PHASE 5: PROJECT DOCUMENTATION & SUBMISSION

SMART WATER SYSTEM

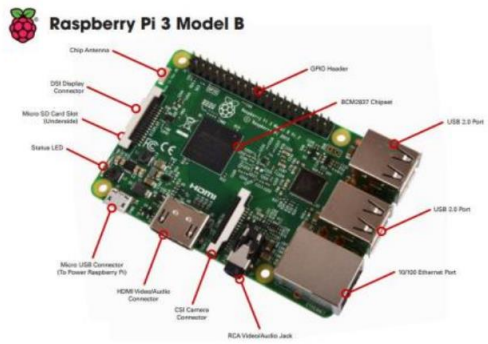
PROJECT'S OBJECTIVES:

The objectives for a Smart Water Management System project are,

1. **Real-time Monitoring**: This objective involves continuously collecting and providing real-time data on various aspects of water usage, quality, and availability. Users can monitor water-related information at any time, helping them stay informed about their water resources.\
2. **Water Conservation**: The goal of water conservation is to promote responsible water usage. By identifying and minimizing wastage, the project aims to reduce the overall consumption of water resources, which is crucial for sustainability.
3. **User-Friendly Interface**: Developing a user-friendly mobile app or interface is essential for easy access to data and control options. It ensures that users can interact with the system effortlessly and access the information they need.
4. **Water Quality Assurance**: Monitoring water quality parameters, such as pH, turbidity, and chlorine levels, helps ensure that the water meets safety and quality standards. Users can be alerted if water quality issues arise.
5. **Leak Detection**: Detecting and notifying users of water leaks or abnormal water usage is essential to prevent water loss and property damage. Timely detection and alerts can help users take prompt action to mitigate potential issues.
6. **Environmental Impact**: Minimizing water waste and reducing the environmental impact of water consumption is a key objective. Responsible water management is essential for preserving natural resources and ecosystems.

IOT SENSOR SETUP:

1. RASPBERRY PI:



- The Raspberry Pi 3 Model B is a compact, single-board computer with the capability to perform a wide range of tasks, equivalent to those typically handled by a standard desktop computer. This includes functions such as spreadsheet work, word processing, internet browsing, programming, and gaming, among others
- Raspberry Pi3ModelB built on the latest Broadcom 2837 ARMv8 64bit processor, the new generation. With built-in wireless and Bluetooth connectivity, it becomes the ideal IoT ready solution
- The Raspberry Pi 3 Model B is equipped with a 1.2GHz quad-core Broadcom BCM2837 64-bit ARMv8 processor. It features built-in BCM43438 Wi-Fi, Bluetooth Low Energy (BLE), 1GB of RAM, 4 USB 2 ports, a 40-pin extended GPIO interface, and both HDMI and RCA video outputs

2. FLOW SENSOR:

- Flow sensors are positioned at water sources or within pipes to gauge the water flow rate and determine the volume of water that has passed through the pipeline.
- Rate of flow of water is measured as litre per hour or cubic meters
- The water flow sensor is composed of a plastic valve that allows water to pass through it. It also incorporates a water rotor and a Hall effect sensor to measure the flow of water.
- The fundamental principle that underlies the operation of this sensor is the Hall effect. In this sensor, the rotation of the rotor induces a voltage difference in the conductor. This induced voltage difference occurs perpendicular to the electric current.

3. ESP8266:

- The ESP8266 can perform two key functions: it can either offload Wi-Fi network management tasks from another application processor or assist in running a separate application.
- The ESP8266 Wi-Fi module is a standalone system-on-chip (SoC) featuring an integrated TCP/IP protocol stack, allowing any microcontroller to seamlessly connect to Wi-Fi networks.

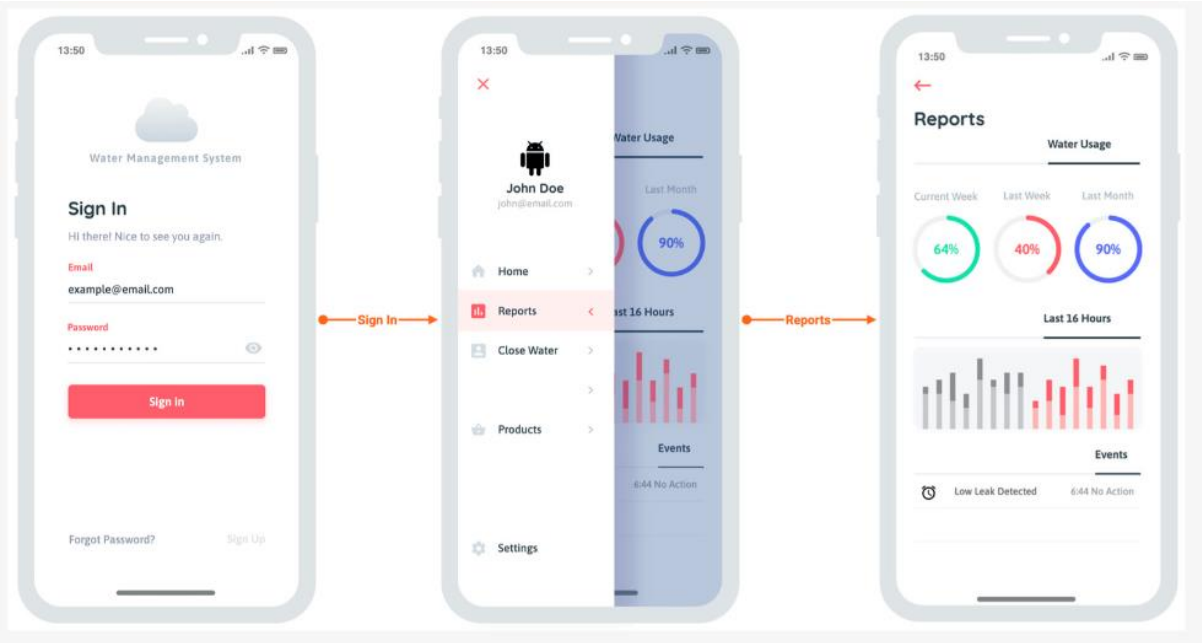
4. ULTRASONIC SENSOR:

