University of Auckland

Department of Electrical, Computer and Software Engineering

COMPSYS 704 – Advanced Embedded Systems, 2024

Project 1 = 60% - 40% GRP and 20% IRP

Designing Advanced Embedded Software Systems - SystemJ Approach

1. Introduction

This document is a project brief for the design of an advanced embedded software system that supports and controls the operation of a high-tech manufacturing firm that produces complex and sensitive liquids packaged into bottles specified in the purchase orders of their customers. The designed system employs sensing and actuating technologies of Industrial Internet of Things (IIoT) to monitor and control the equipment, (1) workstations, which take part in physical transformations and manipulations of the workpieces in manufacturing process, and (2) transport devices that move workpieces from one to the next workstation until the workpiece becomes an instance of the final product. The order in which workstations do their operation on each workpiece is specified in a product recipe. For example a bottle filled with the same mix of the liquids is considered a product, and it differs from another bottle with different mix of liquids. The system control ensures that the workpiece uses proper transport device(s) to reach the next workstation after successfully completing operations on the current workstation, as specified in the product recipe and production plan. Each specific product recipe may require initial configuration of the production line and subsequently dynamic reconfiguration of the production line, including changes of the recipe, depending on the changes in production line resources conditions, state and availability of workstations and transport devices during production process.

Production is launched in batches of identical instances of a product described in a purchase order with a list of products and quantities (number of instances) submitted by the client to the production facility on-line. Each batch has to be completed fully before launching production of a batch of the next product from the list.

The manufacturing system is described by a specification of the core system functionality and potential additional options, which are the main inputs for the design teams to explore and make the final system specification. The design methodology has to allow decomposition of system's behaviour on subsystems/components that enable loading of a sequence of initial workpieces (empty bottles) that are subsequently filled with a mixture of liquids as the final product, closed with a cap/lid, done through a sequence of operations performed by different workstations and packed to the boxes that contain all products from the purchase order.

The whole system will be designed to certain level of abstraction and functionally simulated to allow further feasibility study by the firm. The system operation will be described by applying SystemJ design approach, which is suitable for the design of complex software systems for all kinds of execution platforms, from centralised to highly distributed. The simulation model will contain models of individual devices (workstations and transport devices and may include material/workpiece manipulators such as robots) used in the manufacturing process, as well as their individual controllers (with detailed and/or abstracted behaviours), and the overall control strategy in terms of coordination of the device controllers into overall system control that implements everything required in the purchase order and the product recipe. We will assume that all possible products belong to the same category with recipes describing also variations/differences of individual products each from the other.

2. Project Goals

The main overall goal of this project is to make a preliminary and simplified design of a distributed embedded system that is typically implemented on the infrastructure that combines Internet of Things (IoT, Edge devices, device layer), embedded computers (e.g. routers and more powerful Edge devices/gateways layer) and large computers distributed along local networks and wider Internet (up to Fog layer). The required system functionality will be designed in the first stage as a number of software (sub)systems that have clearly defined inputs and outputs to the physical world and cooperatively perform processing of the inputs to generate required outputs, effectively acting as controllers of the physical world, in our case manufacturing plant, with their ability of interoperation and integration into bigger complex systems. The designed system and its design/modelling methodology allow simulation of practically any element of the system functionality before actual deployment, without changing initially developed system (and its code), while following main ideas of Industrie 4.0 and 5.0 paradigm and one of Industrial IoT (IIoT).

As the main development tool for the system design we are going to use system-level design/programming language SystemJ, which enables modelling and design of control parts of the target systems as well as of simulated parts, e.g. plant, in the same model and top-down design methodology and bottom-up implementation supported by modelling capabilities of SystemJ. SystemJ is underpinned by Globally Asynchronous Locally Synchronous (GALS) formal Model of Computation (MoC). The powerful abstractions of SystemJ that support and utilise concurrency, reactivity, communication and synchronisation of concurrent software behaviours, pre-emptions and logical time, are crucial to enable fast and realistic system prototyping, as well as functional validation and evaluation, where all concurrent behaviours do not need to be developed to the same level of details and still composed and integrated into overall system functionality.

In addition, besides ability to describe the complex system behaviour SystemJ separates the designed functionality from the run-time system and execution platform on which it executes, thus allowing a full system functional design validation without having the whole platform available. SystemJ extends its capabilities with Java and allows use of Java for implementation and re-use of the existing Java software base. This can be further extrapolated on software developed using other programming languages via their interfaces with Java. Also, systems developed in SystemJ enable the same functional descriptions, developed in simulation, to be deployed onto any execution platform without or with a minimal source specification/code change. In case when some parts of a designed system need to operate in hard real-time, they must simply utilise a time-predictable execution platform on which SystemJ program, or its parts, run. Finally, due to formal foundations of the language and the fact it is based on GALS MoC, it promotes correct by design system development, where the designed systems can be checked and/or formally verified against some properties before deployment and thus reduce amount of the traditional software testing.

3. Context

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 physical access and security control. Also, they intend to provide and use interfaces to external business environment, such as sales channels, supply chains, logistics etc, to become Industrie 4.0 compliant.

The company envisages the use of Internet of Things technologies with Edge, Fog and Cloud computing infrastructure, illustrated in Figure 1, to implement the initial systems, which will be the basis of future extensions and development of additional systems. The immediate development requires integration of advanced automated devices (workstations, transport devices and robots) into manufacturing process in order to achieve high extendibility, resilience and adaptability of the process with dynamic reconfiguration as the ultimate goal.

