

DEGREE PROJECT FOR MASTER OF SCIENCE WITH SPECIALIZATION IN ROBOTICS
DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY WEST

Applying digital twin technology in education and research

- A case study of PTC automation line

Abdlkarim Alsaleh



A THESIS SUBMITTED TO THE DEPARTMENT OF ENGINEERING SCIENCE
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE WITH SPECIALIZATION IN ROBOTICS
AT UNIVERSITY WEST
2021

Date:	June 8, 2021
Author:	Abdlkarim Alsaleh
Examiner:	Kristina Eriksson
Advisor:	David Simonsson, University West
Programme:	Distance Master in Robotics and Automation
Main field of study:	Automation with a specialization in industrial robotics
Credits:	120 Higher Education credits
Keywords:	Digital Twin, Visual Commissioning, Automation, Academia
Template:	University West, IV-Master 2.10
Publisher:	University West, Department of Engineering Science S-461 86 Trollhättan, SWEDEN Phone: + 46 520 22 30 00 Fax: + 46 520 22 32 99 Web: www.hv.se

Summary

This thesis work investigates the possibilities and limitations of using digital twin technology to create virtual automation lines which can be used in education and research to conduct automation labs virtually. The PTC automation line at University West has been used as a case study in this thesis. The digital twin created in this work consists of three key parts: a virtual model of the automation line created in Visual Components Premium 4.2, system control (PLC-control program) created in TwinCat 3, and a Beckhoff ADS communication protocol that connects the virtual model with the PLC program.

Using a virtual model of industrial-like lab equipment in place of a real system can bring several benefits. It can increase visibility and safety in the system. It can also increase the accessibility of the system. Conducting virtual labs and experiments can also help in reducing the total cost of the system. The virtual twin of the automation line built in this work can be used to help the users to conduct automation labs and experiments virtually and to test their PLC programs offline.

Preface

This thesis work was carried out in the spring of 2021 as a part of my study in the master program in automation and robotics (distance, 2 years) at University West in Trollhättan, Sweden. First of all, I would like to thank my supervisor David Simonsson and my examiner Kristina Eriksson for providing me with all the needed help and support during the project. I would also like to thank Gabriel Sebastian from the division of production systems at University West for all his help during this work.

Affirmation

This master degree report, *Applying digital twin technology in education and research*, was written as part of the master degree work needed to obtain a Master of Science with specialization in Robotics degree at University West. All material in this report, that is not my own, is clearly identified and used in an appropriate and correct way. The main part of the work included in this degree project has not previously been published or used for obtaining another degree.



Signature by the author

Abdulkarim Alsaleh

June 8, 2021

Date

Contents

Preface

SUMMARY	III
PREFACE	IV
AFFIRMATION	V
CONTENTS	VI
SYMBOLS AND GLOSSARY	IX

Main Chapters

1 INTRODUCTION	1
1.1 BACKGROUND	1
1.2 PROBLEM FORMULATION	1
1.3 AIM AND RESEARCH QUESTION	2
1.4 DELIMITATION	2
1.5 OUTLINE	2
2 RELATED WORK	3
2.1 DIGITAL TWIN	3
2.1.1 <i>Digital twin concept</i>	3
2.1.2 <i>Digital twin and cyber-physical system</i>	3
2.1.3 <i>DT dimensions</i>	4
2.2 DIGITAL TWIN ENABLING TECHNOLOGIES	4
2.2.1 <i>Physical-world enabling technologies</i>	4
2.2.2 <i>Virtual model enabling technologies</i>	4
2.2.3 <i>Data management enabling technologies</i>	5
2.2.4 <i>Services enabling technologies</i>	6
2.2.5 <i>Connections enabling technologies</i>	6
2.3 DT LEVELS OF INTEGRATION	6
2.3.1 <i>Digital model</i>	6
2.3.2 <i>Digital shadow</i>	6
2.3.3 <i>Digital twin</i>	6
2.4 MODELLING	7
2.5 SIMULATION	7
2.5.1 <i>Simulation vs emulation</i>	7
2.6 INDUSTRIAL AUTOMATION	8
2.6.1 <i>Controller</i>	9
2.6.2 <i>Industrial communication</i>	10
2.7 COMMISSIONING	10
2.7.1 <i>Virtual commissioning (VC)</i>	11
2.7.2 <i>Virtual commissioning stages</i>	11
2.7.3 <i>Virtual commissioning workflow</i>	12
3 METHOD	13

3.1	DATA COLLECTING	13
3.2	DATA PROCESSING.....	13
3.2.1	<i>Physical device modelling</i>	14
3.2.2	<i>Logical device modelling</i>	14
3.2.3	<i>System control modelling</i>	14
3.2.4	<i>Communication</i>	15
3.3	RESULT EVALUATION	15
4	DESIGN & IMPLEMENTATION	16
4.1	AUTOMATION LINE SPECIFICATION	16
4.2	AUTOMATION LINE WORKFLOW	17
4.3	MODELLING	18
4.3.1	<i>Physical model building</i>	18
4.3.2	<i>Logical model building</i>	19
4.3.3	<i>System control building</i>	23
5	RESULTS AND DISCUSSION	29
5.1	RESULT ANALYSIS	29
5.2	DISCUSSION	30
6	CONCLUSION	31
6.1	FUTURE WORK AND RESEARCH	31
6.2	CRITICAL DISCUSSION.....	31
6.3	GENERALIZATION OF THE RESULT	32
7	REFERENCES.....	33

List of Figures

Figure 1:	DT three-dimensional model. Adopted from [4].	4
Figure 2:	DT five-dimensional model. Adopted from [4].	5
Figure 3:	Classifying of DT enabling technologies. Adopted from [13].	5
Figure 4:	Different levels of the digital twin. Adopted from [6] and [8].	6
Figure 5:	Automation levels. Adopted from [27] and [29].	8
Figure 6:	Productivity-flexibility relation in different automation systems. Inspired by [26]	9
Figure 7:	PLC internal structure. Adopted from [34].	10
Figure 8:	Virtual commissioning development stages. Adopted from [45].	11
Figure 9:	Model configurations applied in commissioning. Adopted from [47].	12
Figure 10:	Virtual commissioning workflow. Adopted from [48].	12
Figure 11:	Data processing layout. Adopted from [48].	14
Figure 12:	Automation line layout.....	16
Figure 13:	Layout of Automation line communication. Adopted from line specifications.....	17
Figure 14:	State diagram of an assembly station.	18
Figure 15:	An assembly station.....	19
Figure 16:	Modelling behaviours in Visual Components.	20
Figure 17:	Part of the global Python script "helper.py".	20
Figure 18:	Behaviours and features of the AGV.	21
Figure 19:	AGV Python scripts.	22
Figure 20:	Behaviours added to all robots.....	22
Figure 21:	Structure of the PLC program.....	23
Figure 22:	Global variable list.	Fel! Bokmärket är inte definierat.
Figure 23:	Code inside the function "FindCode".	Fel! Bokmärket är inte definierat.
Figure 24:	Flowchart of the control programs.	Fel! Bokmärket är inte definierat.
Figure 25:	HMI used to test the robot-level function blocks.....	Fel! Bokmärket är inte definierat.

Figure 26: Connected variables between Visual Components and TwinCat.**Fel! Bokmärket är inte definierat.**

Figure 27: Part of the sub-program used to test the function blocks.**Fel! Bokmärket är inte definierat.**

Appendices

APPENDIX A: ROBOT-LEVEL FUNCTION BLOCKS

APPENDIX B: MAIN CONTROL PROGRAMS

APPENDIX C: SIMULATION VIDEO LINK

Symbols and glossary

AGV	Automated Guided Vehicle: A mobile robot used mostly in industrial to transport production materials.
CAD	Computer-Aided Design
CPS	Cyber-Physical System
DCS	Distributed Control System
DT	Digital Twin
EPR	Enterprise Resource Planning
FBD	Function Block Diagram: A graphical PLC programming language.
HiL	Hardware in the Loop
HMI	Human-Machine Interface: A human interface use mainly in automation to connect the user to the system
IL	Instruction List: A text-based PLC programming language.
I/O	Input/Output
IoT	Internet of Things
LD	Ladder Diagram: A graphical PLC programming language.
MES	Manufacturing Execution Systems
MiL	Model in the Loop
OPC UA	Open Platform Communications United Architecture: A client-server communication protocol used mainly in automation to connect the machines and controllers
PiL	Process in the Loop
PLC	Programmable Logical Controller
POU	Program Organization Unit
PTC	Production Technology Centre (University West)
SCADA	Supervisory Control and Data Acquisition
SFC	Sequential Function Chart: A graphical PLC programming language.
SiL	Software in the Loop
ST	Structured Text: A text-based PLC programming language.
VC	Visual Commissioning

1 Introduction

This chapter introduces the master's thesis by giving a brief background of the thesis topic, the definition of the research problem, the aim and the research question, and the study delimitation and outline.

1.1 Background

Laboratory work is an essential component in research, education, and engineering training. The majority of the universities and academic centres are equipped with modern laboratories. However, access to these laboratories might be restricted due to various factors. In distance learning and research, for example, the physical labs equipment is not always accessible for students and researchers. Another example of the restriction factors is the restriction caused by the current Covid-19 pandemic over the last two years.

To address this problem, the universities and academic centres need to adopt new approaches in learning and research processes. One of the suggested approaches is to build a virtual representation (a digital twin) of the laboratory equipment. This allows the students and researchers to perform experiments virtually using simulation software.

Most of the studies and research carried out on digital twins focus on their industrial, commercial, and civil applications. However, there is a shortage of research that addresses the applications of digital twin technology in research and education. This project aims to contribute to filling the knowledge gap in this field.

The base of this project is the digital twin technology (DT). The digital twin is an emerging technology, and it is considered a part of Cyber-Physical Systems (CPS). In addition to DT and CPS, this work contributes to other research areas such as automation, virtual commissioning, smart manufacturing, and industry 4.0.

1.2 Problem formulation

The division of Production Systems at University West is rebuilding their Production Technology Centre (PTC) automation line. This automation line is used primarily to conduct certain labs in the master programs in robotics and automation as well as for undergraduate education and in research. In parallel with the rebuilding process, there is a plan to build a virtual laboratory (a digital twin) based on the new design of the automation line. This virtual laboratory will help the distance students and researchers to test and debug their implementations of the labs before conducting them in the real laboratory. For certain labs, it will be sufficient to conduct the labs virtually for gaining the required knowledge.

This thesis work will investigate the possibilities and limitations of using such a virtual laboratory in education and research.

1.3 Aim and research question

The purpose of this master's thesis is to investigate the possibilities to employ digital twin technology in education and research by building and evaluating a virtual model of the automation line in PTC in University West. The result of this project can be extended to other industrial-like lab equipment used in research or/and education. The main goals of this master's thesis are to:

- build a digital twin of the PTC automation line using Visual Components and Twincat,
- and validate and evaluate the virtual model and the logic control of the digital twin.

RQ: What are the possibilities and limitations of applying digital twin technology in education and research to conduct industrial-like labs virtually?

1.4 Delimitation

The focus of this study will be on the virtual commissioning of the new automation line in university West PTC. The experiment used to compare the physical lab performance to the virtual lab performance will be limited to one robotic cell of the automation line. This experiment will be done by Behzad Far as part of his master's thesis work [1]. The practical part of this study will be designed and implemented using Visual Components premium 4.2, AutoCAD 2021 and TwinCat 3. The physical behaviours of the modelled components, such as gravity and inertia, will not be considered in this study.

1.5 Outline

The content of this work is structured in six main chapters. The first chapter provides an overview of the thesis topic and addresses the thesis problem, objective, and limitation. The second chapter is a literature review that surveys books and scholarly articles to provide the required theoretical background information on the thesis topic. In the third chapter, the methods and the work procedure used in this thesis work are presented. The design and implementation of the practical part of the research are presented in the fourth chapter. In the fifth chapter, the outcome of this study is presented and evaluated. The recommendations, improvement and suggested future work are discussed in the sixth chapter.

2 Related work

In this section, the theoretical background of the key concepts in this thesis is presented. The aim is to build the required knowledge by studying and interpreting relevant and most recent literature.

2.1 Digital twin

In the current era of the technological revolution, there are massive volumes of data generated by sensors and smart devices. The need for processing this data has paved the way for several new technologies. Digital twin (DT) is one of those emerging technologies. The key enabling factors for digital twin technology are the internet of things (IoT) and the possibility to collect data in real-time [2]. DT is one of the key milestones in the development of the Industry 4.0 concepts [3], and it can enhance the system by [3], [4], [5]:

- Increasing the visibility in the system.
- Increasing visibility reduces both maintenance cost and energy consumption.
- Increasing efficiency and safety.
- Optimizing of the system operations.
- Fusing different information technologies such as cloud computing, simulation and IoT.

