-end, enabling data retrieval from the Raspberry Pi and user interactions.

Diagrams and Schematics:

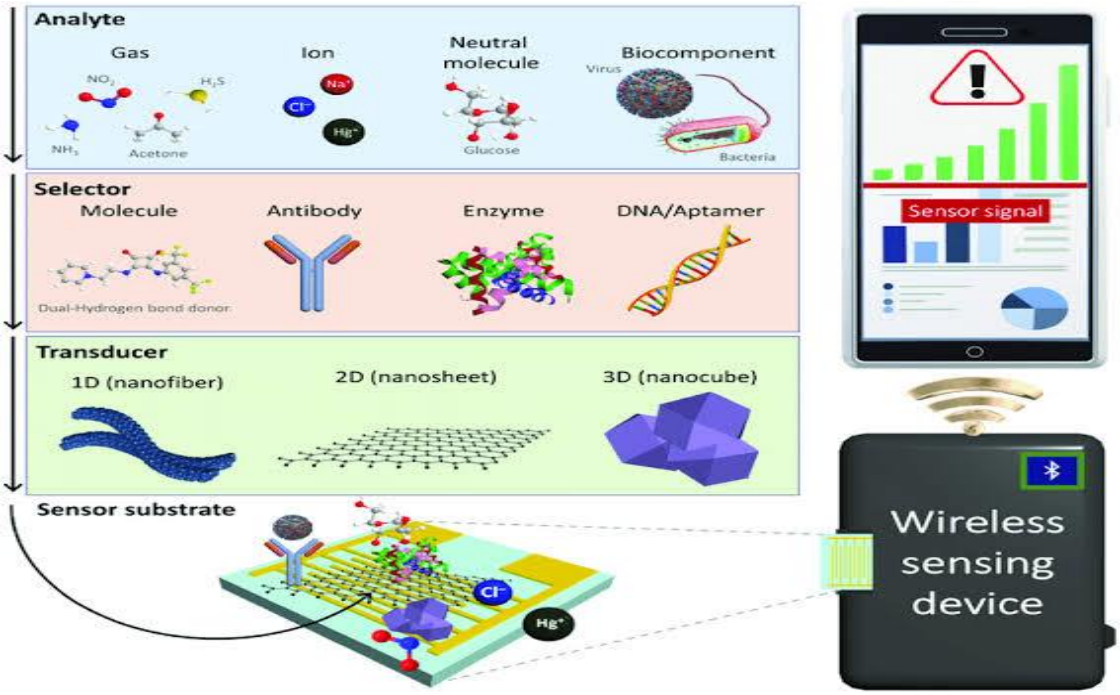
For creating diagrams and schematics, tools like Lucidchart, draw.io, or Adobe Illustrator can be used. You can design diagrams depicting the system architecture, sensor connections, Raspberry Pi integration, and data flow.

Use these tools to illustrate the connections between the IoT sensors, the Raspberry Pi, and the mobile app. Show the flow of data from sensors to the Raspberry Pi and then to the mobile app.

Screenshots of the Mobile App:

If you're developing the mobile app, you can take screenshots directly from the development environment (Android Studio, Xcode, etc.).

Capture screens that display sensor data, graphs showing trends, user interface elements for data visualization, and any other relevant features.



A real-time water consumption monitoring system can significantly contribute to promoting water conservation and sustainable practices in several ways:

1. Awareness and Behavioral Change:

Real-time Data: By providing immediate and accurate information on water usage, individuals and businesses can become more conscious of their consumption patterns.

Behavioral Shift: Access to real-time data encourages users to make informed decisions, leading to more mindful water use.

2. Leak Detection and Prevention:

Early Detection: The system can identify leaks or abnormal water usage instantly, allowing for timely repairs, thus preventing wastage.

Conservation: Detecting and fixing leaks promptly helps in conserving water resources and reducing unnecessary usage.