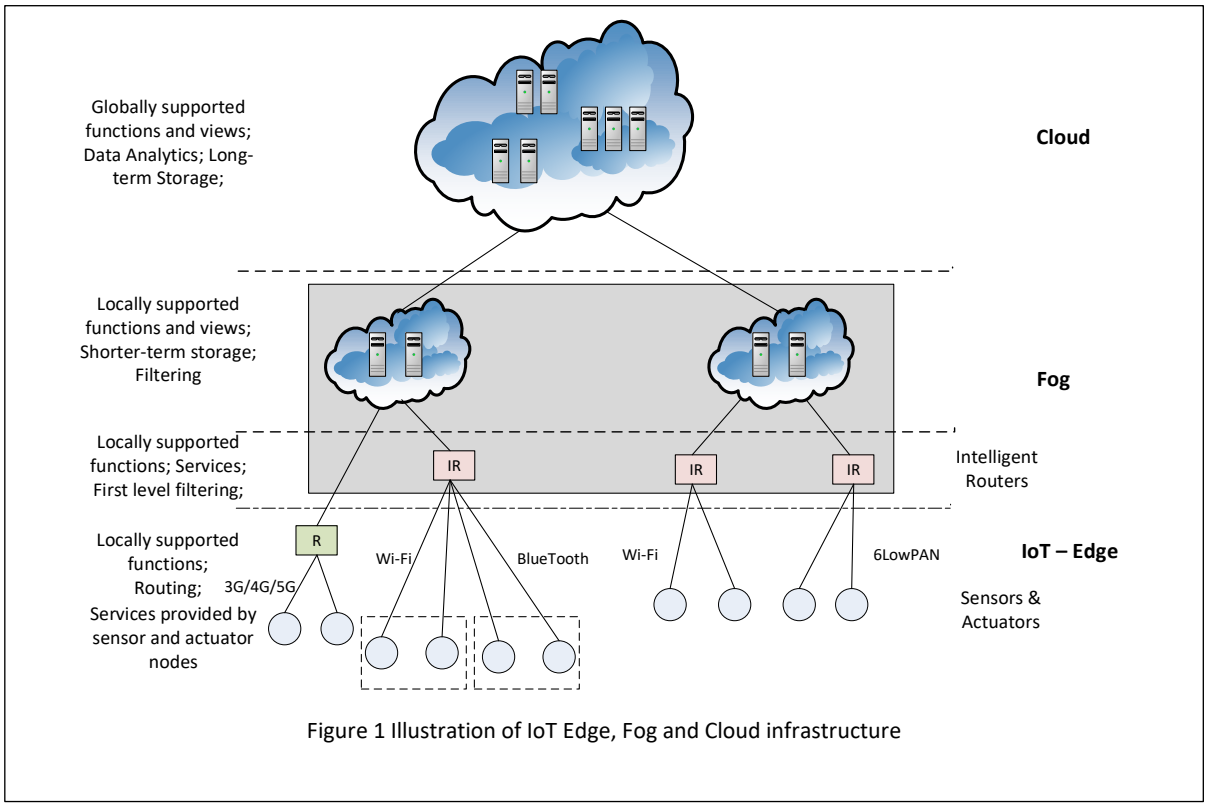


Figure 1 Illustration of IoT Edge, Fog and Cloud infrastructure

An engineering consultancy firm analysed Advantech's needs and how new systems will be integrated into the existing process of producing customised liquids according to customers' recipes and proposed the development of several systems:

1. Automated Bottling System, ABS, (manufacturing component) that bottles the liquids produced by mixing two or more liquids based on customised recipes under strictly controlled environmental conditions and keeps them in strictly controlled climate environment before shipping them to the customers
2. Environment Control System, ECS, that 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
3. Safety and Access and Control System, SACS, that controls presence of people in both major sections of the facility and enforces both safety and security measures according to the adopted regulations and monitors safety of the environment in terms of presence of fire, humans in certain areas, etc
4. Energy Consumption Optimisation System, ECOS, that monitors energy consumption of individual energy consumers with the aim to improve and eventually optimise energy efficiency of the whole facility based on presence of people in the spaces, current climate conditions indoors, time of day etc.
5. Data Analytics System, DAS, which monitors operation of individual machines and operating environment for preventative and predictive maintenance, based on the use of advanced data analytics methods
6. Product Order System, POS, which accepts the orders from the trusted (registered) customers through an automated on-line system and launches and schedules the production automatically.

In this project we focus first on the key system, automated bottling system (ABS), and its interoperation with the purchase order system (POS). For completeness of initial specification we provide rough specification of functionalities of the other systems (2-5) in Appendix 1.

A more detailed specification of system requirements, provided in Section 4, for the system that will be designed in this project is given to multiple groups/design teams to work concurrently and develop a proof of concept. The teams compete each with the other to qualify for the contract for the design of the final solution.

Each team consists of three system designers (students) who need to come up with the design within four weeks, where first week is used to create a conceptual design. The design's scope requires well-coordinated work of the team's members. Methodology of the decomposition of system's functionality on individual system components and their integration and use as the basis of future detailed system development indicates the need for consideration of component/system interfaces that is typical for System of Systems (SoS) design approach.

A detailed list of sensor and actuator nodes envisaged as the part of the infrastructure is listed in Appendix 2, although only some may be used in your project, but you may need to add additional abstract(ed) sensors to achieve project requirements. You are allowed to propose any meaningful alternatives in terms of sensing, actuation and space allocations of sensors and actuators.

4. System Requirements and Descriptions

In this section, major functional features and requirements of the target systems for are given. Further refinement of the functionality of each system is the task for the design team. Automated Bottling system (ABS), described in Section 3.1 and Purchase Order System (POS) described in Section 3.2. are the immediate target design. The described ABS is a core system that should be extended by additional functionalities as indicated in more details in Section 3.1.3.

3.1. Automated Bottling System-ABS

ABS is one of two central parts of the Advantech's manufacturing. They already have a system that uses specific microfluidics processes to produce the liquids as per customer orders ready for filling into bottles under strict conditions. The formula is a niche product, and the company is considering low-volume on-demand production. Each order, specified using an input on-line form, is translated into a recipe that is implemented to produce the required quantity of liquid with specific quality and ingredients. The recipe for each product contains information on each liquid to be filled into a bottle, order in which liquids are filled, and quantity (% of the 100% that indicate full bottle). At most **four different liquids** can be mixed to the final product. Liquids are filled from four different sources using separate dispensing subsystem installed on a rotating carousel or by providing some other physical movement (e.g. linear shifting of individual fillers). A bottle is kept at the same physical position while being filled until all liquids are filled.

Although Advantech has an existing bottling system, they decided to introduce a brand new one that integrates all necessary machines and fully automate process from placing an empty bottle on a feeding/loading end of a conveyor to producing the bottled (sealed) liquid delivered at the output end. The only machine that will be reused is the cap/lid loader which has already been studied in Lab 2. An option to be considered is also to use a robot (Baxter) to fulfil this task. Also, we will assume that there is a loading mechanism/machine that places the bottles on the conveyor of the bottling machine able to communicate with the bottling machine to avoid overloading and synchronise the loading process. A robot could be used to fulfil this and the task of unloading full bottles from the conveyor. Bottle selection and loading onto the conveyor system, unloading and sorting of bottles into separate storage based on the product type will be considered as described in Section 3.1.3.

The goal is to configure existing machines, possibly with extended functionalities, and the robot to create a working, fully automated system as illustrated in Figure 2 and as introduced in further text. Pictures of the real machines and elements of the system are provided in Appendix 3. In this scenario, bottles are made available at the bottle loading end. The bottle loader picks up an empty bottle and places it on the conveyor at the bottle loading point (left end of conveyor). The bottle moves on the conveyor until it reaches the turntable and the presence of the bottle at that point is recognised by a photo eye. The turntable rotates and moves the bottle in steps of 60 degrees; the first step moves the bottle until it is below the liquid filler. The bottle is filled, and the turntable rotates once more, until it is at the lid placement point. The lid loader picks a lid supplied from a magazine to the pick-up point and places it on the bottle. The turntable rotates one more step, where the capper then pushes and screws the lid onto the bottle. Finally, the turntable rotates one more step, where the finished bottle is taken over by the conveyor and moved to the collection point at the right end of the conveyor. The requirement is that this system in its final form can handle multiple bottles simultaneously, where multiple bottles are at different points/locations in the system at the same time (you can determine that capacity and ensure no overloading of the system occurs).

In a real system, the machines are equipped by a mix of microcomputers (programmable controllers) connected to a local network and by a local router that supports both wired (Ethernet) and Wi-Fi connections and enables further connection to Internet. However, due to the time constraints, **the requirement to the design teams** is to make a simulated model of the ABS that will include functionalities of the real machines (plant models), their controllers and simple visualisation of the state of the ABS (statuses of the machines used in and positions of bottles).

3.1.1. Scenario Description

The Automated Bottling Station (ABS) presented in Figure 2 has three major functional parts/intelligent machines: Bottling Station (BS), Lid Loader (LL) Bottle loader and Bottle Unloader, which can be Baxter robot. All the machines, together with the microcontrollers and associated sensing capabilities and software that implement their control algorithms are considered to be intelligent machines. The Lid Loader (LL) is a basic intelligent machine. The Bottling Station (BS) consists of four constituent machines, the Conveyor, Rotary Turntable, Filler and Capper. The bottle loader and unloader are additional machines that can be extended as described in Section 3.1.3 and considered part of individual small-scale project. The same is valid for the POS system.

