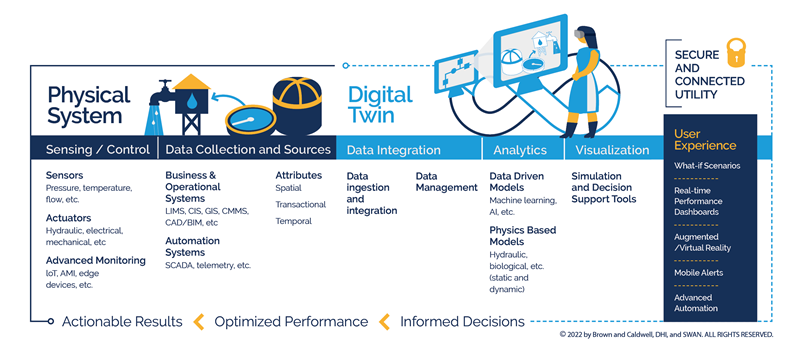
**Progress Report :  
(references mentioned in ppt)**  
DIGITAL TWINS UNDERSTANDING

## ARCHITECTURE:

SWAN Digital Twin Architecture diagram gives a framework for categorizing a digital twin’s main elements. This architecture framework can function as a checklist and lets you see the components and gaps you have to complete your digital twin.  
  
if you want to decide which infrastructural investments to make in your utility for the next five years, you don’t need a live update of sensor data and events. You need aggregated information of which are the critical areas in your infrastructure. However, if you want to make operational decisions like whether to turn off and on a pump and for how long, you want all the live and forecast data you can get your hands on, and you might even want to know what the possible consequences are. These decisions require different dynamic data and different granularity of the digital twin.



## EVALUATE MY DIGITAL TWIN:

## 4 MAIN PILLARS OF DIGITAL TWIN

 We identified four main pillars of a digital twin that can help guide your way to evaluate your digital twin and how to proceed further. These pillars are – Outcomes, Technology and Connectivity, Insights, and Interactions and Actions. 

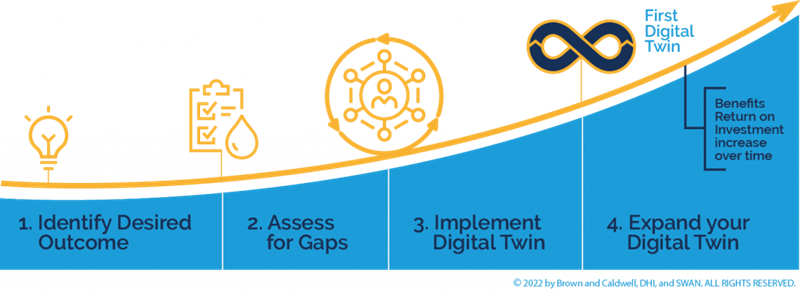
1. Outcomes pillar:  Some examples are improved regulatory compliance, lower-cost operations, and a more reliable and resilient system. These outcomes should be as quantifiable as possible so that they can be used to track digital twins’ performance and plan further development.

2. Tech and connectivity: a utility with high connectivity and high technology will be able to provide a digital twin with more real-time data, making the digital twin more reflective of the actual operational conditions. It will also enable a digital twin to provide more intelligent decision support and advanced automation.

3.Insights: measure of information produced by models in the digital twin. They can be generated by analysis of simulations from data-driven or artificial intelligence (AI), physics-based models, or a combination of them. Insights give you an idea of what is happening, what has happened and what will happen in your system. Some analysis examples are anomaly detection, what-if scenarios, predictive operational parameters

physics-based models (e.g., hydraulics models) excel in simulating expected design operating conditions in a utility, while a data-driven model simulates patterns in actual operating conditions  
4. Interactions and Actions:  these actions can be for staff, including operators, engineers and managers. This support can be suggested setpoints or suggested work orders to create.

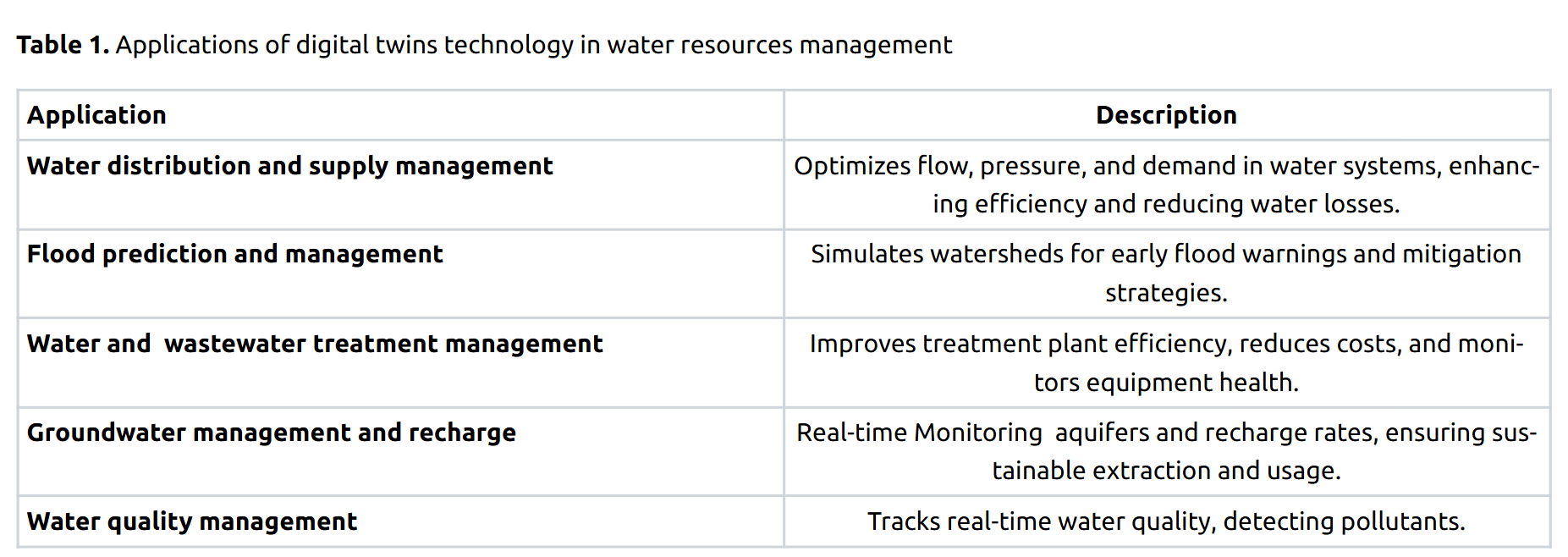
 Digital Twins can supplement programmable logic controllers (PLC) and provide intelligent control automation or a co-pilot to the operations team. This operational co-pilot enables operators to control devices like pumps and valves more dynamically and in more discrete time steps under their direct supervision.  
  
