COMPSYS704:GRP1 – Team 7 Project Compendium

1. Introduction

Advantech Ltd., a company for manufacturing and delivery of sensitive and high value bottled liquids, have decided to build a new manufacturing facility that will automate the manufacturing process within the existing facility, provide advanced system for monitoring and controlling environmental conditions and access and security control.

This project is to develop this solution leveraging IoT concepts, synchronous programming and system-level designing. The chosen language and environment to create this is System J and Java. This is a written compendium detailing to a lower level the components and operation of this full solution.

2. Brief

The solution is facility wide. Incorporated in multiple parts of the physical facility as well as in the purpose of the facility, it has the following requirements:

- A developed Automated Bottling System (ABS), as this is facility's purpose, to output orders of bottled goods
- Be able to monitor who is within the facility as well as specifically where they are
- Be able to provide security in the form of only allowing selected personnel in general and specific areas of the facility (e.g. restricted access to the main office)
- Continuously monitor and adjust the facility's climate factors such as humidity and heat to desired conditions. Even able to adjust the climate of specific areas differently within the facility
- Receive and process orders from registered customers for bottling

These are the high-level requirements from which more detailed and specific requirements stem from.

3. Design Notes

The following section explains the components, structure and operations of the ABS. Design approach and implementation of the solution are also explained in this section as well as any assumptions or matters to note.

Approach & Implementation

In order to implement it was required whatever tools, language and environment to be used allowed the use of IoT concepts, synchronous programming and systems-level design. Prior to beginning the project it was already decided to use System J and Java for implementation.

System J

System J is a synchronous programming language following the concept of creating globally asynchronous locally synchronous (GALS) systems. Its basic structure is that a collection of synchronous programming statements form a reaction. This reaction is able to have signals inputted into it to be utilised by the program statements as well as emit signals that can be broadcasted and received by other reactions. These signals can be pure signals simply representing a TRUE/HIGH or FALSE/LOW or they can be valued, being able to hold various data types and structures such as an array of integers. A collection of reactions all operating on the same clock tick is known as a clock domain. Clock Domains are asynchronous with each other and instead use defined channels to communicate signals and values to each other. A collection of clock domains is considered to be a System J system. This system will have asynchronous output and input of signals to some kind of environment. It is built atop Java, CSP and Esterel.

The benefit and reason for using System J within this project is that all computation and logic can be done on it, with different operations being able to be run at the same time. This concept is used to enable multiple stations with their own independent tasks to run at the same time and not wait for each other to complete as would be the case in an exclusively sequential, asynchronous programming language. Additionally, with it being built upon Java, Java libraries for connection and interfacing can be integrated well with the System J. The exact breakdown of clock domains and use of libraries is explained more as one goes along this document.

Java

Java is an object-oriented programmer that makes use of a compiler and built-in virtual machine. It is structured around variables that when grouped together make an instance or object. A type or collection of objects are defined in class, as well as all its associated methods. The reason for using Java within this project is to leverage the various libraries, namely the Java Swing and Java Socket libraries. Java Swing is a library that allows for the creation of graphic user interfaces (GUIs) that can allows visualisation of a system as it operates as well as allow easier interfacing of a user inputting values and commands. This is library is used to make all GUIs that feed, as well as show information to and from the user of the system. Java's socket library allows for the creation of TCP/IP sockets to communicate locally and externally with other computers and programs. This proves essential to allow for the information on the computation of the System J to be communicated for the GUI to display, as well as when the GUI wants to send collected information from the user.

Design Approach

The approach was taken was to make the overall system of systems decentralized. The full solution is made of the four sub-systems:

- Automated Bottling System (ABS): The manufacturing component that bottles the liquids produced by mixing two or more liquids based on customised recipes.
- Safety and Access and Control System (ACS): Monitors and Controls presence of people in both major sections of the facility and enforces both safety and security measures according to the adopted regulations such as humans in certain areas.
- Environment Control System (ECS): Controls environmental conditions (temperature, humidity and cleanliness, as well as lighting conditions) in manufacturing and office sections of the facility, as well as control the conditions for storage of bottled products before delivery. Additionally, monitors safety of the environment in terms of presence of fire.
- Product Order System (POS): Accepts the orders from the trusted (registered) customers through an automated on-line system and launches and schedules the production automatically.