Operation of the system is supported by a number of photo-eyes that detect the passage of bottles at certain points, as indicated in Appendix 4. The presence of a bottle in front of a photo-eye is used for synchronisation of operation of the ABS. While the Bottling Station is the key component of the ABS, the Lid Loader and Baxter Robot (if chosen for loading) are needed to provide key workpieces for completion of the production cycle that includes placing an empty bottle at the beginning of the Conveyor, filling the bottle with the supplied liquids, supplying a lid and placing it on the top of a bottle, securing the lid with the Capper and delivering a completed product at the right end of the Conveyor. A machine that collects full bottles on the exit point of the conveyor (unloader) is omitted from the figure and you can assume that there is a device/machine that performs this function and indicates that a bottle is collected from the conveyor. You can make further assumptions on the operation of individual machines, use of photo-eyes for detecting passage of the bottles and the use of automated label reader (Bar code, QR code or even camera) to identify the bottle once it is loaded onto the Conveyor.

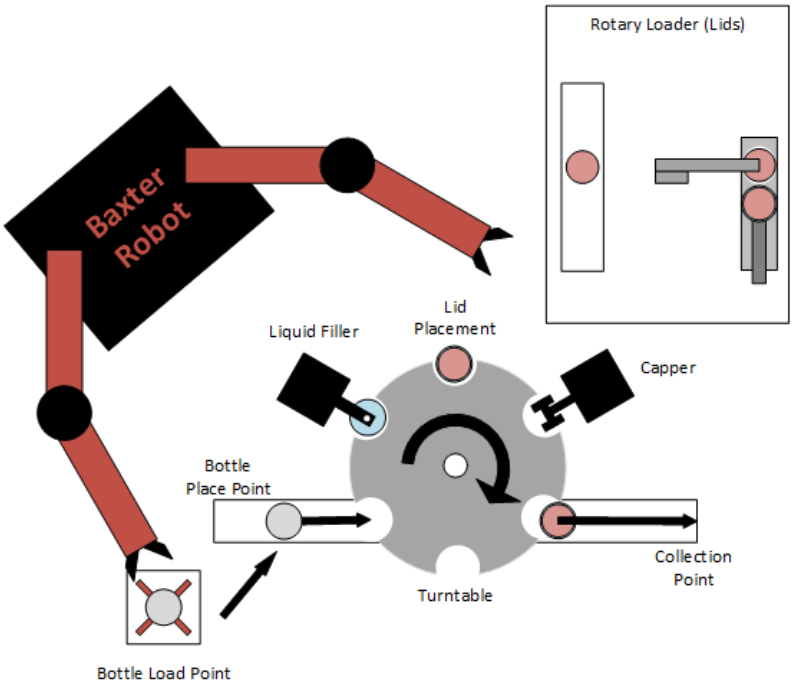


Figure 2 Simplified Overview of the Automated Bottling Station

3.1.2. Objectives for ABS

Your main task is to specify functional operation of each machine, design the model of each machine that includes machine/plant model and its control software, Controller, and integrate all these individual models into a fully functional model of ABS. The model has to be also slightly extended according to the new functional and design

requirements. If in doubt you can talk with the Advantech consultants for clarifications/approvals or you can make your own decisions that will be explained in the reports and compendiums.

The decision has been made that developing a simulation of these machine models will allow you to quickly determine how this system could be designed. The idea is to develop a functional model of each machine that captures core elements of its functionality extended with a visualisation of the machine's operation and status, and then connect that model with a Controller to illustrate Model Driven design approach. The Controller of each machine should be in a form that does not require any change when implemented on embedded microcomputer of the physical machine. To make a full simulation model of each intelligent machine, you will be using SystemJ, as demonstrated in Lab 2 and 3. During the simulation runs, your program **must show operations of the individual controllers in conjunction with plant models** by visualizing the current statuses of the plant in a GUI (e.g. position of the bottles, status of the machine like performing required operation, idle etc). Also, as a preparatory exercise and part of the teams' training, a model of the Lid (or Cap) Loader was developed first (refer to Lab 3), in order to familiarise the design team with the new tools and design approach.

Once individual intelligent machine models are developed, they will need to be integrated into the full ABS simulation model (group project). The purpose of this is to demonstrate to the company how the integral solution of the ABS would look like. At the same time, the developed integrated ABS Controller will be directly executable on both development computer and on the distributed execution platform used in real system. For more details of the envisaged tasks read Section 5 of this document.

The ABS is located in a room that requires certain global environmental and safety conditions to be satisfied. In this project you can assume two major conditions:

- No presence of people in the specified space (detected by human presence sensors) and
- The ranges for ambient conditions (temperature, humidity and light intensity)

We will simplify actions of the ABS if those conditions are not met: (a) if presence of people is detected the process stops and resumes from a predefined point (all opened bottles removed from the system before resumption of operation) and (b) if the upper/lower limits of the environmental conditions are exceeded, the same action is undertaken as in (a). You are allowed to make further assumptions.

3.1.3 Objectives for Extended ABS (EABS)

Extended ABS refers to the functionality which extends the core ABS as described in preceding sections. The design team has to decide which specific extensions will be applied to their solution and get approval from the project consultants (teaching team). The examples of those extensions are given below but not limited to the list:

- Extension of the system to deal with at least two different bottle sizes, e.g. 200 ml and 500 ml (loading and unloading)
- System ability to separate completed products (full bottles) of different sizes as in the above into their own storage as part of unloading process and sorted by batches of separate products
- Inclusion of elements of fault-tolerance (e.g. mitigation of failure of some functional parts, e.g. workstations or transport devices, inclusion of Baxter robot, see Appendix 5)
- Generalisation of the way how the recipe is presented
- Using digital twins of products to implement production process and enable tracking product's after-sales life
- Other, proposed by group members/individual students

The above as well as a POS system, described in Section 3.2 will be considered individual research projects (IRP).

3.2. Product/Purchase Order System - POS

Product order system (POS) allows the customers to launch orders directly on-line from their computers (for example, through a web-based application or some dedicated application). The order should contain information on products, which are differentiated by bottle size and liquid specification (selection of liquids that have to be mixed into each produced bottle), and volume of the product, which defines batch sizes. The order should be submitted to the

manufacturing system directly, processed and production launched for all batches. Tracking of bottles (workpieces) during production and notification to the customers on completion of the purchase order. Design teams (or individual students) have freedom of specifying and implementing a meaningful POS that seamlessly integrates with EABS and aids manufacturing process.

5. Project Tasks

Your main tasks are to make global specification and then SystemJ design of the core ABS system, with its relationship with POS.

Two major milestones are in weeks 6 (interim design also referred to as conceptual design, before study break) and 7 (final design, immediately after the study break) as described in Section 7. The milestone dates are identical for the group and individual project, GRP and IRP, respectively and they are explained in Section 7.

There are several ways of achieving the project goals and you will need to make own design decisions with the type of questions as follows:

- How will you coordinate the entire production system (EABS) and integrate it with POS?
- Will you use a centralised coordinator/orchestrator for EABS, or distribute control and to the what degree? How will you model the state of each machine and develop its plant model?
- How will you present intuitively key operations of the EABS model during a simulation so that the developer can easily modify and debug their overall control algorithms for the ABS?
- How the production will be tracked and identity of instance of the product maintained from the beginning to the end?