Getting started with DT

**The digital twin journey, a scalable and flexible approach to planning, designing and implementing digital twins.**  


**Identify – decide what you want**: We recommend starting small, looking for low-hanging fruits, and making the goal as actionable as possible. This will be the guiding star of the direction of the digital twin.

**Assess – what you have in your utility**: get an inventory of technology, initiatives and projects that can help you achieve the goal you have identified because it provides a framework that covers a high-level overview of the different components of a digital twin.

**Implement –** **an iterative approach and involve different stakeholders :** Set some success criteria and comm. with others to understand the goal and metrics to measure. Review the metrics and iterate the implementation.

###this approach will work for all digital twins in the water sector.  
****

**1. Water Distribution and Supply Management**

**Water distribution systems are vital for providing water for different users and sectors, especially in populated cities with high demand.**

* Helping the operators to select the optimal decisions in real-time by simulating the effect on any operation prior to taking the action in the real system.
* Employ energy-saving strategies, such as using a fewer number of pumps when demand falls below a dynamically chosen limit.( switch one motor on for taking water to a single tank then other if more consumption of water per hour only then, tanks connected to each other, keep a primary tank and a primary source of water to the tank, use mainly that unless required)
* Identifying anomalies to establish an enhanced maintenance system, providing diminishing  maintenance costs and downtime to limit disruptions to end-users.
* Optimizing the operation of the system in order to improve the quality of the service and the water quality.
* Developing emergency response plans and modeling the behavior of the system under emergency conditions for detecting an early warning system against possible contamination into the network.

Pipes and pumps are intricate systems with complex internal mechanisms and exact control requirements, necessitating careful safety measures. They play a crucial role in maintaining the flow, pressure, and control of fluids in various processe. They can also become one of the major sources of water losses, through leakage.

Digital twins technology, in conjunction with machine learning (ML) and the internet of things (IoT),  can control and reduce the water losses, including  flow management, water, and energy monitoring, along with water grid control, to work together to boost the efficiency of water distribution systems.  Using a detection and communication system, water losses can be reduced through intelligent supervision of sensors, telemetry, and actuators that control water pressure and flow at critical network points

The integration of a remote-control platform, powered by big data analytics, allows water-energy network managers to optimize system performance through real-time control and data-driven decisions, progressively improving the overall efficiency of the network.

\

gathers data from sensors in pipes, pumps, and valves, facilitating simulations that predict water flow, pressure, and quality. Early detection of leaks and blockages can be facilitated by this technology, enhancing repair response times and reducing water loss. It supports predictive maintenance, allowing the Public Utilities Board (PUB) to preempt equipment failures, thus minimizing disruption

By analyzing weather data and consumption patterns, it adjusts to demand fluctuations. Additionally, it helps in developing smart water grids, ensuring better resource allocation

2. Groundwater :  
Digital twins for groundwater monitoring can monitor real time data  from sensors distributed across the aquifers, aiding in decision making by identifying areas that require automated water pump drainage and sending signals through the internet to activate the water pumps based on the groundwater table  value, maintaining the table value levels under the surface. The model also includes features such as historical data visualisation based on previous sensor readings and a predictive machine learning model to predict ground water level based on historical precipitation data.

3.Water quality management:  
digital twins that can monitor, predict, and communicate water quality dynamics by combining contaminant fate and transport models, online sensor data assimilation, and real-time visualization and response capabilities. In the context of watershed-scale water quality, digital twins provide innovative features for real-time response, enabling real-time monitoring, prediction, and management of water quality hazards such as algal blooms, chemical spills, and combined sewer overflows

Water management using Digital Twins Research paper:  
this paper systematically discusses the concept and development history of digital twin smart water conservancy, compares its differences with traditional water conservancy models, and further proposes the digital twin smart water conservancy five-dimensional model

The research progress of digital twin smart water conservancy is summarized by focusing on six aspects:

1.digital twin water conservancy data perception, transmission

2.data analysis and processing

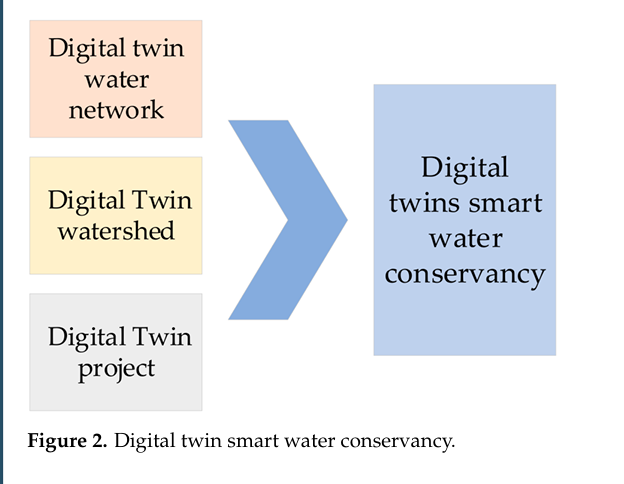
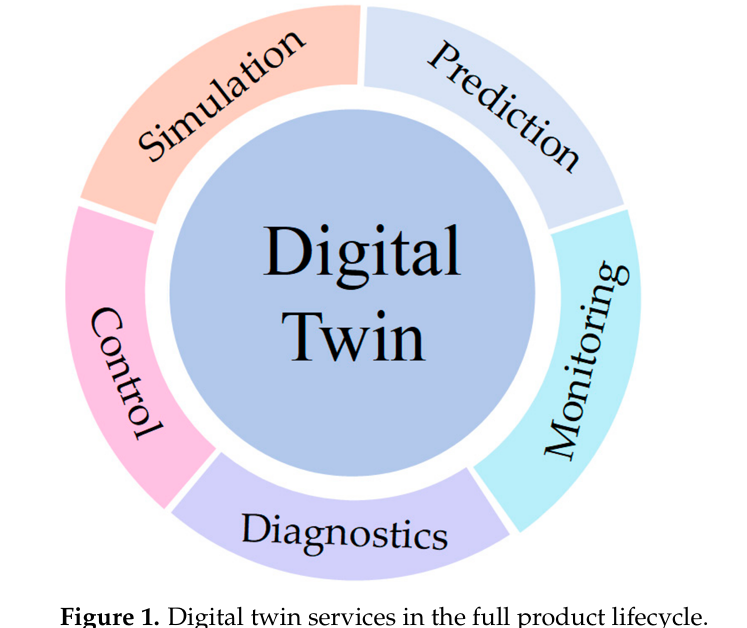
3.digital twin water conservancy model construction

