

Water Management Plant Simulation

By Sergio Verdugo, Baptiste Etroy, Juan Diego Fernández, Fynn Fruhling

and Pedro Torrado

Cloud Computing Final Project

Professor: Juan Grau

Introduction

Objective

Our project aims to develop an advanced cloud-based infrastructure for a water management plant simulation. Utilising cloud computing and Internet of Things (IoT) technologies, we strive to enhance the efficiency, responsiveness, and operational control of water treatment processes.

Approach

We approach this challenge by integrating a robust cloud computing infrastructure with IoT-enabled valves. This system facilitates real-time data processing and remote control capabilities, ensuring a high level of precision and flexibility in water management.

Initial System Design

Overview

The initial system design envisioned a comprehensive setup incorporating IoT devices for real-time water flow control, a cloud-based backend for processing and data storage, and a user-friendly interface for operational command and monitoring.

System Components and Architecture

The initial architecture comprised:

IoT Devices: Valves equipped with sensors to monitor and regulate water flow.

Cloud Services: Cloud-based storage and computing resources for processing data from IoT devices.

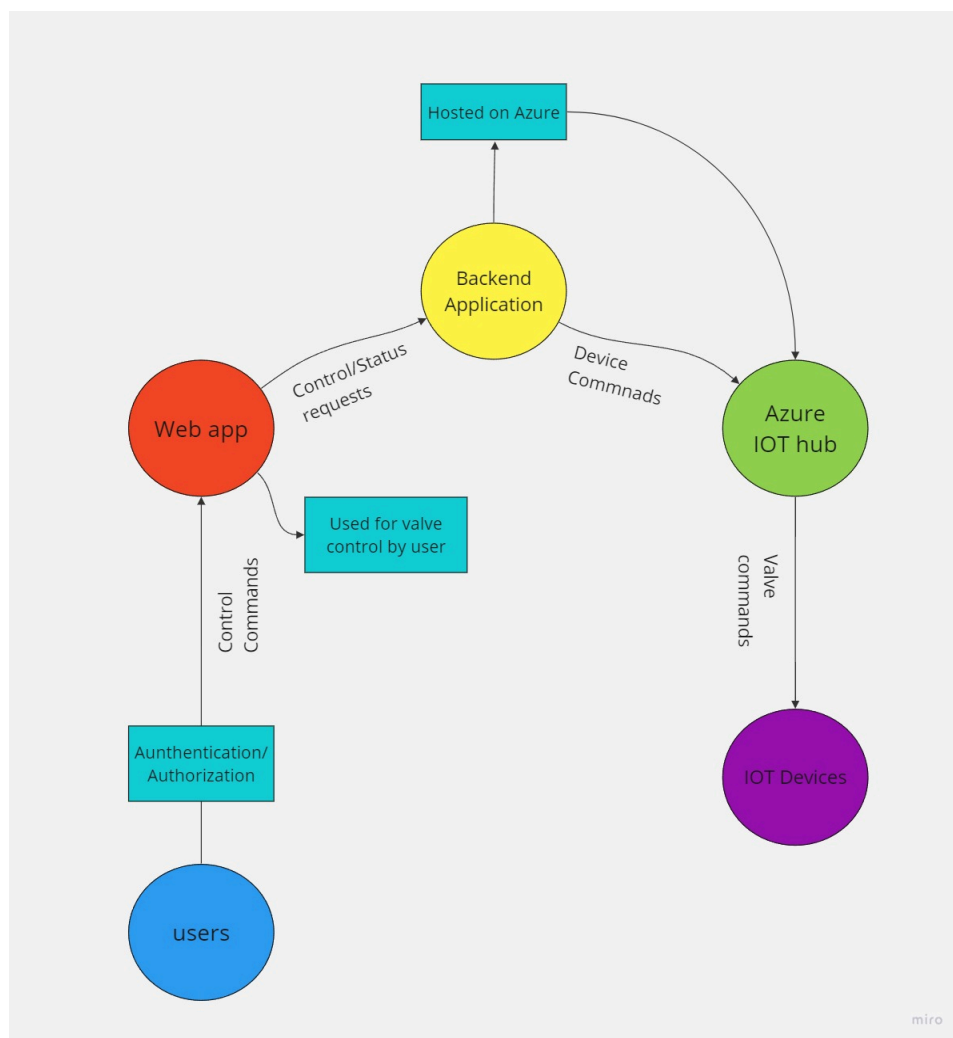


Figure 1: Data flow diagram

Entity Relationship Diagram

Entity-Relationship Diagram for Water Management Plant Infrastructure

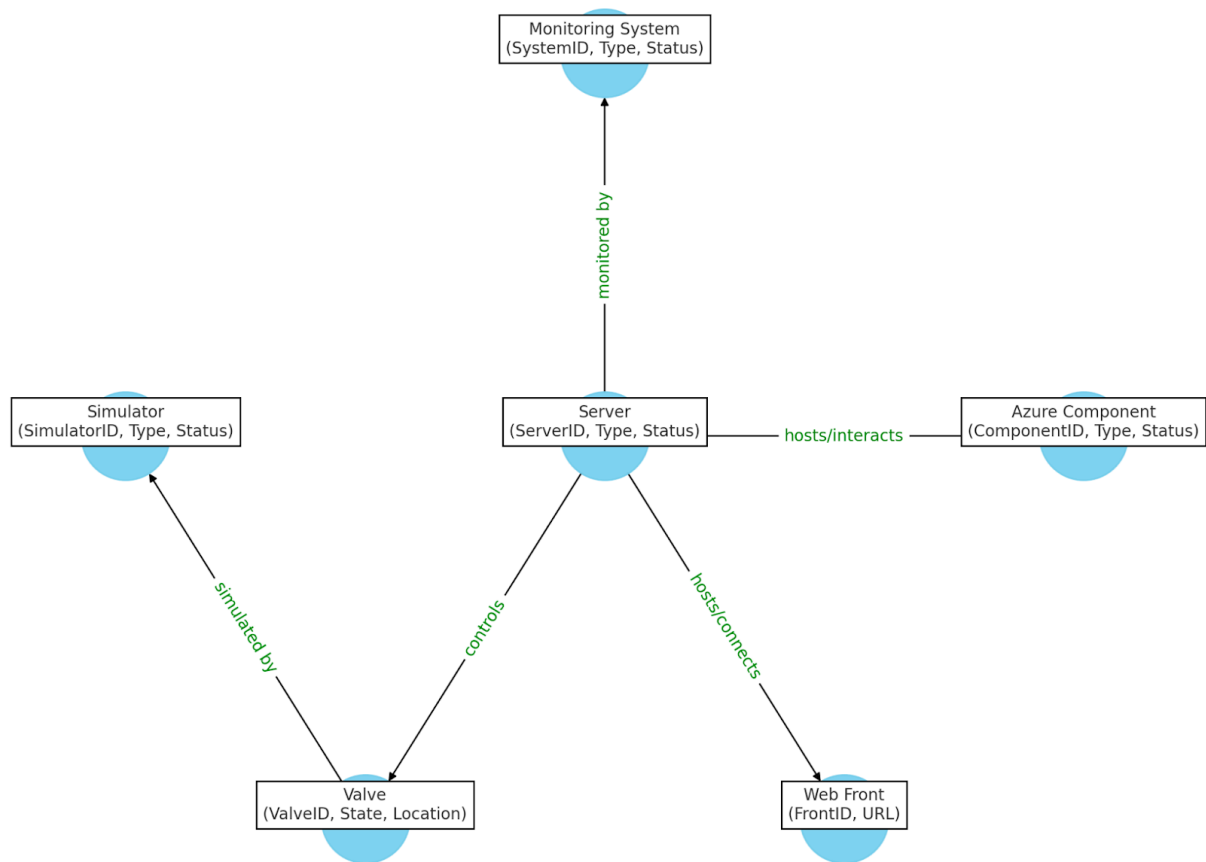


Figure 2: Entity relationship diagram

- **Servers**
Central to the infrastructure, controlling IoT valves and connecting to the web front and Azure components. Attributes include ServerID, Type, and Status.
- **Valves (IoT)**
Represent IoT devices controlling water flow, linked to simulators for testing. Key attributes are ValveID, State, and Location.
- **Web Front**
Interface for manual operation, connected to servers.
- **Azure Components**
Support infrastructure with VMs, Containers, etc.
- **Simulators**
Simulate valve operations, linked to IoT valves.
- **Monitoring Systems**
Oversee server operations and policy implementation.