A possible approach to overall system architecture that includes all elements needed for simulation model and then the use of the Controller part in real system is shown in Figure 3.

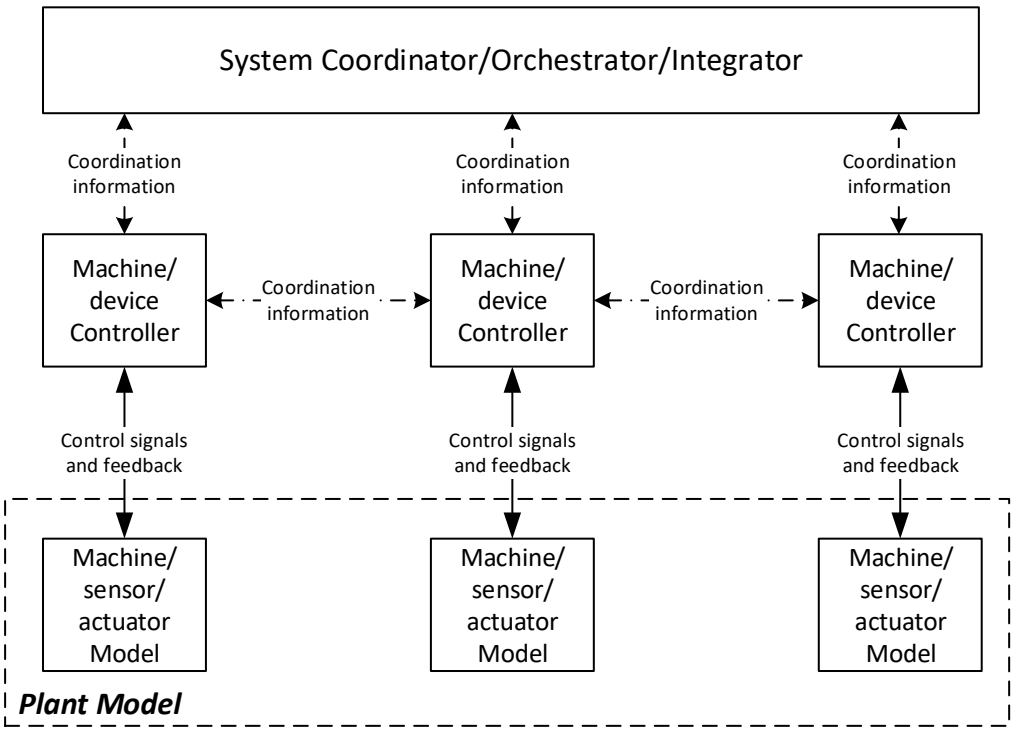


Figure 3. A possible simulation model of individual machines and integrated system

The group project is undertaken in **groups of three students**, the groups have to be created (self-organised) by the end of week 3. In case you have not agreed with others to be a part of any group, you will be allocated to a group

created by teaching staff or added to a group that has less than three students. Each group needs to be well coordinated and the individual design tasks well defined in a collaborative way and then executed by different members of the group. Any problems and issues regarding the collaborative work have to be addressed between members of the group and if not resolved indicated to the teaching staff **immediately, within a single working day**.

When making a conceptual design of your system, you have to identify major tasks, from which each student has to take one as their individual project, while the remaining tasks, which include integration of the Extended ABS (EABS) model, will be treated as part of the group project. These parts can be allocated to the individuals by the group decision.

It is expected that you will come across design challenges and ambiguities which will only be resolved by communication and interaction with your design consultants (teaching staff). The key to success will be to being proactive and not let the problems and decisions accumulate.

Each member of each group has to submit an individual peer assessment of the team members' contributions, as well as to clearly acknowledge the parts of the group work they designed. The details of peer assessment will be provided on Canvas.

The project (both GRP and IRP) is divided into two major phases (milestones) in order to ensure that the intermediate milestones and goals are discussed and identified before proceeding towards the project finalisation. These phases are also referred to as Interim and Final phase/milestone. Those phases are time-aligned for both GRP and IRP.

6. Design Tools and Execution Platform

For the real manufacturing system, each intelligent machine is associated with a microcontroller/microcomputer that can execute control algorithm software, which we refer to as the Controller. The role of each machine's Controller is to sense signals relevant for the operation of the machine, generate control signals that are issued to perform required operations and sequences of operations, provide status information to the Controllers of surrounding (other) machines, receive statuses from other machines and provide statuses, and to and communicate with the EABS control orchestrator/coordinator that ensures integrity of the EABS.

However, within this project, the systems, including all the controllers, **will only be run (simulated) on a desktop computer**. The controllers of the simulated individual machines will execute together with their plant models (i.e. the model of the intelligent machines) on a desktop (development) computer. The concept will be explained in lectures and implemented in Lab 3 on the example of the Lid/workpiece Loader.

The main design components and modules of the design will be encapsulated to clock domains and reactions of SystemJ, according to the reasons and justifications of your design principles and approach. They communicate each with the other using signals and channels as the SystemJ abstract objects , which can be of any Java type (built-in or defined). When developing the models of individual systems and their parts, you may use the approach of separating controller and plant (machines, sensors, actuators) models as shown in Figure 3. The details of this approach will be shown (are given) on the example of Lid (Cap) Loader as a machine used in ABS and will be covered in a training session (Lab 3). Also, some aspects of the core ABS are refined further and shown in Appendix 4.

The models will be developed in SystemJ and augmented, as the need be, with Java code. It is allowed to use other development tools (languages, libraries etc.) to develop visualisation and augment your SystemJ models. However, in that case you will need to provide interfacing with SystemJ models (including Java parts) and will not be subject to any support from the teaching team.

Innovative approaches to the overall project or its parts will be rewarded with the bonus marks, but the total number of marks for the GRP and IRP are capped to 40 and 20, respectively.

7. Additional Project Details

7.1. Project Milestones

7.1.1. Milestone 1 deliverables and assessment (20%, 10% group work+5% individual work)

Due date: WEEK 6 Mon, interim milestones for GRP and IRP, Interview in the same week lab time, TBA.

The deliverables (additional explanations will be given in lectures/via Canvas, if necessary):

- Conceptual design of overall system (EABS+POS). The preliminary conceptual design of the whole system provided in diagrammatic form with short explanations of its parts and overall control strategy (centralized, decentralized, mixed). Short report on completion of group and individual parts. A list of the tasks and allocation of the tasks to individual members of the design team within group part will be provided within the report. The report has to contain a diagram of the EABS (preferably on A3 page) with short descriptions (minimum 3 A4 single-spaced pages, 10 point font)
- Conceptual designs of individual (sub)projects with a diagram(s) and short report. The report has to contain a diagram of a proposed ABS extension (one of those will be POS) with short descriptions (minimum 2 A4 single-spaced pages, 10 point font)
- A demonstration of one of your simulated machines of the ABS (machine, sensor and actuator with the controller, i.e. plant model and its controller), **other than** the Lid/Cap Loader, where the device can be controlled and used by emitting signals from its controller and delivering status signals to the controller. Your plant/controller model should be able to print informative logs on a terminal to indicate its current state and operation. A simple visualisation (GUI) has to be developed that indicates how the plant model communicates with GUI.