2.1.1 Digital twin concept

Presently, there is no common standard definition of DT in both research and industry, and the definitions vary based on the area of application [6]. In addition to the lack of a standard definition of the digital twin concept, the digital twin is usually confused with other similar concepts such as simulation and cyber-physical system (CPS) [7].

Tao, Zhang and Nee in [4] listed the most used definitions of the digital twin in academia and industry. Based on these definitions they described the digital twin “*as a virtual representation that interacts with the physical object throughout its lifecycle and provides intelligence for evaluation, optimization, prediction*”.

According to Agalinos et al. in [3], a digital twin system consists of a physical entity, a virtual model, and connections between the physical and virtual spaces. Fuller et al. described the digital twin as seamless and bidirectional integration of data between the physical and digital spaces of a system [8].

2.1.2 Digital twin and cyber-physical system

DT and CPS are two closely related concepts as both systems aim to achieve seamless integration between the virtual and the physical spaces [4]. However, the borders between the two systems are not completely clear in academia [7]. Furthermore, Tao, Zhang and Nee [4] considered DT as a focused application of CPS.

According to Tao et al. in [9], a CPS system is defined as “*the integration of computational and physical processes*”. Whereas DT is described as a real-time optimization of a physical system by using a digital replica of the system.

The main focus in CPS is on communication, computing, and control [4], [9]. However, DT technology focuses more on virtual modelling. Sensors and actuators are the key components in CPS while data and models are the main components of DT [9].

2.1.3 DT dimensions

Tao et al. [10] found that there are two models of DT in research: the three-dimensional model and the five-dimensional model. In the three-dimensional model, DT has three core components: physical space, virtual space, and connection. This model is illustrated in Figure 1.

In addition to the core components of the three-dimensional model, the five-dimensional model has two extra components: data and service [11], [12], [13]. The structure of the five-dimensional model is shown in Figure 2.

The virtual model represents the mapping of the physical system properties, behaviours, and rules in the virtual world. The services in DT are used to encapsulate the complicated functions of DT into simple and user-friendly modules [4], [12]. Data in this model is considered the main driver of the digital twin model. Data in a five-dimensional model can be obtained from physical systems. Data can also be collected from virtual models and services. Connections are used to allow data exchange between DT different core components [4], [13].

2.2 Digital twin enabling technologies

Tao, Zhang and Nee in [4] as well as Qi et al. in [13] classified enabling technologies of DT into five categories based on the core components of the five-dimensional framework shown in Figure 2. The key technologies applied to DT are shown in Figure 3.

2.2.1 Physical-world enabling technologies

These technologies are used mainly for perceiving data and cognizing the physical entities. Some examples of the technologies in this category are sensing, measurement and process technologies [13].

2.2.2 Virtual model enabling technologies

Modelling is the cornerstone of digital twin technology, and it aims to build and process a virtual replica of the physical entities. The key modelling technologies used in DT are geometric modelling and physics simulation [13].

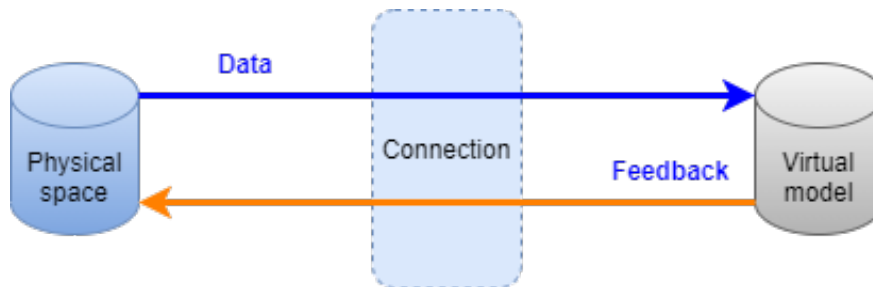


Figure 1: DT three-dimensional model. Adopted from [4].

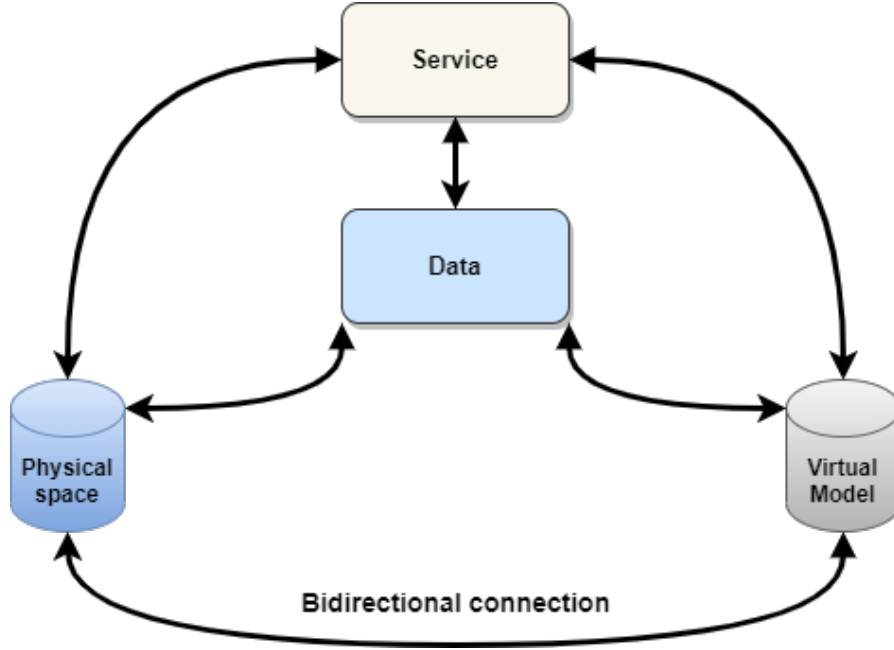


Figure 2: DT five-dimensional model. Adopted from [4].

2.2.3 Data management enabling technologies

The key functions of data management technologies are to collect, store and process system data, other functions are to fuse the data from different sources and to visualize this data. The key technologies in the data management category are storage technology and fusion technology [13].

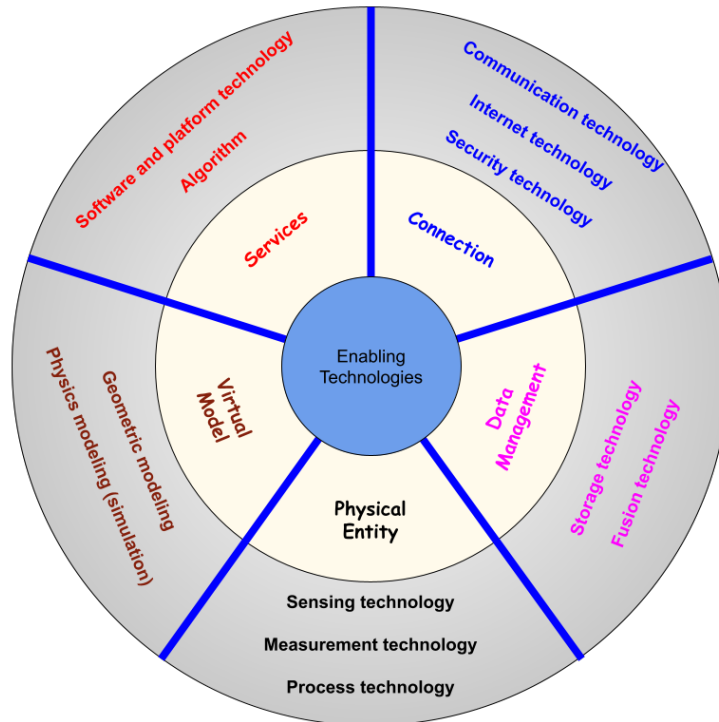


Figure 3: Classifying of DT enabling technologies. Adopted from [13].

2.2.4 Services enabling technologies

Digital twin integrates various technologies which enable monitoring, simulation, and diagnosis of the system. Two examples of the services enabling technologies are software and platform technology and algorithm [13].

2.2.5 Connections enabling technologies

Effect data exchange between the core components of DT requires several assistive technologies such as communication technology, internet technology and security technology [13].

2.3 DT levels of integration

Based on how the data is exchanged between the physical and virtual spaces, the digital twin technology is classified into three categories: digital model, digital shadow, and digital twin [6], [8]. Figure 4 shows DT different levels of integration.

2.3.1 Digital model

In this model, there is no automated flow of data between the physical space and the virtual space, and the data is updated manually in both directions. The changes in the physical system do not affect the virtual model automatically and vice versa [6] [8]. According to Lu et al. [7], the digital model is a simulation rather than a digital twin system.

2.3.2 Digital shadow

If there is an automated flow of data from the physical space to the virtual space, the model is defined as a digital shadow. In this model, changes in the status of the physical system have a direct impact on the virtual model. The changes in virtual model status, however, have no impact on the physical system.

2.3.3 Digital twin

In this model, the real-time data can flow in both directions and the changes in the status of the physical system can directly affect the virtual model and vice versa [3], [6], [8]. The flow of real-time data makes it possible to create effective digital twin systems with the capability to predict and model the dynamic changes in the system [3].

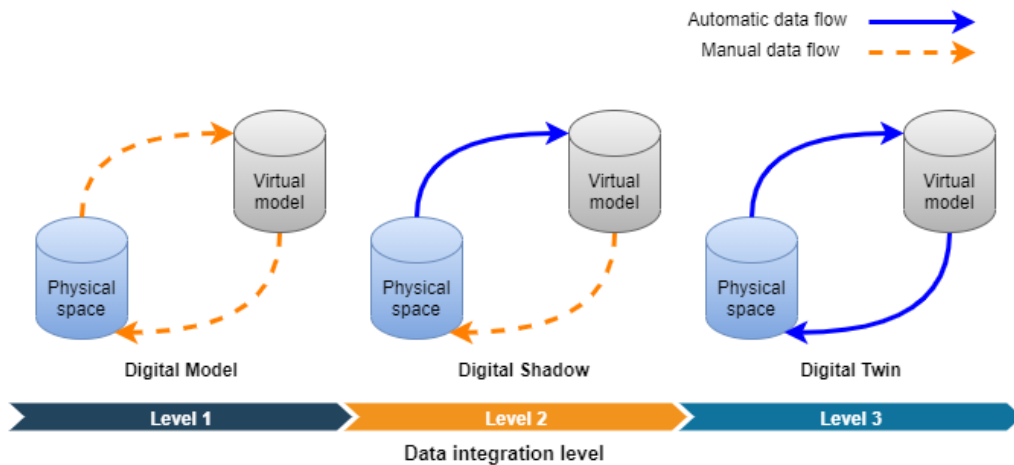


Figure 4: Different levels of the digital twin. Adopted from [6] and [8].

2.4 Modelling

White and Ingalls defined a model in [14] as “*an entity that is used to represent some other entity for some defined purpose*”. A model, in general, is an abstraction of a real complex system and it is used mainly to study the behaviour of the real system when the system under control cannot be observed directly due to the high cost of experimenting with the real system or because of unsafety issues [14], [15]. There are three basic types of models: physical models, mathematical models, and computer models.

Loper referred to physical models in [15] as iconic models. Physical models can be static or dynamic. A static physical model is a scale model of the real system being modelled. Static physical systems do not change with respect to time while dynamic physical models do change with time [16]. A typical example of a static physical model is the scaled-down model of a building under construction. The wind tunnel is a typical example of a dynamic physical model.

Dym in [17] defined the mathematical model as “*a representation in mathematical terms of the behaviour of real devices and objects*”. Mathematical models are usually represented using a set of mathematical equations involving a set of variables and they are classified into two basic categories: continuous models and discrete models. In continuous models, the variables “*vary continuously in space and time, while the variables in discrete models vary discontinuously*” [18].

Depending on how they change over time, the mathematical models can also be divided into static mathematical models and dynamic mathematical models. When mathematical models are implemented and manipulated using computers, they called computer model [16].

2.5 Simulation

Simulation and modelling are very closely connected concepts. According to Singh in [16], mathematical modelling can be considered as a simulation, and mathematical simulation can be considered as modelling.

Simulation is a practical process that aims to conduct experiments on a model of a real system to study the behaviour of the system of interest [14]. Simulation, according to White and Ingalls, is a practical method used to study a real system by developing a model of the system and performing experiments on this model to understand the behaviour of the system under control [14].

Based on the nature of the system variables, Sokolowski and Banks in [19] classified simulation models into three categories: discrete event simulation, continuous simulation, and Monte Carlo simulation. In continuous simulations, the system variables are continuous functions of time while in discrete event simulations, the variables are discrete functions of time. Monte Carlo simulation uses probabilities to model the system behaviour. The outcomes of the model are predicted based on randomly generated samples of input variables.

2.5.1 Simulation vs emulation

Simulation models are abstractions of real systems and they hold different levels of simplification and approximation and therefore, there is a notable difference between the performance of the real system and the model. This difference is proportionally related to the complexity of the system and the level of the simplification of the model. Emulation models can help to narrow this performance gap [20].