User Interaction and Control

User Interface: A web-based application for real-time monitoring and manual control of the valves.

Users were expected to interact with the system through a web interface, allowing them to manually operate valves and view real-time data. The interface was designed to be intuitive and accessible

Problems Encountered

Challenges in Implementation

Our project encountered significant technical challenges, particularly with Azure services and the Raspberry Pi simulator, which necessitated creative problem-solving and alternative approaches.

Azure Services - Policy Problem

The most critical issue we faced was with Azure. Due to a policy problem at the organisational level, we were unable to create any resources within Azure's environment. This unexpected constraint severely impacted our project as it disabled our access to vital Azure services, which were integral to our original design.

The nature of the policy problem was complex. It stemmed from administrative restrictions set at the organisational account level, which inadvertently prevented us from provisioning new resources. This issue was not just a technical hurdle but also an

administrative one, requiring coordination with account administrators and a deep understanding of Azure's policy and governance framework.

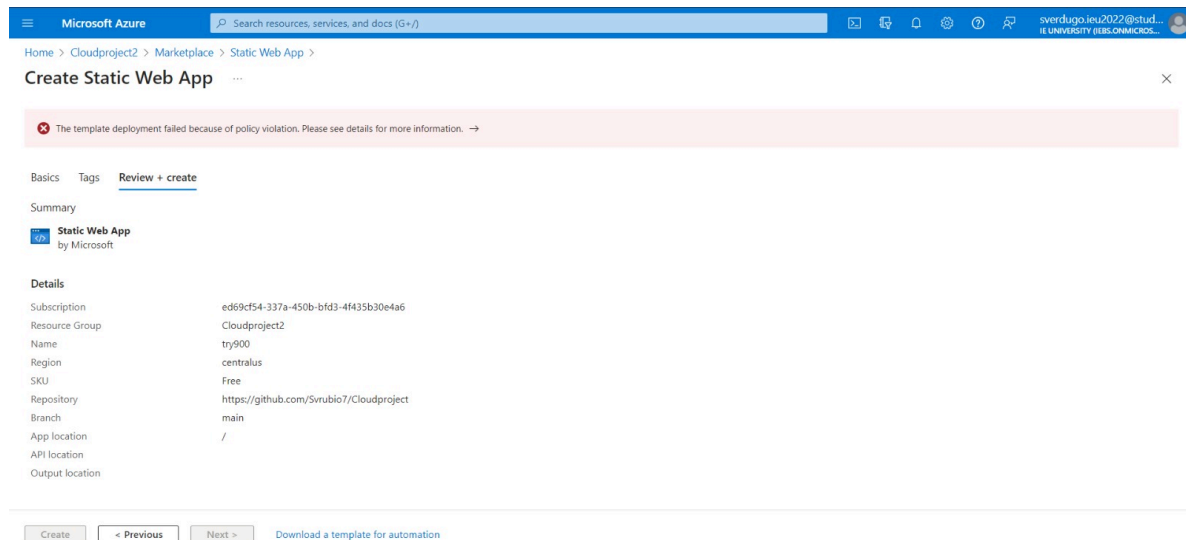


Figure 3: template deployment error

As a workaround, we pivoted to alternative services. With Azure services off-limits, we adapted our strategy and utilised GitHub for hosting our application components. This shift required a significant redesign of our infrastructure and deployment strategy. We had to ensure that the GitHub-hosted components could communicate effectively with our other system elements, maintaining the integrity and functionality of our design.