7.1.2. Milestone 2 deliverables and assessment (40%, 30% group work+15% individual work)

Due date: WEEK 7 Mon, see on Canvas, Final deliverables for GRP and IRP, Demo/Interview in the same week lab time, TBA

As a final task, you will complete the simulation models of the integrated EABS and POS. You will need to develop the models in SystemJ that will accurately represent the operation of the designed systems. Your simulation environment must be able to clearly present a current state of the system operation (e.g. via a GUI visualisation and additional terminal printings where appropriate) to the designer so that they can validate the correct operations of their designed system.

The final deliverables include:

- The final design designed systems in diagrammatic form (based on your conceptual design from Milestone 1) accompanied with a short report (minimum 8 A4 single-spaced pages, 10-point font) for GRP, and separate diagrammatic representation and short report (minimum 4 A4 single-spaced pages, 10-point font) of the IRP designed system.
- Demonstration of the integrated system (EABS, POS) designed using SystemJ (in the Lab) in week 7, time/venue TBA (lab time) that includes interviews with each member of team.
- Short presentation slides (not more than 10) to present to the design consultants your solution for the simulation model of the overall system and individually designed parts.
- A compendium (describing your design approach, design decisions and description of the implementation) that differ from the initial specification given in this document.
- Short descriptions and documentations (you can use pseudo-codes and diagrams) for all controllers, plant models, and system integration accompanied with programs (code) and instructions how to run it
- A single zip compressed file with the solutions, group work report, and readme file (PDF) that gives instructions how to run your simulation model.
- A single zip compressed file with the solutions of the individual work and compendium with the code and explanations how to run it
- Some further details will be provided/agreed separately.
- A confidential peer assessment (TBA).

7.2. Team/Group Issues

Occasionally, issues will arise between group members. If this does happen, please raise the issue with the teaching staff immediately or *as early as possible* so an amicable resolution can be reached.

8. Readings

1. Salcic, Z. & Malik, A. (2010), SystemJ Technology – Programming Without Borders, White paper, ECE Department, University of Auckland
2. H. Park & Z. Salcic: Designing Software Systems Using SystemJ, (2019) Embedded Systems Research Group, ECSE Department, University of Auckland, available via Canvas for COMPSYS 704
3. Malik, A., Salcic, Z., Roop, P. S., & Girault, A. (2010). SystemJ: A GALS language for system level design. Computer Languages, Systems and Structures, 36 (4), 317-344. doi:10.1016/j.cl.2010.01.001
4. Malik, A., Salcic, Z., Chong, C. & Salman, J. (2012).System-level approach to the design of a smart distributed surveillance system using SystemJ, ACM Transactions on Embedded Computing Systems (TECS), vol. 11, Issue 4, doi:10.1145/2362336.2362344
5. Dwi Atmojo U, Salcic Z, Wang KI-K, Park H. (2014). System-level approach to the design of ambient intelligence systems based on wireless sensor and actuator networks, Journal of Ambient Intelligence and Humanized Computing, Springer, DOI 10.1007/s12652-014-0221-3
6. S. S. Setty, H. Yaqoob, A. Malik, I-K. Wang, Z. Salcic, and H. Park (2015) A Unified Framework for the Design of Distributed Cyber-Physical Systems – Industrial Automation Example, ICIEA 2015
7. Dwi Atmojo, U., Salcic, Z. & Wang, I-K. (2015), Extending SOSJ Framework for Reliable Dynamic Service-oriented Systems, Service Oriented Computer Architectures, SOCA 2015
8. Park H., Salcic, Z. Wang, K. I.-K., Atmojo, U. D., Sun, W.-T. and Malik, A. (2013) A New Design Paradigm for Designing Reactive Pervasive Concurrent Systems with an Ambient Intelligence Example," in ISPA, 2013 IEEE 11th International Symposium, Melbourne, Australia
9. Salcic, Z., & Malik, A. (2013). GALS-HMP: A heterogeneous multiprocessor for embedded applications. ACM Transactions on Embedded Computing Systems, 12(1s). doi:10.1145/2435227.2435254
10. Salcic Z, Park H, Teich J, Malik A, Nadeem M , NoC-HMP: A Heterogeneous Multicore Processor for Embedded Systems Designed in SystemJ, ACM Transactions on Design Automation of Embedded Systems, 2017, Volume 22 Issue 4, June 2017
11. Salcic, Z., Dwi Atmojo, U., Park, H., Chen, A. -Y., & Wang, K. -K. (2017). Designing dynamic and collaborative automation and robotics software systems. IEEE Transactions on Industrial Informatics. doi:10.1109/TII.2017.2786280
12. Sorouri, M., Patil, S., Salcic, Z., & Vyatkin, V. (2015). Software composition and distributed operation scheduling in modular automated machines. IEEE Transactions on Industrial Informatics, 11(4), 865-878. doi:10.1109/TII.2015.2430836
13. Dwi Atmojo U, Salcic Z, Wang KI-K. (2018) Dynamic Reconfiguration and Adaptation of Manufacturing Systems Using SOSJ Framework, IEEE Transactions on Industrial Informatics 14(6):2353-2363
14. Dwi Atmojo U, Salcic Z, Wang KI-K & Vyatkin, V. A (2019) Service-Oriented Programming Approach for Dynamic Distributed Manufacturing Systems, IEEE Transactions on Industrial Informatics
15. Kang, O, Salcic Z (2023) Towards Interoperability of Industrial Automation Systems Using SystemJ and TwinCAT, 18th Conference on Industrial Electronics and Applications (ICIEA)

Academic integrity notice

The University of Auckland will not tolerate cheating, or assisting others to cheat, and views cheating in coursework as a serious offence. The work that a student submits for grading must be the student's own work, reflecting his or her learning. Where work from other sources is used, it must be properly acknowledged and referenced. This requirement also applies to sources on the world-wide web. A student's assessed work may be reviewed against electronic source material using computerised detection mechanisms. Upon reasonable request, students may be required to provide an electronic version of their work for computerised review.

Appendix - 1 Functional Specification of Other Systems

A1.1. Environment Control System - ECS

Environment Control System (ECS) has several goals. First, it has to maintain temperature and humidity at the target levels in each zone. Here, we assume that the same temperature is to be maintained for zones 1 and 7, zones 2 and 3 and zones 4, 5 and 6, therefore there are three actuators (air-conditioners in manufacturing section) and heaters/fans in office section. That means each zone environmental conditions (temperature and humidity) have to be monitored by the sensor nodes and the current measurements have to be taken into account when turning ON/OFF air conditioners (with the set temperature), heaters and fans. The decisions on the use of air conditioners, heaters and fans are based not only on the current measurements indoors, as well as on time of the day (work hours vs hours when there are no employees in the office or manufacturing section), occupancy of the spaces and other conditions that need to be taken into account. The ECS also has responsibility to monitor and control lighting conditions (light intensity) in each of the zones, depending on the time of the day and occupancy of the spaces. A strategy has to be developed to automatically turn the lighting elements (one per each zone) with the required intensity depending on the presence of people, use of timers and time of the day/week. A part of the environment control system is dedicated to detection of fire/smoke in the premises and control of the conditions within the scope of fire extinguishing system, as well as of alerting the personnel of the situation, issue of instructions what to do in case of fire and communication with the fire service.