McGregor defined an emulation model as a model “*where some functional part of the model is carried out by a part of the real system*” [21]. Emulation is employed basically in the development and validation phases of the system life cycle [22].

The key difference between emulation and simulation models is the purpose of the modelling. With simulation models, the primary aim is to analyse the system behaviour and performance by trying and testing different scenarios and parameters [21] [23]. Emulation models aim to test the system logical control with various operating conditions. Emulation models can be used as a risk-free training tool. Another important difference is that emulation models should be run almost in real-time while simulation models can run faster than the real-time of the modelled system [21].

2.6 Industrial automation

Industrial automation is the combination of “*the process control and information systems*”. The industrial automation system aims to increase productivity, low production cost and enhance product quality [24]. An automation system consists mainly of two components: the controlled process (the plant) and the controller [25]. According to Hawkins in [26] the key components of an industrial automated system are machinery, control, computing, and communication.

The structure of a traditional automation system, according to Durkop et al. in [27], can be divided, in general, into five different levels as shown in Figure 5. The structure is derived from the automation pyramid model of the International Society of Automation standard (ISA-95) [28]. In this model, the upper three levels are optional. Level 4 is the highest, and it is responsible for the management of the whole business and logistic process of the enterprise. The plant management level is responsible for the management of the entire manufacturing process. The operator level is used to monitor and control the system devices and process using a set of HMIs. On the control level (level 1), the system controllers (PLCs) manipulate field level I/O signals such as sensors and actuators signals [27], [29].

According to Hawkins in [26] and Groover in [30], there are three types of industrial automation: fixed industrial automation, programmable industrial automation, and flexible industrial automation. A process in a fixed industrial automation system is mainly set to produce one type of products. It is often very expensive to change the product

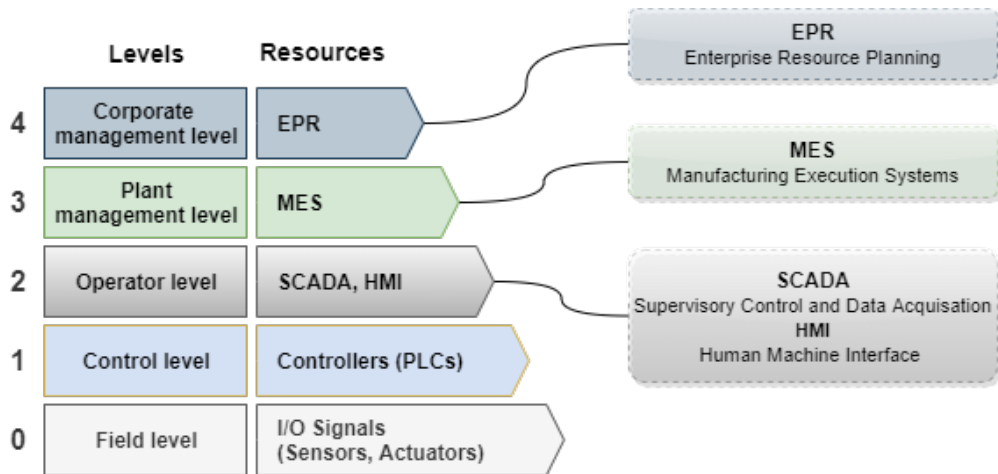


Figure 5: Automation levels. Adopted from [27] and [29].

in fixed automation. With programmable automation, the system can be used to produce new products by re-programming it and without changing the current equipment. The cost of changing the product in programmable automation is considerably low compared to fixed automation systems. With flexible industrial automation, it is possible to use the same programming and equipment to produce a variety of products. The relation between productivity and flexibility for the previous automation systems is shown in Figure 6.

2.6.1 Controller

Controllers are considered to be the key parts of the industrial automation system as they carry out the computing and the management of I/O signals in the system [31]. There are essentially four basic types of industrial controller: mechanical, pneumatic, hydraulic, and electronic controllers. Electronic controllers can take two forms: a distributed control system (DCSs) and programmable logic controllers (PLCs) [24]. DCSs are mainly used in process control and it is beyond the scope of this study.

Programmable logic controller (PLC)

Bolton in [32] and [33] defined a programmable logic controller as a digital electronic “*microprocessor-based*” controller that can store instructions in its memory and processes these instructions to control industrial processes. The key components of a PLC system are a central processing unit, memory, power supply and IO interfaces. The basic structure of a PLC system as well as its external connections with peripheral equipment is shown in Figure 7.

PLC programming

The majority of key PLC manufactures follow international standard IEC 61131-3 which is considered as a guideline for programming modern PLC systems. IEC 61131-3 defines five PLC programming languages. Three of these programming languages are graphical and two are text-based. The graphical programming languages are Function Block Diagram (FBD), Ladder Diagram (LD), and Sequential Function Chart (SFC), while the text-based programming languages are: Structured Text (ST) and Instruction List (IL) [34].

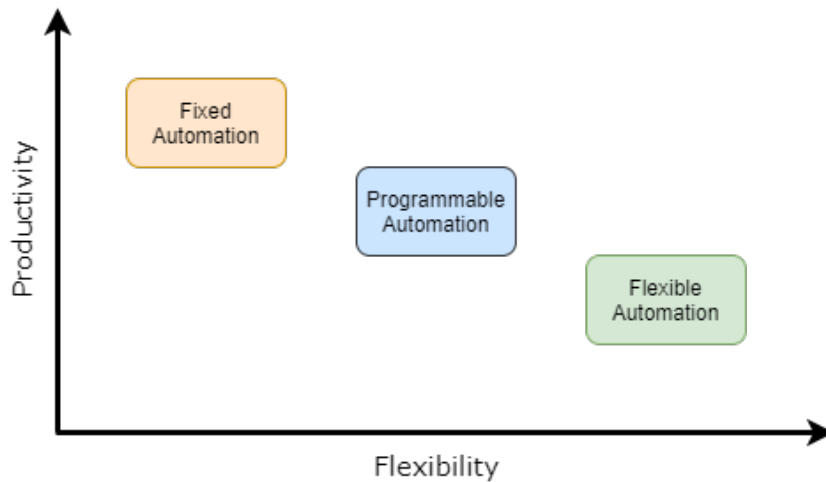


Figure 6: Productivity-flexibility relation in different automation systems. Inspired by [26]

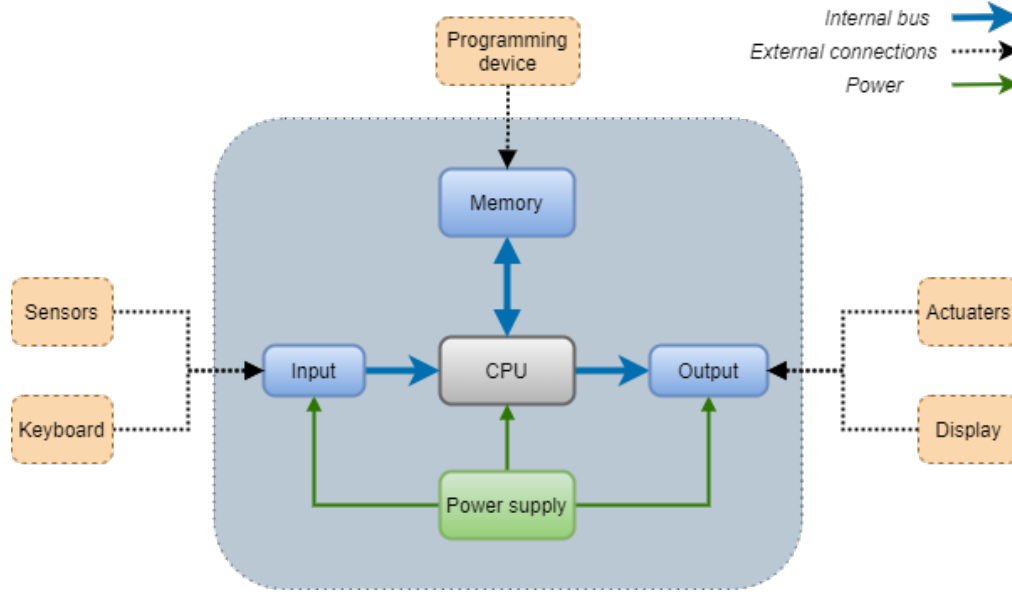


Figure 7: PLC internal structure. Adopted from [34].

ST is a high-level C-like PLC programming language and this makes it very useful in programming complex control tasks. IL, on the other hand, is an Assembly-like low-level language. and IL is used mostly for retaining compatibility with older PLC systems. LD is the most used PLC programming language because of its simplicity and also due to its similarity to the traditional electrical relay diagram. SFC is very effective in programming state-based control and/or sequential control. SFC, however, cannot be used as a stand-alone language and it is used always with another standard language such as ST or LD [34].

2.6.2 Industrial communication

An industrial communication system is a network used to transfer data between controllers and other devices in the system such as sensors, I/O devices, and actuators [35]. The transferred data can be of different forms. It can be textual or numerical data. It can be in form of digital or analogue I/O signals [31].

There are currently many different industrial communication protocols and technologies applied in automation systems. This includes DeviceNet, OPC UA, Profinet to name but a few. Profinet is a real-time protocol for communication over industrial networks in automation systems. It is based on the EIC 61158 standard [36]. DeviceNet is a low-end device-oriented network designed primarily to interface devices in the lower level of the automation system (sensors, actuators) with system controllers. A single DeviceNet network can connect up to 64 nodes [37]. OPC Unified Architecture (OPC UA) is “a platform-independent communication protocol for industrial automation, developed by OPC Foundation” [38].

2.7 Commissioning

American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) has defined the commissioning process as “a quality-oriented process for achieving, verifying, and documenting that the performance of facilities, systems, and assemblies meets defined objectives and criteria” [39].

Commissioning is the last phase in the life cycle of a system building or upgrading and it works as a bridge between the design and the operation of the system. Commissioning is performed usually by a dedicated commission team in cooperation with the operators and other technical staff [40].

Commissioning has typically three phases: pre-commissioning, core commissioning and Start-up [41]. The pre-commissioning is the whole activities performed during the system building phase to prepare the system for the core commissioning phase. The core commissioning phase is to put the system equipment and sub-systems into their “initial operation”. In the start-up phase of commissioning, the system is put into its actual planned operation.

2.7.1 Virtual commissioning (VC)

With real commissioning, the system cannot be tested and validated before it is built. This often results in delays and increase the cost during the start-up phase of the system. Virtual commissioning (VC) is used mainly to overcome this problem by decreasing the time required during the commission phase [42]. Virtual commissioning enhances real commissioning by detecting system errors and bugs in the early stage before the real commissioning is started [43].

Virtual commissioning, according to Vermaak and Niemann in [44], is “*the simulation of a system in a virtual environment*”. The simulation model is used for the validation of the system. Virtual commissioning starts in the early stage of system development before the physical system is built.

Using a commissioning virtual environment makes it possible to reconfigure and upgrade an existing system easily. This can be achieved due to the ability to perform the virtual commissioning of the system in parallel with the production and assembly processes.

2.7.2 Virtual commissioning stages

There are several techniques used during the development and test of automation systems. This includes a model in the loop (MIL), a process in the loop (PIL), hardware in the loop (HIL) and software in the loop (SIL). During the VC development lifecycle, however, three of these models are mainly used. These models are MIL, SIL, and HIL [45]. Figure 8 shows the development stages of VC while Figure 9 shows the configurations of different VC models.

Model in the loop (MIL)

Model in the loop is the first stage in the VC development process [45]- [46]. In MIL both the controlled process and the controller are replaced with simulation models [25].

Software in the loop (SIL)

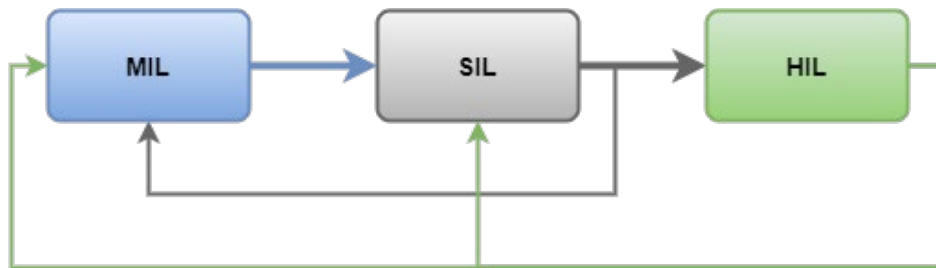


Figure 8: Virtual commissioning development stages. Adopted from [45].

