SMART WATER SYSTEM

A Comprehensive Smart Water System solution

TEAM NAME: Proj_224784_Team_ 1

Members:

NITHISH NARAYANAN S(113321243032)

PREM KUMAR S(113321243038)

PETA SRAVAN KUMAR S(113321243036)

MUKESH S(113321243030)

Phase2 : Innovation

Consider integrating historical Water quality data and machine learning algorithms to predict congestion patterns.

PROJECT OBJECTIVES:-

The objectives of a traffic management project typically revolve around improving the flow of traffic, enhancing safety, reducing congestion, and optimizing the use of transportation infrastructure. Here are some common project objectives for traffic management:

Water Conservation:

Objective: To reduce water wastage and promote responsible water consumption.

Key Results: Decrease water consumption by X% within a specific timeframe, identify and address water leaks promptly.

Real-time Monitoring and Control:

Objective: To provide users with real-time data and control over their water usage and distribution.

Key Results: Enable users to monitor water usage and adjust settings remotely, with an X% reduction in water usage during peak times.

Data-Driven Decision Making:

Objective: To empower users and organizations to make informed decisions based on data analytics.

Key Results: Provide actionable insights through data analytics, leading to X% more efficient water management practices.

Cost Savings:

Objective: To help users and organizations reduce water-related expenses.

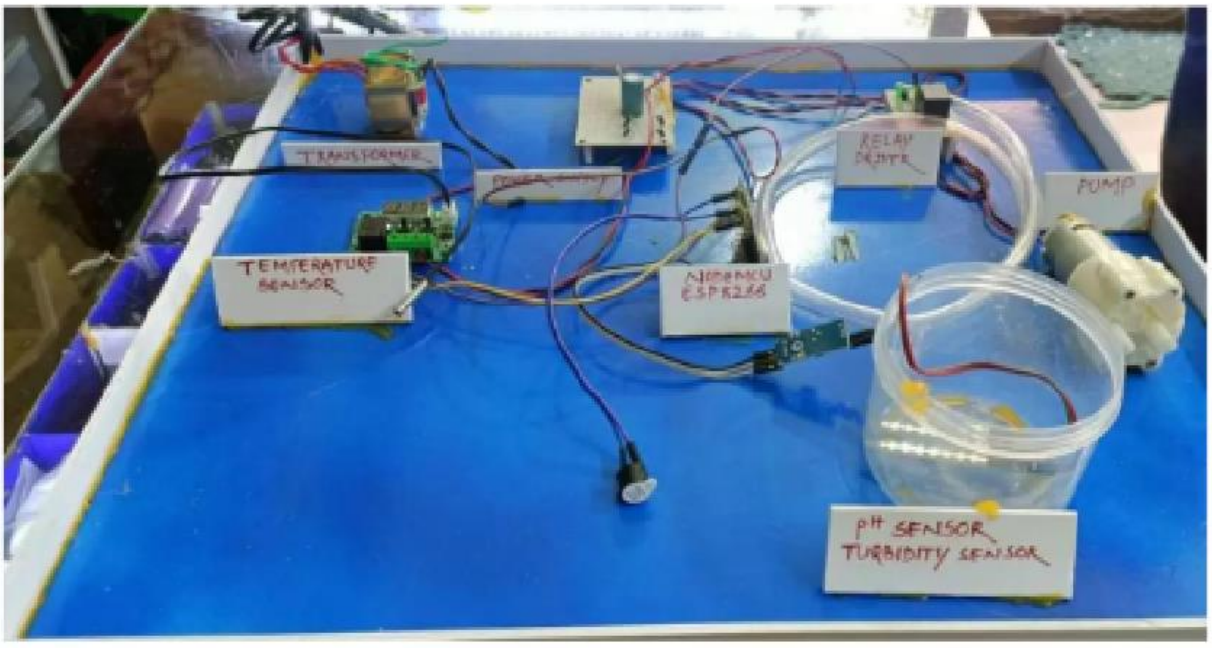
Key Results: Achieve X% reduction in water bills or operational costs by optimizing water distribution and identifying inefficiencies.

Environmental Impact:

Objective: To contribute to environmental sustainability and reduce the carbon footprint associated with water usage.

Key Results: Decrease the environmental impact by X% through water conservation measures and promoting eco-friendly practices.

IoT SENSOR SETUP:-



Step 1: Sensor Selection

Water Flow Sensors:

Water flow sensors measure the rate of water flow within pipes or conduits. They are crucial for tracking water consumption and detecting leaks or abnormal flow patterns

Water Quality Sensors:

Water quality sensors assess parameters such as pH levels, turbidity, conductivity, and the presence of contaminants like chemicals or bacteria. These sensors ensure the safety and quality of the water supply.