3. Goal Setting and Accountability:

Set Targets: Users can set consumption targets or goals based on the real-time data received. This encourages individuals or organizations to work towards specific water-saving objectives.

Accountability: Regular updates and notifications can keep users accountable for their usage, fostering a more responsible approach toward water conservation.

4. Data-Driven Decision Making:

Insights and Analytics: Real-time and historical data analysis can offer insights into usage trends, peak times, and areas of high consumption.

Optimization: This information can aid in optimizing water usage strategies, determining where and how to reduce consumption effectively.

5. Education and Community Engagement:

Educational Tool: The system serves as an educational tool, teaching users about their water usage habits and the importance of conservation.

Community Engagement: Sharing collective data or community goals can foster a collaborative approach, encouraging everyone to contribute to water conservation efforts.

6. Resource Management and Environmental Impact:

Efficient Resource Management: Optimizing water usage benefits the community and larger water supply systems, contributing to their sustainability.

Environmental Impact: Reduced water consumption translates to a positive environmental impact, conserving local water sources and supporting ecosystems

integration of IoT Sensors with Python:

Example using Python and Raspberry Pi to collect sensor data:

```
# Sample code to collect data from IoT sensors (e.g., temperature sensor)


import time

import board

import adafruit_dht


# Initialize the sensor

dht_sensor = adafruit_dht.DHT22(board.D4)


while True:

   try:

      temperature_c = dht_sensor.temperature

      humidity = dht_sensor.humidity

      if temperature_c is not None and humidity is not None:

         print("Temperature: {:.1f}°C, Humidity: {}%".format(temperature_c, humidity))
```

```
        else:
            print("Failed to retrieve data")
        time.sleep(2)
    except RuntimeError as error:
        print(error.args[0])
        time.sleep(2)
        continue
```

Basic Transit Information Platform using Python (Flask):

Backend:

```
from flask import Flask, jsonify

app = Flask(_name_)

# Sample transit data
transit_data = {
    "bus_1": {
        "route": "A",
        "current_location": "Bus Stop X",
        "arrival_time": "10:00 AM"
    },
    "bus_2": {
        "route": "B",
        "current_location": "Bus Stop Y",
        "arrival_time": "10:15 AM"
    }
}
```

```
@app.route('/transit-data', methods=['GET'])
def get_transit_data():
    return jsonify(transit_data)


if _name_ == '_main_':
    app.run(debug=True)
```

Frontend: For the frontend, you'd use HTML/CSS along with JavaScript or a JavaScript framework like React or Vue to fetch data from the backend and display it in a user-friendly manner.

Integration: For integrating sensor data into the transit platform, modify the backend to receive and process the IoT sensor data. This could involve creating API endpoints or functions to handle and incorporate sensor data into the transit information provided by the platform.

Raspberry Pi Data Transmission Output:

Assuming a Raspberry Pi gathers data from IoT sensors, the output could be in the form of printed or logged sensor data. For instance, considering temperature and humidity sensor data:

plaintext

Copy code

Temperature: 25.4°C, Humidity: 55%

Temperature: 26.0°C, Humidity: 54%

Temperature: 25.9°C, Humidity: 53%

This data could be displayed or logged on the terminal or stored in a file, a database, or transmitted to an API for further processing or display on a dashboard or mobile app.

Mobile App UI Example:

Transit Information Display:

Homepage:

Welcome screen with the transit system's logo and a brief description.

Navigation options for different sections of the app.

Real-Time Bus Tracking:

Map interface displaying bus routes.

Real-time bus locations marked with icons.

Clicking on an icon shows bus details, such as route, arrival time, and current location.

Temperature and Humidity Sensing:

Separate section displaying temperature and humidity readings.

Graphical representation of historical trends (line chart or graph).

Current readings displayed prominently (e.g., "Temperature: 25°C, Humidity: 60%").

Settings and Alerts:

User preferences for alerts or notifications related to bus schedules or environmental changes.

Options to set personal goals for transit use or environmental targets.