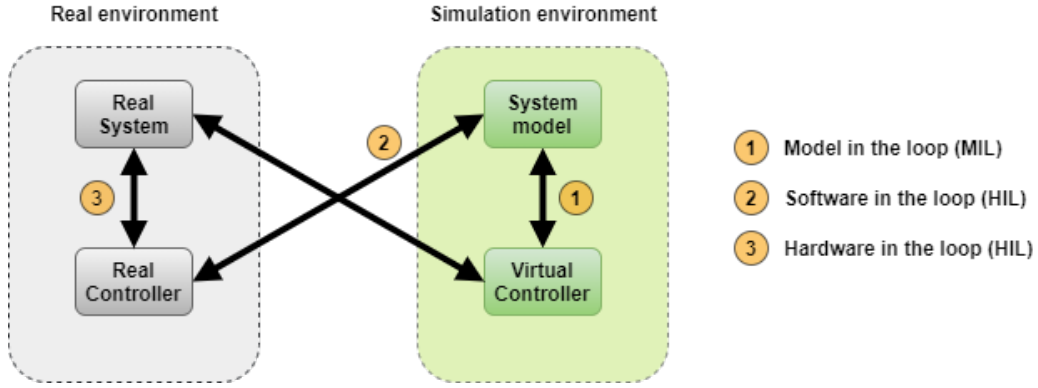


Figure 9: Model configurations applied in commissioning. Adopted from [47].

According to Cech et al. in [47], the software in the loop stage is the integration between the simulation model of the system and the real controller.

Hardware in the loop (HIL)

As demonstrated in [45], the hardware in the loop is the last stage in virtual commissioning. In this stage, the real hardware and software are used to test the system [47]. However, Lee and Park in [48] referred to this stage as real commissioning rather than HIL. They considered that HIL involves a virtual system and a real controller.

2.7.3 Virtual commissioning workflow

Modelling in virtual commissioning can be divided into three sub-tasks: physical device modelling, logical device modelling and system control modelling [49]. Physical device modelling is used to transfer the physical design of the device to a digital representation. The physical device modelling includes both the geometric and kinematic attributes of the modelled system [49], [48]. The logical device modelling is used to represent the behaviour of the modelled system in response to the control program [48]. System control modelling is to build the control program (PLC program) required to control the logic of the system [49]. Figure 10 describes the workflow of the virtual commissioning technology.

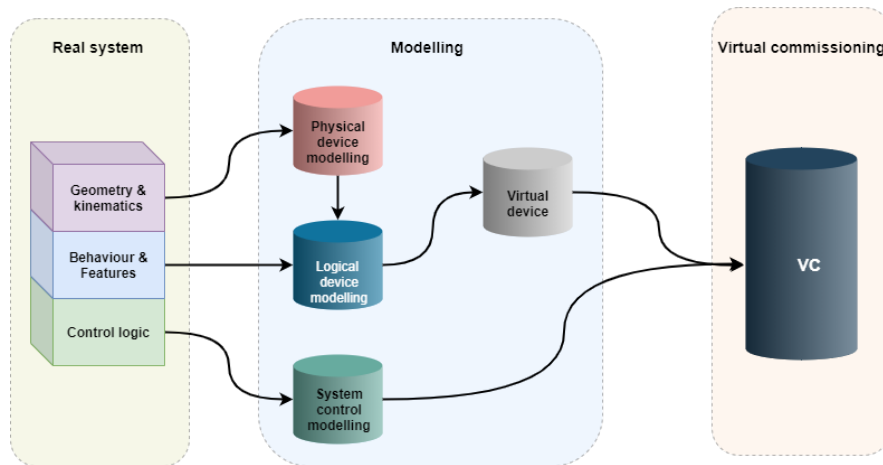


Figure 10: Virtual commissioning workflow. Adopted from [48].

3 Method

The method section describes the procedures and techniques used to solve the study problem. This includes identifying the problem as well as collecting and processing the data. This includes also analysing the result of the study [50].

3.1 Data collecting

The digital twin of the automation line is built using Visual Components premium 4.2 and TwinCat 3. The data required to build the digital twin can be divided into two key parts: geometrical data and behavioural data. Geometrical data include all required geometrical information such as CAD models of the key parts of the automation line. Behavioural data describe how the line component behave in response to control commands and signals. The geometrical and behavioural data of the automation line are obtained from the division of production systems at University West.

The majority of geometrical data are obtained from a layout created in RobotStudio which is a simulation and offline programming tool from ABB. This layout is used to get the information required to build a similar layout of the automation line in Visual Components. Some of the geometrical data are provided in form of Standard Triangle Language (STL) files which can be imported directly to the Visual Component. This is especially true for the products parts (cylinders, octagon, plates, containers).

The behavioural data of the automation line are provided in the project description. The description gives detailed information about the automation line suggested work. This includes the following:

- A brief introduction to the new automation line.
- Communication layout
- Description of the states of the stations
- Description of assembly flow in the stations
- Description of the function blocks required to control the robots
- Description of the stations' input/output signals

3.2 Data processing

To process the collected data and convert it into the needed digital twin, three key modelling types are required, namely, physical device modelling, virtual device modelling, and system control modelling. In this work, Visual Components premium 4.2 is used for both physical device modelling and logical device modelling while control system modelling is performed using TwinCat 3 software. Since the models that build the digital twin are located in two different software programs, a communication protocol is needed to integrate these models. The processing layout is shown in Figure 11.

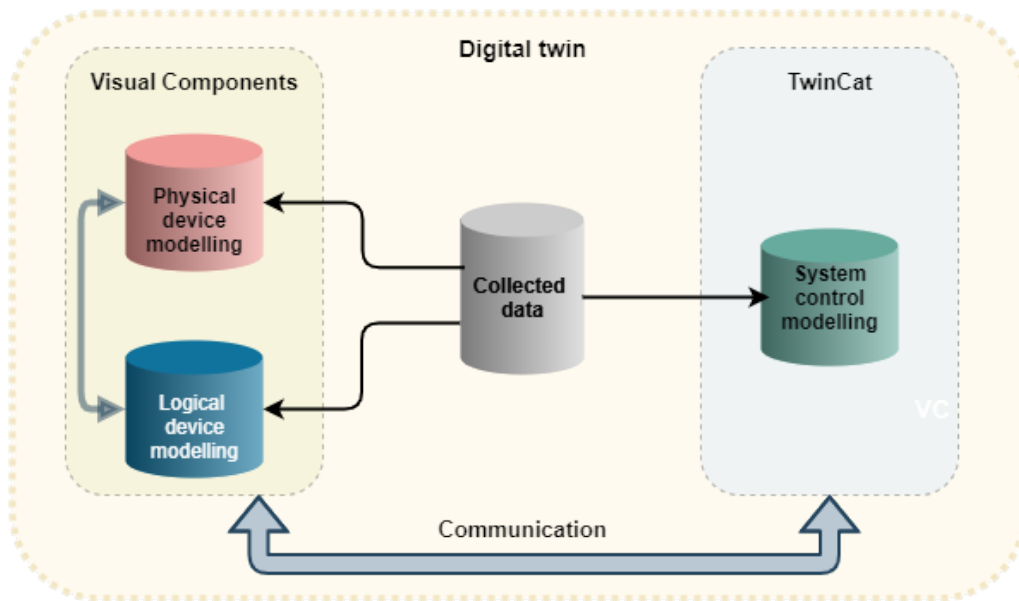


Figure 11: Data processing layout. Adopted from [48].

3.2.1 Physical device modelling

Physical device modelling is the process of transferring the physical design of a device to a digital representation. The physical device modelling includes both the geometric and kinematic attributes of the modelled device.

The majority of the layout components can be added directly from the Visual Components library. This is especially true for the standard components in Visual Components such as robots, fences, and tables. Non-standard components, which are not provided are modelled using AutoCAD software when they cannot be modelled easily using Visual Components.

3.2.2 Logical device modelling

Logical device modelling is the process of defining the behaviour of the modelled device in response to the control logic and signals. The most commonly used approach to defining and controlling the behaviour of the modelled component is to add a Python script to the component. Python scripts make it possible to control the component behaviour in a very effective and flexible way. Adding signals and sensors to the modelled components are other examples of logical device modelling.

3.2.3 System control modelling

System control modelling is the process of building the control logic (PLC program) required to control the system. System control modelling includes generally the following key steps:

- Defining and connecting the global variables
- Building the functions blocks and other helper sub-programs
- Building the main program
- Designing the HMIs

3.2.4 Communication

The key parts of the digital twin of the automation line reside in different software programs. A communication protocol is, therefore, required to allow these parts to communicate and exchange data. There are two options of communication protocol available to connect Visual Components models and TwinCat projects. These options are Beckhoff ADS and OPC UA. In this work, however, the Beckhoff ADS protocol will be used due to its simplicity in both implementation and configuration.

3.3 Result Evaluation

The practical work and the result of this study will be tested and evaluated continuously during the implementation of the model. In the physical modelling phase, the modelled components of the automation line will be checked to ensure that they are modelled according to the specifications. During the logical modelling phase, the modelled components will be tested by using Python scripts test code. In the system control modelling phase, the PLC's function blocks, and the other program organization units will be tested using test program organization units (POUs) and test HMIs. The virtual model created in this work will also be tested by Behzad Far as part of his master's thesis work [1].

4 Design & Implementation

This work aims to investigate the possibilities and limitations of building a digital twin of automation lab equipment and using the digital twin to conduct automation labs and experiments virtually. The new PTC automation line at University West was used as a case study in this work. More specifically, the automation line project in the course Automation System (ATM700) was used for modelling and testing the digital twin. In the course project, the automation line is designed to produce three similar types of products. Three colour codes (red, blue, and green) are used to distinguish between these products. Each product is assembled of three key parts of the same colour. The parts are a plate, a cylinder, and a screw. Special pallets (assembly containers) are used in production as fixtures during the assembly process. Other pallets (storage containers) are used to store the assembly parts in the buffers of the assembly stations.

4.1 Automation line specification

Figure 12 shows the layout of the automation line. The line consists of nine robotic cells (STN100 – STN900) and one automated guided vehicle (AGV). The automation line uses a master/slave configuration where STN500 is the master and the other cells (assembly stations) are slaves. The master (STN500) is an input/output station, and it is designed to perform the main control of the production line. STN500 is also used to supply and direct the assembly pallets to the different assembly stations via AGV and also to store the finished products. Each station in the line is controlled by a dedicated PLC. All assembly stations as well as the AGV are connected to the I/O station through an OPC UA communication server as illustrated in Figure 13. PTC automation line is constructed to work according to the following specifications and requirements:

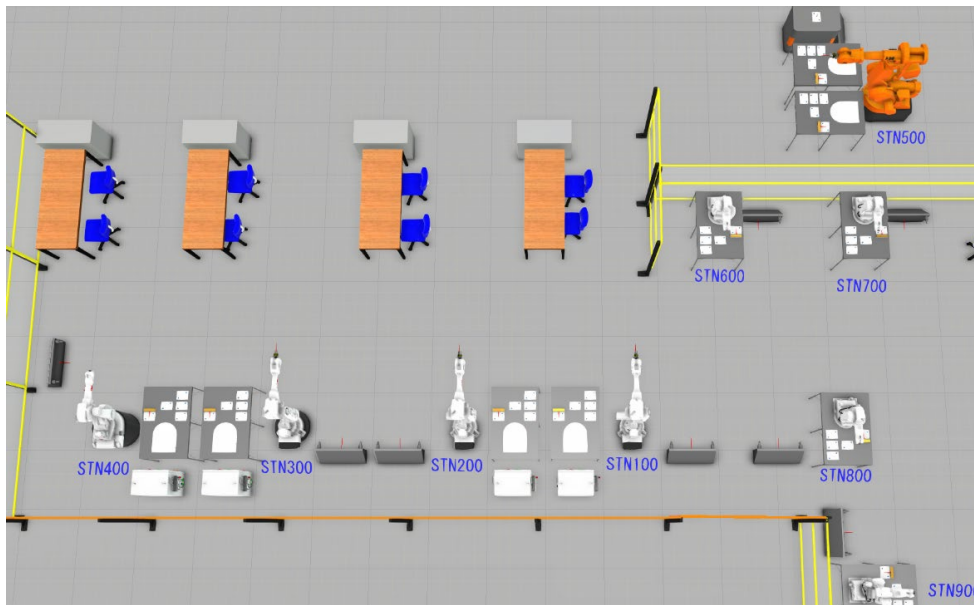


Figure 12: Automation line layout

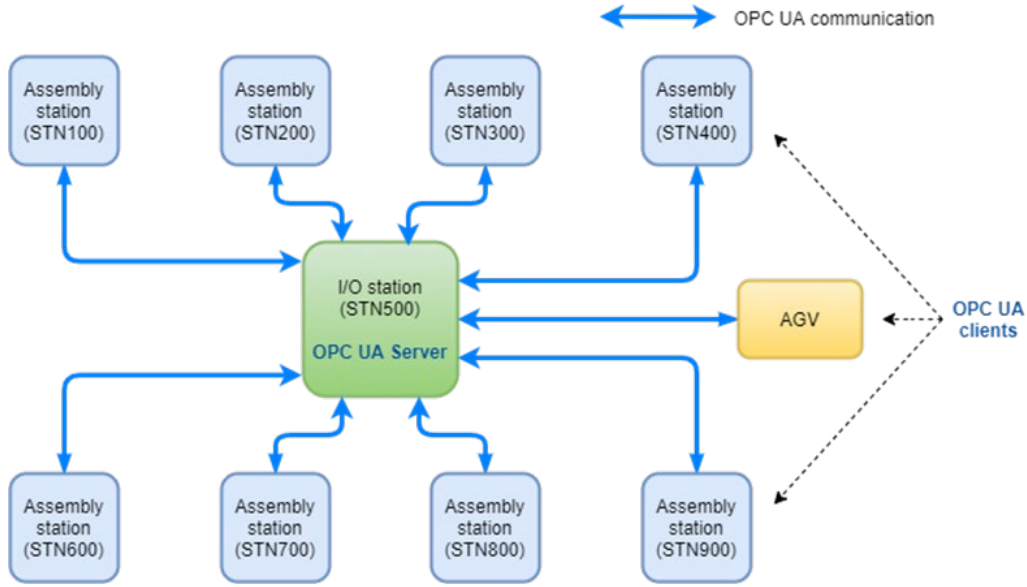


Figure 13: Layout of Automation line communication. Adopted from line specifications.