4. digital twin water conservancy interaction and collaboration

5.digital twin water conservancy service application

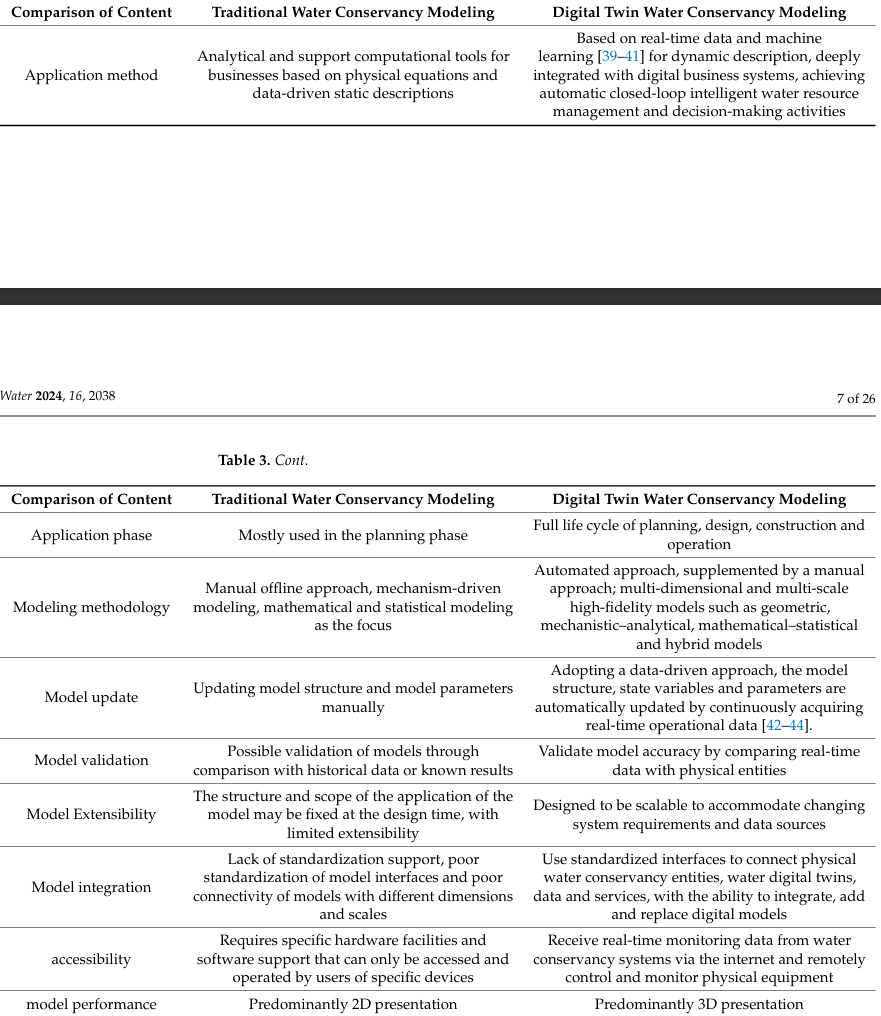
6.the challenges and problems of digital twin technology in the application of smart water conservancy

