Project 1 Individual Report

Yugyeong Hong

yhon106/436666227

Abstract

This report outlines the design and development of a high-tech company's specialised liquid production process simulation using advanced embedded systems. As this is a personal report, this document highlights the sections I directly worked on, such as the design and development of components of the automated bottling system and the entire safety and access control system. The project was developed by integrating SystemJ with Java interfaces to produce advanced simulations.

Introduction

According to the design brief, our team aimed to develop a streamlined design of a distributed embedded system using SystemJ. We were tasked with simulating a high-tech company's process for producing specialised liquids tailored to customer orders. Our responsibility involved designing and executing a functional simulation of various stages of the bottling, manufacturing, and overall ecosystem.

This simulation includes:

- Automated Bottling System - ABS
- Product Ordering System - POS
- Environment Control System - ECS
- Safety and Access Control System - SACS

Based on the brief, each team member was assigned specific parts of the ABS and an entire system to handle individually. I was tasked with the Conveyor and Rotary table of the ABS, as well as the SACS.

The ABS components were naturally modular, allowing for a straightforward distribution of tasks among team members. The brief identified five distinct parts of the ABS that could be developed independently, requiring periodic testing to confirm that all components functioned correctly.

The SACS represents the safety and access of the bottling process environment. This system encompasses access cards, permitting only authorised individuals to access the facility, and manages human presence, enabling the ECS to adjust parameters like humidity, light, and temperature in each zone. Additionally, this system notify the ABS to control potential hazards associated with bottling operations. A crucial feature of the SACS is to halt any bottling activities when staff are present in the manufacturing area, particularly in zone 5. Further details on this will be elaborated later in the report.

Design

When designing the SACS, various system components had to be taken into account. The main objectives of the ACS are to permit specific cards access to particular areas and to halt the bottling operation when a human is detected in the manufacturing zone. While there was consideration to employ badges that convey an individual's location within any zone, the decision was made to use human presence sensors in each zone for both interrupting the bottling process and regulating the environment.

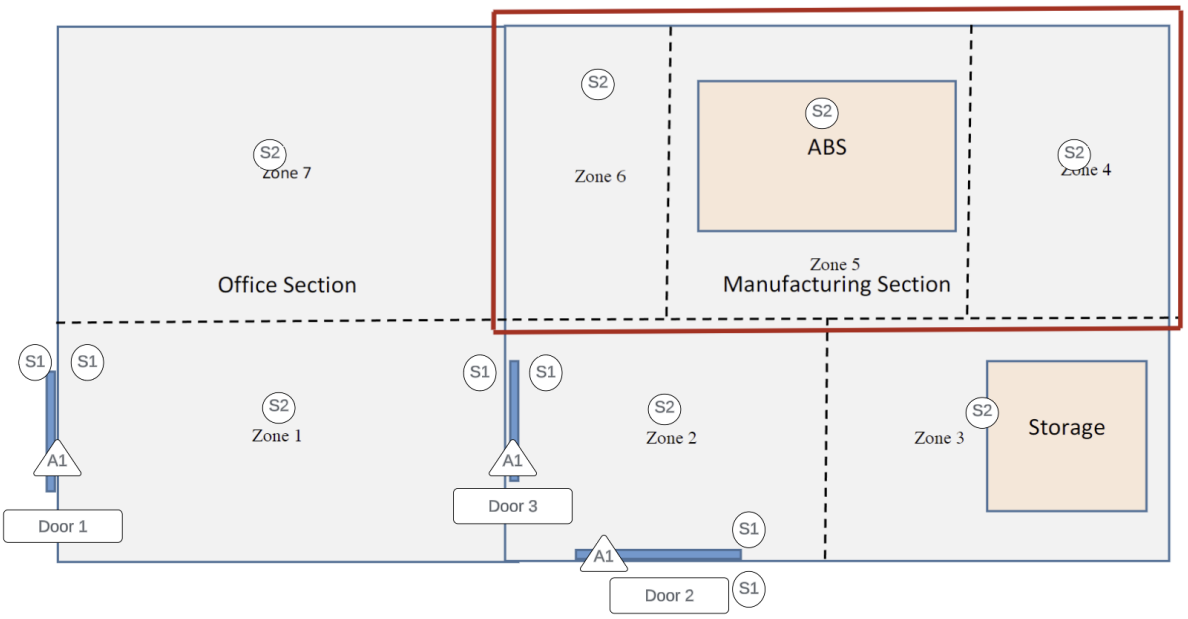

Figure 1. Manufacturing facility layout with sensors and actuators for ACS

This Access Control System (ACS) can be managed through sets of sensors and actuators, which are linked and interact with a SystemJ program using input and output signals. In the preliminary design, Proxy Card Readers and Fingerprint sensors were implemented to permit access for authorised individuals by unlocking the doors and to add an extra layer of access verification. However, based on feedback from the Milestone 1 interview, Fingerprint sensors were deemed unnecessary for an additional layer of access verification. Consequently, only Proxy Card Readers were utilised, with 2 levels of authentication, under the assumption that these readers can store diverse authorization data.

During emergencies, such as a fire, all three doors automatically unlock to ensure safe evacuation. Human presence sensors are positioned across every individual zone (Zone 1 - 7), with the assumption that a single human presence sensor can effectively encompass and monitor each respective zone. Additionally, the presence of a human in Zone 5 results in the immediate suspension of the automated bottling system. Figure 1 visually illustrates the strategic placement of sensors and actuators within the ACS framework. In the diagram, the Sensors and Actuators are denoted as S1 and S2, and A1 and A2 respectively.

- S1: **Proxy Card Reader**, Reads information on an ID card
    - Assume that when a card is presented to the reader, the reader communicates with the database to determine whether the cardholder is authorised to access the area or system.
- S2: **Human presence sensor**, Indicates movement of persons in Its range
    - Assume that one human presence sensor can fully cover one zone

- A1: **Door/Gate lock**, authorise entry for approved individuals and secure the door in the presence of a fire.

The ACS major components:

1. Door 1, which is an external door connected to zone 1. The Door 1 has a proxy card reader on each side of the door which reads information on ID cards and it unlocks the Door 1 for the office section authorising personnel to access.

2. Door 2, which is an external door connected to zone 2. The Door 2 has a proxy card reader on each side of the door which reads information on the ID card. Door 2 is unlocked for the manufacturing zone authorising personnel to access.

3. Door 3, which is an internal door connected between zone1 and zone 2. The Door 3 has a proxy card reader on each side of the door which reads information on the ID card. Door 3 is unlocked for the manufacturing zone authorising personnel to access.

4. Communicating with ECS and ABS systems for overall system control.

In this case the operation of the Controller depends on its interactions with the ACS components. Therefore, in order to test a program, manual control keys will be used to provide input stimuli to and receive control signals that change the state of individual components from the Controller.

SystemJ Implementation

Within the ACS, both a controller and plant model are present, each accompanied by a .sysj and .xml file. The controller contains the system's logic, whereas the plant caters to the simulator. The .xml files detail the clock domains, subsystems, and all input and output signals. A central clock domain is integrated within the controller, overseeing the system's logic, including access permissions

The controller contains the following inputs and outputs:.

Input Signals

- **accessDoor1** : indicates the request to access.
- **accessDoor2** : indicates the request to access.
- **accessDoor2** : indicates the request to access.
- **officeAuthorised** : indicates the authorisation for the office section.
- **ManufacturingAuthorised** : indicates the authorisation for the manufacturing section.
- **smokeDetected** : indicates smoke detected.

Output Signals

- **open_door1** :control signal to unlock door 1.
- **open_door2** :control signal to unlock door 2.
- **open_door3** :control signal to unlock door 3.
- **close_door1** :control signal to lock door 1.
- **close_door2** : control signal to lock door 2.
- **close_door3** :control signal to lock door 3.

Figure 2 represents a loop for controlling access to a door, presumably Door 1, based on certain conditions.

```
loop{
    await(officeAuthorised && accessDoor1 && !smokeDetected); // Wait until both signals are present
        trap(T){

            {sustain open_door1;} || {waitl(3.0s); exit(T);} // Sustain open_door1 for 3 seconds

        }
        emit close_door1;

        abort(officeAuthorised && accessDoor1) {
            await(!open_door1);
        }


}
```

Figure 2. Controlling access for Door 1

1. loop: The entire block of code runs continuously in a loop.

2. await(officeAuthorised && accessDoor1 && !smokeDetected): The code waits for three conditions to be met:

- officeAuthorised signal is present, which might mean that there's authorization from the office for access.
- accessDoor1 signal is present, possibly indicating a request to access Door 1.
- smokeDetected signal is NOT present, ensuring safety from potential hazards like fire.