- I/O station works as the communication medium for all other stations in the line including the AGV. A station in the line can therefore communicate directly only with the I/O station.
- All the product parts are made of metal, and they can trigger the conductive sensors used in the robot tools to detect and identify the assembled parts in the assembly containers.
- I/O station performs the line main control logic, and it assigns tasks to all stations in the line.
- The empty assembly containers are supplied to the assembly stations by the I/O station.
- All storage buffers in the assembly stations are initially loaded by the staff with random pallets of parts.
- The pallets of parts have a predefined quantity of items.

4.2 Automation line workflow

The assembly process is performed according to the following steps. First, a plate is placed in its position in the assembly container. A cylinder then is added to the plate. Lastly, a screw is added to the product. All parts should have the same colour as the assembly container.

The production process starts by manually loading the buffers with assembly containers and parts. Every station scans its buffer storages and stores the storages content details. The I/O station supplies the line with an empty assembly container. The container is directed to the first assembly station that has the required part. The workflow of an assembly station in the automation line is organized according to the general algorithm shown in List 1. The state diagram of an assembly station is shown in Figure 14.

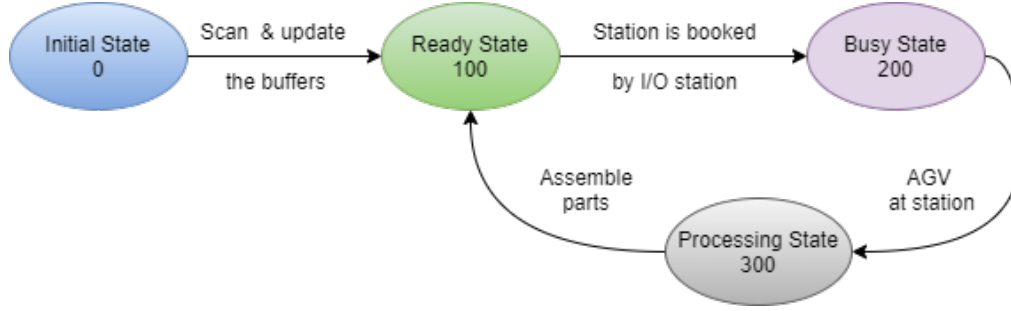


Figure 14: State diagram of an assembly station.

```
1:      Scan plates in station buffer
2:      Update details of buffer
3:      State = 100 (Ready state)
4:      WHILE there are orders Do
5:          IF station is booked by IO station THEN
6:              State = 200 (Busy state)
7:              IF AGV is at station THEN
8:                  State = 300 (Processing state)
9:                  Scan to find colour of pallet provided by AGV
10:                 Scan previously assembled parts on pallet.
11:                 Assemble missing parts in assembly pallet
12:                 Update buffer details
13:                 Send order for next assembly
14:                 Place pallet on AGV
15:                 Set station to ready state (100)
16:                 Wait for next production order
```

List 1: Assembly station workflow.

4.3 Modelling

Modelling in this work comprised physical and logical models building as well as the development of the system control. Through the model building, the following assumptions were made:

- The line is designed to produce three products in each production cycle.
- The products are assembled in a sequence which means that there is only one assembly station working at any time of the production cycle.

4.3.1 Physical model building

The physical modelling phase was performed in Visual Components Premium 4.2, and it involved building a model that possesses the geometric and kinematic attributes of the modelled automation line. The dimensions and location of a component in the line are good examples of these geometric attributes, while the acceleration, deceleration and linear speed of the moving components are typical examples of the kinematic attributes. Three methods were used to physically model the automation line parts in Visual Components. The methods involved adding components from the built-in library, creating geometries from scratch, and/or importing existing geometries.

Most of the components required to build the model were found in the Visual Components standard library. Some of these components, such as ABB robots, were used directly without any modification, while the other components, e.g., working tables,

required some modifications to their default properties before using them. Figure 15 shows the key geometries in an assembly station.

Some of the required automation line components were not provided in the standard library of Visual Components but rather provided as pre-made 3D cad models. This was especially true for the containers and the assembly parts (plates, cylinders, octagons). These models were imported to the Visual component and even added to the library.

The components that were neither found in the Visual Components standard library nor created in other software were created from scratch. The components with simple geometries were built directly in Visual Components Premium using its modelling features. For example, the fixtures on the working table shown in Figure 15 were built using this approach. The components with more complicated geometries required special CAD software to build them. One instance of these components is the docking station of the AGV (Figure 15) which was created in AutoCAD 2021.

Validation in this phase was done by drawing a comparison between the modelled layout and the physical layout of the automation line. This involved checking that the geometric and kinematic attributes of the modelled components were set according to the specifications.

4.3.2 Logical model building

The physical models created in the previous phase are usually passive. In other words, they cannot respond directly to the logic control commands and signals. Through logical modelling, certain behaviours and features were added to these components to make them active. Figure 16 shows the available behaviours in the Visual Component software. The most commonly used behaviours and features in this work are signals, sensors, Python scripts and frames.

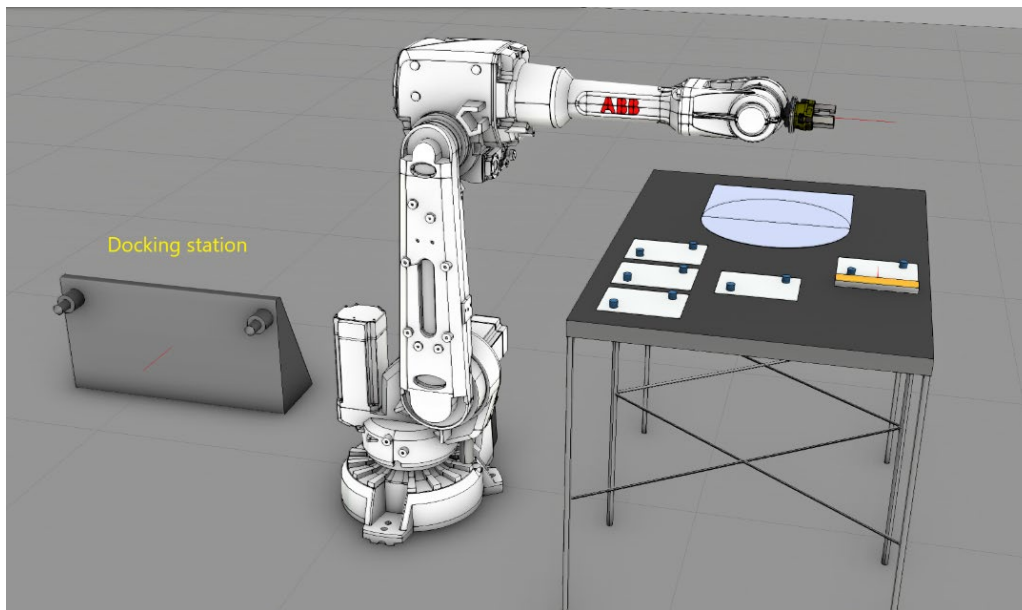


Figure 15: An assembly station.

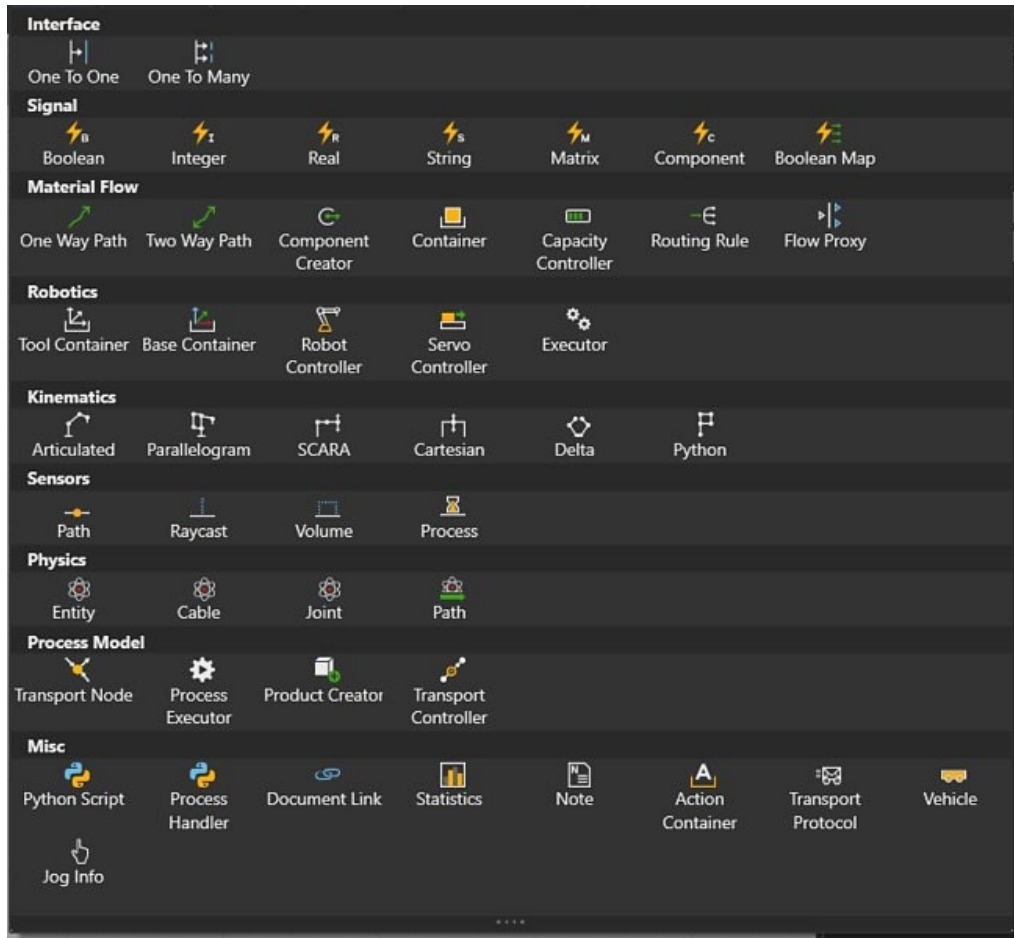


Figure 16: Modelling behaviours in Visual Components.

Python scripts were used in logical modelling to add internal logic to the components of the line. This logic allows the components to behave effectively in response to the external and internal commands and signals. Most of the Python scripts used in this work were defined locally inside their related components. Some Python scripts were however defined globally outside any component. In this case, their code could be imported and used by all local scripts. Figure 17 shows a part of the global code used in this work.

```
35
36 def placePart(robot, pallet, x = xOffset, y = yOffset, z = 15, R = 0):
37     robot.place(pallet, Approach = 100, Tx = x, Ty = y, Tz = z, Rz = R)
38
39
40 def setSignal(signal):
41     signal.Value = True
42     signal.signal()
43
44
45 def resetSignal(signal):
46     signal.Value = False
47     signal.signal()
48
```

Figure 17: Part of the global Python script “helper.py”.

Signal behaviours, on the other hand, were used to give the components and their behaviours the ability to interact with each other. The sensors were especially useful for detecting and identifying certain components in the automation line. The frames were added mostly to define the positions of the robots in the virtual environment.

It was not required to model all automation line components logically. This kind of modelling was only required for the parts that involve directly in the production processes. This includes the AGV, the robots, the sensors, and the dock stations.