Each System has its own collection of clock domains for computation as well as GUI that is linked to help with interfacing with and showing the user.

Specifically, towards the ABS, the approach was taken to have everything centred around a single plant. This plant is the simulator of the entire ABS operation. The central plant receives signals from multiple controllers, one controller for each station of the ABS system. Once it receives these signals, the next step/changes occur within the plant. The result of these changes or next steps are communicated in channels back to the controllers so they can make the next computation and signals are sent to the ABS GUI to provide it with the information it requires to portray graphically what is occurring. The Gui itself is able to feed values back to the plant. The reason for this approach is allows for each station to be represented by a controller, each made in its own clock-domain. This allows the stations to operate independently of each other and at the same time. The plant being in the centre serves as the component that makes sure everything is done at the right timing to ensure a clean flow of operations. The controllers handle external signals from the other sub-systems so as to have the incoming data already sent to where it is being used. For a graphical representation for this, refer to the report which is included within the package alongside the compendium.

Specification differences

It must be noted that there are some deviations made from the initial specifications provided. These include:

- The ABS only makes use of one plant, as said before this is to leverage the benefit of centralizing where all the controller signals go and thus be able to manipulate timing of their operations from one place.

- The ABS will receive arrays from the environment. This is to reduce the amount of socket connections required to bring all the data required for the operation of the ABS. The result of this is that setting up before running the solution requires an extra step. It has been decided an extra short step is better than having more connections to handle.

Assumptions & Notes

Overall our design represents a large amount of work and effort. However confusion about the approach to the plant & controller model led to the final design not operating properly. We attempted to integrate both the plant and controller into the individual clock domains, so that each element is simulating its own operation. While we did start an approach of the Plant and Controller model towards the end of the project, this did not get completed.

ABS Clock Domains

Below is the detailing of all the clock domains within the ABS system, as well as a further breakdown of their signals, variables, any functional reactions as well the operation steps each one takes.

Capping Station

Description

The capping station screws the lid that has been previously placed onto the bottle tightly.

1. Waits for a bottle to arrive at position 4.
2. Clamps the bottle until the last step. Lowers the gripper.
3. When the gripper is lowered, grip the cap.
4. Twist the gripper.
5. Release the cap.
6. Untwist the gripper.
7. Raise the gripper.
8. Unclamp the bottle.

Input signals

- Signal bottleAtPos4 – Present when a bottle is at position 4
- Signal gripperZAxisLowered – Present when the gripper/capper unit is fully lowered
- Signal gripperZAxisLifted – Present when the gripper/capper unit is fully lifted
- Signal gripperTurnHomePos – Present when the gripper is at the initial position
- Signal gripperTurnFinalPos – Present when the gripper is fully turned

Output signals

- Signal cylPos5ZaxisExtend – brings the gripper down (absence of this signal will bring the gripper up)
- Signal gripperTurnRetract – untwists the gripper
- Signal gripperTurnExtend – twists the gripper
- Signal capGripperPos5Extend – Grips the cap (absence of this signal will release the cap)
- Signal cylClampBottleExtend – Clamps the bottle (absence of this signal will unclamp the bottle)

The capping station uses parallel reactions as well as liberal use of await statements to time each step. This is because multiple signals need to be emitted across multiple steps, and the signals received from the machine/plant can be used to move to the next steps.

Conveyor Belt

Description

The conveyor belt moves empty bottles from the loading zone onto the rotary table, and full bottles off the rotary table into the collection point. Bottles are shifted along the conveyor belt until they collide with the table or collector, so there is no need to immediately stop the belt. With that in mind, the conveyor belt logic is at follows.