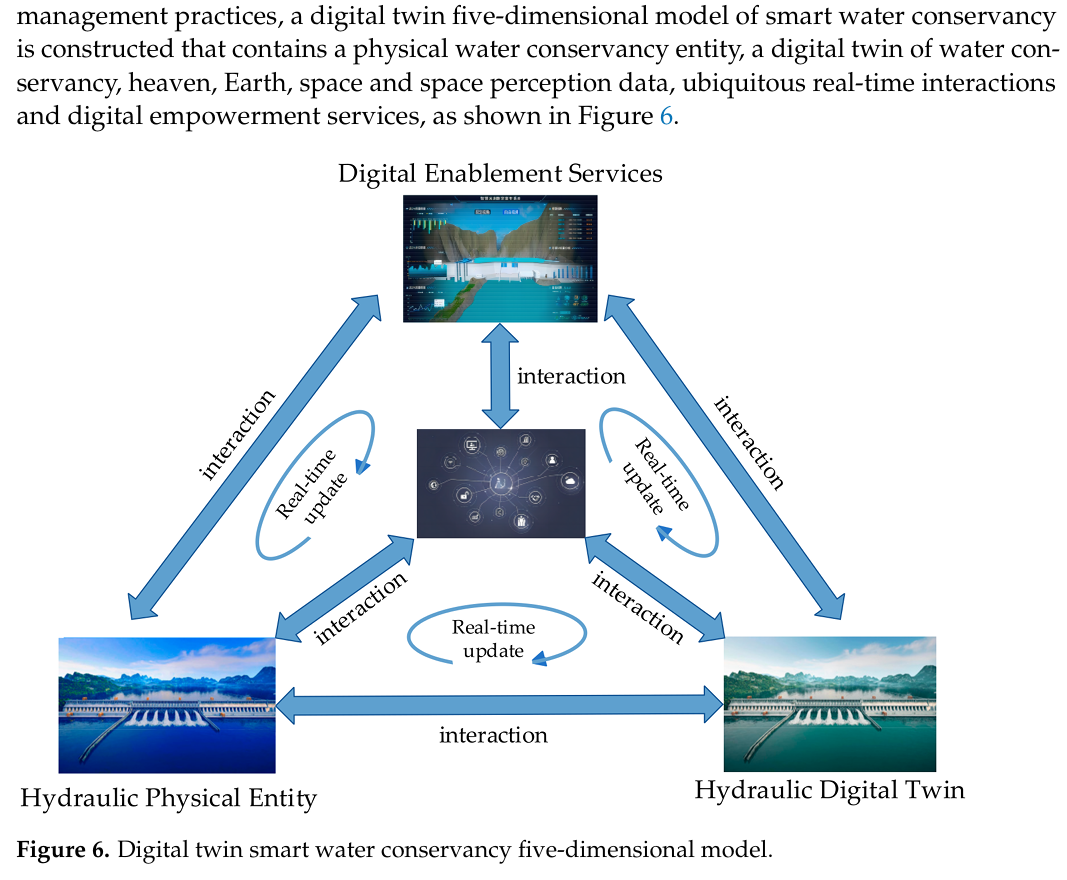
digital twin is oriented to the whole life cycle process of the product, which is a kind of technology that makes full use of the model, data and intelligence and integrates multiple disciplines to play the role of a bridge and link connecting the physical world and the information world to provide more real-time, efficient, and intelligent services

proposes the digital twin smart water conservancy five-dimensional model, which contains five dimensions of water conservancy physical entities, water conservancy perception data, ubiquitous real-time interaction, and digital empowerment services. Based on this model, this study reviews the research progress of digital twin smart water conservancy in six dimensions: water conservancy data perception, data transmission, data analysis and pro cessing, digital twin water conservancy model construction, digital twin water conservancy interaction and collaboration and service application 

Start of the paper\*:

Ministry of Water Resources has proposed a digital twin water conservancy project, digital twin water network and top-level design, so that the digital twin watershed, digital twin water network and digital twin water conservancy project come together to form a digital twin smart water conservancy series—the three factors are physical watersheds, physical water networks, and physical water conservancy projects in the digital space of mapping; the three relationships determine the relationships between the three physical entities; and the three physical entities are inter-related. The relationship between the three is determined by the inter-relationship of the three physical entities, which are not alternative to each other, have their own focus, are relatively independent, are interconnected, and conduct information sharing ?????





 **Water Conservancy Physical Entity**:  
Real-time data from physical water conservancy infrastructures (topography, equipment, construction impact areas) are collected using integrated air, sky, and ground monitoring systems. These include IoT sensors and devices feeding data into digital twin systems.

 **Water Conservancy Digital Twin**:  
A comprehensive virtual model of the water conservancy system is built using geometric, analytical, statistical, behavioral, and visualization models to represent the physical entity dynamically across multiple dimensions and time scales.

 **Heaven, Earth, and Space Perception Data**:  
Sensor-collected data, essential for digital twin operations, may have errors due to environmental influences and transmission loss. Thus, preprocessing and data fusion techniques are required for accuracy and integration.

 **Ubiquitous Real-Time Interaction**:  
A networked infrastructure (including IoT, internet, control systems, and protocols) enables real-time, secure, and efficient data exchange between physical systems and their digital counterparts for collaborative and optimized control.

 **Digital Empowerment Services**:   
Application layer of the digital twin system serving various stakeholders (authorities, operators, public) with solutions in water resource management, flood control, project monitoring, and risk early warning.  
  
created a ppt of my understanding of digital twins for water management

# Digital Twin Implementation Summary: Campus Water Consumption

## 1. Overview of Sample Data

- The uploaded files represent water consumption logs from different campus blocks: A1MD, A1FF, A1FD, A2MFF, BTTF.

- Each record contains:   
 • Date/Time (timestamp of reading)  
 • Totalizer (cumulative water usage in liters)  
 • Consumption (incremental water used in liters)

- This data is critical for real-time monitoring and can be directly used to build a digital twin system.

## 2. Mapping the Data onto a Digital Twin Framework

A Digital Twin of the campus water system involves the following layers:

1. A. Physical Entity (Campus Infrastructure):

* - Buildings and zones equipped with water meters.  
  - Each meter represents a physical node in the water network.

1. B. Perception Layer (Sensors):

* - Flow meters log total and incremental water usage.  
  - Data is timestamped for chronological tracking.

1. C. Data Layer (Integration and Storage):

* - Stream data into a central server or cloud.  
  - Use time-series databases (e.g., InfluxDB, TimescaleDB).  
  - Apply ETL (Extract-Transform-Load) for data cleaning and fusion.

1. D. Virtual Entity Layer (Simulation Models):

* - Construct digital models of buildings using BIM or GIS.  
  - Associate sensor points with model nodes for real-time updates.

1. E. Interaction Layer (Control & Visualization):

* - 3D dashboard showing live consumption across campus.  
  - Visual alerts for leak detection, overuse, or anomalies.

1. F. Services Layer (Applications):

* - Predictive maintenance, water budget planning, sustainability tracking.  
  - Emergency response simulations and planning support.

## 3. Recommendations

- Ensure all key buildings have connected meters.  
- Centralize data acquisition with timestamp alignment.  
- Implement machine learning models for consumption forecasting and leak detection.  
- Use a modular approach to scale the digital twin block-by-block.

**1. Physical Layer: Instrument the Infrastructure**

* **What to do**:
  + Map out all water-consuming zones (e.g. A1MD, A1FF, Labs, Hostels, Admin).
  + Ensure **flow meters** are installed and working at key junctions/pipelines in each zone.
* **Goal**: Each building/block becomes a “node” in the water network.

