Smart Water Management Using IoT: Real-World

Examples :

It’s easier to understand climate change when you experience challenges

yourself. Consider unpredictable water supply, worsening water scarcity, and

water pollution. Whether you’re an agricultural firm or city administration, all

these affect you. And if you’re looking to become more efficient and boost

your green credentials, you might not know where to turn.

This is where smart water management using IoT could make all the

difference.

Among the many benefits of IoT technology, it helps keep water quality high.

Smart sensors can provide peace of mind that equipment like pumps and

pipelines is highly functional, and with IoT services, you can dispose of

wastewater safely and in line with regulations.

Let's take a closer look at the most common real-world examples that illustrate

the impressive advantages of smart water management systems.

Smart City Water Management:

City administrators need to keep a close eye on water supply, consumption,

and equipment. With IoT, the whole water supply chain can become more

transparent and easier to control.

With the help of sensors, a smart city water management system can enable

you to collect real-time data—information that helps you visualize water

distribution across the network. Residents with smart meters can make more

informed decisions as a result, leading to a more sustainable city overall.

Water waste and disrupted water supply chains are a drain on the city’s

budget. IoT can help you watch the health of water equipment and detect

problems, like leaks in pipes. This allows operators to receive alerts and start

fixing issues immediately. In the meantime, AI predictions allow you to nip

problems in the bud by preventing failures before they cause severe incidents.

With AI, city administrators can also watch the watershed and predict which

areas are likely to flood, information that will help local authorities warn

residents, manage traffic, and keep the city on its feet.

Real-World Example: Smart Irrigation of City Parks

Cartagena, a city in Columbia, has smart irrigation in its municipal parks and

gardens. The solution calculates the amount of water each area needs

depending on the state of the soil, weather forecast, and irrigation calendar. If

something goes wrong, such as a leak, the authorities are alerted right away

and they’re even shown the location.

Main benefits

• Better transparency in water management

• Fewer incidents

• Enhanced control over the water supply

• Saved city budget

• Improved city sustainability

Water Quality Management System

Watching the quality of water that comes into our houses is crucial. Rivers,

lakes, and reservoirs may contain contaminants that are dangerous to us, and

the increasing world population combined with urbanization has also

worsened water quality. In our changing world, IoT can help monitor and

analyze distributed water and ensure it complies with regulatory standards.

A water quality management system using IoT can deal with quality issues

effectively. You only need to consider a simple comparison to appreciate the

difference: Without IoT, water samples need to be collected and analyzed

manually. This process is costly and time-consuming because it requires large

equipment and an expensive workforce. In contrast, IoT sensors can measure

a variety of parameters like temperature and turbidity. Operators receive

regular data from multiple samples, enabling them to remotely perform quality

control on water reserves.

Real-Life Example: Watching the Quality of River Water

A solution from Ericsson and AT&T monitors water quality for the city of

Atlanta, Georgia, where four million citizens get drinking water from the

Chattahoochee River. IoT helps authorities check the quality of water, while

sensors measure its conductivity, turbidity, temperature, and thermometry.

Main benefits

• Increased water quality

• Saved budget on manual analysis of water samples

• Smaller workforce involved

• Remote quality control

• Compliance with regulatory requirements

Looking for industrial IoT experts?

Learn more about our IIoT solutions

Water Level Monitoring and Dam Management:

Dams bring water to livestock and irrigation and supply many industries. They

also play a pivotal role in flood control and can assist river navigation, so it’s

crucial that dams and reservoirs function properly and their water levels are

safe. The trouble is that traditional monitoring methods are time-consuming

and complex.

Water level monitoring and management of dams using IoT can improve this,

using ultrasonic, vibration, and pressure sensors to help monitor dam function.

With pressure sensors, in particular, you can detect leaks in pipes and receive

instant alerts. Predictive technologies ensure dam operators get early

warnings and are able to keep watch over water availability in each reservoir.

This may be particularly helpful for irrigation.

A smart solution can also give you remote control over the movement of

gates, so there’s no need to send staff to the site in severe weather conditions

like floods or storms. If the water reaches a certain level, the system can

decide to open or close the gate.

Real-Life Example: Smart Dam Monitoring

A ThingsLog level monitoring solution helps dam owners in Bulgaria to

manage more than 100 dams in the region. IoT sensors remotely watch water

levels at each dam site. The system sends instant alerts if flooding is possible.