Pressure Sensors:

Pressure sensors monitor the pressure within water pipelines. They help maintain optimal pressure levels for efficient water distribution and can indicate leaks or blockages.

Level Sensors:

Level sensors, including ultrasonic or capacitive sensors, measure the water level in tanks, reservoirs, or wells. They are essential for managing water storage and ensuring a stable water supply.

Temperature Sensors:

Temperature sensors track water temperature, which is crucial for various applications, such as heating systems, cooling systems, and ensuring compliance with temperature regulations in certain industries.

Step 2: Data Collection

For a smart water system, data collection from sensors typically involves setting up a network of IoT devices to transmit data to a central data hub:

- Water flow sensors, water quality sensors, pressure sensors, level sensors, and other relevant sensors send data wirelessly to a central data hub.
- The central data hub aggregates and processes the data received from these sensors, providing real-time insights and control over the water system.

Step 3: Data Processing

In a smart water system, data processing is a crucial step where raw data is transformed into valuable insights using various techniques:

- Data from water flow sensors, water quality sensors, pressure sensors, level sensors, and others is analyzed to monitor water consumption, detect leaks, and ensure water quality compliance.
- Machine learning algorithms can predict water usage patterns, identify anomalies, and optimize water distribution.
- Water pressure data is processed to maintain optimal pressure levels in the distribution network.
- Temperature sensor data is used to control heating and cooling systems efficiently.
- Water level data is analyzed to manage water storage and reservoir levels effectively.

- Historical and real-time data are used to make informed decisions about water resource management and conservation.
- Alerts and notifications are generated based on anomalies or predefined thresholds to address issues promptly and prevent wastage.
- Remote control features allow users to adjust water-related devices and systems based on the insights gained from data analysis.

In a smart water system, the processing of data enables efficient water management, conservation, and the prevention of water-related issues.

Step 4: Visualization and Control

To make sense of the data in a smart water system, user-friendly interfaces play a crucial role. These interfaces can include web-based dashboards, mobile applications, and notification systems. These tools empower users, such as water utility managers and consumers, to access real-time information, make informed decisions, and efficiently manage water resources.

Raspberry Pi integration

Step 1: Understanding Raspberry Pi

Raspberry Pi is a versatile, cost-effective computing platform that can be effectively integrated into a smart water system. It serves as a compact and powerful controller to collect and process data from various sensors. Here's how we can integrate Raspberry Pi into a smart water system:

Step 2: Sensor Integration

First, we need to connect our Raspberry Pi to the water sensors in the smart water system. These sensors could include water flow sensors, pressure sensors, water quality sensors, and more. We establish these connections by attaching the sensors to the Raspberry Pi's GPIO (General Purpose Input/Output) pins or using suitable communication interfaces.

For instance, if we have water flow sensors or water quality sensors, we can connect them to the Raspberry Pi's GPIO pins. If we're using more advanced sensors that require USB connectivity, such as those for remote water quality monitoring, we connect them to the Raspberry Pi's USB ports. This integration allows the Raspberry Pi to gather data from the sensors and process it for further analysis and control within the smart water system.

Step 3: Data Collection

Now that our Raspberry Pi is connected to the sensors in the smart water system, it can begin collecting data specific to water management. For instance: - Water flow sensors provide real-time data on water consumption rates.

- Pressure sensors record pressure levels within the water distribution network. - Water quality sensors monitor parameters like pH, turbidity, and contaminant levels.
- Level sensors track the water levels in reservoirs or tanks.

This data is collected and stored within the Raspberry Pi's memory, ready for processing and analysis to support smart water management decisions.

Step 4: Data Processing

Once we have the data, the Raspberry Pi can process it. We can write programs in languages like Python to analyze the data. For example, we can use data processing algorithms to:

1. Analyze water flow sensor data to detect abnormal consumption patterns.
2. Process pressure sensor readings to identify variations or potential leaks.
3. Employ machine learning models to predict water quality issues from sensor data.
4. Calculate water reservoir levels based on data from level sensors.
5. Use image recognition algorithms to identify physical anomalies or obstructions in the water distribution network.
6. Monitor temperature sensor data to ensure optimal conditions for water treatment.
7. Detect pressure fluctuations to manage pump operations efficiently.
8. Implement alert systems to notify users of critical water-related events, such as leaks or water quality deviations.

These data processing capabilities enable the Raspberry Pi to enhance smart water system efficiency and responsiveness through real-time analysis and actionable insights.