AGVs in the standard library of VC are initially passive. They require additional behaviours and features to be able to interact with their environment. Figure 18 shows the behaviours and features added to the AGV, and Figure 19 shows a part of the Python script used to control the AGV. The vehicle behaviour was used to define the AGV as a moving component that can be controlled by a Python Script. The NextStation signal is an integer sent by the PLC program to indicate the ID of the station the AGV is moving toward. MoveAGV is a function used to moves the AGV between two positions using a set of pre-defined paths between fixed points on the automation line floor. The surface of the AGV is equipped with a sensor that is triggered when the AGV is loaded with an assembly container. This sensor was modelled using a ray cast sensor which is a behaviour in Visual Components used mainly to measure distances and detect components in Visual Components 3D models. Three signals may be used in combination with a raycast sensor behaviour. These signals are a range signal, a components signal and a Boolean signal. The Boolean signal is triggered when there is a component within the detection threshold of the sensor. The component signal used to identify what component detected by the sensor and the range signal is used to store the distance between the sensor and the detected component.

The main difference between the I/O robot and the other robots in the line is that there are no assembly processes performed in the I/O station. Despite this difference, these robots have almost the same added behaviours and features. The key behaviour required for all robots is the inductive sensor. The inductive sensor is attached to the

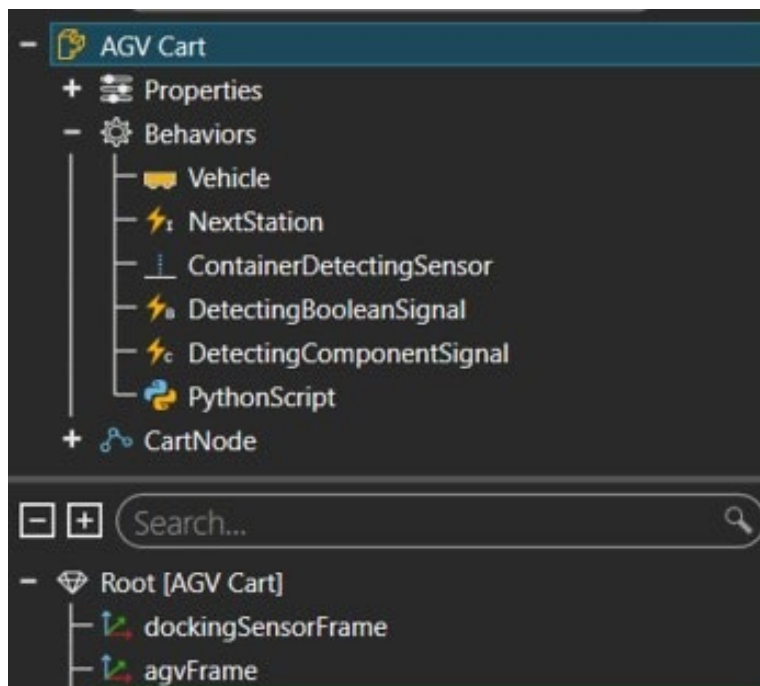


Figure 18: Behaviours and features of the AGV.

```

28
29 # Initialize the AGV
30 def OnStart():
31     vehicle.clearMove()
32     vehicle.Acceleration = 2000
33     vehicle.Deceleration = 2000
34     vehicle.MaxSpeed = 1000
35     vehicle.Interpolation = 0
36
37 # The main method
38 def OnRun():
39     while app.Simulation.IsRunning:
40         triggerCondition(lambda : getTrigger() == plateSignal and plateSignal.Value == True)
41         detectedComponent = plateComponentSignal.Value.Name
42         delay(delayTime)
43         if "Container" in detectedComponent:
44             currentStationID = findCurrentStation()
45             nextStationID = nextStationSignal.Value
46             if nextStationID != 0:
47                 moveAGV(currentStationID, nextStationID, vehicle)
48

```

Figure 19: AGV Python scripts.

robot arm and is used to check for the pre-assembled parts on an assembly container. In Visual Component, the inductive sensors were modelled using the raycast sensor behaviour, a Boolean signal, and a component signal. The other behaviours required by the robots were the order-related and position-related signals. Order-related signals were used to receive the control orders from the PLC-control program and to send the current status of the robot to the PLC program. Position-related signals were used to set the different positions of the robot arm. Figure 20 shows the behaviours added to all robots in the line.

Every dock station in the automation line is equipped with a sensor to detect if the AGV is docked at the station or not. The behaviour of this sensor was modelled in Visual Components in the same manner used to model the robot tool inductive sensor. The same applies to the scanner stations attached to the working tables where raycast

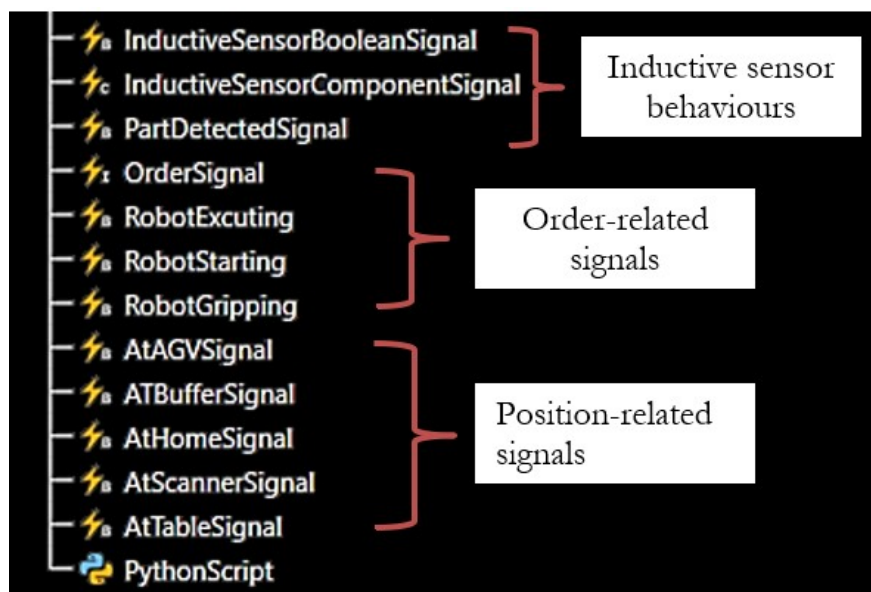


Figure 20: Behaviours added to all robots.

sensors were used along with the appropriated signals to detect the type of the scanned container.

4.3.3 System control building

The system control logic (PLC-control program) plays a key role in monitoring and controlling the physical automation line. As with the real system, a PLC-control program is required to monitor and control the virtual model. The PLC program, in this work, was developed using Beckhoff TwinCat version 3, and it was built based on a pre-defined template. This template contains the data structures and the function blocks required to control the robots in the real automation system.

In the real environment, every station has its own controller and, therefore, its own PLC-control program. OPC UA communication architecture is used to allow the stations to exchange data. In this configuration, the I/O station is working as an OPC UA server and the assembly stations and the AGV as OPC UA clients. Although this configuration was still applicable in the virtual environment, it was not implemented in this work where a one-PLC configuration was used in its place. With the new configuration, One PLC program has been used to monitor and control the whole automation line. The PLC program was organised using different types of program organization units (POUs) according to the structure shown in Figure 21. The PLC program was built using the following steps:

- Defining the global variables.
- Building the function blocks and the helper functions.
- Building the program organisation units (POUs).
- Building the human-machine interfaces (HMIs).

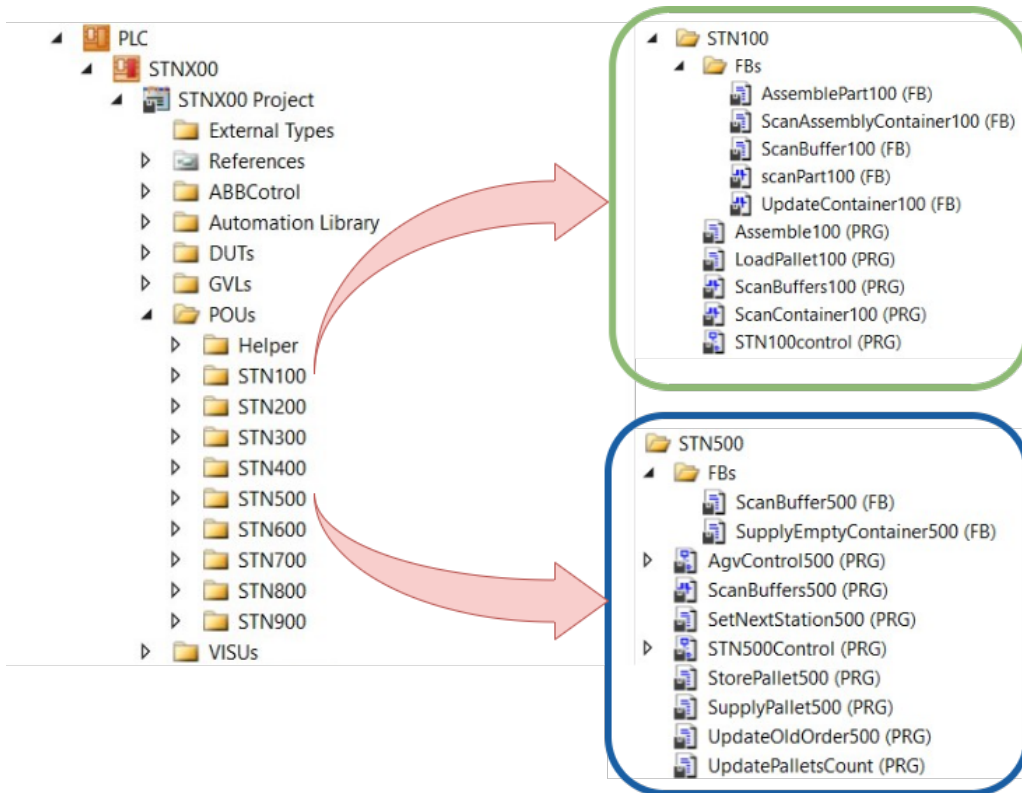


Figure 21: Structure of the PLC program.

- Connecting the PLC program variables with the Visual Components signals.
- Testing and validating the program.

Global variables

The global variables are required to exchange data between the PLC program and the virtual model. They can also be used to change data between different POU's of the program. In this work, the global variables were organised into global variable lists (GVL). Figure 22 shows the global variables defined in every assembly station.

Functions and Function blocks

Functions and function blocks (FB) are used to perform tasks inside PLC programs. They help to write a well-organised PLC program. The key difference between the functions and the functions block is the number of their output. A function block can have more than one output, while a function can only have one output. Figure 23 shows the FindCode function code, which converts the Boolean values of its inputs to scanner sensors to an integer value. The output integer value of the function is used to identify the type of scanned container based on the signals of the scanner sensors.

Many function blocks have been implemented in this PLC program. These function blocks can be categorised into the following three levels: robot-level, station-level, and system-level function blocks. The function blocks used at the system level are necessary to control and/or interact with the whole automation line. The function block used to control the AGV is an example of the function blocks in this category.

Station-level function blocks are used mainly to control the operations of the whole automation cell. SupplyEmptyContainer500 and ScanAssembledParts100 (Figure 21) are two examples of the function blocks in this category. The first function block is used by the I/O station (STN500) to supply the empty containers to the system, while

```
2  VAR_GLOBAL
3      // Order for the next assembly part. The part name should be sent as p
4      qsOrderForNextAssembly: STRING := '0';
5      // Parts in your Station buffer. Each buffer have 3 storage locations.
6      BufferProducts: Station;
7      // Signal to AGV, a pallet is placed from Station to the AGV.
8      qxPalletPlacedinAGV: BOOL;
9      // Signal From AGV, that it have docked in your station
10     ixAGVatStation: BOOL;
11     //100:Ready to Work $ 200: Getting Booked $ 300: Booked and Busy $ 400
12     mwStationStatus: INT;
13     // Emergency Stop is ok and safety is clear.
14     ixEmergencyStopOK:BOOL := TRUE;
15     // Inductive sensor reading. from scanning station
16     ixInductiveSensor1:BOOL;
17     // Inductive sensor reading. from scanning station
18     ixInductiveSensor2:BOOL;
19     // Inductive sensor reading. from scanning station
20     ixInductiveSensor3:BOOL;
21     // Inductive sensor reading. from scanning station
22     ixInductiveSensor4:BOOL;
23     // Inductive sensor reading. from the sensor in robot Tool
24     ixInductiveSensorinTool:BOOL;
25     // Station ID of the station you are working with.
26     mwStationID: INT;
27 END_VAR
```

Figure 22: Global variable list.

```
1 FUNCTION FindCode : SINT
2 VAR_INPUT
3     mxBit1: BOOL;
4     mxBit2: BOOL;
5     mxBit3: BOOL;
6     mxBit4 :BOOL;
7 END_VAR
8 VAR
9 END_VAR
10 FindCode := BOOL_TO_SINT(mxBit1)*1 + BOOL_TO_SINT(mxBit2)*2 +
11             BOOL_TO_SINT(mxBit3)*4 + BOOL_TO_SINT(mxBit4)*8;
12
```

Figure 23

the other is used by the first assembly station (STN100) to check for the assembled parts in the currently received assembly container.

