COMPSYS 704 Project 1 Individual Report

By Phoenix Doolan

Phoenix Doolan / pdoo984 / 451938550

Abstract

This report documents the planning and development of a working simulation of a high-tech firm's customised complex liquid production process using advanced embedded systems. As this is an individual report it will cover the aspects of the project that I personally contributed to including the designing and implementation of parts of the automated bottling system and the whole of the safety and access control system. The development of this project was done using SystemJ and Java interfaces together to create advanced simulations.

Introduction

According to the design brief, my team's goal was to create a simplified design of a distributed embedded system using the programming language SystemJ. The design we were creating was a simulation of a high-tech firm's liquid production. The aforementioned firm produces customised complex liquids that are ordered by the consumer. Our job was to design and implement a working simulation of multiple parts of the bottling processes, manufacturing, and ecosystem.

This simulation includes:

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

According to the brief our team was to designate the development of parts of the ABS as well as one whole system to each individual member. I was designated with the lid loader of the ABS and the SACS.

The tasks relating to the ABS could easily be divided among team members due to how modular the ABS is as a system. The brief listed five different components of the ABS which could be developed in isolation to the others with occasional testing necessary to ensure everything is working.

The SACS represents the safety and access of the environment around the bottling process. This includes access cards that allow workers to navigate through the facility and badges that transmit their location. The access that a worker has and their location are both very important for safety reasons as the bottling processes can potentially be dangerous. Because of this, one aspect of the SACS is to suspend any bottling processes when personnel are nearby. This will be discussed later on in the report.

Design

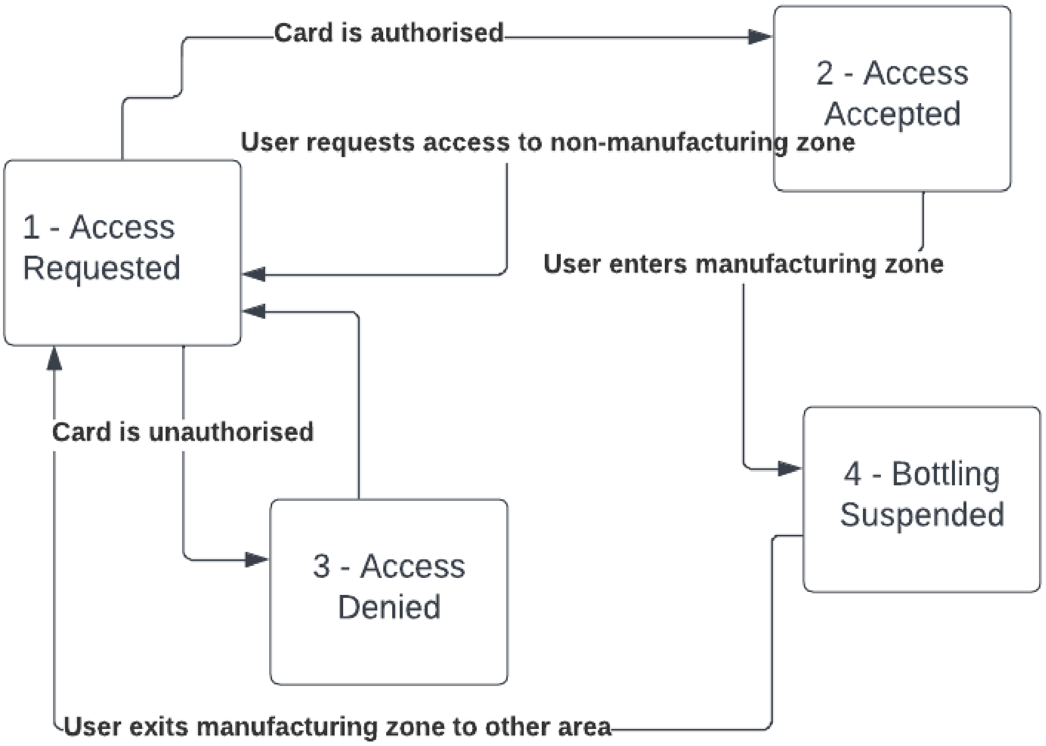

Card is authorised
2 - Access Accepted
User requests access to non-manufacturing zone
1 - Access Requested
User enters manufacturing zone
Card is unauthorised
4 - Bottling Suspended
3 - Access Denied
User exits manufacturing zone to other area

Figure 1: Understanding of SACS

When designing the SACS I had to take into consideration the various parts of the system. The ACS's main goals are to allow certain cards to access certain areas and to suspend the bottling process when the card is present in zone 4, 5 or 6. Entry and exit from the facility is authorised through access cards and biometric information, the latter being optional. Along with the card system there must be a badge system that transmits the user's location.