With smart capabilities, there’s no need to send staff to measure water levels

on site. The system even has pre-programmed formulas that replace manual

calculations.

Main benefits

• Real-time water level monitoring

• Better dam functionality

• Enhanced dam reliability

• Faster decision-making

• Saved time and resources

• No human involvement

Smart Water Management for Agriculture

The world population has exploded in recent decades, and more people mean

more food. But that’s not the only thing that’s new. Food consumption patterns

have also changed, leading to increased global crop production that requires

savvy water usage. Water scarcity can negatively affect yields as much as

water oversupply. IoT is helping to make this process more efficient and smart

than ever before.

A smart water management system for agriculture using IoT can improve crop

fields providing farmers with the oversight they need to avoid water waste.

Sensors monitor multiple parameters, like temperature, humidity, and soil

moisture to calculate how much water crops need. These sensors are

connected to the field and sprinkler in sprinkler irrigation systems, and farmers

receive regular updates on their smartphones.

With AI on board, you can plan agricultural activities wisely and in advance.

Crop water management using IoT helps farmers use less water to grow the

same amount of crops. The technology also enables them to use less

fertilizer, save energy on pumping less water, and save time and money on

labor. IoT solutions also make it possible to see the water level in tanks.

Real-Life Example: Smart Irrigation Management:

The Galileo System from Galcon aids and optimizes irrigation in open

farmlands and greenhouses. An open-field version has about 200 irrigation

programs. Farmers can watch up to 50 main irrigation lines, change flow

intensity, and schedule their activities. What’s more, the software shows a

realistic picture of the watered field.

Main benefits

• Boosted productivity of agricultural activities

• Better quality of products

• Prevented water waste

• More efficient use of water, fertilizer, and energy

• Automated activities

• Optimized farm labor

Perspectives:

With such expansive and varied use cases, the potential of smart water

technologies is clear to see. We also have optimistic estimates on the global

smart water management market value which is developing rapidly.

Some of the key factors driving this market include new laws and regulations

on reduced water consumption that aim to meet sustainability objectives.

Water-related standards, in particular, ensure local authorities and water

suppliers provide safe and high-quality water. For example, the Safe Drinking

Water Act sets the standards for drinking water quality in the US. IoT

technology will help companies comply with this and many other regulations

while achieving sustainability goals.

Afterword:

IoT smart water management brings transparency and optimized control to

the whole water supply chain, helping industries and cities use healthy water

efficiently and follow regulations. With IoT capabilities, you can even collect

and recycle wastewater.

If you need a trusted consultant to assist you in optimizing water production,

distribution, and consumption, Softeq is here to help. We have unrivaled

expertise in hardware design, software, and embedded system developments,

all under one roof.

Smart water management in IoT (Internet of Things) using Python involves monitoring and controlling water-related systems efficiently. Here’s a high-level overview of how you can approach this project:

**1. Hardware Setup:**

• Use IoT devices like water level sensors, flow meters, and actuators to monitor and control water systems.

• Ensure these devices are connected to a microcontroller or single-board computer like Raspberry Pi.

2. **Data** **Collection**:

• Use Python libraries like `Adafruit\_IO`, `MQTT`, or `Blynk` to collect data from sensors.

• Read sensor data such as water levels, flow rates, and temperature.

3. **Data** **Processing**:

• Use Python to process and analyze the collected data. You can use libraries like `pandas` and `numpy`.

• Implement algorithms to detect anomalies or trends in water usage.

4. **Data** **Storage**:

• Store data in a database (e.g., SQLite, MySQL, or InfluxDB) to keep historical records.

• Use libraries like SQLAlchemy or InfluxDB-Python to interact with the database.

5. **Real**-**time** **Monitoring** **and** **Alerts**:

• Implement real-time monitoring of water systems.

• Send alerts and notifications through email, SMS, or push notifications using Python libraries like `smtplib` or `Twilio`.

6. **Visualization**:

• Create a web-based or mobile app to visualize water usage data. Use frameworks like Flask or Django for web apps.

• Use Python libraries like Plotly, Matplotlib, or Seaborn for data visualization.

7. **Control**:

• Implement control mechanisms to actuate pumps, valves, or other devices based on sensor data.

• Ensure that you have fail-safe mechanisms to prevent water wastage or system malfunctions.

8. **Energy** **Efficiency**:

• Optimize energy consumption of IoT devices by scheduling sensor readings and controlling actuators based on energy-efficient algorithms.

9. **Security**:

• Ensure that your IoT system is secure. Use encryption and authentication methods.