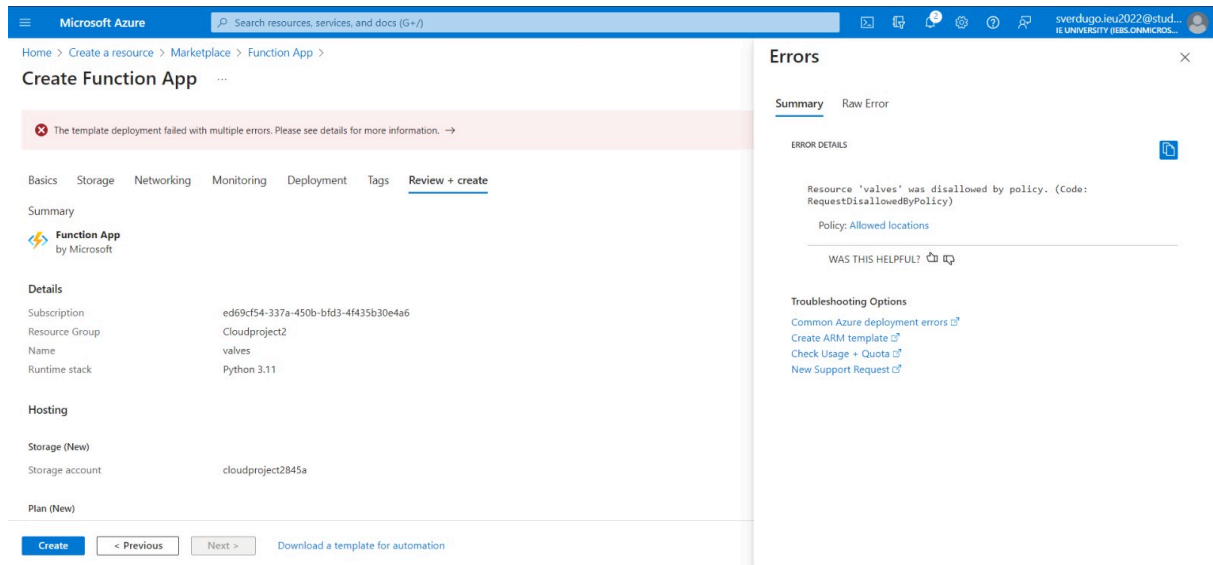


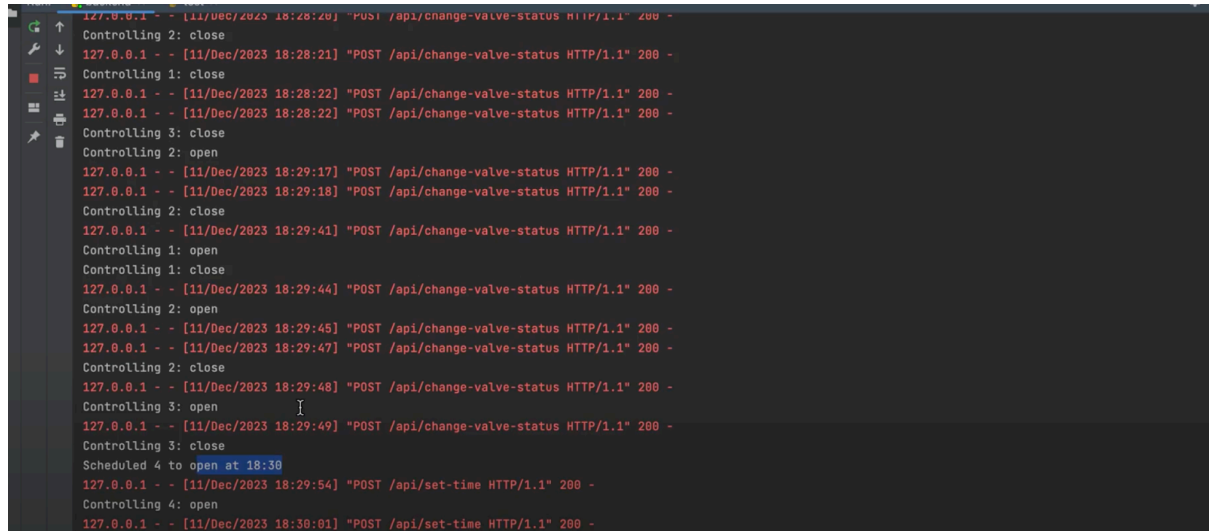
Figure 4: Error output

Raspberry Pi Simulator - Custom Development

Another major challenge was the Raspberry Pi simulator. Our initial plan was to use a pre-existing simulator for testing the IoT interaction with our system. However, due to compatibility and functionality issues, we found that existing simulators did not meet our project's specific needs.