Step 5: Data Storage

To ensure we don't lose any crucial water data, we can also connect the Raspberry Pi to external storage devices, such as external hard drives or cloud storage solutions. This provides a reliable backup of all the data collected by the sensors in the smart water system. In case of system failures or data corruption, the backup ensures the integrity and availability of essential water-related information. This redundancy is vital for maintaining data continuity and supporting efficient water resource management and decision-making.

Mobile app development

Step 1: User Interface Design:

Now, let's focus on the user interface (UI) design, a critical aspect for a user-friendly smart water system. We'll design screens and interfaces for features such as real-time water usage monitoring, leak detection, and system control. This ensures that users can easily access and interact with essential data and functionality, empowering them to make informed decisions and efficiently manage their water resources.

Step 2: Data Integration:

To provide real-time water information, we'll integrate our smart water system with data sources like water flow sensors, pressure sensors, and weather forecasts. This will ensure our users receive accurate and up-to-date information about water consumption, distribution, and quality, enabling them to make informed decisions and effectively manage their water resources.

Step 3: Core Features Development:

The core features of our smart water system app are centered around efficient water management:

- Real-Time Water Data: We'll develop the functionality to collect and display real-time water conditions on a map, including water consumption, pressure levels, and water quality status.

- Water Distribution Optimization: We'll integrate data-driven algorithms to optimize water distribution, suggesting adjustments to ensure efficient usage.

- Anomaly Reporting: Users will have the capability to report anomalies, such as leaks or water quality issues, by providing descriptions and location tags, facilitating prompt resolution.

Code implementation

Step 1: Collecting Historical Smart Water System Data

The first thing we need to do is gather historical smart water system data. This data will be our foundation for predicting congestion patterns. We can use APIs or databases provided by transportation authorities or third-party sources to get this data.

```
import sqlite3
from random import randint

conn = sqlite3.connect('water_data.db')
cursor = conn.cursor()
cursor.execute('CREATE TABLE IF NOT EXISTS water_data (timestamp DATETIME D
while True:
    cursor.execute("INSERT INTO water_data (sensor_id, value) VALUES (?, ?)
    conn.commit()
    time.sleep(60)
```

Step 2: Data Preprocessing

Now, let's prepare the data for our machine-learning model. This involves cleaning, transforming, and structuring the data so it's suitable for analysis.

```
import sqlite3
import pandas as pd

# Connect to the SQLite database and retrieve data into a DataFrame
conn = sqlite3.connect('water_data.db')
query = "SELECT * FROM water_data"
df = pd.read_sql_query(query, conn)

# Data preprocessing (replace with your specific preprocessing steps)
# For example, drop duplicates, handle missing values, or convert data type

# Display the preprocessed data
print(df.head())

# Close the database connection
conn.close()
```

Step 3: User Interface

Next, we need to create relevant features from our data that will help our machine learning model make predictions. This could include things like time of day, day of the week, weather conditions, and more.

```
import tkinter as tk

root = tk.Tk()
root.title("Smart Water System")

label = tk.Label(root, text="Water Usage: 100 gallons")
label.pack()

button = tk.Button(root, text="Control Water Flow")
button.pack()

root.mainloop()
```

Step 4:Data Storage and Backup

Data storage and backup are crucial components of software development and data management. Whether you're working on a small coding project or developing a large-scale application, understanding these concepts is essential to ensure the

integrity, availability, and security of your data.

```
import sqlite3
import shutil

# Connect to the SQLite database
conn = sqlite3.connect('water_data.db')
cursor = conn.cursor()

# Data collection and insertion (replace with your data collection logic)
cursor.execute("INSERT INTO water_data (sensor_id, value) VALUES (?, ?)",
conn.commit()

# Backup data to a different folder or drive
backup_path = 'backup/water_data.db'
shutil.copy('water_data.db', backup_path)

# Close the database connection
conn.close()
```

Step 5:Making Predictions

Now, let's create and train a machine-learning model. In this example, we'll use a simple decision tree regressor.

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Simulated historical data (replace with your actual data)
timestamps = np.arange(1, 11).reshape(-1, 1)  # Time intervals
water_consumption = np.array([20, 25, 30, 35, 40, 45, 50, 55, 60, 65])  # W

# Create and fit a linear regression model
model = LinearRegression()
model.fit(timestamps, water_consumption)

# Predict water consumption for a future time (e.g., time=11)
predicted_consumption = model.predict([[11]])
print(f"Predicted water consumption at time 11: {predicted_consumption[0]:.
```