A1.2. Safety and Access Control System – ACS

Access Control System (ACS) controls movement of the personnel in the space. First, it authorises entry/exit from the Facility by the use of access cards and biometric information, the latter being only an option. Also, all personnel and visitors are issued a carry-on badge that enables identification and location of the personnel by using RF-based localisation system. Different access rights are assigned for Office section and Manufacturing section. Presence of personnel in the Manufacturing section is strictly controlled. Also, presence of personnel in the zones that directly associated with the manufacturing (ABS and around ABS) is detected and bottling process suspended as there must be no humans in the vicinity of the machines and the bottling process. Virtual boundaries, implemented using laser beams, are enforced around the ABS.

A1.3 Energy Consumption and Optimisation System - ECOS

Upon completion of the aforementioned systems, another system that may use the already established infrastructure (sensing, actuation) and results of operation of the previously developed systems, needs to be added. It is Energy Consumption and Optimisation System, ECOS. The system has the goal of monitoring operational conditions of all machines, presence of personnel and environmental conditions, to make the decisions regarding energy usage (electricity in our case). The system can directly collaborate with all abovementioned systems to make decisions to turn ON and OFF some of the consumers (air conditioners, heaters, fans, lighting units and machines). As the first approximation we will assume that individual electricity consumers have constant load when they are switched on. As the Facility has access to the energy supplied from power utility, as well as local solar energy generation and batteries that can be recharged, the optimisation of energy usage becomes possible. We assume that all actuation devices can be controlled through the network by the ECOS, as well as that ECOS has access to the necessary information from all three other systems. Thus, some principles of System of Systems (SoS) approach need to be applied in this case.

A1.4 Data Analytics System (DAS)

As the Facility will generate large amount of data during its operation, data will be stored and used for longer term decisions and providing feedback to all existing systems, for example for fine-tuning processes, establishing better systems settings, optimisations of energy consumption etc. Advantech envisage establishment of Data Analytics System (DAS) that will help close some of the loops for decision making and tuning of processes. As the volume of the produced data is usually increasingly-large and geographically-distributed, the data processing analytics paradigms should be highly efficient and scalable. We envisage the use of data processing frameworks such as MapReduce for batch (off-line) processing of data collected from the sensors, as well as framework for stream data processing (on-line/real-time) such as Apache Spark.

Appendix 2 Working with the simulated and real sensors and actuators – list of some common functions

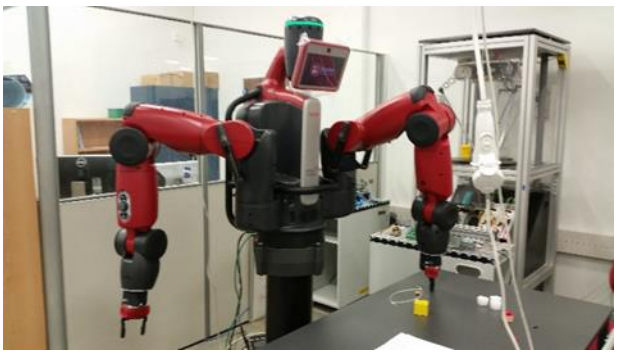
Table A2.1 Examples of sensors and actuators used in manufacturing facility (use these and/or add new as you need)

Device name	Functionality	Associated signals and comments
Sensors		
Temperature sensor	Measures the temperature in vicinity of the device and delivers readings in degrees Celsius	From a sensor node
Humidity	Measures the humidity in vicinity of the device and delivers readings in %	From a sensor node
Light intensity	Measures the light intensity in vicinity of the device and delivers readings in lumens	From a sensor node
Human presence	PIR – Passive Infrared – Indicates movement of persons in its range	From a sensor node
Proxy Card Reader	Reads information on an ID card	From a sensor node
Smoke alarm	Sets off when smoke detected Health status can be checked upon request	From a sensor node
Location of a sensor node	Reads the location of the wearable node with the given ID	From PC where the location system runs
RSSI of WSN node (if wireless sensor node used)	Reads RSSI values of the nodes (with IDs) found in vicinity	From WSN node
Actuators		
Lightbulb	Can be switched ON with a given light intensity	Wi-Fi node
Power plug	Can be switched ON/OFF	Wi-Fi node
Heater	Can be switched ON/OFF	Via Wi-Fi enabled power plug
Electric fan	Can be switched ON/OFF	Via Wi-Fi enabled power plug
AC device	Can be switched ON/OFF with the target temperature setting	Via power plug/option from/to Wi-Fi node
Siren	Can be switched ON/OFF	From associated computer node
Door/gate lock	Can be LOCKED/UNLOCKED	From associated computer node
Window/blinds opener	Can be closed or opened to certain position	From associated computer node (given opening)

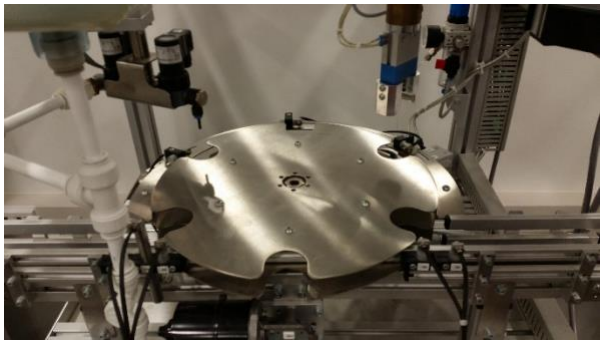
When using these signals give them appropriate name in SystemJ.

Also, you may change the input and output signals and add additional input and output signals if they are necessary for your specific implementation. This allows introduction of abstract interfaces of additional sensors and actuators.

Appendix 3 – Workstations of ABS

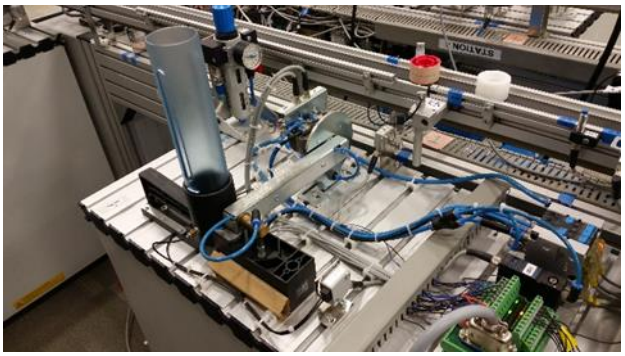


Baxter Robot



Bottling Machine and its parts

Bottle with Cap



Cap Loader

Figure A1.1 Functional parts of the core ABS

Appendix 4 - The core bottling machine

The bottling machine is designed to fulfil the following four tasks:

1. Send bottles to position 1 via the conveyor,
2. Fill the bottles with the predefined amount (depending on the type of bottle recognised).
3. Place a lid/cap on the bottle.
4. Screw and press caps onto the bottles.
5. Send bottles via conveyor to the next stage (e.g. labelling or packaging).