• Regularly update and patch your system to protect against vulnerabilities.

10. **Scaling**:

• Plan for scalability in case you need to monitor and manage water systems across a larger area.

• Consider using cloud-based services for scalability.

11. **Machine** **Learning** (**Optional**):

• If you have a large dataset, you can apply machine learning models to predict water usage patterns and optimize resource allocation.

12. **Documentation** **and** **Testing**:

• Keep thorough documentation of your project for maintenance and future improvements.

• Test the system rigorously to ensure it works as expected.

Remember to adapt these steps according to your specific use case and the types of water systems you are managing. Also, consider the power source for your IoT devices, as they might need to run continuously.

# In every route which requires login , just put if logged\_in:

# Logout button that send request to /logout

from flask import Flask, render\_template, url\_for

from flask import jsonify, request

from flask import flash, redirect,abort

import datetime as d

app = Flask(\_\_name\_\_)

lastOffTime = d.datetime.now()

netlitres = 0

prevtank1 = 10

diff = 0

x = 0

logged\_in = False

power =0

tank1data = 40

tank2data = 40

current =  "OFF"

option = ""

ontime = 0

stat1 = ""

stat2 = ""

stat3 = ""

'''@app.route('/',  methods=['GET'])

def index():

    global tank1\_level

    return render\_template('index\_gauge.html')  '''

@app.route('/')

def home():

    if logged\_in:

        return render\_template('home.html')

    else:

        return redirect('/login')

    lastOffTime = d.datetime.now()

@app.route('/login', methods=['GET'])

def login\_page():

    return render\_template('login.html')

@app.route('/logout', methods=['GET'])

def logout():

    global logged\_in

    logged\_in = False

    return render\_template('login.html')

@app.route('/login', methods=['POST'])

def check\_login():

    global logged\_in

    if request.form['password'] == 'password' and request.form['username'] == 'admin':

        logged\_in = True

        return redirect('/')

    else:

        return render\_template('login.html')

@app.route('/water', methods=['GET'])

def water():

    if logged\_in:

        return render\_template('index\_gauge.html')

    else:

        return redirect('/login')

@app.route('/energy', methods=['GET'])

def energy():

    if logged\_in:

        return render\_template('energy.html')

    else:

        return redirect('/login')

'''@app.route('/power/<int:p>', methods=['GET'])

def power():

    global power

    power = p

    return 'ok' '''

@app.route('/deptho/<int:depth\_cm1>', methods=['GET'])

def show\_post1(depth\_cm1):

    global tank1data

    global netlitres

    global prevtank1

    global diff

    if depth\_cm1 < prevtank1:

        diff = (prevtank1 - depth\_cm1)

    tank1data = depth\_cm1

    prevtank1 = depth\_cm1

    netlitres = netlitres + diff

    return 'ok'

@app.route('/power/<int:p>', methods=['GET'])

def power(p):

    global power

    power = p

    return 'ok'

@app.route('/change/<string:switch>', methods=['GET','POST'])

def change(switch):

    global option

    option = switch

    return option

@app.route('/stat/<int:p>', methods=['GET'])

def status(p):

    global ontime

    global tank1data

    global tank2data

    global current

    global option

    global stat1,stat2,stat3

    global power

    power = p

    s1 =""

    s2 =""

    s3 = ""

    if tank1data<20 and tank2data>20 and option == "auto":

        s1 = "xyz"

        stat1="a"

        current = "On"

    elif option == "on":

        s2 = "abc"

        stat2 = "a"

    elif option == "off":

        s2 = ""

        stat2 = ""

    elif option == "a2on":

        s3 = "pqr"

        stat3 = "a"

    elif option == "a2off":

        s3 = ""

        stat3 = ""

    elif option == "a1on":

        s1 = "xyz"

        stat1 = "a"

        current = "On"

    elif option == "a1off":

        s1 = ""

        stat1  = ""

        current = "Off"

    pfinal = s1 + s2 + s3

    if stat1 =="a":

        pfinal = pfinal + "xyz"

    if stat2 == "a":

        pfinal = pfinal + "abc"

    if stat3 == "a":

        pfinal = pfinal + "pqr"

    final = pfinal + "$"

    return final

@app.route('/depths/<int:depth\_cm2>', methods=['GET'])

def show\_post2(depth\_cm2):

    global tank2data

    tank2data = depth\_cm2

    return 'ok'