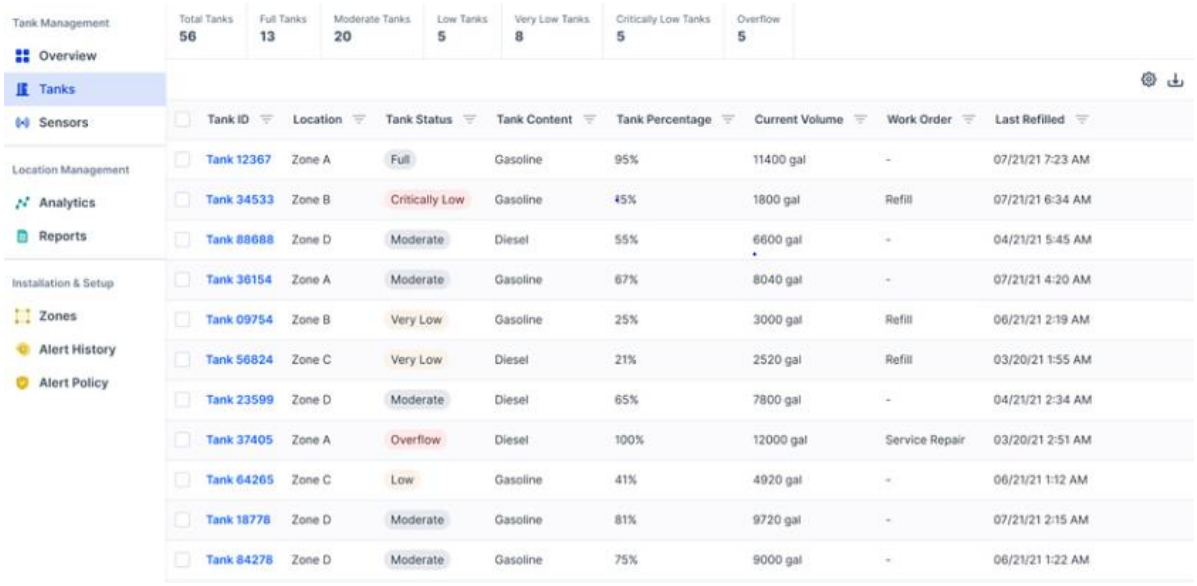
- The HC-SR04 ultrasonic module is a module that can provide noncontact measurement within the range of 2cm to 400cm with a ranging accuracy that can reach 3mm. It works on the principle of echolocation.

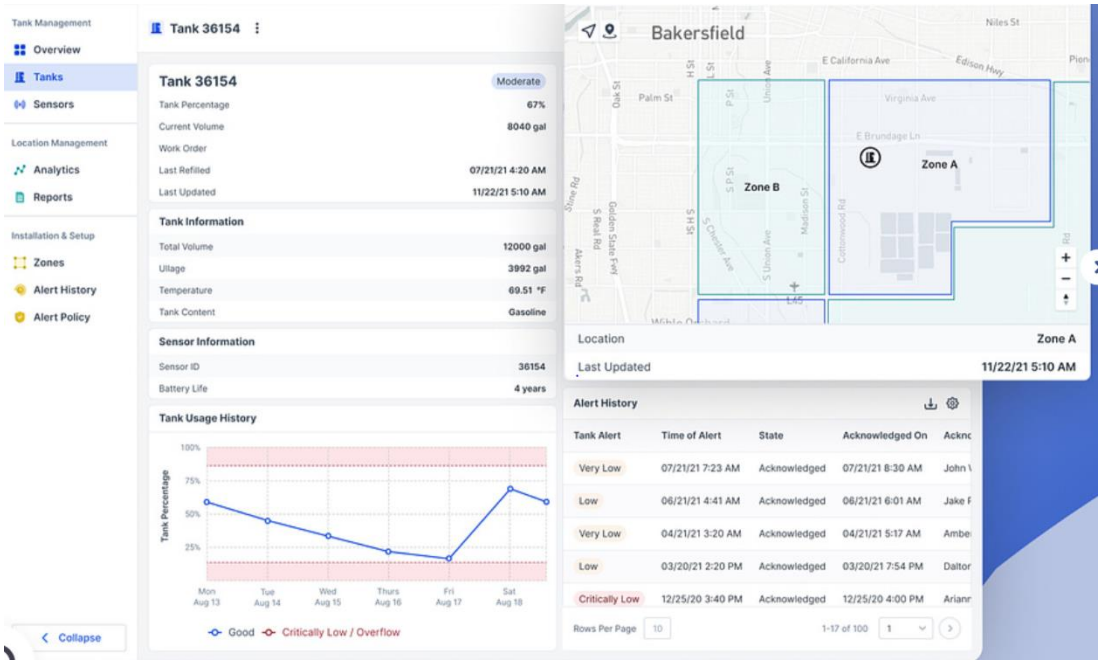
5. PRESSURE SENSOR:

- A pressure sensor, also known as a pressure transducer or pressure transducer, is a device that measures the pressure of a gas or liquid. It typically converts the pressure into an electrical signal, which can be analog or digital, and is proportional to the pressure applied.
- Pressure sensors are commonly used in various applications, including industrial processes, automotive systems, medical devices, and in IoT projects for monitoring water systems and more.
-

MOBILE APP DEVELOPMENT:







RASPBERRY PI INTEGRATION:

Integrating a Raspberry Pi into a smart water management system can add significant processing power, data storage, and connectivity capabilities to the project.