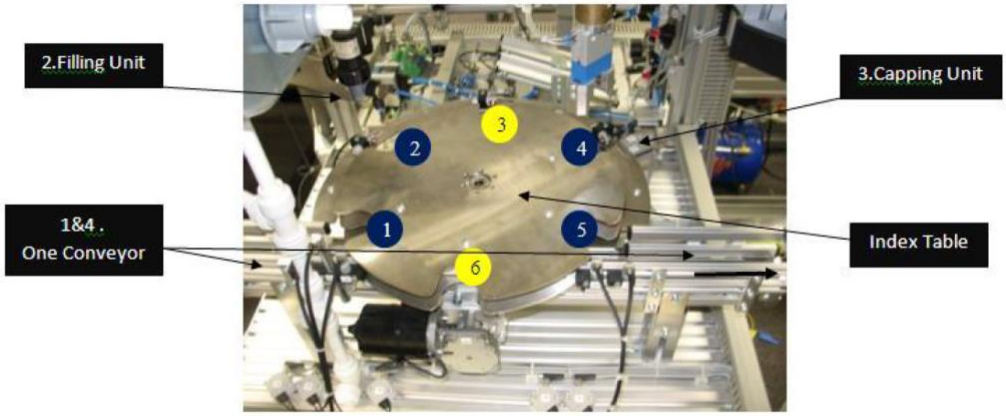


Figure A4.1. Bottling machine main components

As it is shown in the Figure A4.1, the index table has 6 positions (position 6 is unused). In order to shift bottles to the next station, each time the index table makes a 60 degree rotation.

There is a set of actuators and sensors for this machine (as listed in Table A4. 1):

A4.1 Actuators (motors, pneumatic cylinders, electric valves)

- A conveyor belt: to bring empty bottles in and at the end take the capped bottles out (at positions 1 & 5). See also Figure A4.2.
- A rotary table (index table): to guide and move bottles through different machine stations (at position 1,2, 3, 4 & 5)
- A filling station: to fill bottles with a certain volume of liquid (at position 2): including inlet valve, pressure canister and injection valve
- Lid/cap loader (refer to Lab 3)
- A capping station: to keep bottles tight and cap them (at position 4): including three pneumatic cylinders for z axis movement, rotation and gripper

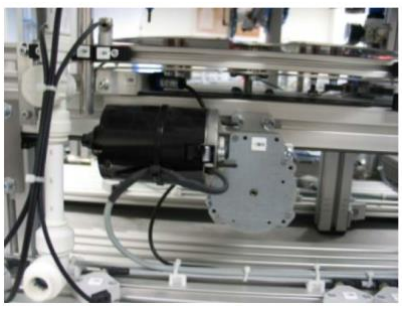


Figure A4. 2. Sample of an actuator (Gearing motor for conveyor)

Table A4. 1. Sensors and actuators associated with each index table position

Position	Sensors	Motor/Pneumatic Cylinder/valve
1	bottleAtPos1	-
1	capOnBottleAtPos1	-
2	-	Inlet valve
2	-	Pressure canister
2	-	Injection valve
2	bottleAtPos2	-
2	Pressure canister high	-
2	Pressure canister low	-
4	-	Gripper z axis movement
4	-	Gripper rotation
4	-	Bottle clamp
4	bottleAtPos4	-
4	Gripper at 0 degree (initial pos)	-
4	Gripper at 270 degree (final pos)	-
4	Gripper at top	-
4	Gripper at bottom	-
4	Gripper close	-
4	Bottle clamp open	-
4	Bottle clamp close	-
5	bottleAtPos5	-
5	bottleLeftPost5	-
Conveyor	-	Conveyor motor
Rotary table	-	Rotary table motor
Rotary table	Rotary table at position	-

A4.2 Sensors

They provide required information about machine's condition and operation)

- 4 sensors to identify presence of bottle in each of the 4 positions (Positions 1, 2, 4, and 5)
- 1 sensor to detect whether there is a cap on bottle at position 1
- 1 sensor to show whether a bottle has successfully left position 5 (placed next to the position 5)
- 1 sensor to identify that the index table is in the right position (i.e. to check whether the bottle is in line with the sensor)
- 8 other sensors to show the initial and final positions of pressure canister, bottle clamp, gripper's rotary and z axis movement cylinder, and one sensor to detect the gripper extension (one way cylinder)

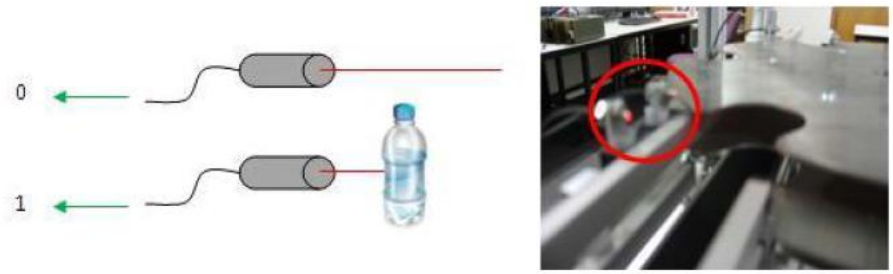


Figure A2.3. Sample of a sensor used in the machine

A4.3 Examples of SystemJ clock-domains for controlling the bottling machine

There should be **at least four** clock-domains that need to be implemented: conveyor clock-domain, rotary clock-domain, capper clock-domain, and filler clock-domain. Each of them has a predefined set of input and output signals to control the corresponding component of the bottling machine. Details of the interface signals and clock-domain operations are described as follow.

Conveyor clock-domain

- Input signal(s)
    - bottleAtPos1 – Present when a bottle is arrived at position 1 of the index table.
    - bottleLeftPos5 – Present when there is a bottle, which left from position 5 of the index table, still on the conveyor
- Output signal(s)
    - motConveyorOnOff – Enabling/disabling the conveyor motor.

Basic operation of the conveyor clock-domain would be:

1. Wait until there are no bottles on the far end of conveyor via signal bottleLeftPos5
2. Enable the conveyor motor
3. Check whether the bottle is arrived at position 1 via signal bottleAtPos1
4. Wait until the bottle is successfully left position 5 via signal bottleLeftPos5
5. Stop the conveyor motor

Rotary Table clock-domain

- Input signal(s)
    - tableAlignedWithSensor – Present when the positions of the index table are aligned with the sensors (photo-eyes).
    - bottleAtPos5 – Present when the bottle is at position 5
    - capOnBottleAtPos1 – Present when there is a cap on bottle at position 1
- Ouput signal(s)
    - rotaryTableTrigger – Turns the rotary table while a status of this signal is true

Basic operation of the rotary clock-domain would be:

1. Check if a bottle at position 1 has already a cap on it. If so, wait until the bottle is manually removed from the table.
2. Rotate the table 60 degrees via sustaining signal rotaryTableTrigger for about 0.5 second
3. Ensure the table positions are aligned with the sensors via tableAlighnedWithSensor

Capper clock-domain

- Input signal(s)
    - bottleAtPos4 – Present when a bottle is at position 4
    - gripperZAxisLowered – Present when the gripper/capper unit is fully lowered
    - gripperZAxisLifted – Present when the gripper/capper unit is fully lifted
    - gripperTurnHomePos – Present when the gripper is at the initial position
    - gripperTurnFinalPos – Present when the gripper is fully turned
- Output singal(s)
    - cylPos5ZaxisExtend – brings the gripper down (absence of this signal will bring the gripper up)
    - gripperTurnRetract – untwists the gripper
    - gripperTurnExtend – twists the gripper
    - capGripperPos5Extend – Grips the cap (absence of this signal will release the cap)
    - cylClampBottleExtend – Clamps the bottle (absence of this signal will unclamp the bottle)

Basic operation of the capper clock-domain would be:

1. Check if a bottle is at position 4, if not, wait until it arrives
2. Clamp the bottle and lower the gripper
3. Wait until the gripper is lowered, and grip the cap
4. Twist the gripper
5. Release the cap
6. Untwist the gripper

