SMART WATER MANAGEMENT

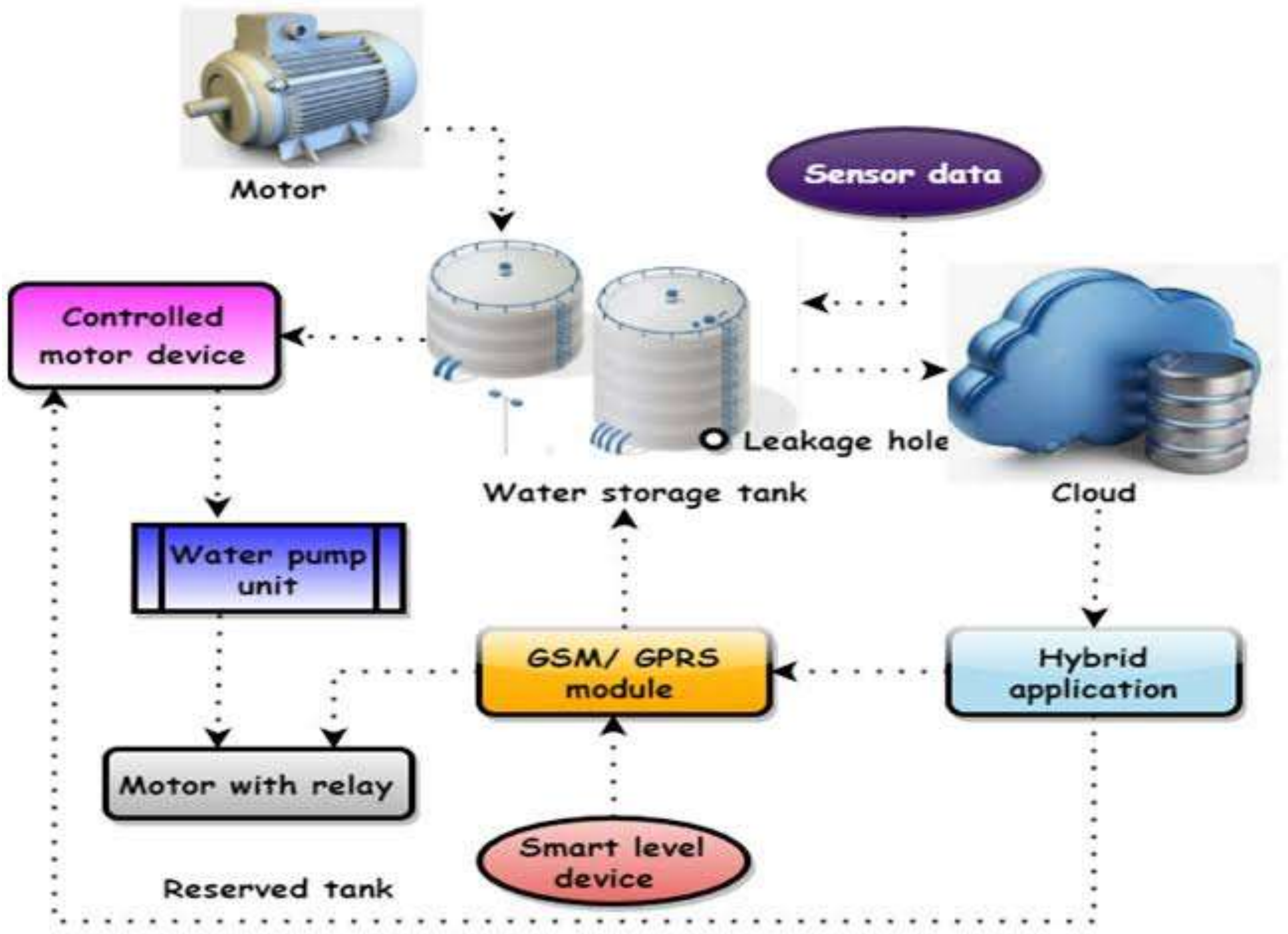
USING IOT

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)
```