As a result, we took on the task of developing our own custom simulator using Python. This endeavour was both ambitious and technically demanding. We had to create a simulator that accurately mimicked the behaviour of Raspberry Pi-controlled valves in a cloud computing environment. This involved:

This was a huge challenge that we managed to overcome by creating this simulator with which the app interacts and is dedicated to manage the state of the valves including changing it when it is scheduled.



```
127.0.0.1 - - [11/Dec/2023 18:28:20] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 2: close  
127.0.0.1 - - [11/Dec/2023 18:28:21] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 1: close  
127.0.0.1 - - [11/Dec/2023 18:28:22] "POST /api/change-valve-status HTTP/1.1" 200 -  
127.0.0.1 - - [11/Dec/2023 18:28:22] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 3: close  
Controlling 2: open  
127.0.0.1 - - [11/Dec/2023 18:29:17] "POST /api/change-valve-status HTTP/1.1" 200 -  
127.0.0.1 - - [11/Dec/2023 18:29:18] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 2: close  
127.0.0.1 - - [11/Dec/2023 18:29:41] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 1: open  
Controlling 1: close  
127.0.0.1 - - [11/Dec/2023 18:29:44] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 2: open  
127.0.0.1 - - [11/Dec/2023 18:29:45] "POST /api/change-valve-status HTTP/1.1" 200 -  
127.0.0.1 - - [11/Dec/2023 18:29:47] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 2: close  
127.0.0.1 - - [11/Dec/2023 18:29:48] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 3: open  
127.0.0.1 - - [11/Dec/2023 18:29:49] "POST /api/change-valve-status HTTP/1.1" 200 -  
Controlling 3: close  
Scheduled 4 to open at 18:30  
127.0.0.1 - - [11/Dec/2023 18:29:54] "POST /api/set-time HTTP/1.1" 200 -  
Controlling 4: open  
127.0.0.1 - - [11/Dec/2023 18:30:01] "POST /api/set-time HTTP/1.1" 200 -
```

Figure 5: Record of messaging with simulator

Revised System Design

Modifications and Improvements

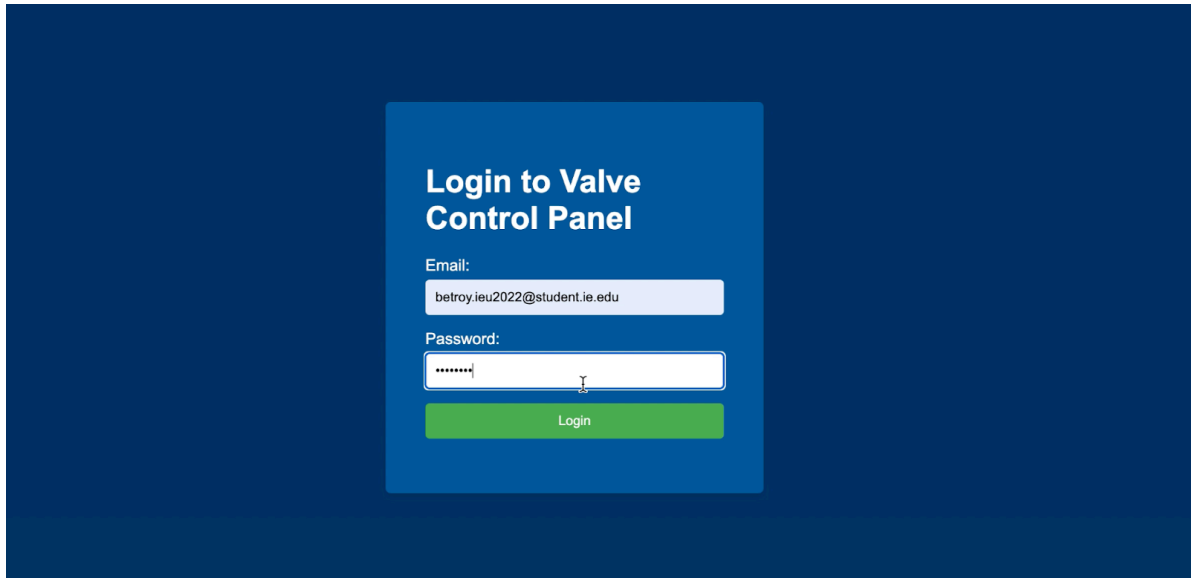
Due to the problems already described, we had to pivot and change our implementation design while still using cloud services.