- **Hardware Setup**:
  - Connect the Raspberry Pi to the power supply and ensure it's connected to the internet, either via Wi-Fi or Ethernet.
- **Sensor Connectivity**:
  - Interface the Raspberry Pi with the water sensors (e.g., flow sensors, water quality sensors, pressure sensors) using appropriate communication protocols. This may involve GPIO pins, I2C, SPI, or other interfaces, depending on the sensor types.
- **Data Acquisition**:
  - Develop or configure software on the Raspberry Pi to read data from the sensors at regular intervals. This data can include water flow rates, quality measurements, and pressure data.
- **Data Processing**:
  - Utilize the processing power of the Raspberry Pi to perform data processing tasks. This can involve real-time calculations, data filtering, or other data manipulation as needed for our application.
- **Data Storage**:
  - Store the collected data in a local database on the Raspberry Pi or in a cloud-based solution, depending on our data retention and access requirements.
- **IoT Communication**:
  - Establish communication between the Raspberry Pi and the IoT platform or cloud service. This enables us to send data to the cloud for remote access and analysis.
- **Cloud Integration**:
  - Integrate your Raspberry Pi with cloud services like AWS, Azure, or Google Cloud to store and process data remotely. This allows us to access the data from anywhere with an internet connection.
- **User Interface**:

- Develop a user interface using the Raspberry Pi to display data, control water-related devices, and provide an easy-to-use dashboard for users.

- **Alerts and Notifications**:
  - Set up alerting and notification systems on the Raspberry Pi to inform users or administrators of critical events, such as water leaks or quality issues.

- **Security Measures**:
  - Implement security measures to protect the data and communication between the Raspberry Pi and the cloud, ensuring data privacy and system integrity.

- **Data Analytics:**
  - Use the processing capabilities of the Raspberry Pi to perform data analytics, generate reports, and provide insights into water usage patterns and trends.

CODE IMPLEMENTATION:

PYTHON CODE FOR SENSORS:

```
import time
import jwt
import paho.mqtt.client as mqtt
import ssl
import random
from gpiozero import MCP3008
import RPi.GPIO as GPIO
# Define the Google Cloud IoT Core project ID and cloud region
project_id = "your-project-id"
cloud_region = "your-cloud-region"
# Define the device information
device_id = "your-device-id"
registry_id = "your-registry-id"
mqtt_bridge_hostname = "mqtt.googleapis.com"
mqtt_bridge_port = 8883
# Define the path to your private key and certificate files
private_key_file = "path-to-your-private-key-file"
cert_file = "path-to-your-certificate-file"
# Create a function to generate a JWT token for device authentication
def create_jwt():
 token = {
```

```
 'iat': time.time(),
 'exp': time.time() + 60 * 20,
 'aud': project_id
 }
 with open(private_key_file, 'r') as f:
 private_key = f.read()
 return jwt.encode(token, private_key, algorithm='RS256')
# Define the callback function for MQTT connection
def on_connect(client, userdata, flags, rc):
 print("Connected with result code " + str(rc))
 client.subscribe("/devices/{}/config".format(device_id))
# Define the callback function for MQTT message reception
def on_message(client, userdata, msg):
 print("Received message '{}' on topic '{}'".format(msg.payload, msg.topic)
# Set up the MQTT client
mqtt_client = mqtt.Client(client_id="projects/{}/locations/{}/registries/{}/devices/{}".format(
 project_id, cloud_region, registry_id, device_id))
mqtt_client.username_pw_set(username="unused", password=create_jwt())
mqtt_client.tls_set(certfile=cert_file, cert_reqs=ssl.CERT_REQUIRED,
tls_version=ssl.PROTOCOL_TLSv1_2, ciphers=None)
mqtt_client.on_connect = on_connect
mqtt_client.on_message = on_message
# Connect to Google Cloud IoT Core
mqtt_client.connect(mqtt_bridge_hostname, mqtt_bridge_port)
# Start the MQTT loop
mqtt_client.loop_start()
# Function to read data from the flow sensor
def get_flow_data():
 return round(random.uniform(0.0, 10.0), 2)
# Function to read data from the level sensor
def get_level_data():
 with MCP3008(channel=0) as adc:
 voltage = adc.value
 level_data = map_voltage_to_level(voltage)
 return level_data
```

```
# Function to read data from the pressure sensor
def get_pressure_data():
 with MCP3008(channel=1) as adc:
 voltage = adc.value
 pressure_data = map_voltage_to_pressure(voltage)
 return pressure_data
# Simulate sending sensor data to Google Cloud IoT Core
while True:
 flow_data = get_flow_data()
 level_data = get_level_data()
 pressure_data = get_pressure_data()
 payload = {
 "flow": flow_data,
 "level": level_data,
 "pressure": pressure_data
 }
 mqtt_client.publish("/devices/{}/events".format(device_id), str(payload))
 time.sleep(10)
```

MOBILE APPLICATION:

HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Water App</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <div class="login-container">
        <h1>Login</h1>
        <form id="login-form">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required>
            <label for="password">Password</label>
            <input type="password" id="password" name="password" required>
            <button type="submit">Login</button>
```

```
        </form>
    </div>
    <div class="dashboard-container">
        <h1>Water Usage Dashboard</h1>
        <p id="user-welcome"></p>
        <div id="usage-data">
            <!-- Water usage data will be displayed here -->
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

CSS:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

.login-container, .dashboard-container {
    display: none;
    padding: 20px;
}
h1 {
    text-align: center;
}
form {
    text-align: center;
}
input[type="text"], input[type="password"] {
    width: 100%;
    margin: 10px 0;
    padding: 10px;
    border: 1px solid #ccc;
```

```
}
button {
    background-color: #0074d9;
    color: #fff;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
}
#usage-data {
    /* Style for displaying water usage data */
}
```

JS:

```
// Simulated data for demonstration purposes
let waterFlowRate = 0; // Initial flow rate (liters per minute)
let waterPressure = 0; // Initial pressure (PSI)
// Function to simulate data update from IoT devices
function updateIoTData() {
    // Simulating data updates from IoT devices
    waterFlowRate = Math.random() * 10; // Simulate flow rate changes
    waterPressure = Math.random() * 50; // Simulate pressure changes
    // Send data to Raspberry Pi for processing and communication with the cloud
    sendToRaspberryPi(waterFlowRate, waterPressure);
    // Repeat the update every 5 seconds (simulated real-time)
    setTimeout(updateIoTData, 5000);
}
// Function to send data to Raspberry Pi
function sendToRaspberryPi(flowRate, pressure) {
    // Simulate sending data to the Raspberry Pi
    console.log(`Sending data to Raspberry Pi: Flow Rate - ${flowRate} L/min, Pressure - ${pressure} PSI`);
}
// Function to send data to ESP8266 for IoT communication
function sendToESP8266(data) {
    // Simulate sending data to the ESP8266
```