@app.route('/return\_global', methods=['GET'])

def return\_global():

    global tank1data

    global tank2data

    global current

    global ontime

    global lastOffTime

    global netlitres

    global x

    global power

    x = x+ 1

    if current == "OFF":

        lastOffTime = d.datetime.now()

    elif current == "On":

        if  0==0:

            f= 0

            diff = d.datetime.now() - lastOffTime

            lastOffTime = d.datetime.now()

            f = diff.microseconds

            ontime = ontime + f

    return jsonify(tank1 = tank1data , tank2 = tank2data, stat = current, time = ontime/1000000, net= netlitres)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host='0.0.0.0', port=8080, debug=True)

IoT- Smart Water Management Systems:

1. Introduction of Smart Water Management Systems:

One of the essential elements in the universe is water. Nowadays, consumers continuously seek

methods to simplify their lives. Monitoring water quality is critical to ensuring the planet’s health and longterm viability. Water is the source of many infectious illnesses, and garbage thrown by residents and

environmental disasters from industrial enterprises pollute most of the nearby freshwater supplies in SA.

Drinking water can be stored in an overhead tank. The principal causes of water quality deterioration in

residential buildings are the development of microbes in overhead tanks and distribution networks,

corrosion of pipe material, and the non-replacement of existing pipes. To avoid catastrophic health

implications, it is necessary to continuously and remotely check the quality parameters of the water system

in real-time.

The main contributions of this paper are:

\*Smart water management gives a greater understanding of the water system, including flaw detection,

preservation, and water management.

\*A comprehensive database of regions with water losses or unlawful connections can be built with the

introduction of smart water system technology by public service corporations.

\*Smart water grids can save costs by conserving water and energy while improving the quality of

service to consumers. Wireless data transfer allows consumers to assess their water use to reduce

water costs in other circumstances.

2. Relevant Survey:

Thermal characteristics have significantly impacted the environmental footprint and life-cycle

evaluation of buildings. It is proposed in this study that the opacity of a façade element could be adjusted

year-round in response to seasonal variations using the smart water-filled glass (SWFG) control technology

described in this research. As a result of SWFG and WFG’s climate-based design strategy, glass buildings

are no longer seen as climate change liabilities and instead as opportunities for sustainability .According to

a new study, the Internet of Things (IoT) could monitor renters’ and landlords’ electricity and water use (IoT).

One technique has made it simpler to read meters and water meters through the internet, and renters and

landlords can see the data gathered through an Android application.Tenants and landlords have found the

gadget and IoT-based Android app dependable, accurate, useful, and easy to use.

3. Proposed Method:

In Saudi Arabia, water supply and sanitation are marked by problems and successes. Water shortage

is a major issue. Water desalination, water distribution, sewerage, and wastewater treatment have received

significant investments to combat water shortages .Today, desalination provides roughly half of the

country’s drinking water, groundwater mining provides 40%, and surface water supplies in the mountainous

southwest account for 10% .A desalinated water pipeline runs from the Arabian Gulf to Riyadh, Saudi

Arabia’s capital and largest city. Consumers get water nearly at no cost. Despite these advances, the quality

of service remains low, for instance, in terms of supply continuity. The public sector in Saudi Arabia is

characterized by a widespread lack of institutional capability and governance. The Saudi vision 2030

addresses this challenge to be resolved.

4. Numerical Outcome:

Internet of Things-based smart water systems can assist prevent these scenarios from occurring and

repair the harm that has already been done due to the careless use of water resources. From a freshwater

reservoir to the collection and recycling of wastewater, smart water technology makes the entire water

supply chain more transparent and controllable. This section includes applications for water management,

IoT devices for water management, and smart water treatment methods. Finally, a list of several vital

qualities of these systems is framed based on the survey, all of which should be integrated into future

designs. As shown in Figure .modern IoT-based water quality monitoring solutions have been thoroughly

surveyed. All articles focused on meeting the minimum WHO/USEPA standards for potable water, which

need systems to be low-cost, portable, low-power, Internet-enabled, real-time, and compliant with these

standards.

Figure :Water quality monitoring analysis.

5. Conclusions:

The water sector has been grappling with creating an efficient and long-lasting water system. It is

included in the IoT-SWM. People intend to broadcast more data to the cloud and analyze it further to

construct some algorithm to determine the tank’s lifespan and the proper aspects of leaking. Procedures

and actions are determined depending on the threshold, capital cost, and the accessibility of equipment

and materials. Even though statistically minimal water savings can be achieved using in-line flow restrictors,

they can be much more cost-effective than water-efficient taps in certain situations. If they have been

installed as part of normal maintenance visits, the expenses would be lower. They are a low-cost alternative

to outdated toilets and are unlikely to save a lot of water. When it the time comes to renovate restrooms,

installing water-saving toilets should be considered.