3. trap(T): This initiates a trap construct, which allows the inner operations to run until the exit condition for the trap is met.

4. {sustain open_door1;} || {waitl(3.0s); exit(T);}: The door is commanded to remain open (sustain open_door1), and concurrently, A timer waits for 3 seconds (waitl(3.0s)). Once this timer elapses, it triggers the exit(T) command, which breaks out of the trap construct.

5. emit close_door1: Once out of the trap, the command to close Door 1 is issued.

6. abort(officeAuthorised && accessDoor1): If the conditions officeAuthorised and accessDoor1 are both met at any point, the actions within the abort block will execute. This provides a mechanism to interrupt normal processing if certain conditions are met.

7. await(!open_door1): The loop will pause here until Door 1 is confirmed to be closed. This is a safety check ensuring that the door is closed before any subsequent iterations of the loop.

In summary, this represents a safety mechanism for Door 1 access. It ensures that the door can only be opened when there's office authorization, an access request, and no smoke detection. For further improvement, the system can be designed to handle subsequent access requests even when the door is already open.

The behaviour of Door 2 and Door 3 mirrors that of Door 1, except they utilise the manufacturingAuthorised signal as an input.

The figure 3 represents the Emergency Smoke Detection control.

```
loop{
    await(smokeDetected); // Wait for smokeDetected signal

    trap(T5){
        {sustain open_door3;} || {sustain open_door2;} || {sustain open_door1;} || {await(!smokeDetected); exit(T5);}
    }
    {emit close_door1;} || {emit close_door2;} || {emit close_door3;}

}
```

Figure 3. Emergency Smoke Detection control

1. Waits for the smokeDetected signal.
2. If detected, it will sustain open_door1, open_door2 and open_door3 and will continue doing so until the smoke detection signal is off.
3. Upon the detection of smoke, all three doors are signalled to close (close_door1, close_door2, close_door3).

In the ACS controller, all these loops work concurrently. In other words, all these described behaviours for the three doors and emergency smoke detection happen simultaneously and independently of each other.

Figure 3 displays the simulation results during an emergency situation, while Figure 4 illustrates the results when there's an access request for Door 2 by someone authorised for the manufacturing section.

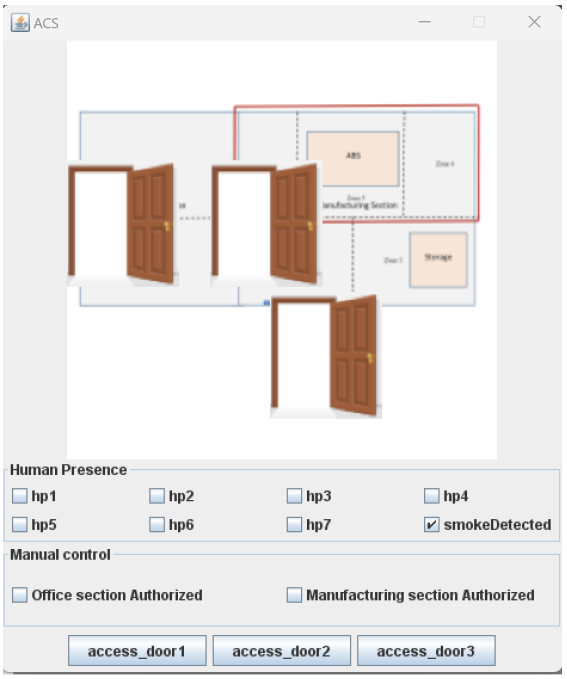

Figure 3. Emergency detection simulation

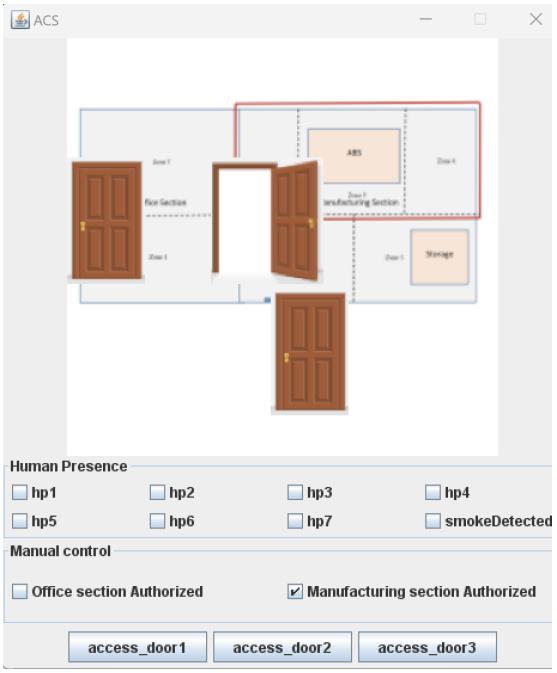

Figure 4. Door access simulation

As shown in Figure 5, ACS communicates human presence in every zone to ECS, and specifically, the presence in the manufacturing zone to the ABS Coordinator. Consequently, when a human presence signal is detected, it is forwarded to other systems for additional control measures.

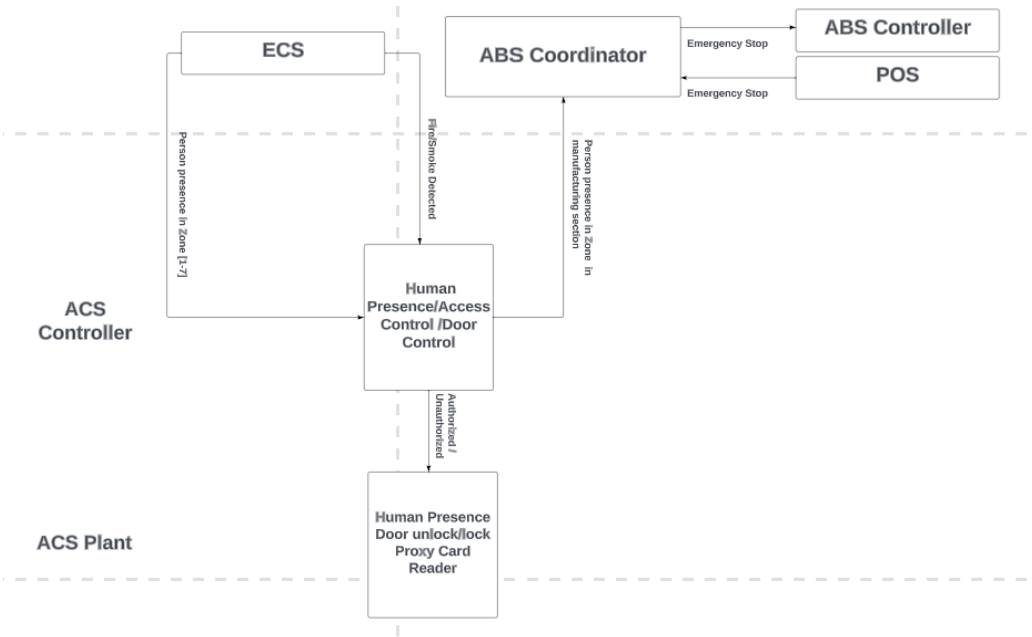

Figure 5.An overview of the ACS model with relationship between other systems

Discussion

Originally, the design comprised three distinct controllers: one for door control, another for access control, and the third for detecting human presence. Each of these controllers had its respective plant model. Additionally, a coordinator was incorporated to facilitate communication between these controllers and other external systems. However, the design was later streamlined to consist of a singular controller and plant model, eliminating the need for a separate coordinator. This unified controller now manages door access, verifies authorizations, and directly transmits signals to other interconnected systems. This modification presented several advantages:

- Simplified Architecture: By consolidating multiple controllers into one,the logic and operations are centralised, making the overall system more straightforward, making it easier to manage, modify, and troubleshoot.

- Reduced Communication Overhead: Eliminating the coordinator means there's less inter-module communication required, enhancing the system's efficiency and potentially reducing latency.

- Consistent Data Handling: With a single controller, data and signals follow a consistent path and processing method, reducing the risk of inconsistencies or misinterpretations that can arise from multiple processing points.

Conclusion

The report illustrates the SACS among various systems, providing simulation outcomes that depict the system's operational capabilities in diverse contexts, such as emergencies and door entry requests. The SACS, as designed, operates in line with the given specifications. It simulates an access mechanism that offers different levels of facility access to personnel. Moreover, it interfaces seamlessly with other systems to achieve advanced controls, such as interrupting the bottling process and managing lighting functions.