```
    console.log(`Sending data to ESP8266: ${data}`);
}

// Start simulating data updates
updateIoTData();
```

Real time water consumption monitoring system can promote water conservation and sustainable practices

A real-time water consumption monitoring system can play a crucial role in promoting water conservation and sustainable practices by providing accurate and up-to-the-minute data on water usage. Here's how such a system can contribute to these goals:

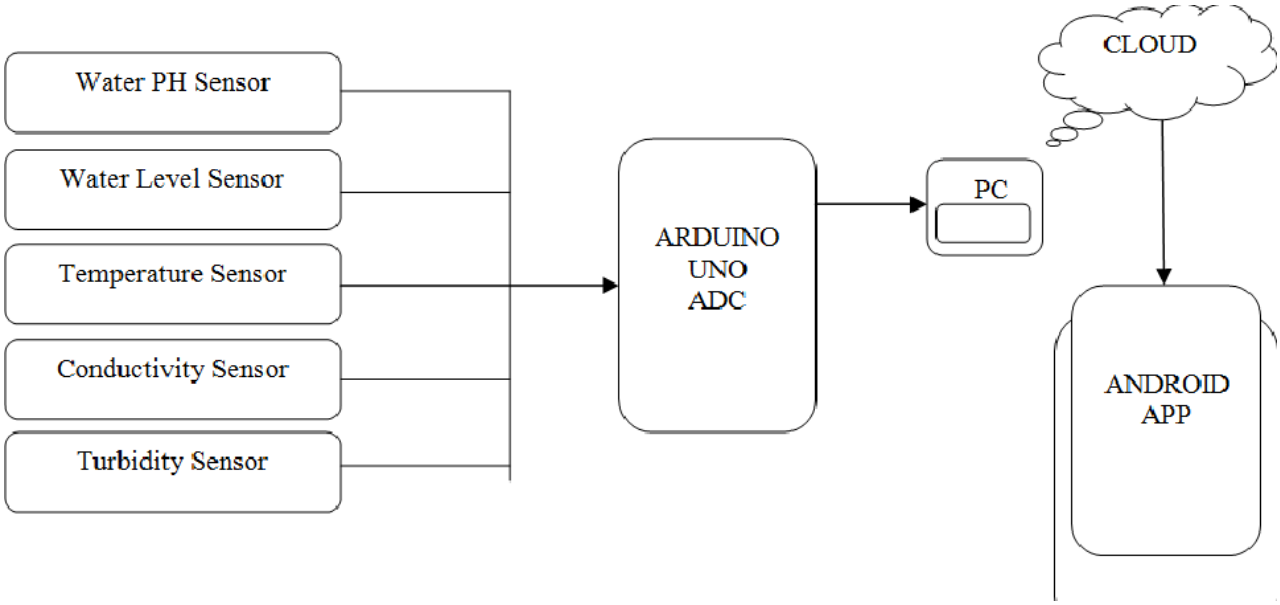
1. Awareness and Education:
    - Real-time monitoring systems make people more aware of their water usage patterns. When individuals or organizations can see how much water they are using in real-time, they become more conscious of their consumption.
    - This awareness can lead to education and behavioral change as people learn about the importance of water conservation and how their actions impact water resources.
2. Leak Detection and Prevention:
    - These systems can identify water leaks as soon as they occur. Leaks are a major source of water wastage, and quick detection can prevent water from being wasted and also help avoid property damage.
    - By addressing leaks promptly, water users can save money on their bills and contribute to overall conservation efforts.
3. Goal Setting and Targets:
    - Real-time data allows individuals and organizations to set specific water usage goals and targets. They can track their progress toward these goals and make adjustments as needed.
    - This goal-oriented approach encourages more efficient water use and encourages a sense of responsibility for water conservation.
4. Incentives and Rewards:
    - Some real-time monitoring systems offer incentives or rewards for achieving water conservation goals. This can be particularly effective in motivating individuals and organizations to reduce their water consumption.
    - Incentives might include discounts, rebates, or recognition for outstanding conservation efforts.
5. Data Analytics and Trend Analysis:
    - Real-time monitoring systems collect data over time, which can be used for trend analysis and identifying usage patterns. This information is valuable for making informed decisions about water management and conservation strategies.
    - Water utilities, businesses, and policymakers can use this data to identify areas of high consumption and implement targeted conservation measures.
6. Timely Alerts and Notifications:
    - These systems can send alerts and notifications when water usage exceeds a predefined threshold or when unusual usage patterns are detected. Users can then take immediate action to address the issue and conserve water.

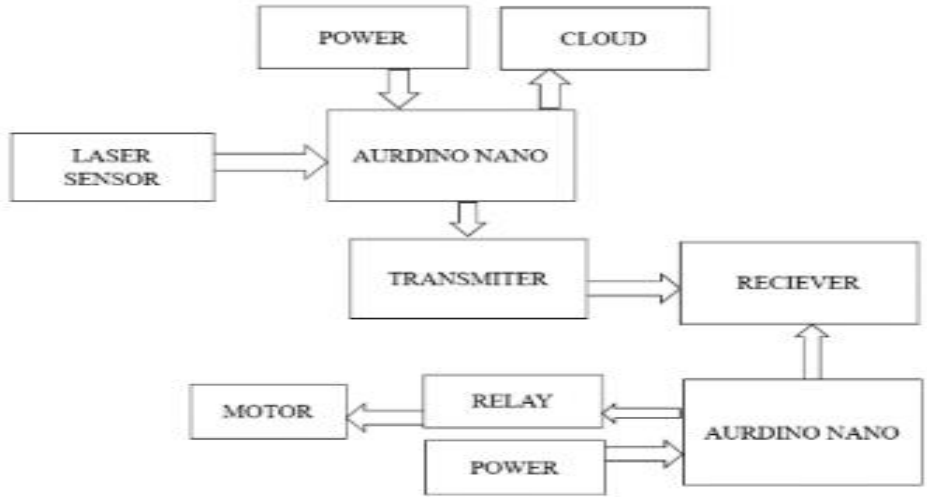
- Timely alerts are especially useful for commercial and industrial users who can prevent excessive water usage during non-production hours.

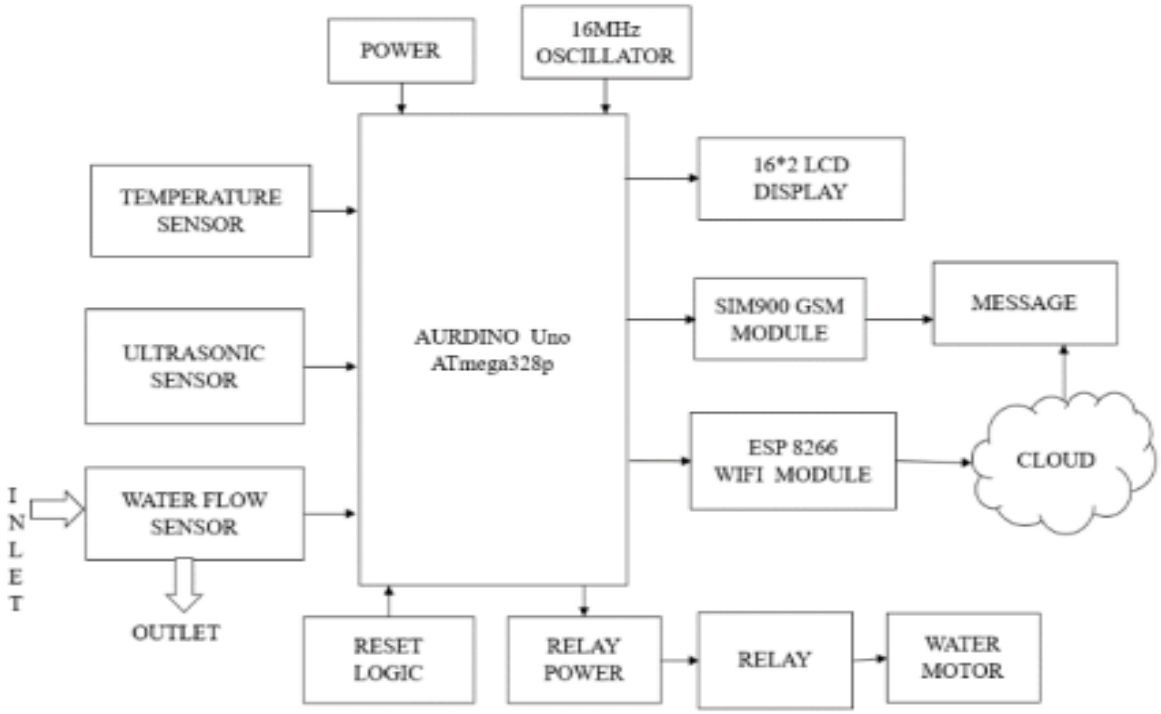
7. Improved Water Resource Management:
    - Real-time data can help water utilities and authorities manage their resources more efficiently. They can allocate water resources more effectively and plan for infrastructure upgrades based on usage data.
    - This can lead to a more sustainable and resilient water supply system.
8. Public and Private Sector Collaboration:
    - Real-time water monitoring systems can facilitate collaboration between the public and private sectors. Businesses and industries can work with local authorities to implement conservation practices, reduce water waste, and contribute to the sustainability of water resources in the region.

In summary, a real-time water consumption monitoring system empowers individuals, businesses, and communities to actively participate in water conservation and sustainable practices. By providing accurate and timely information, it fosters awareness, encourages responsible water use, and helps prevent wastage. It also supports informed decision-making and collaboration between stakeholders to ensure the long-term sustainability of water resources.

DIAGRAMS AND SCHEMATICS:







REPLICATION OF THE PROJECT:

Equipment and Materials:

Gathering the necessary equipment, materials, and IoT sensors, such as water quality sensors, flow sensors, temperature sensors, IoT development boards, and the hardware for the transit information platform.