Coding:

In every route which requires login , just put if logged\_in:

# Logout button that send request to /logout

from flask import Flask, render\_template, url\_for

from flask import jsonify, request

from flask import flash, redirect,abort

import datetime as d

app = Flask(\_\_name\_\_)

lastOffTime = d.datetime.now()

netlitres = 0

prevtank1 = 10

diff = 0

x = 0

logged\_in = False

power =0

tank1data = 40

tank2data = 40

current = "OFF"

option = ""

ontime = 0

stat1 = ""

stat2 = ""

stat3 = ""

'''@app.route('/', methods=['GET'])

def index():

global tank1\_level

return render\_template('index\_gauge.html') '''

@app.route('/')

def home():

if logged\_in:

return render\_template('home.html')

else:

return redirect('/login')

lastOffTime = d.datetime.now()

@app.route('/login', methods=['GET'])

def login\_page():

return render\_template('login.html')

@app.route('/logout', methods=['GET'])

def logout():

global logged\_in

logged\_in = False

return render\_template('login.html')

@app.route('/login', methods=['POST'])

def check\_login():

global logged\_in

if request.form['password'] == 'password' and request.form['username'] == 'admin':

logged\_in = True

return redirect('/')

else:

return render\_template('login.html')

@app.route('/water', methods=['GET'])

def water():

if logged\_in:

return render\_template('index\_gauge.html')

else:

return redirect('/login')

@app.route('/energy', methods=['GET'])

def energy():

if logged\_in:

return render\_template('energy.html')

else:

return redirect('/login')

'''@app.route('/power/<int:p>', methods=['GET'])

def power():

global power

power = p

return 'ok' '''

@app.route('/deptho/<int:depth\_cm1>', methods=['GET'])

def show\_post1(depth\_cm1):

global tank1data

global netlitres

global prevtank1

global diff

if depth\_cm1 < prevtank1:

diff = (prevtank1 - depth\_cm1)

tank1data = depth\_cm1

prevtank1 = depth\_cm1

netlitres = netlitres + diff

return 'ok'

@app.route('/power/<int:p>', methods=['GET'])

def power(p):

global power

power = p

return 'ok'

@app.route('/change/<string:switch>', methods=['GET','POST'])

def change(switch):

global option

option = switch

return option

@app.route('/stat/<int:p>', methods=['GET'])

def status(p):

global ontime

global tank1data

global tank2data

global current

global option

global stat1,stat2,stat3

global power

power = p

s1 =""

s2 =""

s3 = ""

if tank1data<20 and tank2data>20 and option == "auto":

s1 = "xyz"

stat1="a"

current = "On"

elif option == "on":

s2 = "abc"

stat2 = "a"

elif option == "off":

s2 = ""

stat2 = ""

elif option == "a2on":

s3 = "pqr"

stat3 = "a"

elif option == "a2off":

s3 = ""

stat3 = ""

elif option == "a1on":

s1 = "xyz"

stat1 = "a"

current = "On"

elif option == "a1off":

s1 = ""

stat1 = ""

current = "Off"

pfinal = s1 + s2 + s3

if stat1 =="a":

pfinal = pfinal + "xyz"

if stat2 == "a":

pfinal = pfinal + "abc"

if stat3 == "a":

pfinal = pfinal + "pqr"

final = pfinal + "$"

return final

@app.route('/depths/<int:depth\_cm2>', methods=['GET'])

def show\_post2(depth\_cm2):

global tank2data

tank2data = depth\_cm2

return 'ok'

@app.route('/return\_global', methods=['GET'])

def return\_global():

global tank1data

global tank2data

global current

global ontime

global lastOffTime

global netlitres

global x

global power

x = x+ 1

if current == "OFF":

lastOffTime = d.datetime.now()

elif current == "On":

if 0==0:

f= 0

diff = d.datetime.now() - lastOffTime

lastOffTime = d.datetime.now()

f = diff.microseconds

ontime = ontime + f

return jsonify(tank1 = tank1data , tank2 = tank2data, stat = current, time = ontime/1000000, net= netlitres)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=8080, debug=True)