**2. Perception Layer: Sensor Data Collection**

* **Use**:
  + Sensors you already have (as shown in your CSVs) logging Totalizer and Consumption.
* **Enhancements**:
  + Add **pressure sensors** or **quality sensors** if needed.
  + Ensure consistent **timestamped data** (uniform intervals).

**3. Data Layer: Centralized Storage & Processing**

* **Set up**:
  + A **Time-Series Database (TSDB)**: InfluxDB, TimescaleDB, or Firebase Realtime DB.
  + A lightweight **ETL pipeline** that:
    - Cleans the data (removes duplicates, aligns timestamps).
    - Merges data from all buildings into a unified dataset.
* **Optional**: Use edge devices for preliminary filtering before sending to the cloud.

**4. Virtual Layer: Create the Digital Twin Model**

* **Tools**:
  + Use **BIM (Building Information Modeling)** or **GIS** tools to create a virtual map of your campus.
  + Tools like Unity/Unreal or open-source platforms like CesiumJS can be used for 3D dashboards.
* **Mapping**:
  + Link each sensor to its digital counterpart (e.g., sensor in A1MD → virtual node A1MD).
  + Incorporate water flows, connections, and tanks/reservoirs if any.

**5. Interaction Layer: Control & Real-Time Visualization**

* **SCADA or IoT platform**:
  + Integrate control capabilities like turning off a valve or adjusting pressure if overuse is detected.
* **Dashboard**:
  + Use platforms like Grafana, Power BI, or a custom web interface to:
    - Visualize flow by zone.
    - Trigger alerts for unusual behavior (e.g., sudden spike overnight).
    - View historical consumption patterns.

**6. Services Layer: Intelligence & Decision Support**

* **Analytics Applications**:
  + **Leak Detection**: Sudden consumption spike without occupancy.
  + **Forecasting**: Use machine learning to predict tomorrow’s usage per block.
  + **Scenario Planning**: Simulate water demands during fests, droughts, or fire drills.
* **Sustainability Tracking**:
  + Show % water reused, compare consumption to sustainability goals.

**Tools You Can Use**

| **Layer** | **Suggested Tools** |
| --- | --- |
| Sensor Input | Arduino + flow sensors, LoRaWAN |
| Data Storage | InfluxDB, Firebase, PostgreSQL |
| Modeling (3D) | SketchUp + GIS, BIM, CesiumJS |
| Visualization | Grafana, Power BI, Unity Dashboards |
| Control | Node-RED, MQTT Broker |
| AI/Analytics | Python + Scikit-learn, TensorFlow |

**Minimal Viable System (MVP)**

To get started:

1. Choose 2-3 buildings (e.g., A1MD, A1FF, A2MFF).
2. Set up a database and a simple Python script to stream sensor data.
3. Create a basic Grafana dashboard to show real-time consumption.
4. Add alert logic: if Consumption > 20 L/hr during unoccupied hours → alert.
5. Expand the system building-by-building.

Workflow:

[ IoT Sensors ] → [ External Vendor ] → [ Daily CSV Logs ]

↓

[ Data Ingestion Script ]

↓

[ Time-Series Database (PostgreSQL/InfluxDB) ]

↓

[ Analysis Layer (Python, ML, Leak Rules) ]

↓

[ Visualization (Grafana/Streamlit/Dashboard) ]

↓

[ Alerts & Forecasts + Control Hooks (future) ]

**Water sensors right side :**

Installed here :

|  |  |  |
| --- | --- | --- |
| Block B - 1st Floor – Female Toilet – Domestic | B1FD | Aryabhatta Block B DCU2 |
| Block B - 1st Floor – Female Toilet – Flush | B1FF | Aryabhatta Block B DCU2 |
| Block B - 1st Floor – Male Toilet – Domestic | B1MD | Aryabhatta Block B DCU2 |
| Block B - 1st Floor – Male Toilet – Flush | B1MF | Aryabhatta Block B DCU2 |
| Block B – 2nd Floor – Male&Female Toilet – Domestic | B2MFD | Aryabhatta Block B DCU2 |
| Block B – 2nd Floor – Male&Female Toilet – Flush | B2MFF | Aryabhatta Block B DCU2 |
| Block B – Ground Floor – Female Toilet – Domestic | BGFD | Aryabhatta Block B DCU2 |
| Block B – Ground Floor – Female Toilet – Flush | BGFF | Aryabhatta Block B DCU2 |
| Block B – Ground Floor – Male Toilet – Domestic | BGMD | Aryabhatta Block B DCU2 |
| Block B – Ground Floor – Male Toilet – Flush | BGMF | Aryabhatta Block B DCU2 |
| Block B – Terrace – Tank – Domestic | BTTD | Aryabhatta Block Roof Top DCU3 |
| Block B – Terrace – Tank – Flush | BTTF | Aryabhatta Block Roof Top DCU3 |

CSV logs to a **completely real-time pipeline**, here’s a clear step-by-step roadmap for how to set this up using the **local server/DCU → cloud → dashboard** architecture.  
[Water Meter Sensors]

↓

[Local Data Concentrator Unit (DCU)]

↓

[Local Server/API Broker]

↓ (e.g. MQTT / REST API)

[Cloud Database / Storage]

↓

[Real-Time Streamlit Dashboard]

**System Architecture**

* **24 water meters** deployed:
  + Domestic lines
  + Flush lines
* Connected to **3 Data Concentrator Units (DCUs)**:
  + Block A DCU
  + Block B DCU
  + Terrace DCU
* DCUs forward packets to a **local server**.
* From there:
  + Data transferred via **SCP/SSH** to my laptop every 1-2 hours.
  + I store them as CSV logs for further processing.

*(→ shown in my deployment and architecture diagrams in PPT slides 3, 4, etc.)*

**Data Pipeline Implementation**

**Scripts I developed:**

**import\_data.py**

* Reads daily logs from local folders
* Merges them into a single master CSV:

CopyEdit

combined\_water\_data.csv

* Output has:

sql

CopyEdit

Meter ID, Consumption (Liters), Date/Time, Building

**Packet Conversion Script**

* Wrote several versions until finally working:
  + Reads JSON-style packet logs.
  + Converts them to rows in combined\_water\_data.csv.
* Learned a ton about JSON parsing, regex, and CSV writing.