Sensor Deployment:

Strategically deploy sensors in locations near water sources to collect relevant data.

IoT Platform and Connectivity:

Choose an IoT platform for data collection and remote monitoring. Set up connectivity, like Wi-Fi, LoRa, or cellular, depending on your deployment location.

Sensor Programming and Configuration:

Develop code for the IoT devices to collect data from the sensors. Program the IoT development boards to send data to the IoT platform via MQTT, HTTP, or other protocols.

Data Collection and Transmission:

Verify that the IoT devices collect data accurately and transmit it to the IoT platform.

Dashboard and Visualization:

Create a user-friendly dashboard for monitoring sensor data in real-time. You can use tools like Grafana, Tableau, or custom web development for this purpose.

Alerts and Notifications:

Implement an alerting system to notify relevant stakeholders when certain water quality parameters are out of range or when anomalies are detected.

Transit Information Platform Development:

Develop the transit information platform, which includes collecting data from various sensors, processing it, and presenting it in a user-friendly format.

User Authentication and Access Control:

Implement user authentication and access control to manage who can access and interact with the transit information platform.

Field Deployment:

Deploy the IoT sensors in the field, ensuring proper installation and calibration. The transit information platform components should also be set up and connected to the sensors.

Testing and Validation:

Thoroughly test the entire system, including sensors, data transmission, data storage, visualization, and the transit information platform to ensure it functions as intended.

Data Security and Privacy:

Implement security measures to protect data, devices, and systems from unauthorized access and data breaches.

Regulatory Compliance:

Ensure that the project complies with environmental regulations, data privacy laws, and any other relevant legal requirements.