9. Wait for the bottle at position 5 to leave the rotary table using bottleLeftPo5
10. Enable the conveyor by sustaining motConveyorOnOff
11. Check if a bottle has arrived at pos 1 using bottleAtPos1
12. Check if the bottle has left position 5 using bottleLeftPos5
13. Stop the conveyor motor

Input signals

- Signal bottleAtPos1 – Present when a bottle is at position 1 of the table
- Signal bottleLeftPos5 – Present when there is a bottle still on the conveyor past position 5

Output signals

- Signal motConveyorOnOff – Enabling/disabling the conveyor motor.

This uses a simple state machine in a single reaction. It waits for the appropriate signals in each state, while also emitting the appropriate signals for that state.

Filler

Description

Clock Domain that controls the filler station of the ABS. Each filler is it's own reaction, with some additional reactions to help with the storing of order values and timing.

Input signal(s)

- signal bottleAtPos2 – Present when the bottle is at position 2
- signal dosUnitEvac – Present when a pressure canister is at bottom
- signal dosUnitFilled – Present when a pressure canister is at top
- signal int[] order – signal holding the next order to be done

Output signal(s)

- signal valveInjectorOnOff – Turns on or off the valve injector (absence of this signal will turn off the injector)
- signal valveInletOnOff – Opens the inlet valve (absence of this signal will close the inlet)
- signal dosUnitValveRetract – brings the pressure canister to top
- signal dosUnitValveExtend – brings the pressure canister to bottom
- signal fillerReady – communicates with plant to tell if filler has completed a bottle, and is waiting for next order

Internal Signals & Variables

- signal fXdone – 4 signals to tell rest of clock domain that the corresponding filler had finished pouring.
- signal timeupX – 4 signals to signify the expiring of timers within the clock domain.
- signal timeXStart – 4 signals to send to the command to start timers.
- int Maxtime – value representing how much time it would take for pressure cylinder to fully retract
- int orderCount – value representing how many iterations of a mix must be done to complete full order
- int fXAmount – variable that holds value representing the portion that the corresponding filler will fill the bottle.

**-Please not X is a placeholder for a integer value*

Reactions / Synchronous programming

The following is detailing the operation of each reaction in the Clock Domain

Reactions 1, First Filler machine controller

1. Check if a bottle is at position 2, if not, wait until it arrives

2. Communicate to plant, that filler station is not ready for a new bottle.
3. Emit signal to start timer counter for this filler.
4. Turn on the valve injector.
5. Check if a pressure canister is at bottom, if so, bring it up to fill the bottle with liquid.
6. Wait for timer to send signal for “time up” stop pulling cylinder.
7. Turn off the injector.
8. Open the inlet.
9. Force down the pressure canister.
10. Wait until the cylinder is fully extended (i.e., at bottom)
11. Close the inlet.
12. Signify this filler is done so next one can fill its portion of bottle.
13. Wait for fourth filler to be complete.
14. Send signal to plant to signify filler is ready for next bottle.
15. Wait for bottle to be moved along to the next station of ABS.

Reactions 2-4, Remaining Filler controllers

1. Check if a bottle is at position 2 and that previous filler is done if not, wait until it arrives.
2. Communicate to plant, that filler station is not ready for a new bottle.
3. Turn on the valve injector.
4. Check if a pressure canister is at bottom, if so, bring it up to fill the bottle with liquid.
5. Wait for timer to send signal for “time up”, once detected stop pulling cylinder.
6. Turn off the injector.
7. Open the inlet.
8. Force down the pressure canister.
9. Wait until the cylinder is fully extended (i.e., at bottom)
10. Close the inlet.
11. Signify this filler is done so next one can fill its portion of bottle.
12. Wait for bottle to be moved along to the next station of ABS.

