Smart Water Management Phase-3

Kanimozhi.S

au420321104011

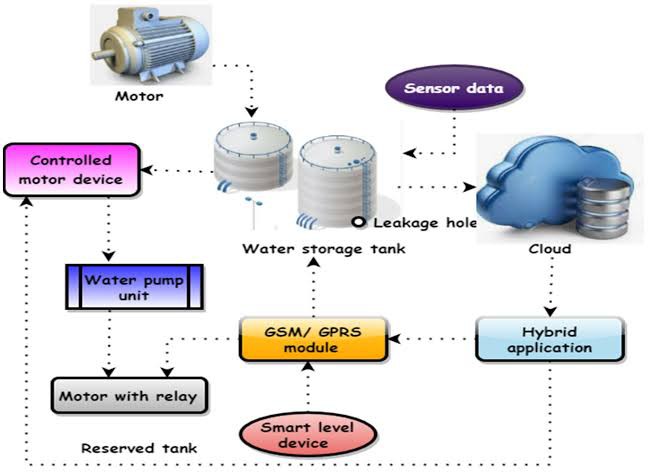
guhansubramanian981@gmail.com

Development Part 1

1. Introduction :

IoT-enabled water management systems use sensors, controllers, meters, and other devices connected to mobile, web apps, and data processing and analysis tools. All this creates a platform for efficient water supply management, freshwater quality checking, pollution detection, and more.

# Block DIAGRAM :



## Defining Project Requirements:

Begin by clearly defining the objectives of your Smart Water Management project. What specific data are you looking to collect and analyze? What problems are you trying to solve? Understanding your project's goals is essential.

## IoT Device Deployment:

Choose the appropriate IoT devices for your project.

These devices should be capable of collecting, processing, and transmitting data to a central server or cloud platform.

## Data Analysis and Visualization:

Set up a data analysis platform that can receive data from your IoT devices and perform relevant analyses. Tools like Python libraries (e.g., Pandas, Matplotlib, Seaborn) can be used for data analysis and visualization.

## Remote Monitoring and Alerts:

Implement remote monitoring and alerting systems to notify relevant stakeholders when specific water parameters go beyond predefined thresholds. This can be done through emails, SMS, or mobile apps.

# Program:

import serial import time import requests import random

# Define your ThingSpeak API key and channel URL api\_key = "G1RSE98QHVUFT626"

url = f"https://api.thingspeak.com/update?api\_key={api\_key}"

# Initialize the serial connection to Arduino

arduino\_port = '/dev/ttyACM0' # Update with your Arduino's port ser = serial.Serial(arduino\_port, 9600)

# Function to read data from Arduino def read\_arduino\_data():

data = ser.readline().decode().strip()

return data

# Simulated sensor data (replace with actual sensor readings) def read\_sensor\_data():

temperature = random.uniform(20.0, 30.0)

water\_level = random.uniform(0, 100) return temperature, water\_level

while True: try:

# Read data from Arduino arduino\_data = read\_arduino\_data()

# Read sensor data

temperature, water\_level = read\_sensor\_data()

# Prepare data to send to ThingSpeak data = {

'field1': temperature,

'field2': water\_level, 'field3': arduino\_data

}

# Send data to ThingSpeak

response = requests.post(url, data=data) print("Data Sent to ThingSpeak")

except Exception as e: print(f"Error: {e}")

# Set the data upload interval

time.sleep(300) # Upload data every 5 minutes (adjust as needed)