Final System Components and Architecture

The final design included:

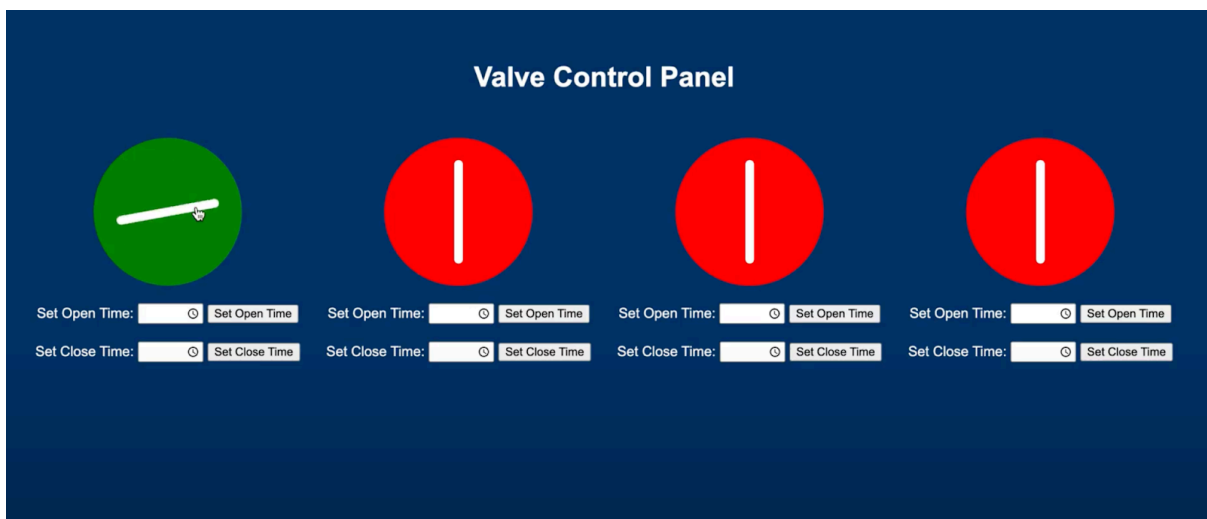
- An app with a detailed user interface.
- An authentication system that checks if you are authorised to modify the state of the valves.
- A custom simulator made with python that controls the state of the valves.

Final user interface:



The image shows a login interface for a valve control panel. It features a dark blue background with a central light blue box containing the title "Login to Valve Control Panel". Below the title are two input fields: "Email:" with the value "betroy.ieu2022@student.ie.edu" and "Password:" with masked characters "*****". A green "Login" button is positioned below the password field.

Figure 6: User authentication panel



The image displays the main control panel for valve manipulation. It has a dark blue background with the title "Valve Control Panel" at the top. Below the title are four circular indicators: a green circle with a white lever icon on the left, and three red circles with white vertical bar icons on the right. Under each indicator are two input fields labeled "Set Open Time:" and "Set Close Time:", each followed by a small circular icon and a "Set" button.

