GNANAMANI COLLEGE OF TECHNOLOGY(Pachal,Namakkal)

DEPARTMENT OF BIOMEDICAL ENGINEERING

(Third Year)

TITLE : SMART WATER MANAGEMENT

TEAM MEMBERS:

POOVARASI R(620821121082)

DEEPIKA N(620821121017)

NAVEENA D(620821121075)

SABEETHA P(620821121095)

POOVARASI R(620821121081)

BY

R.POOVARASI(620821121081)

SMART WATER MANAGEMENT

PROBLEM:

Inefficient Water Usage In A Garden

DESCRIPTION:

Imagine you have a garden with automated irrigation,but it waters your plants on a fixed schedule,regardless of weather conditions.This leads to water wastage during rainy days or over watering during periods of high humidity.

SOLUTION:

Create a Smart Watering System using IoT and Arduino.

COMPONENTS NEEDED:

1. Arduino Board(e.g.,Arduino Uno)
2. Soil Moisture Sensor
3. Water Pump
4. Relay Module
5. Rainfall Sensor(optional)
6. Wifi Module(e.g.,ESP8266)
7. Smart Phone or Computer for monitoring

STEPS TO IMPLEMENT:

SOIL MOISTURE MONITORING:

Connect the soil moisture sensor to the Arduino.Program the Arduino to read soil moisture levels periodically.Set a moisture threshold to determine when the plants need water.

WATER PUMP CONTROL:

Connect the water pump to a relay module.Program the Arduino to control the relay based on soil moisture readings.When the moisture level falls below the threshold,activate the pump to water the garden.

WEATHER DATA INTEGRATION(OPTIONAL):

Connect a range sensor to the Arduino to detect rainfall.integrate weather data from the internet using the wifi module.If it's raining or for caste predicts rain,suspend watering to avoid over watering .

REMOTE MONITORING AND CONTROL:

Set up wifi connectively and the Arduino to send data to the cloud .Create a web or mobile app to monitor soil moisture and control the system remotely.Receive alerts or notifications when the system waters the garden or when a issues arise.

BENEFITS:

WATER EFFICIENCY:

The system waters the garden only when neccesary,reducing water wastage.

REMOTE CONTROL:

Monitor and control the system from anywhere,ensuring optimal plant care.

WEATHER INTEGRATION:

Prevent over watering during rainy periods,saving water and money.

THIS SIMPLE PROJECT DEMONSTRATES HOW IOT AND ARDUINO CAN BE USED TO SOLVE A COMMON WATER MANAGEMENT PROBLEM BY MAKING IRRIGATION SMARTER AND EFFICIENT.

PHASE-2

1. **IoT Connectivity:**

   - Utilize IoT modules (such as ESP8266 or ESP32) with Arduino to connect the system to the internet.

   - Enable bidirectional communication, allowing the system to send data to the cloud and receive commands or updates.

2. **Soil Moisture Sensing:**

   - Implement soil moisture sensors in key locations to measure the moisture content of the soil.

   - Use capacitive soil moisture sensors for accurate readings, and calibrate them to specific soil types.

3. **Data Transmission:**

   - Establish a secure connection to an IoT platform (like ThingSpeak, Blynk, or AWS IoT) to transmit real-time data.

   - Ensure data encryption for privacy and security.

4. **Cloud-Based Analytics:**

   - Implement cloud-based analytics to process and analyze the collected data.

   - Utilize machine learning algorithms to predict future soil moisture levels based on historical data, weather forecasts, and other relevant parameters.

5. **Mobile Application:**

- Develop a user-friendly mobile app for farmers or users to monitor and control the system remotely.

- Include features such as real-time soil moisture levels, historical data graphs, and the ability to adjust irrigation settings.

6. **Automated Irrigation Control:**

- Implement an automated irrigation system that adjusts water flow based on real-time sensor data.

- Include features like scheduling, threshold alerts, and emergency shutdown in case of sensor malfunctions or extreme conditions.

7. **Energy Efficiency:**

- Design the system to be energy-efficient by using low-power components and optimizing the communication protocols.

8. **Scalability:**

- Ensure that the system is scalable, allowing users to expand the coverage area or add more sensors as needed.

9. **Weather Integration:**

- Integrate weather APIs to incorporate forecast data into the decision-making process.

- Adjust irrigation schedules based on upcoming weather conditions to avoid unnecessary watering during or after rainfall.

10. **Community and Data Sharing:**

- Allow for community-based data sharing where users can contribute anonymized data for broader analysis.

- Promote a collaborative approach to water management, especially in regions facing water scarcity.

PHASE-3

Development-1

1. Choose Components:

- Select an Arduino board compatible with IoT modules (e.g., Arduino Uno or Arduino MKR series).

- Choose a water level sensor appropriate for your application.

2. Connect Level Sensor to Arduino:

- Connect the level sensor to the Arduino using appropriate pins.

- Ensure proper power supply and ground connections.

3. Integrate IoT Module:

- Add an IoT module (e.g., ESP8266, ESP32) to enable wireless communication.

- Connect the IoT module to Arduino, considering serial communication or other interfaces.

4**. Write Arduino Code**:

- Develop Arduino code to read data from the water level sensor.

- Implement code for the IoT module to send this data to a cloud platform

5.Cloud platform itegration:

- Choose an IoT cloud platform (e.g., Thing Speak, Blynk, AWS IoT).

- Set up an account and create a project to receive and visualize data.

6. Data Transmission:

- Use relevant communication protocols (MQTT, HTTP) to send data from Arduino to the cloud.

- Implement security measures such as encryption if needed.

7. Rule-Based Automation:

- Set up rules on the cloud platform to trigger actions based on water level data.

- For example, automate alerts, notifications, or control devices like pumps.

8. Mobile App or Web Interface:

- Develop a user interface to monitor and control the system.

- Consider creating a mobile app or a web-based dashboard.

9. Power Management:

- Optimize power consumption for long-term operation, especially if the system is battery-powered.

- Implement sleep modes for the Arduino or IoT module when not actively transmitting data.

10. Testing and Calibration:

- Test the entire system under various scenarios to ensure reliability.

- Calibrate the water level sensor for accurate measurements.

11. Documentation:

- Document the system design, connections, and code for future reference or troubleshooting.

Remember to consider environmental conditions, scalability, and security throughout the development process. Regularly update and maintain the system to address any issues or improvements.

PHASE-4

DEVELOPMENT-2

Feature Engineering:

1. Data Collection: Gather data from IoT sensors connected to Arduino devices. This data may include water flow rates, water quality, temperature, humidity, and more.

2. Feature Selection: Identify relevant features from the collected data. Not all sensors or data points may be equally important for your specific application. Prioritize features that have a significant impact on water management.

3. Feature Extraction: Create new features or transform existing ones. For example, you can calculate daily averages, maximum values, or trends in the data to capture important patterns.

4. Normalization and Scaling: Ensure that all features are on the same scale, which is important for some machine learning algorithms. Normalize or scale data as needed.

5. Feature Engineering for Time Series: If your data is time-series data, consider lag features, rolling statistics, and other time-related transformations.

Model Evaluation:

1. Data Split: Divide your dataset into training, validation, and test sets. This is typically done to train, tune hyperparameters, and evaluate the model's performance.

2. Model Selection: Choose an appropriate machine learning algorithm for your specific problem. Common choices for time-series data include linear regression, decision trees, random forests, or more advanced methods like recurrent neural networks (RNNs) or Long Short-Term Memory (LSTM) networks.

3. Model Training: Train your selected model using the training dataset. Tune hyperparameters to optimize model performance. For example, you might adjust the learning rate, the depth of decision trees, or the number of LSTM units.

4. Validation: Use the validation dataset to assess the model's performance during training. This helps in avoiding overfitting and making necessary adjustments to the model.

5. Model Evaluation: Finally, evaluate the model's performance on the test dataset using appropriate metrics. Common metrics for regression tasks include Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared. For classification tasks, metrics like accuracy, precision, recall, and F1-score are relevant.

6. Cross-Validation: Depending on your dataset size, consider using k-fold cross-validation to ensure robust model performance assessment.

7. Model Interpretation: Understand how the model is making predictions. For example, for decision tree-based models, you can visualize feature importance.

8. Deployment: Once you're satisfied with your model's performance, deploy it to your IoT and Arduino setup for real-time or near-real-time water management.