7. Raise the gripper
8. Unclamp the bottle

Filler clock-domain (for a single liquid filler, need to be expanded to four individual liquids fillers)

- Input signal(s)
  - bottleAtPos2 – Present when the bottle is at position 2
  - dosUnitEvac – Present when a pressure canister is at bottom
  - dosUnitFilled – Present when a pressure canister is at top
- Output signal(s)
  - valveInjectorOnOff – Turns on or off the valve injector (absence of this signal will turn off the injector)
  - valveInletOnOff – Opens the inlet valve (absence of this signal will close the inlet)
  - dosUnitValveRetract – brings the pressure canister to top
  - dosUnitValveExtend – brings the pressure canister to bottom

Basic operation of the filler clock-domain would be:

1. Check if a bottle is at position 2, if not, wait until it arrives
2. Turn on the valve injector
3. Check if a pressure canister is at bottom, if so bring it up to fill the bottle with liquid
4. Wait until the cylinder is fully retracted (i.e. at top)
5. Turn off the injector
6. Open the inlet
7. Force down the pressure canister
8. Wait until the cylinder is fully extended (i.e. at bottom)
9. Close the inlet

In 2024 project you are required to create mix of up to four liquids in a bottle with the proportion of liquids specified in a purchase order. For that purpose you will need to create a new filler system with four fillers that operate as specified above and make sure that there will be no liquid overflow (hint: add signals that allow this; you may assume the liquids are filled one after another). This will result in "your design" of the liquid filling machine.

Note: You are not going to design algorithms for bar/QR-code and image capture, processing and recognition, but make a conceptual design that as a result of image capture recognises type of the bottle and transmits that information to the filler controller before the filling starts. The filler then adjusts amount of the liquid filled to the type of the bottle. Also, if the Baxter is not used, then the loaders of workpiece and parts will be used to place them input them into the process.

Appendix 5 – Baxter Service Robot

A5.1 Functional operation and main characteristics of Baxter

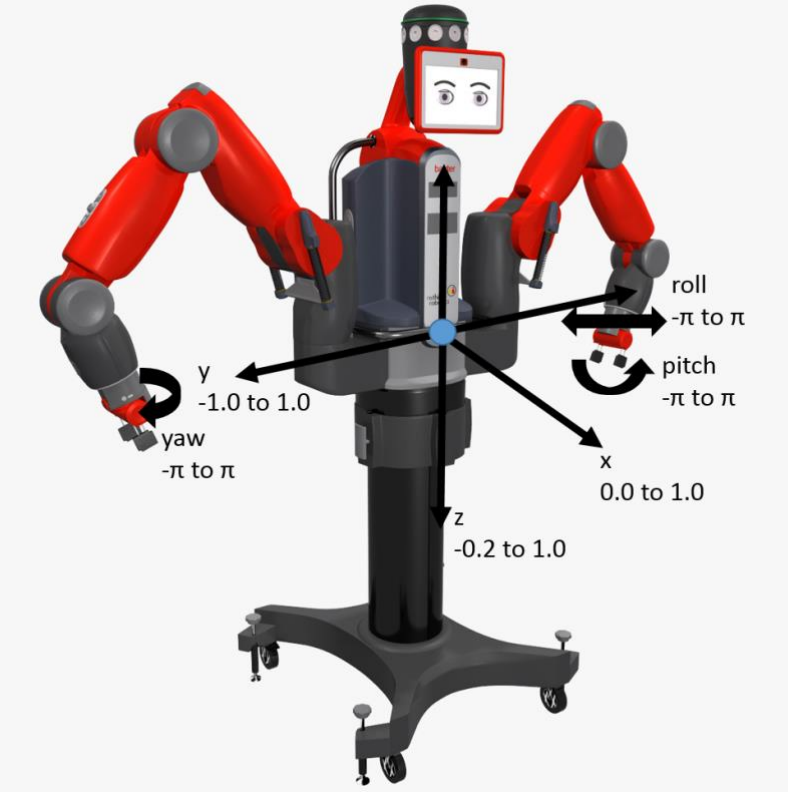
This specification is given for completeness of the ABS if you decide to include it into your design. The Baxter is an optional transport device. In case of not using Baxter an abstracted machine that loads empty bottles, and removes full bottles from the conveyor need to be integrated into the simulation model. Alternatively you can abstract Baxter arms operation with the commands as described below in this section. The presence and absence (removal) of the bottle on the conveyor needs to be detected by using sensors.

The Baxter robot is used to do two tasks:

1. Load empty bottles into the bottling station
2. Place caps onto the bottles

It achieves this by using two robot arms, each with 7 degrees of freedom. Interacting with Baxter is actually reasonably simple, as most of the difficult math, particularly the path planning and inverse kinematics, has been abstracted away for you. Essentially, each arm:

- can open or close the gripper
- can move to a specified end pose, where the end pose is described as x, y, z, roll, pitch, yaw of the end effector. These values are abstracted by a set of alphabet letters that represents fixed positions the arms can move to.



A5.2.Clock-domains for controlling Baxter

There should be **at least two** clock-domains that need to be implemented: bottle loading clock-domain and bottle removing clock-domain, which are associated with two arms of the Baxter. Each of them has input and one output signals to control the corresponding arm of the Batxer.

Bottle loading clock-domain controls the loading arm of the Baxter by taking the empty bottles from a loading table to the loading end of the conveyor.

- Input signal(s)
  - [pure] bottleAtPoint – Present when a bottle is on the loading table
  - [pure] CMDfb – Present when the arm has reached its destination
- Output signal(s)
  - [valued (String object)] CMD – Sends a command to the Baxter

Basic operation of the bottle clock-domain could be:

1. Check if a bottle is on the loading table, if not, wait until it arrives
2. Lower the arm slightly to the right height (Position B)
3. Close the gripper
4. Move the arm to the turntable, position 1 (Position C)
5. Lower the arm to the right height (Position D)
6. Open the gripper
7. Move the arm back to the pickup location (Position A)

Similarly, you can abstract the operation of the other arm that removes (unloads) full bottles at the other end of the conveyor. In a real ABS implementation, there is a Python server running on the Baxter PC that can receive and execute a set of commands for controlling the robot. Commands can be sent to the server as:
function_name component_name arguments with a space between each word. In SystemJ, the command is encoded in a String object and transmitted via valued output signal. In this project you only need **two** commands to control the gripper and the robot arms as shown below:

Command(arguments)	Description
limb_moveto(component_name, pos)	Moves the arm to the specified position
limb_gripper(component_name, value)	Opens or closes the gripper

Command 1.

`limb_moveto(component_name, pos)`

Example:

```
CD(output String signal CMD; input signal CMDfb;)->{
      emit CMD("limb_moveto left_limb B"); // Moves the arm to the position B
      await(CMDfb);           // Wait until the arm has reached position B
      // Continue operation
}
```

Command 2.

`limb_gripper(name, value)`

Set the value of the gripper (open or close, 1 or 0).

Example:

```
CD(output String signal CMD; input signal CMDfb;)->{
      emit CMD("limb_gripper left_limb open");  // Opens the gripper
      await(CMDfb);                        // Wait until the gripper has closed
      // Continue operation
}
```

In project you can abstract Python server by a simple plant/robot model that communicates by aforementioned signals with the controller.