Figure 7: Valve management system, valve manipulation

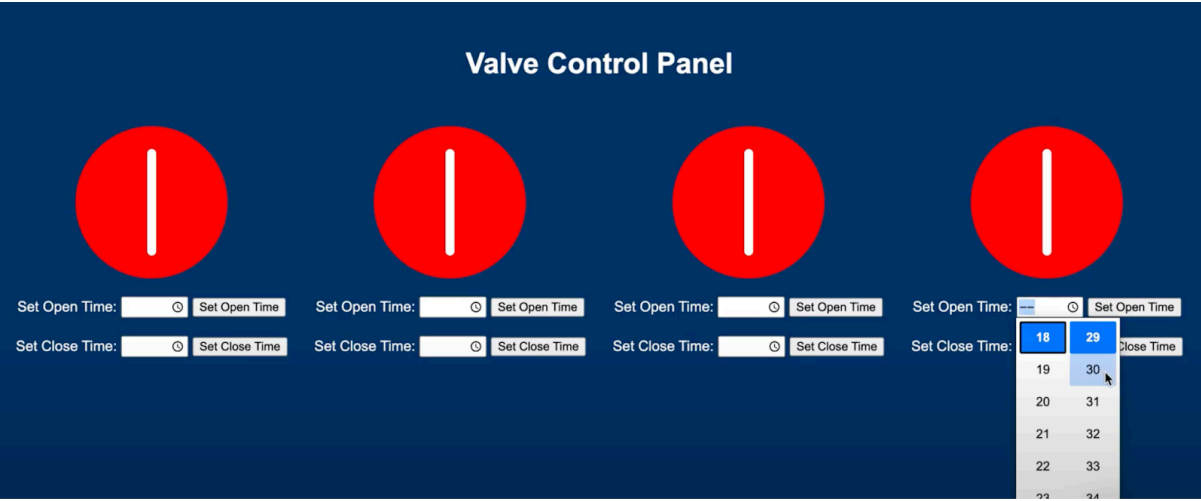


Figure 8: Valve scheduling

Conclusion

Achievements

Our project successfully integrated cloud computing and IoT technologies to create a responsive and efficient water management simulation system. The system allows for precise control and real-time monitoring of water treatment processes.

Future Enhancements

Future enhancements could include implementing AI algorithms for predictive maintenance, expanding the system for larger scale operations, and incorporating advanced security protocols.

Bibliography

(No date) Google cloud documentation | documentation. Available at:

<https://cloud.google.com/docs> (Accessed: 11 December 2023).

Craigshoemaker (no date) Azure static web apps documentation, Microsoft

Learn. Available at: <https://learn.microsoft.com/en-us/azure/static-web-apps/>

(Accessed: 11 December 2023).

Documentación de Python - 3.12.1 (no date) 3.12.1 Documentation. Available

at: <https://docs.python.org/es/3/> (Accessed: 11 December 2023).

Kcpitt (no date) Azure documentation, Microsoft Learn. Available at:

<https://learn.microsoft.com/en-us/azure/?product=popular> (Accessed: 11

December 2023).

Kgremban (no date) Conexión del Simulador web de raspberry pi a azure iot hub

(node.js), Conexión del simulador web de Raspberry Pi a Azure IoT Hub

(Node.js) | Microsoft Learn. Available at:

[https://learn.microsoft.com/es-es/azure/iot-hub/iot-hub-raspberry-pi-web-simula](https://learn.microsoft.com/es-es/azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started)

tor-get-started (Accessed: 11 December 2023).

MozDevNet (no date a) CSS: Cascading style sheets: MDN, MDN Web Docs.

Available at: <https://developer.mozilla.org/en-US/docs/Web/CSS> (Accessed: 11

December 2023).

MozDevNet (no date b) HTML: Hypertext markup language: MDN, MDN Web

Docs. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTML>

(Accessed: 11 December 2023).

Msangapu-Msft (no date) Quickstart: Deploy a python (django or flask) web app to Azure - Azure App Service, Quickstart: Deploy a Python (Django or Flask) web app to Azure - Azure App Service | Microsoft Learn. Available at: <https://learn.microsoft.com/en-us/azure/app-service/quickstart-python?tabs=flask%2Cwindows%2Cvscode-aztools%2Cvscode-deploy%2Cdeploy-instructions-azportal%2Cterminal-bash%2Cdeploy-instructions-zip-azcli> (Accessed: 11 December 2023).

Welcome to flask¶ (no date) Welcome to Flask - Flask Documentation (3.0.x). Available at: <https://flask.palletsprojects.com/en/3.0.x/> (Accessed: 11 December 2023).