At the robot level, the function blocks are required to control the operations of the robots. These operations include, for example, moving the arms of the robots to pre-defined positions in the cell, picking and placing parts and containers, and assembling parts. The function blocks which control the robots in the virtual environment have the same structure as the function blocks used to control the robots in the real system. This can help to smoothly transfer the PLC program between the two environments. A full list of the robot-level function blocks can be found in Appendix A.

Program organisation units (POUs)

The system control logic used in this work was organised using the structure shown in Figure 21. In this configuration, one PLC project is used to control and monitor the whole automation line. The project uses a directory structure to store and organise its POUs in a hierarchical manner. Every station in the line has its own root folder in the structure. The root folder of a station contains all the functions, function blocks and subprograms needed to control the station. Every station, as well as the AGV, has its main control program. The control program of a station is designed to control all the station subprograms, and it is defined to be executed continuously as long as the PLC program is running. The logic used to control stations varies based on the function of the controlled station. There are thus three different types of control programs used in this PLC project. Figure 24 illustrates the flowcharts of these three control programs. The implementation of these control programs in TwinCat can be found in Appendix B.

Human-machine interfaces (HMIs)

Human-machine interfaces are used to connect the automation line staff and users to the system. HMIs are not compulsory parts in this work and the virtual automation line can be built and run perfectly without using any kind of user-machine interfaces. However, for testing purpose, one HMI was implemented in the PLC program. This HMI was used mainly to test and validate the implementations of the robot-level function blocks. As shown in Figure 25, the HMI is divided into two parts. The upper part is used to test the function blocks controlling the moving and picking/placing actions. The lower part is used to test the scanning and assembling function blocks.

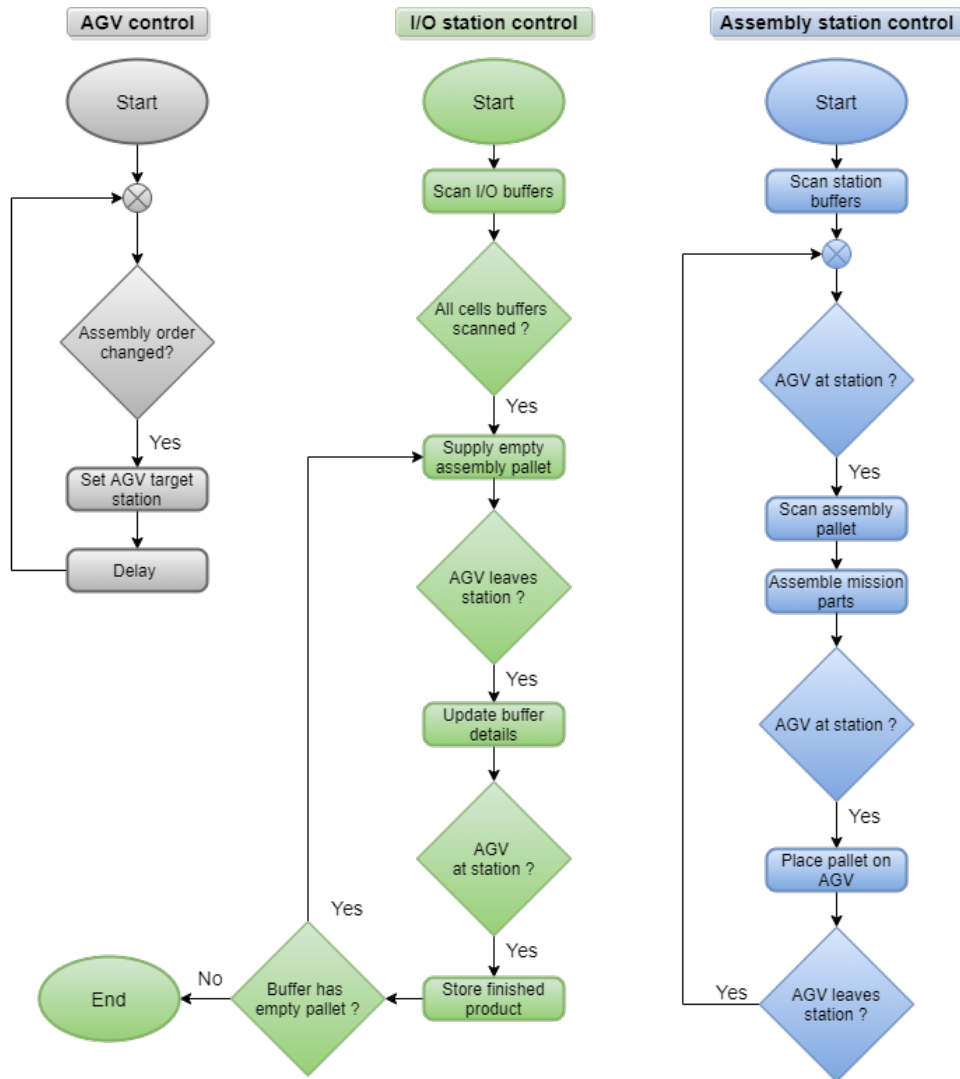


Figure 24: Flowchart of the control programs.

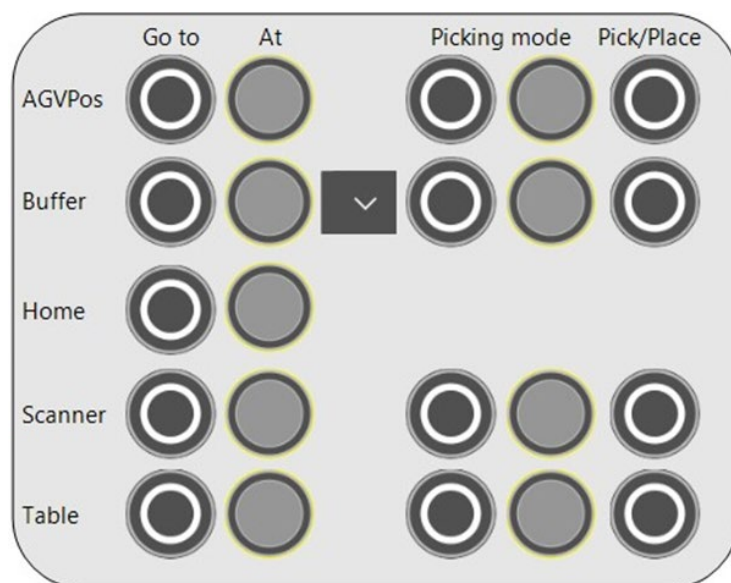


Figure 25: HMI used to test the robot-level function blocks.

Connectivity

Beckhoff ADS (Automation Device Specification) interface was used in this work to connect the PLC program to the virtual model. The key rationale behind choosing Beckhoff ADS was its simplicity in comparison to the OPC UA interface.

The connection between the 3D model and the PLC program was established by pairing the global variables in TwinCat with their related signals in Visual Components using the connectivity feature in VC. Two sets of variables are used in this connection, simulation-to-server variables, and server-to-simulation variables. As the names suggest, the simulation-to-server variables refer to the Visual Components signals connected to their related input variables in the PLC program, while the server-to-simulation variables refer to the PLC program variables connect to their related input signals in Visual Components. Establishing a connection between Visual Components and TwinCat required the following steps: adding the server, editing, and testing the connection, and pairing the variables. In the first step, a Beckhoff ADS server was added using the Visual Components connectivity configuration panel. The next step was to set the ADS port and test the connection providing that the default port used by TwinCat 3 is 851. The last step was to add the required simulation and server variables to the server and to pair them with their related variable in the PLC program. Figure 26 shows a part of the connected variables between the Visual Components layout and the TwinCat PLC program.

Validation

The last step in system control modelling was to test and validate the model. This was done in two levels, POU's level and project level. At the first level, all the PLC project POU's, such as the functions and the function blocks, were tested individually with the

Connected Variables								
Structure	Simulation variable	Simulation type	Simulation value	Latest value	Status	Server variable	Server type	
BeckhoffServer								
Server to simulation								
NextStation	AGV Cart.NextStation	Integer	500	500	✓	GVL500.qwNextAGVPos	INT	
OrderSignal	Robot100.OrderSignal	Integer	7	7	✓	GVL100.qwRobotOrder	INT	
OrderSignal	Robot500.OrderSignal	Integer	14	14	✓	GVL500.qwRobotOrder	INT	
OrderSignal	Robot200.OrderSignal	Integer	0	0	✓	GVL200.qwRobotOrder	INT	
OrderSignal	Robot300.OrderSignal	Integer	0	0	✓	GVL300.qwRobotOrder	INT	
RobotStarting	Robot100.RobotStarting	Boolean	FALSE	FALSE	✓	GVL100.qxStartABB	BOOL	
RobotStarting	Robot500.RobotStarting	Boolean	TRUE	TRUE	✓	GVL500.qxStartABB	BOOL	
RobotStarting	Robot200.RobotStarting	Boolean	FALSE	FALSE	✓	GVL200.qxStartABB	BOOL	
RobotStarting	Robot300.RobotStarting	Boolean	FALSE	FALSE	✓	GVL300.qxStartABB	BOOL	
Simulation to server								
AGVAtStation	Dockstation100.AGVAtStation	Boolean	FALSE	FALSE	✓	GVL100.ixAGVAtStation	BOOL	
AGVAtStation	Dockstation500.AGVAtStation	Boolean	TRUE	TRUE	✓	GVL500.ixAGVAtStation	BOOL	
AGVAtStation	Dockstation200.AGVAtStation	Boolean	FALSE	FALSE	✓	GVL200.ixAGVAtStation	BOOL	
AGVAtStation	Dockstation300.AGVAtStation	Boolean	FALSE	FALSE	✓	GVL300.ixAGVAtStation	BOOL	
AtAGVSignal	Robot100.AtAGVSignal	Boolean	TRUE	TRUE	✓	GVL100.ixRobotAtAGV	BOOL	
AtAGVSignal	Robot500.AtAGVSignal	Boolean	FALSE	FALSE	✓	GVL500.ixRobotAtAGV	BOOL	
AtAGVSignal	Robot200.AtAGVSignal	Boolean	FALSE	FALSE	✓	GVL200.ixRobotAtAGV	BOOL	
AtAGVSignal	Robot300.AtAGVSignal	Boolean	FALSE	FALSE	✓	GVL300.ixRobotAtAGV	BOOL	
AtBufferSignal	Robot100.AtBufferSignal	Boolean	FALSE	FALSE	✓	GVL100.ixRobotAtBuffer	BOOL	
AtBufferSignal	Robot500.AtBufferSignal	Boolean	TRUE	TRUE	✓	GVL500.ixRobotAtBuffer	BOOL	
AtBufferSignal	Robot200.AtBufferSignal	Boolean	FALSE	FALSE	✓	GVL200.ixRobotAtBuffer	BOOL	
AtBufferSignal	Robot300.AtBufferSignal	Boolean	FALSE	FALSE	✓	GVL300.ixRobotAtBuffer	BOOL	
Average update time: 22.4 ms Max update time: 62.0 ms Pairs with errors: 0 Average plugin time: 11.7 ms Max plugin time: 15.9 ms Errors on this run: 0								

Figure 26: Connected variables between Visual Components and TwinCat.

help of the Visual Components model. This was especially true for the robot-action function blocks where a test program and a test HMI (Figure 25) were used to test them. A part of the test program is shown in Figure 27.

At the project level, the whole PLC project was tested. The I/O station (STN500), the AGV and three assembly stations (STN100-STN300) in the virtual model were used to perform the required test experiments. During the test process, different scenarios of buffers contents were tried, and the behaviour and the performance of the connected virtual model were observed and evaluated.