Reaction 5, Storing the Order

1. Wait for order
2. Temporarily store order signal value in local array
3. Store first element in orderCount variable
4. Store next 4 elements of order array in corresponding FxAmount variables
5. Await for filler 4 to be done, decrement orderCount variable
6. Once orderCount reaches 0, emit orderTaken to signify this order has been completed

Reaction 6, Timers

Within this reaction are 4 synchronous statements with the following operation

1. Wait for timerXStart signal
2. Sleep for proportioned time
3. Emit time up

Lid Loader

Description

The lid loader picks up a lip from the magazine to put it on the bottle. As the code was re-used from the lab, there is a manual mode, though it is not used in the project, so it will be ignored.

1. Waits until there is a request to put a lid on a bottle.
2. Swivels the arm to the extended position to move it out of the way.

3. Extends the pusher to move the lid from the magazine, then retracts it.
4. Swivels the arm to the lid.
5. Vacuums the lid so that the arm grips it.
6. Swivels the arm to the extended position.
7. Drops the lid.

Input signals

- Signal pusherRetracted – Present when pusher is retracted.
- Signal pusherExtended – Present when pusher is extended.
- Signal WPgripped – Present when the lid is gripped.
- Signal armAtSource – Present when the arm is not extended.
- Signal armAtDest – Present when the arm is extended.
- Signal empty – Present when there are no more lids in the magazine.
- Signal request – Present when there is a new request for a lid.

Output signals

- Signal pusherExtend – Extends the pusher (absence of this signal will retract the pusher)
- Signal vacOn – Turns the vacuum on (absence of this signal will turn the vacuum off)
- Signal armSource – Swivels the arm to the unextended location
- Signal armDest – Swivels the arm to the extended location

This runs through the steps one-by-one while waiting for the appropriate signals. Parallel reactions are used for the vacuum, which needs a sustained signal across multiple steps.

Rotary Table

The rotary table moves the bottles between the different stations of the ABS machine.

Input signals

- Signal tableAlignedWithSensor – Present when the positions of the index table are aligned
- Signal bottleAtPos5 – Present when the bottle is at position 5
- Signal capOnBottleAtPos1 – Present when there is a cap on bottle at position 1

Output Signals

- Signal rotaryTableTrigger – Turns the rotary table while a status of this signal is true

Running Logic

1. Wait until there is a bottle present at position 1
2. Check if there is a cap on bottle 1 with the signal capOnBottleAtPos1. If there is, wait for it to be manually removed.
3. If there is a bottle present at position 5, wait for it to be removed by the conveyor belt.
4. Turn the table by sustaining rotaryTableTrigger until the table is aligned with the sensor.

These 4 steps repeat.

Connections/Interfacing

There are connections from the ABS are to external systems, as the stations in each of the parts of the ABS are almost wholly independent.

The connections from the ABS to other systems include:

- The POS sending orders
- The ACS sending a halt signal if there are unauthorized personnel
- The ECS sending a halt signal if there is a fire

Plant & Controller

The plant models were an oversight in our design as we implemented a messy approach where the relevant code was spread through the ABS clock domains themselves. Through use of awaits and state machines, the clock domains are emulating their own operation. The Plant system that we did develop is an accurate representation of how the physical system works, providing the real-life signals that are needed to run the ABS system. As well as feeding signal inputs to the ABS, the plant is also responsible for feeding outputs to the GUI

Plants for the rotary table, conveyor, filler, and capper were made.

The rotary plant tracks the position of any bottles in the table as well as if that bottle has a cap on it or not.

The capper plant implements a one-second delay for the actions to better emulate physical reality.

Conclusion

In the end, the solution for Advantech Ltd., that was set out to be implemented using System J and Java as the two main tools was conceptualized and for the most part made. A system of system was made that overall was decentralized across the 4 biggest sub-systems, whilst having aspects of a more central design in underlying components. Justifiable changes were made as this project was developed to make operations less complex and reduce the number of components to monitor.

Appendix

ABS Diagram