Mini Pulp Process

System Documentation

Version 1.2.0.0

Eero Eriksson

Document state: Complete Modified: 24.3.2025

CONTENTS

1 INTRODUCTION

This document is the system documentation concerning the Mini Pulp Process simulator. This is specifically about the software's server side of the process known as BatchProcSimulator. The purpose of this document is to explain the system structure of the simulator software. With this document in unison with the source code and other documentation, the user should get a broader understanding of the system.

2 SOFTWARE ENVIRONMENT

In this chapter the development environment is defined. With this information, the development team should be able to recreate the environment to continue the development. Additionally, the tools that were used in the development of the software are defined so that they are easy to find.

2.1 Requirements

The simulator must be easily usable and available to the end users. Main purpose of the simulator is to simulate the pulp process with enough accuracy to be used as a training tool during course AUT.420. Preferably the simulator doesn't need any additional setting up other than downloading the package from the course platform.

2.2 Approach

The simulator is made with JavaScript using Node.js. The parts of the software are made into separate entities. The backend uses classes as much as possible to make the system more understandable. For some reason, the opcua server does not like to be in a class. OPC UA server uses port 8087 whereas the express server uses port 8088 of the localhost. Each of the components of the software has a distinct task in the system, making it easy to make changes to the system without breaking the other parts. The frontend and the backend are separated so changes can be made to either one without having to change the other. The simulation is run every 100 ms. The software is run by typing "node app.js" to the terminal in the project folder.

2.3 Versions

These are the versions of the software that are used in the development:

- VSCode: 1.98.2
- Node.js: 27.9.0
- Express: 4.21.2
- Express-sse: 0.5.3
- Jest: 29.7.0
- Node-opcua: 2.148.0

3 SYSTEM STRUCTURE

In this chapter, the system structure is shown and the interactions between the parts of the software.

3.1 Class diagram

Here is the class diagram of the backend of the simulator in figure 1.

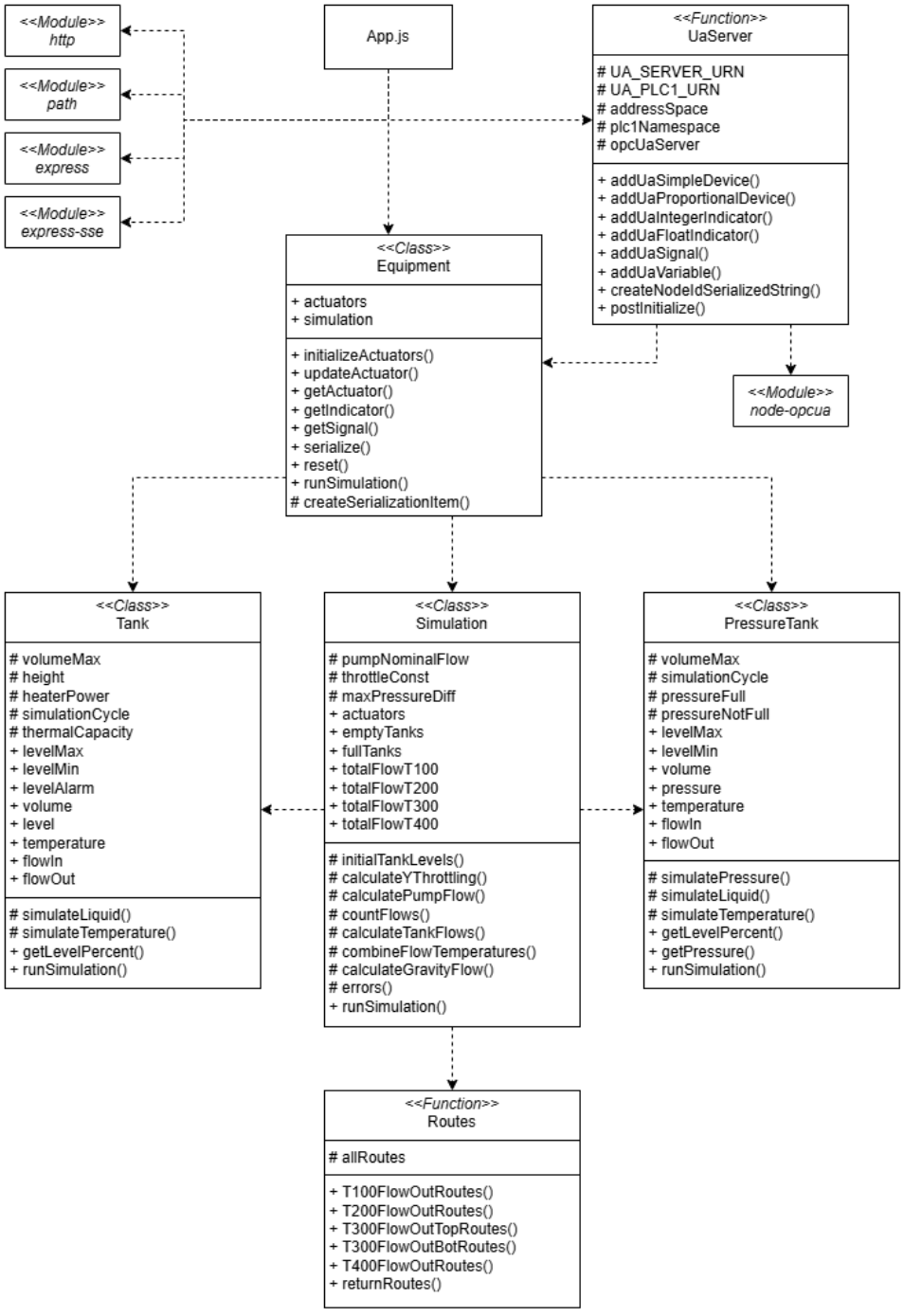

Figure 1. The class diagram

The arrows show the relationship between the different components. The + signs indicate that the variable and method are public whereas the # signs indicate that they are private. Here is what is the responsibility of each class or function:

- App: Starts the application and the express server.
- UaServer: Creates the OPC UA server with appropriate parameters.
- Equipment: Handles all the actuators and methods to use them.
- Tank: Represents a basic tank and its variables.
- PressureTank: Represents a pressure tank and its variables.
- Simulation: Calculates the changes in each simulation cycle.
- Routes: Checks the system for every available route for the liquid.