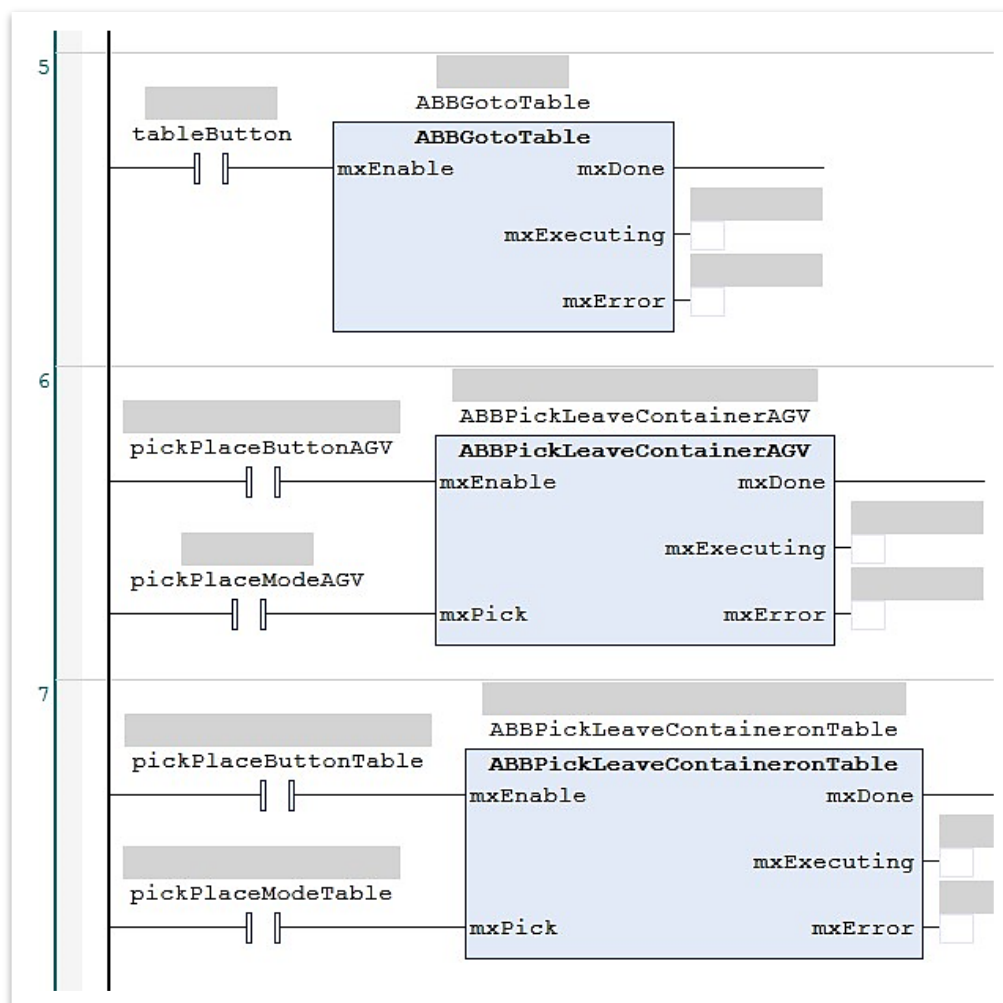


Figure 27: Part of the sub-program used to test the function blocks.

5 Results and discussion

In this chapter, the results of building the virtual model are presented, analysed, and discussed.

5.1 Result analysis

The main objective of this work was to investigate the possibilities and difficulties of applying the digital twin technology in academia and research to enable conducting of automation labs remotely. This was conducted by building a digital twin of the PTC automation line and testing the digital twin by trying different scenarios and configurations based on the initial contents of the buffers of the cells shown in Table 1. The main goal of these experiments was to evaluate the model performance and to investigate if the model works according to the provided specifications and requirements.

Table 1. Description of the initial contents of the buffers of the stations in three testing scenarios.

Station	Buffer	Scenario1	Scenario2	Scenario3
STN100	1	Red plates	Blue screws	Red cylinders
	2	Red cylinders	Green cylinders	Green plates
	3	Red screws	Red plates	Red plates
STN100	1	Green plates	Green screws	Blue plates
	2	Green cylinders	Red cylinders	Green cylinders
	3	Green screws	Blue plates	Red screws
STN100	1	Blue plates	Red screws	Green screws
	2	Blue cylinders	Blue cylinders	Blue screws
	3	Blue screws	Green plates	Blue cylinders
STN100	1	Red assembly container		
	2	Green assembly container		
	3	Blue assembly container		

In the first scenario, all assembly processes required to produce one product is performed completely in one assembly station. In the second scenario, the assembly processes required for every product are performed in three different assembly stations. In the last scenario, the buffers are filled randomly, and the assembly sequence required to complete one production cycle varies based on the contents of the buffers. The YouTube link to the third scenario simulation video can be found in Appendix C.

Running the virtual automation line using the three previous configurations gave almost the same results. The model was able to complete the production cycle smoothly and without any issues. This was especially true when the simulation speed factor in Visual Components was equal to or less than 1. With higher values of simulation speed factor, the model was mostly able to complete the production cycle, but the model performance was not always stable. The model did not run smoothly and there were time lags between some assembly processes in the experiments. This behaviour of the model is explicable in terms of the latency in the connection between the Visual Components and TwinCat. Another reason for this behaviour is that increasing the simulation speed factor affects only the simulation time of the model inside Visual Components, and it has nothing to do with the running speed of the PLC program. Some

signals sent from the PLC program to the Visual Components program will not be received at the time when they are needed. This problem could be solved by increasing the delay time between functions calls in the Python scripts in Visual Components, but this solution increased the total time required to perform one production cycle.

Behzad Far in [1] has used the virtual models built in this project in an experiment as a part of his master thesis work. The experiment involved creating a PLC program to produce one product using one assembly station. According to Behzad Far, the assembly process was tested in both manual mode and auto mode and the virtual mode functioned perfectly in both modes.

5.2 Discussion

The model in this work was built using three main applications, namely, Visual Components 4.2, TwinCat 3 and AutoCAD 2021. Building a well-functioning model requires therefore good knowledge of these applications. Although there are a lot of suitable resources to learn AutoCAD, that is not the case when it comes to learning Visual Components and TwinCat. The Courses and lessons offered by Visual Components Academy are useful resources to learn the basic of modelling in Visual Components, but they are probably not sufficient to acquire in-depth knowledge of the application. It is also very hard to navigate through and find information in the TwinCat help system which is the main source of information about the application and its programming languages.

Writing, testing, debugging and troubleshooting the code that controls the virtual model was extremely hard in this project. First of all, it was not easy to choose the proper languages that suit the different tasks in TwinCat. It was also very hard to debug the code in TwinCat especially for the code written in the ST language. The logic used in this work to control the virtual model is split into two parts, the PLC program code created in TwinCat and the Python scripts code created in Visual components. These two parts are connected using a communication protocol. This structure makes it even harder to debug and troubleshoot the code. It was also very tricky to use some function blocks inside sub-programs written in ST language. This was especially true for the timers and for the function blocks that contain timers.

6 Conclusion

At the end of this work, it has been proved that it is possible to use the digital twin technology to build virtual representations of the industrial-like lab equipment used in education and research. The virtual models can be used to conduct certain labs virtually. This answers the first parts of the research question of this work. The second part of the RQ, related to the limitations of using these virtual twins, is answered in the following sections.

6.1 Future Work and Research

The model developed in this thesis work is a digital model which is a digital twin with the lowest level of data integration. This is because in this model, there is no automated flow of data between the physical system and the virtual system, and the data is updated manually in both directions. Future extra work is required to convert this model to a real digital twin with full data integration where the real-time data can flow in both directions and the changes in the status of the physical system can directly affect the virtual model and vice versa.

There are two communication architectures available to connect TwinCat PLC programs with Visual Components models. These architectures are Beckhoff ADS and OPC UA. In the literature, there are not enough resources that address these two architectures. Further research might be required to investigate these architectures and/or to make a comparison between them.

This virtual model uses the Beckhoff ADS architecture, and it is highly recommended to test the model with the OPC UA architecture. This is important to investigate the difference between the two architectures and to evaluate their performance when they are implemented using the same PLC program and Visual Component model. This can help to choose the architecture that gives the best performance with the least time delay.

The current implementation of the PLC program includes all the code required to control the I/O station and all other assembly stations involved in the production cycle. When the virtual automation line will be used in education to help the students to test their PLC program offline, it is crucial to remove the code of the station/stations assigned to the students and to encrypt the code of the other stations.

This virtual model has not been tested yet in a practical way. It is therefore very helpful to find a system to collect and analyse feedback from the model users and to use the feedback to improve the model.

6.2 Critical Discussion

Although the virtual model of the PTC automation line can be used directly at least for educational purpose, some improvements may be done to make it more practical and effective. First of all, the content of the buffers of the stations is hard-coded inside the Visual Components model. This means that the content of the buffers is predefined, and it is, therefore, impossible to change the content of these buffers without modifying

the model. An effective solution for this problem will be utilizing an algorithm to randomly fill the buffers with the pallets and parts.

In the real automation line, every station has its controller and thus its PLC program. However, the system control implemented in this work uses only one PLC program to control all the stations in the automaton line. This configuration was very helpful in simplifying the implementation of the system control and its connection with the virtual model. This can, however, be seen as a drawback because, with the current configuration, it is hard to isolate or take control over individual stations in the line and also to assign tasks to the virtual model users.

6.3 Generalization of the result

Using virtual automation lines in education and research instead of the real ones can bring several common benefits. The virtual models increase visibility and give a better overview of the system components. The safety of the system is improved by using virtual models in comparison with the real automation lines. The virtual automation lines are environmentally friendly as they consume extremely less power and materials than the physical automation lines [51] and [52].

In summary, this work proves that it is very beneficial to use virtual automation labs in academia, especially when they are used in parallel with physical labs. The digital models increase the accessibility of modelled systems. This is especially crucial when the access to the physical systems is restricted due to time, location or other constraints such as the Covid pandemic restrictions in the last two years. The virtual automation labs are very useful tools for testing and debugging the system control programs before they are uploaded to the real controllers. They can also be used to optimize the modelled systems by using what-if scenarios.

The modelling processes and technique used in this work can be applied to all similar industrial-like lab equipment especially for those used in education. In some research labs where more accurate and realistic results are required, the model needs to fulfil extra requirements. Among other things, the latency in signal transfer between the virtual model and the control program should be as small as possible. For certain applications, it is also very crucial to build an extremely accurate physical model where the dimensions and positions of the modelled components are very close to those in the real system. Other factors should be considered while building the model. Examples of these factors the effect of inertia and gravity force of the modelled components on the model.

7 References

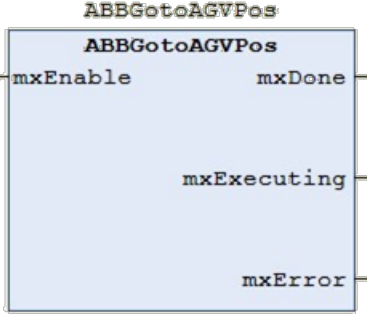
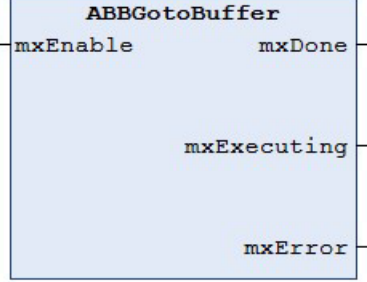
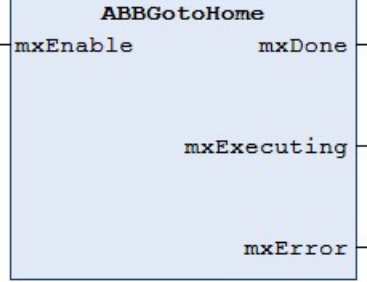
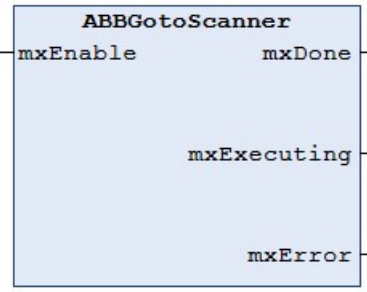
- [1] S. Behzad Far, *Digital Twin of Automation Line- Seamless Transfer*, Trollhättan: University West, 2021.
- [2] R. Parmar, A. Leiponen and L. D. Tomas, “Building an organizational digital twin,” *Business Horizons*, vol. 63, no. 6, pp. 725-736, 2020.
- [3] K. Agalinos, S. T. Ponis, E. Aretoulaki, G. Plakas and O. Efthymiou, “Discrete Event Simulation and Digital Twins: Review and Challenges for Logistics,” *Procedia Manufacturing*, vol. 51, p. 1636–1641, 2020.
- [4] F. Tao, M. Zhang and A. Nee, *Digital Twin Driven Smart Manufacturing*, London: Academic Press, 2019.
- [5] A. Rasheed, O. San and T. Kvamsdal, “Digital Twin: Values, Challenges and Enablers From a Modeling Perspective,” *IEEE Access*, 2020.
- [6] W. Kritzinger, M. Karner, G. Traar, J. Henjes and W. Sihn, “Digital Twin in manufacturing: A categorical literature review and classification,” *IFAC (International Federation of Automatic Control)*, 2019.
- [7] Y. Lu, C. Liu, K. I.-K. Wang, H. Huang and X. Xu, “Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues,” *Robotics and Computer-Integrated Manufacturing*, vol. 61, 2019.
- [8] A. Fuller, Z. Fan, C. Day and C. Barlow, “Digital Twin: Enabling Technologies,” *IEEE Access*, 23 June 2020.
- [9] F. Tao, Q. Qi, L. