MAHENDRA INSTITUTE OF ENGINEERING AND TECHNOLOGY

COLLEGE CODE ­– 6115

SMART WATER MANAGEMENT

INNOVATION

TEAM:PROJ\_223284\_TEAM\_1

TEAM ID :565

TEAM MEMBERS;

1.KANNAN.S

2.KATHIR KUMAR.S

3.KISHORE.S

4.KAMESH.V

5.KARUPPUSAMY.M

FACULTY MENTOR NAME :

SANTHANARAJ . M

**1.INTRODUCTION**

Water, our planet’s essential component, is needed for agriculture, manufacturing, electricity production, and sustaining human health. Approximately one billion people throughout the world do not have access to safe drinking water. Water transmission pipes lose 20 percent to 30 percent of the water that passes through them on a regular basis, with losses exceeding 50 percent in older systems, particularly those that have suffered from inadequate maintenance. Water loss in transmission pipelines can be caused by a variety of factors, including leaks, metering problems, public use, and theft. Water and energy loss, as well as significant property damage, are among the issues. Due to the lengthy time and high expense associated with leak detection in the present situation, not all water leaks can be identified; as a result, some leaks continue to occur, resulting in the concerns listed above. Water leak management can be improved if leaks can be detected quickly and repaired quickly. The large disparity between supply and demand for water resources is mostly caused by global population increase and urbanization. Furthermore, aging water infrastructure is putting a pressure on the world's water supply. Cities must change into Smart Cities as the population and urbanization grow. Water is an essential resource for human life, hence smart water management systems play an important role in smart cities. Smart water management technology can improve the effectiveness and stability of a water delivery system, cutting costs and enhancing durability.

1. **METHODS** 
   1. **Water Level Indication**

To measure the water quantity and identify the water level present in the water tank, this Water Level Indication system is implemented. The definition of the water level indicator - it is an indicator that tells you how much water is present in your tank.

**How does the water level indicator work?**

* + - To know the water level present in the tank, we divided the water level of the tank into three different levels: low level, medium level and high level. So, by doing this, our indicator will help us to know the water level present in the tank easily and according to that we can refill the water in the tank. We can also save water instead of overflowing.
    - We used a few LEDs, battery and some wire connection to build this water level indicator and inside the tank we put wires in different lengths for different levels which will show water level by sending a signal back to LEDs.

**Fig. 1**: Level-wise indication using LEDs

**The advantages of this indicator include:**

* + - The consumption of electricity will be less because the indicator helps to stop the unnecessary running of the motor after the tank gets filled. ● Overflow of water is stopped/avoided.
    - The cost of implementation is less, so we can save money.
    - The design planning of the indicator is simple.

**Components You'll Need:**

* Arduino board (e.g., Arduino Uno)
* Sensors (e.g., water level sensor, water quality sensor)
* Actuators (e.g., water pump, solenoid valve)
* Display (e.g., LCD or OLED)
* Power source
* Relay modules (if needed for controlling pumps or valves)
* Wi-Fi or GSM module for remote monitoring (optional)

**Programming Steps:**

1. **Setup Arduino:** Ensure you have the Arduino IDE installed on your computer. Set up your Arduino board, connect the sensors and actuators, and install any necessary libraries for your components.
2. **Define Pin Configuration:** Define the pins to which you have connected the various sensors and actuators. This is done in the **setup()** function.

const int waterLevelPin = A0; // Analog pin for water level sensor

const int waterQualityPin = A1; // Analog pin for water quality sensor

const int pumpPin = 7; // Digital pin for water pump

const int valvePin = 8; // Digital pin for solenoid valve

**Initialize Sensors and Actuators:** In the **setup()** function, initialize the sensors, actuators, and other components.

void setup() {

pinMode(pumpPin, OUTPUT);

pinMode(valvePin, OUTPUT);

// Initialize any libraries or modules here

}

**Read Sensor Data:** In the **loop()** function, continuously read sensor data and store it in variables.

void loop() {

int waterLevel = analogRead(waterLevelPin);

int waterQuality = analogRead(waterQualityPin);

// You may need to convert sensor values to meaningful data

// Check thresholds, calibrate, and filter data as necessary

}

void loop() {

int waterLevel = analogRead(waterLevelPin);

int waterQuality = analogRead(waterQualityPin);

// You may need to convert sensor values to meaningful data

// Check thresholds, calibrate, and filter data as necessary

}

**Make Decisions:** Use conditional statements to make decisions based on the sensor data. For example, if the water level is too low, turn on the pump. If water quality is poor, send a notification.

if (waterLevel < threshold) {

digitalWrite(pumpPin, HIGH); // Turn on the water pump

} else {

digitalWrite(pumpPin, LOW); // Turn off the water pump

}

if (waterQuality > poorQualityThreshold) {

// Send a notification or activate the valve to redirect the water

}

1. **Display Data (Optional):** You can use a display to show real-time data.
2. **Remote Monitoring (Optional):** If you want to monitor and control your system remotely, you can integrate Wi-Fi or GSM modules and create a user interface or connect to an IoT platform.
3. **Error Handling and Logging (Optional):** Implement error handling and logging to track system performance and detect anomalies.
4. **Power Management:** Optimize power consumption if the system runs on batteries.
5. **Testing:** Test your system thoroughly, ensuring it responds correctly to changes in water level and quality.

Remember to adjust the code and components to match your specific water management needs and the complexity of your project. Additionally, consider safety measures and regulations when working with water systems.

const int waterLevelPin = A0; // Analog pin for water level sensor

const int waterQualityPin = A1; // Analog pin for water quality sensor

const int pumpPin = 7; // Digital pin for water pump

const int valvePin = 8; // Digital pin for solenoid valve

void setup() {

pinMode(pumpPin, OUTPUT);

pinMode(valvePin, OUTPUT);

// Initialize any libraries or modules here

}

void loop() {

int waterLevel = analogRead(waterLevelPin);

int waterQuality = analogRead(waterQualityPin);

if (waterLevel < threshold) {

digitalWrite(pumpPin, HIGH); // Turn on the water pump

} else {

digitalWrite(pumpPin, LOW); // Turn off the water pump

}

if (waterQuality > poorQualityThreshold) {

// Send a notification or activate the valve to redirect the water

}

}

1. **Variable Declarations:**
   * **waterLevelPin** and **waterQualityPin** are defined to specify the analog pins where you've connected the water level sensor and water quality sensor, respectively.
   * **pumpPin** and **valvePin** are defined to specify the digital pins where you've connected the water pump and solenoid valve, respectively.
2. **setup() Function:**
   * In the **setup()** function, you set up the pins for the water pump and valve as OUTPUT pins using **pinMode()**. This prepares the pins for controlling the actuators.
   * This is also where you can initialize any libraries or modules you might be using for additional functionality.
3. **loop() Function:**
   * The **loop()** function is the main execution loop for your Arduino. It runs continuously.
4. **Sensor Reading:**
   * Inside the **loop()** function, you read data from the water level and water quality sensors using the **analogRead()** function. This function reads the analog voltage on the specified pins and converts it to a digital value.
5. **Control Logic:**
   * The code uses conditional statements to make decisions based on the sensor data:
     + If the water level is below a certain threshold (you should define **threshold** earlier in your code), it turns on the water pump by setting the **pumpPin** to HIGH. If the water level is above the threshold, it turns off the pump by setting the **pumpPin** to LOW.
     + If the water quality is poor (above a certain threshold defined as **poorQualityThreshold**), you can send a notification or activate the solenoid valve to redirect the water to a different location.
6. **Additional Features (Optional):**
   * If you have additional components, like a display or communication modules, you would incorporate them into the code to display information or send notifications as required.
7. **Looping:**
   * The code continuously loops through the **loop()** function, constantly monitoring and controlling the water system based on the sensor data.

THANK YOU

[View publication stats](https://www.researchgate.net/publication/360269606)