*(This part took many tries because the JSON strings inside the CSV were messy and required fixing.)*

**Dashboard Development**

I made several versions:

**(1) Basic Dashboard**

* Streamlit dashboard.
* Shows:
  + Line chart of daily consumption.
  + Leak alerts (night leaks, spikes).
  + Demand forecast table.
  + Simulated valve control buttons.

**(2) Image Overlay Dashboard**

* Super cool overlay:
  + Floorplan image.
  + Draw sensor readings directly onto image.
* Added click events:
  + Click on a meter → see pop-up graph.
* Initially done with Pillow and Streamlit images.
* Moved to Plotly for clickable dots.

**(3) Plotly Interactive Dashboard**

* Sensors plotted as scatter points over the layout.
* When clicked:
  + Pops up a time-series graph.
  + Shows valve toggle buttons.
* Integrated:
  + Admin login screen.
  + Alerts with different colors based on severity.

**Leak Detection Logic**

Initially wrote simple rules:

* **Night Leak Detection**:
  + If water flow > 100 L between 0 AM – 5 AM.
* **Spike Detection**:
  + Rolling average for each building.
  + Flag if current hour > 3 × rolling average.

Then improved it with **ML**:

* Implemented:
  + Isolation Forest.
  + Used it for detecting outliers in hourly consumption.
* Eventually replaced some parts of threshold-based logic.

→ Generates:

CopyEdit

night\_leak\_alerts.csv

spike\_alerts.csv

**Demand Forecasting**

I tried several models:

* Linear Regression
* Then switched to **Prophet** for seasonality:
  + Weekends have lower consumption.
  + Holidays drop sharply.
* For future improvements:
  + LSTM models for timeseries.

Generates:

CopyEdit

demand\_forecast.csv

**Cloud Logging**

* Integrated Google Sheets for:
  + Logging valve events.
  + Storing leak alerts for admin review.

Implemented using:

* Google APIs
* gspread library

**Simulation Modules**

**Leakage Simulation**

* Inject synthetic data for:
  + Small leaks at night.
  + Gradual leaks.
* Test if dashboard detects them.

**Clog Simulation**

* Sudden or partial drop in consumption.
* Should trigger anomaly alert.

**Attack Detection Simulation**

* Simulated:
  + Spoofing packets.
  + Replay attacks.
  + Fake high usage.

Built **test data generator**:

* Outputs fake packets in same JSON-in-CSV format.

**Valve Control Design**

* Drafted mechanism diagrams in Draw.io.
* Planning:
  + Control valves via Raspberry Pi.
  + Use GPIO relay boards.
* Simulated in dashboard:
  + Valve toggle buttons for each sensor.

Next steps:

* Implement HTTP or MQTT to send commands from dashboard to Pi.

**Graphs & Analysis**

* Generated:
  + Plots of hourly/daily consumption.
  + Spike detection graphs.
  + Forecast charts for next 3 days.
* Added Plotly hover tooltips for clarity.

**Things I Learned**

* How to parse messy JSON strings from CSV.
* How to overlay sensor readings on images.
* Plotly vs Pillow vs Streamlit charts.
* SSH/SCP data transfers.
* ML models for time-series prediction.
* Google Sheets API integration.
* Debugging Tensorflow and Torch models!
* Designing valve circuits for water systems.

**Next Steps**

* Fully automate SCP transfer to run every hour.
* Migrate from 2D overlays to:
  + 3D digital twin model (Three.js).
* Add mobile view to dashboard.
* Integrate real valve control hardware.
* Complete:
  + Anomaly simulator.
  + Attack simulator.
* Conduct:
  + Functional testing.
  + Performance testing.

**Raw Data: CSV Logs**

The external company provided daily logs of water consumption in .csv format. These were per-building meter logs, named things like:

* Water\_History\_A1FD\_2025-05-16.csv
* Water\_History\_A2MFF\_2025-05-16.csv

I started with cleaning and combining these. Wrote a Python script to iterate over folders and files, match formats, append them into a single combined\_water\_data.csv file with columns:

* Meter ID / Building
* Date/Time
* Consumption (Liters)

Done: import\_data.py

**Phase 1: Data Ingestion and Cleaning**

Set up the folder campus\_digital\_twin/, with subfolders:

* /data: All logs go here.
* /scripts: Processing scripts.
* /dashboard: Streamlit front end.

Created a YAML config file for building list and input formats. My initial challenge was just understanding and automating log collection and formatting. Used pandas, glob, os, and some basic error handling.

Later, added a version that *auto-appends* new packet logs without rewriting the full combined file. Took care of formatting mismatches (trailing headers, weird quotes, encoding issues, etc.). Also ensured timestamps were parsed to pandas datetime format consistently.

**Visualization Basics**

Got started with streamlit. Initially, a simple dashboard:

* Title
* Select a building
* Plot line chart of daily usage

Used groupby on Date/Time.dt.date, plotted sum of daily consumption. All basic. But it gave me visibility, and that was the first win.

Done: basic\_dashboard.py

**Deployment Diagram Overlay (Streamlit + PIL)**

Now came the exciting part: putting water meter readings onto the **campus map image**.

Used:

* PIL.ImageDraw to draw readings on an image
* Hardcoded sensor coordinates
* For every hour, show readings near sensor positions
* Leak detection turned the labels RED if spike was detected

Added click events to show time-series plots per sensor. Initially tried using st.button and image map-like hacky overlay, but eventually switched to **Plotly Scatter + customdata + Streamlit event handling**.

Done: plotly\_overlay\_dashboard.py

**Near Real-Time Updates**

Wrote script near\_real\_time\_dashboard.py with:

* Auto-refresh every 5 mins
* Pulls latest combined CSV
* Hourly consumption diff via .groupby().max().diff()
* Red alerts for leak: if current hour is 3x the rolling average of past 3 hours

Also added:

* Valve toggle buttons (simulated)
* Sidebar sensor selection
* Mini graph per sensor

**Leak Detection Logic**

There are two kinds of alerts:

1. **Night leaks** — water flow detected between 12am–5am above 100L
2. **Spikes** — sudden jump compared to rolling average

I saved both to:

* night\_leak\_alerts.csv
* spike\_alerts.csv

Then included these in the dashboard’s alert section. Clicked sensors turned red if flagged.