As you can see from the class diagram, the Equipment class acts as a separator between the servers and the simulation cycles and communications are highly restricted between the different components as there are very few public methods in the components.

3.2 User interface

The user interface is shown in figure 2. The actuators change color based on the state of the system in 500 ms cycles by server by sending a JSON string to the frontend that parses the message into variables. The frontend is found in the folder “public”.

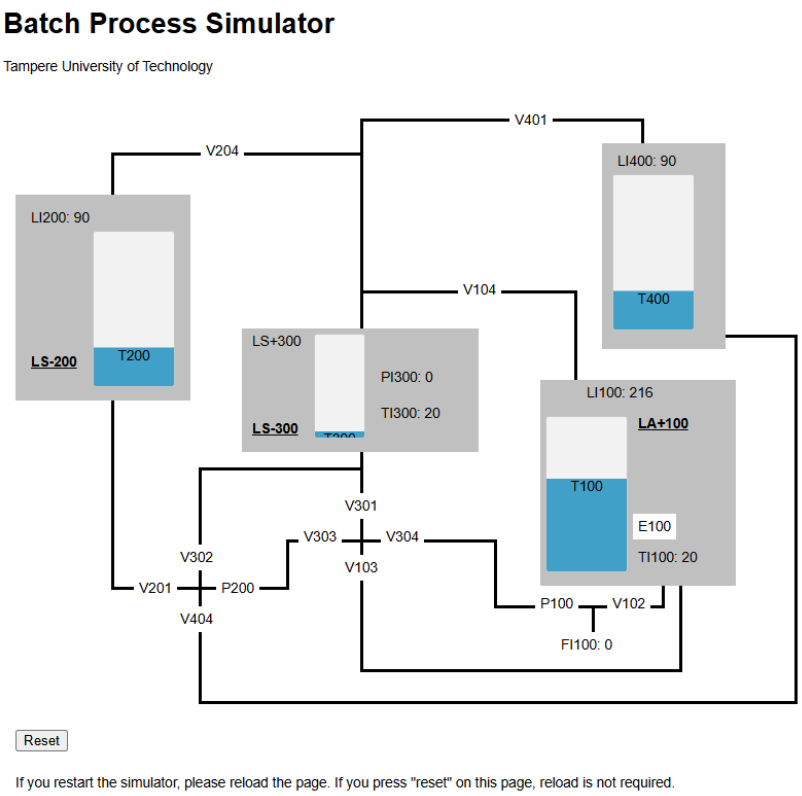

Figure 2. The user interface

The user interface doesn't have all the actuators visible. The backend, however, has all the physics added to actuators and the information is available to the frontend and the user's control system. These should be easy to add to the frontend in the future. They are included in the JSON string sent by the backend.

4 COMPONENTS

Here are small explanations about the components of the simulator and why they are critical to keep it functional.

4.1 App.js

This part of the software is probably the simplest. It contains the express server that runs the user interface in address http://127.0.0.1:8088/. The server uses server-side events that are streamed. It is the first part to be initiated so it creates the equipment class and the OPC UA server.

4.2 Function UaServer

This function creates the OPC UA server. The server is made using the node-opcua module. It is critical to the server that the following parameters are correct to make the connection to the client:

- Server URN: urn:CX-19788E:BeckhoffAutomation:TcOpcUaServer:1
- PLC1 URN: urn:CX-19788E:BeckhoffAutomation:Ua:PLC1
- Hostname: 127.0.0.1
- Port: 8087

Without these parameters the client cannot find the address and the namespaces of the server and therefore will not work. Function has subfunctions for each of the datatypes used in the connection to add the actuators correctly to both sides of the connection.

4.3 Class Equipment

Equipment class handles all the actuators used in the simulation. This class has many public methods because the OPC UA server needs access to update these actuators based on the user's controls. The express server's requests like resetting, running the simulation and creating the JSON string sent to the frontend. This acts as a separator between the servers and the simulation. It makes the system more modular and easier to maintain.

4.4 Class Tank

Tanks represent the basic tanks used in the pulp process. These tanks simulate the changes in temperature and liquid amounts after the simulation has calculated the changes in the whole system. The formulas are included in the comments of the source code for future reference. If the tank signal is marked as null, it means the tank doesn't have an indicator on that spot.

4.5 Class PressureTank

Tanks represent the pressure tanks used in the pulp process. These tanks simulate the changes in temperature, pressure and liquid amounts after the simulation has calculated the changes in the whole system. The formulas are included in the comments of the source

code for future reference. If the tank signal is marked as null, it means the tank doesn't have an indicator on that spot.

4.6 Class Simulation

Simulation calculates the changes in the system during a single simulation cycle that is every 100 ms. It has calculations on whether there are multiple pumps in use, or the flow has multiple endpoints. These will also impact on the temperature and the pressure in the system. The goal is to have the flow in and out of each tank calculated before the values of the tanks are updated. The formulas are included in the comments of the source code for future reference.

4.7 Function Routes

Routes simply check for every single possible route that the liquids can take in the system and return them to the simulation for calculations. These are made based on the P&ID. These are separated by the start tank to make it easier to follow.