Maintenance and Monitoring:

Regularly maintain and update the system to ensure its long-term functionality. Monitor data quality and system performance continuously.

PYTHON BASED INTEGRATION FOR THE PROJECT:

Data Integration:

Merge or combine the data from the Smart Water Management System (water quality, flow, temperature data) and the Transit Information Platform (transit data) as required.

Data Storage:

Store the integrated data in a suitable database system. You can use popular Python libraries like SQLAlchemy for relational databases or MongoDB for NoSQL databases.

Data Analysis:

Implement data analysis algorithms in Python to derive insights from the integrated data. This may include statistical analysis, machine learning, AND data visualization.

Visualization:

Create visualizations or dashboards to present the integrated data in a user-friendly format. Python libraries such as Matplotlib, Seaborn, and Plotly helpS with data visualization.

API Development:

To expose the integrated data to external systems, develop RESTful APIs using Python frameworks like Flask or Django REST framework.

Testing:

Thoroughly test the integrated system to ensure that it works as intended. Implement unit tests and integration tests to catch bugs and issues.

Deployment:

Deploy the integrated system to your chosen environment. You can use platforms like AWS, Azure, or host it on your own servers.

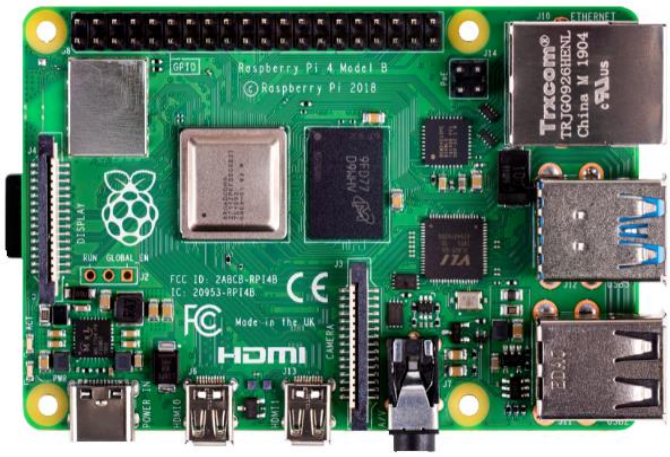
OUTPUT:

RASPBERRY PI DATA TRANSMISSION:

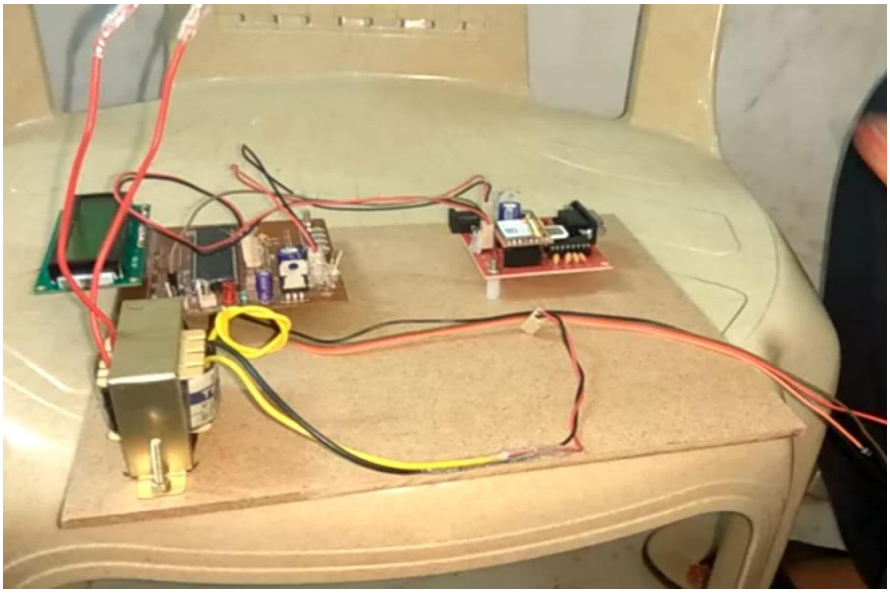
LCD Display Message:

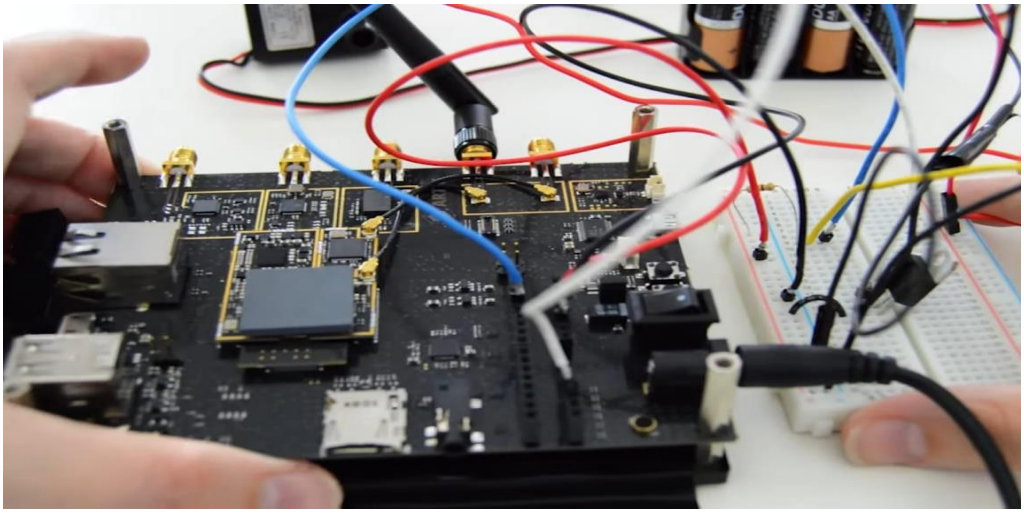
PH Sensor:

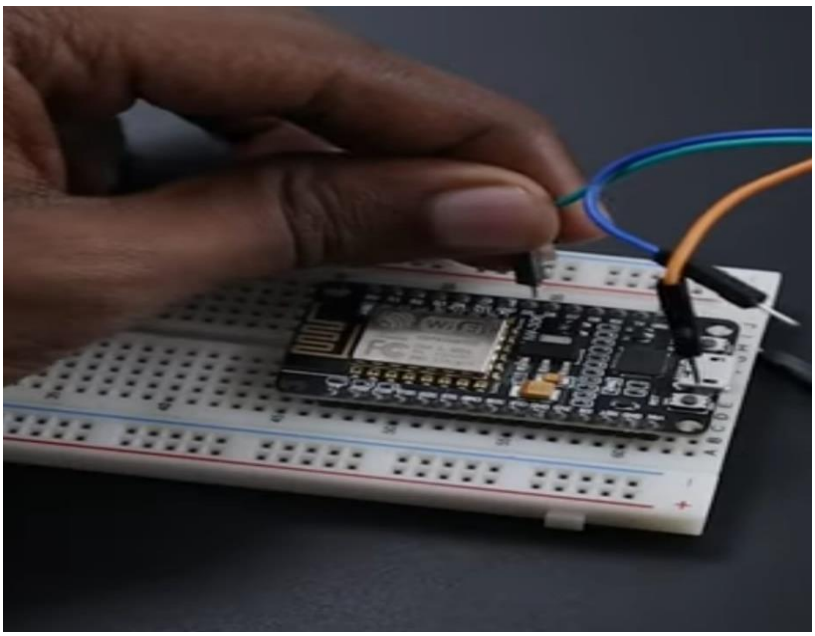
Raspberry Pi 4:



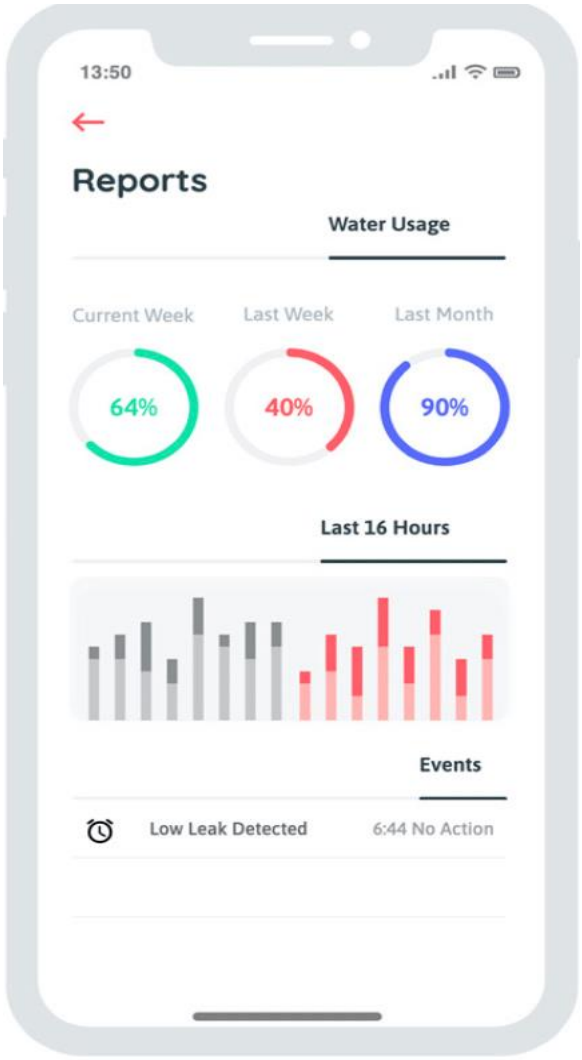
LIVE IMPLEMENTATION:

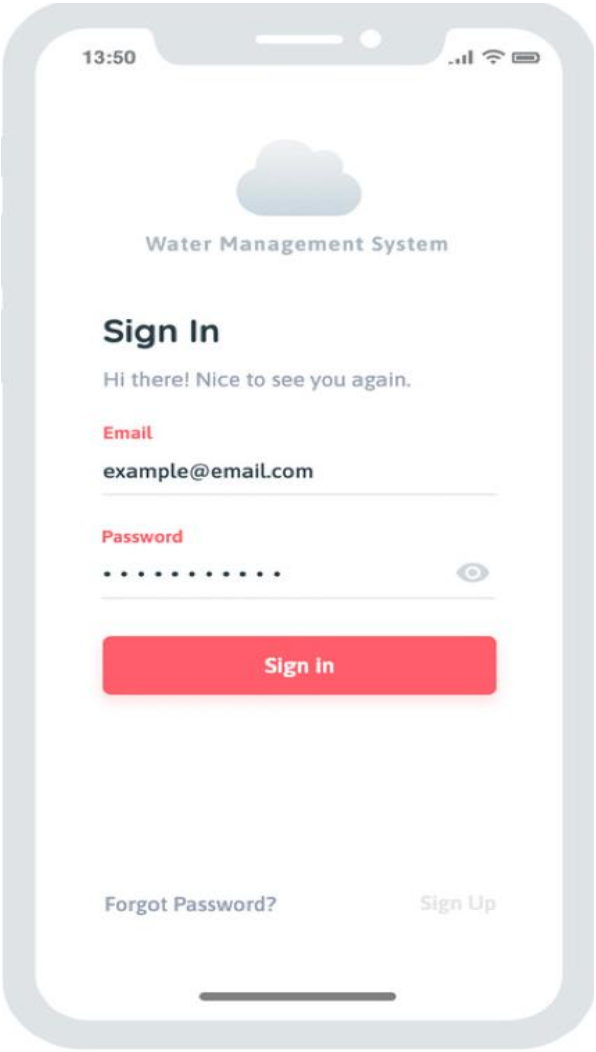


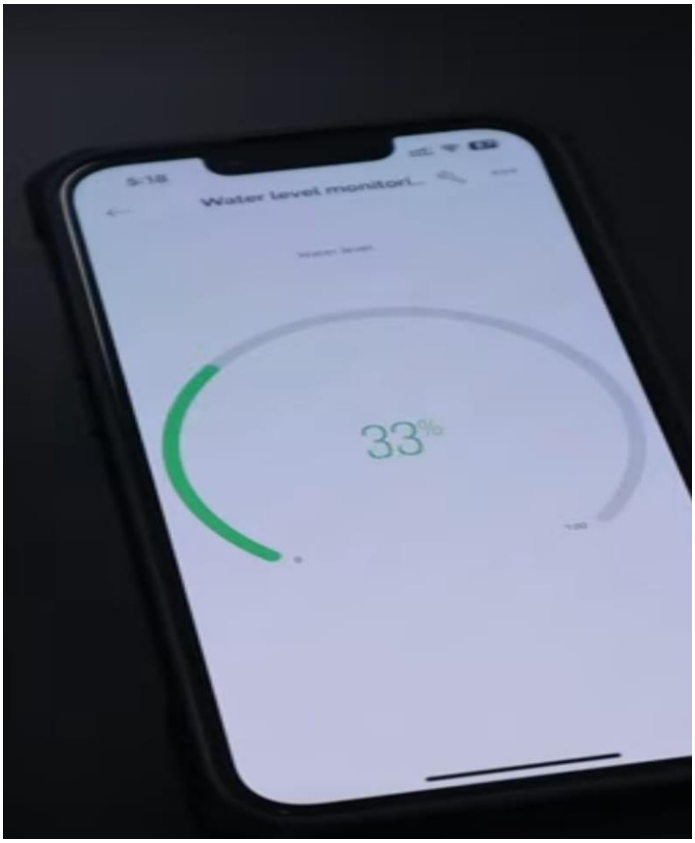




Mobile App UI:


13:50
Reports
Water Usage
Current Week
Last Week
Last Month
64%
40%
90%
Last 16 Hours
Events
Low Leak Detected
6:44 No Action


13:50
Water Management System
Sign In
Hi there! Nice to see you again.
Email
example@email.com
Password
Sign in
Forgot Password?
Sign Up


33%