Later, added ML-based spike prediction using Isolation Forest:

python

CopyEdit

from sklearn.ensemble import IsolationForest

ML-backed anomaly detection now part of simulation backend.

**Demand Forecasting**

Linear regression wasn’t enough. Weekend patterns differed, so used Prophet by Facebook:

* Converted daily consumption to time series per building
* Added binary feature for is\_weekend
* Forecasted for next 3 days per building

Generated CSV: demand\_forecast.csv

Also included this in dashboard table.

Done: demand\_forecasting.py with Prophet

**Local Server Integration & Real-Time Logging**

Until now, I was simulating via static logs. But eventually needed to receive **live sensor data from local server/DCU**.

Two approaches:

1. **SCP (Option 1):** Raspberry Pi (local) pushes new logs to laptop via pscp every hour.
2. **HTTP API (Option 2):** Pi sends logs via simple Flask API every X mins.

Chose SCP for now (simpler, more stable). Added batch script on Pi side and polling on laptop side.

Done: sync\_from\_server.bat and auto-log parser

**Cloud Logging with Google Sheets**

Enabled Google Sheets API using a service account (dt-iiitb.json), linked to:

* Sheet: Campus water logs
* Worksheet: Logs

Logs updated on:

* Every leak detected
* Valve toggled
* Anomaly spike

Used:

python

CopyEdit

import gspread

from oauth2client.service\_account import ServiceAccountCredentials

Added audit logs for all alerts

**Admin Login Mode**

Used st.session\_state and st.text\_input with password masking:

* Normal mode: viewer
* Admin mode: controls, leak logs, toggle buttons

Also changed layout based on admin\_logged\_in flag.

**Simulations: Anomaly, Leak, Clog, Attack**

major addition. As per instructions:

**Specs:**

* Leak simulator: random abnormal flow for building
* Clog simulator: gradual reduction in flow
* Attack simulator: tampering packet logs

**Design:**

* Use tshark for raw packet data emulation
* Simulate logs by modifying r (reading) values
* Inject anomalies via custom function
* Timestamp all injected events

Create simulator.py with CLI options

**Enhanced Dashboard UI**

Integrated everything into one enhanced\_dashboard.py:

* Title + login
* Full deployment map on top
* Hourly readings drawn near meter (24x)
* Leak = red marker
* Click = show line chart + valve toggle
* Sidebar filters
* Admin-only: simulate valve close, alerts
* Demand forecast
* Leak logs

Also used icons, spacing, full-width layout, and light/dark compatibility.

**Functional & Performance Testing**

Functional:

* Leak detection tested with real + simulated data
* Forecasts matched historical patterns
* Sensor click + valve toggle tested

**Entity Diagram, DFD, Use Case, Testing**

* Prepared system **entity diagram** showing:
  + Sensors → Local Server → Twin (Laptop) → Visualization & Control
* **Data Flow Diagram**: multiple levels (raw data → combined → alerts → cloud)
* **Use Cases**:
  + View sensor data
  + Detect anomaly
  + Forecast usage
  + Admin control
* Testing split into:
  + Functional: Each feature tested in isolation
  + Performance: Logged refresh time, API transfer dela

W**hat’s Next**

* Add control circuit hardware (actual valve with relay & Pi GPIO)
* Integrate MQTT or HTTP API for actual control
* Improve simulation interface
* Add usage prediction with LSTM
* Auto SMS alert via Twilio (if time permits)

**Enhancing UI/UX & Admin Tools**

To make the dashboard more polished:

* Added admin login screen
* Severity color codes for sensors (normal, warning, critical)
* Interactive tooltips, hover effects
* Planned 3D version using three.js or plotly 3D

Also created a “Popup Panel” on click — showing mini dashboard per sensor.  
current tasks:

Currently waiting for the real time packets to be received at the local server as there is some issues in the transmission side, antenna height is to be increased...

Once the packets start arriving without any junk / errors then I have to create a ssh protocol over local Wi-Fi to send real time packets from local server to my laptop and then use a python script to convert the packets into a readable and clean csv.

Meanwhile working on:

Digital twin simulations:

Leakage, Clog, and Anomaly Simulation for Digital Twin Water Management

# Introduction

In a Digital Twin-based water management system, simulated scenarios like leakage, clogging, and anomaly injection are critical for validating alert systems, resilience, and data-driven operations. This document presents the design and approach for simulating these conditions to rigorously test the twin's monitoring capabilities.

# Leakage Simulation

Leakage simulation is used to mimic real-world pipe leaks in the digital model to test detection logic.

* Types of leaks to simulate:
* • Constant Leak: Low, steady flow during inactivity (e.g., 0.1–0.2L/hr at night).
* • Night-time Leak: Triggered only during 12 AM to 5 AM.
* • Increasing Leak: Gradually rising leak rate over several hours.

Detection methods include threshold violation, continuous flow beyond active hours, and deviation from expected hourly profiles.

# Clog Simulation

Clog simulation reproduces partial or full blockages in pipes.

* Types of clog patterns:
* • Sudden Drop: From high flow to near zero.
* • Partial Drop: 30–70% decrease in consumption.
* • Intermittent Clog: On/off flow disruption during typical usage hours.

Digital twin can detect clogs by comparing expected vs actual flow and using machine learning classification of under-consumption behavior.

# Test Data Simulator

The simulator is responsible for producing synthetic water meter readings in standard packet format (CSV/JSON) including both normal and abnormal cases.

* Features:
* • Control over building ID, timestamp, and anomaly type.
* • Adjustable frequency and duration of anomalies.
* • Batch file generation for automated test runs.

# Anomaly Simulator

Injects random or rule-based data inconsistencies into the dataset for robustness testing.