The approach for the SACS is to develop various signals that signal where the card holder is using their card and if they have access to that zone. For the badge part of the SACS there will be signals corresponding to both the user's badge ID and the location of the badge.

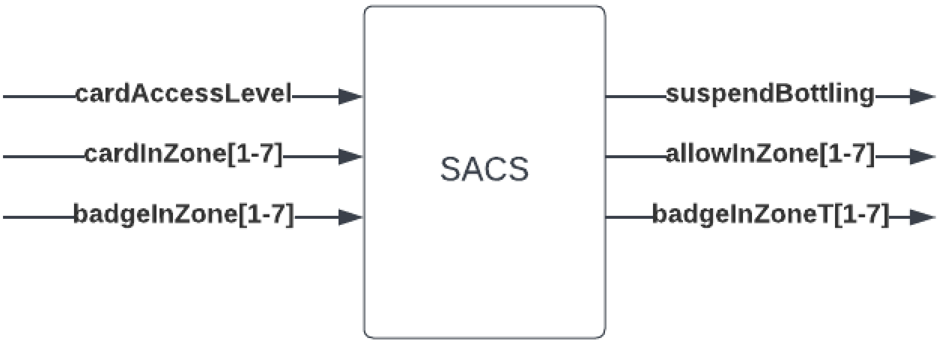


Figure 2: SACS Design

The design seen in figure 2 takes in the access level of the user's card as well as what zone the card is being used in to determine if the user is allowed in the zone and if the bottling process should be suspended (if access is allowed in zone 4, 5 or 6). The badge in zone signal signals the location of the badge and the badge's ID number so the user can easily be identified. The location of the badge can also suspend the bottling process if the user is in zone 4, 5, or 6.

SystemJ Implementation

For the SACS there is a .sysj and .xml file for the controller and a .sysj and .xml file for the plant. The controller contains the logic for the system while the plant is the simulator. The .xml files specify the clock domains, sub systems and all the input and out signals.

There is a central clock domain for the controller which controls the logic for the system. This includes allowing access, transmitting the location of the badge, and suspending the bottling process.

The .sysj file also directly communicates with the ABS as the bottling process must be suspended through the SACS. This is an interface between the SystemJ file and some of the other Java files in the project.

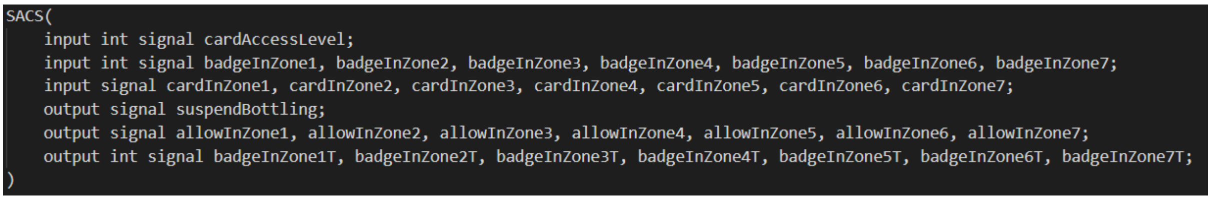
The controller file contains the follow inputs and outputs:

Inputs:

- cardAccessLevel - Indicates the access level of the card
- badgeInZone[1…7] - Indicates where the badge is in relation to the zone's sensors
- cardInZone[1…7] - Indicates where the user is trying to gain access to

Outputs:

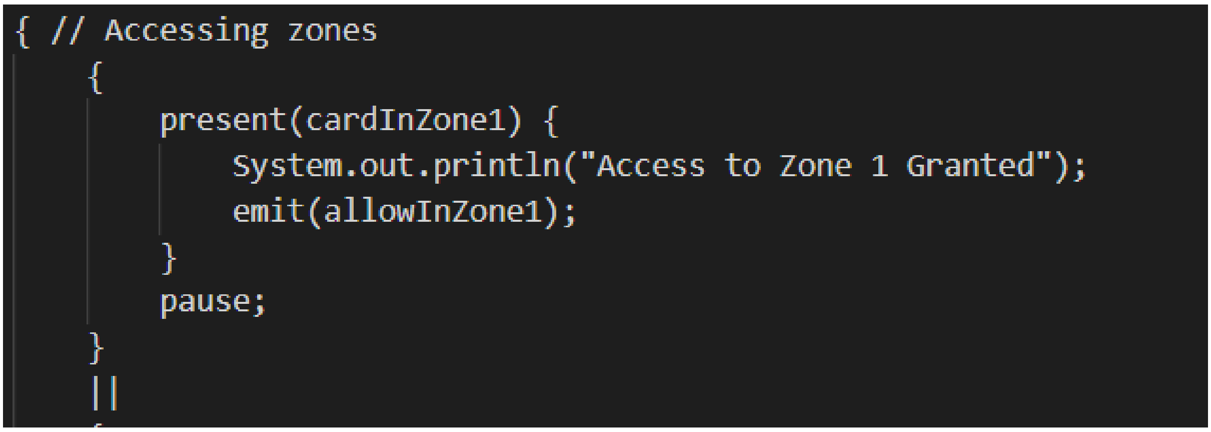
- suspendBottling - Indicates that the bottling process must be suspended
- allowInZone[1…7] - Indicates that the user has gained access to the desired zone
- badgeInZone[1…7]T - Indicates the location of the user



```
SACS(
    input int signal cardAccessLevel;
    input int signal badgeInZone1, badgeInZone2, badgeInZone3, badgeInZone4, badgeInZone5, badgeInZone6, badgeInZone7;
    input signal cardInZone1, cardInZone2, cardInZone3, cardInZone4, cardInZone5, cardInZone6, cardInZone7;
    output signal suspendBottling;
    output signal allowInZone1, allowInZone2, allowInZone3, allowInZone4, allowInZone5, allowInZone6, allowInZone7;
    output int signal badgeInZone1T, badgeInZone2T, badgeInZone3T, badgeInZone4T, badgeInZone5T, badgeInZone6T, badgeInZone7T;
)
```

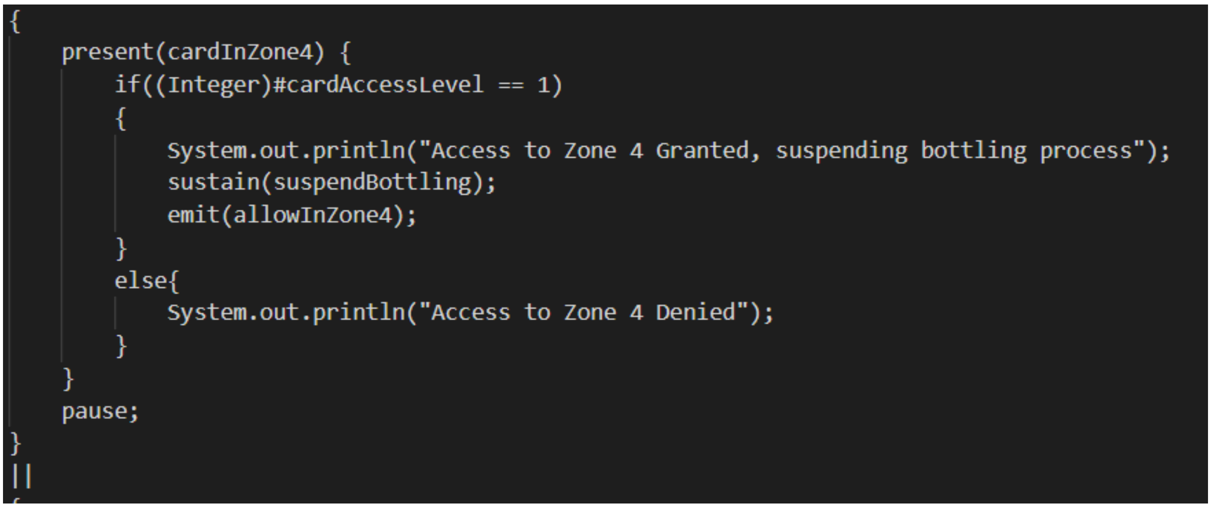
Figure 3: Inputs and Outputs

These signals are all managed using **present** statements containing **emit/sustain** statements all within **while(true)** statements. This system allows constant surveillance of where users are trying to enter and allows the entries that are valid while taking care of the worker's safety.



```
{ // Accessing zones
    {
        present(cardInZone1) {
            System.out.println("Access to Zone 1 Granted");
            emit(allowInZone1);
        }
        pause;
    }
    ||
```

Figure 4: Allowing entry into zone without requirement



```
{
    present(cardInZone4) {
        if((Integer)#cardAccessLevel == 1)
        {
            System.out.println("Access to Zone 4 Granted, suspending bottling process");
            sustain(suspendBottling);
            emit(allowInZone4);
        }
        else{
            System.out.println("Access to Zone 4 Denied");
        }
    }
    pause;
}
||
```

Figure 5: Checking validity of access to Zone 4

As seen in figure 5, when entry needs to be restricted there is a check for if the user’s card has access to that specific zone.

Discussion

For my contribution, the SACS implemented also works according to the brief, simulating an access system that would allow personnel different levels of access throughout the facility and a badge system that transmits the user's location to ensure safety. The bottling process is successfully suspended if any situation that may lead to danger does arise.

Conclusion

My team successfully designed and implemented a working simulation of a complex liquid bottling process using SystemJ and Java. I personally contributed to the team's development of the ABS and also developed the safety and access control system that simulates the navigation and safety of the environment around the manufacturing. The simulated system successfully prevents dangerous situations to ensure safety of the workers.