* Types of anomalies:
* • Sensor Freeze – Repeating same value for hours.
* • Random Spikes – Sudden, isolated consumption jumps.
* • Drift – Gradual increase/decrease over time.
* • Missing Data – Gaps in timestamps or consumption.

All anomalies are labeled for evaluation against detection logic in the dashboard.

# Conclusion

Simulating leakage, clogs, and anomalies within a digital twin helps validate system performance, ensure robust leak detection, and fine-tune predictive controls. These tools enhance readiness for real-world operational issues, driving smarter infrastructure management.

Attack Detection in Digital Twin-based Water Management System

In modern IoT-based water management systems, sensor-level cyber-physical attacks pose serious threats to reliability and safety. This document outlines how attack detection and simulation can be integrated into a Digital Twin (DT) system, enhancing both monitoring and resilience capabilities.

# Can Water Meter Sensors Be Attacked?

* 1. Spoofing – Injecting fake consumption values to mask leaks or trigger false alarms.
* 2. Replay Attacks – Re-sending old data to hide real-time changes.
* 3. Denial of Service (DoS) – Overloading the server with excessive packet data to cause failure.
* 4. Tampering with valve controls – Unauthorized commands to open/close valves, potentially causing damage.

# Can a Digital Twin Detect or Simulate These Attacks?

* 1. Anomaly-Based Detection – Real-time values are compared with learned patterns to flag deviations.
* 2. Redundancy Cross-Validation – DT checks adjacent sensors for inconsistent behavior.
* 3. Simulation Injection – Simulate spoofed or delayed packets to test alert mechanisms.
* 4. Signature Matching – Identify repeated or recognizable malicious behavior (e.g., midnight packet flood).

# Recommended Simulated Attacks for Testing

|  |  |
| --- | --- |
| Attack Type | Simulation Logic |
| Spoofing | Inject abnormal readings (e.g., 9999.9 L) for 2 hours. |
| Replay | Copy historical data and resend at a later time. |
| DoS | Send 1000 identical packets in a short interval. |
| Packet Loss | Drop packets from selected sensors for 2+ hours. |

# Conclusion

The integration of attack simulation and detection into a digital twin environment allows for robust testing and greater operational security. This approach not only identifies vulnerabilities but also trains the system to respond intelligently to cyber-physical threats.  
  
DESIGN:

**Test Data Simulator**

**Design Overview**

**How It Will Work:**

* Create a data generator that:
  + Randomly produces readings for all meters.
  + Supports normal, peak, and outage modes.
  + Writes data in the same format as live CSV logs.

**Features:**

* Adjustable frequency (e.g., every 5 min)
* Time range configuration
* Modes:
  + Normal: baseline consumption
  + High Load: surge usage
  + Low Load: minimal usage

**Components**

1. **Data Generation Script**
   * Command-line options:

css

Copy code

python simulate\_data.py --mode high\_load --start 2025-06-25 --duration 24h

1. **Simulator Config**
   * YAML to define patterns.
2. **Output Logs**
   * Stored as simulated\_data.csv for ingestion.

**Attack Detection Simulation**

**Design Overview:**

**How It Will Work:**

* A script will generate “fake” or tampered packets (e.g., negative consumption, unrealistically high usage).
* These packets will be injected into your existing combined\_water\_data.csv or a separate *simulated\_attack\_data.csv*.
* The dashboard leak detection + anomaly detection logic will process them and show alerts.
* You can validate whether alerts were triggered.

**Data Manipulation Examples:**

* Random spikes (1000x normal consumption)
* Timestamp offsets (e.g., future timestamps)
* Gaps/missing readings
* Repeated identical readings (replay attack)

**Components**

1. **Attack Simulation Script**
   * Python script to generate and append tampered data.
2. **Attack Data Log**
   * Separate CSV to keep track of injected attack packets.
3. **Triggering Alert Logic**
   * Use your ML anomaly detection logic or simple thresholding.
4. **Dashboard Visualization**
   * Show flagged entries in red with a label: “ Attack suspected.”

**Leakage Simulation**

**Design Overview**

**How It Will Work:**

* Generate synthetic consumption logs with continuous, elevated readings (e.g., constant 5L every 15 min overnight).
* Feed them into the main dataset.
* Observe whether:
  + Night leak alerts fire.
  + Consumption graphs clearly show sustained flow.

**Data Manipulation Examples:**

* Add constant baseline usage for all hours.
* Use realistic timestamps (e.g., simulate 3 consecutive nights).

**Components**

1. **Leak Simulation Script**
   * Create a script to inject constant consumption over chosen intervals.
2. **Leak Scenario Config**
   * YAML/JSON file defining:
     + Meter ID
     + Start date/time
     + Duration
     + Leak rate
3. **Leakage Report**
   * CSV summary listing simulated leaks for traceability.

**Clog Simulation**

**Design Overview**

**How It Will Work:**

* Insert intervals with zero or very low consumption (e.g., 0.05L for many hours).
* Compare with typical usage baseline.
* Observe dashboard for underuse flags or operator alerts.

**Data Manipulation Examples:**

* Set readings to zero during normal operation hours.
* Introduce partial flow intermittently.

**Components**

1. **Clog Simulation Script**
   * Python script to inject low readings.
2. **Clog Scenario Config**
   * Define:
     + Affected meters
     + Start/stop times
     + Reduction factor
3. **Clog Simulation Log**
   * CSV file of all changes.

**Anomaly Simulator  
 Design Overview**

**How It Will Work:**

* Inject patterns such as:
  + Sudden spikes
  + Sustained high readings
  + Random missing data
  + Duplicate readings

**Features:**

* Reproducible: same seed generates same data
* Multiple meters at once
* Custom severity levels

**Components**

1. **Anomaly Generation Script**
   * Similar to attack simulation, but broader.
2. **Scenario Definitions**
   * YAML specifying:
     + Meter IDs
     + Anomaly types
     + Timing
     + Magnitude
3. **Dashboard Integration**
   * Show flagged records in red/yellow based on severity.

>Use tshark good api for anomaly and other sims

Done with the specifications and the design phase..

Going to start code and testing.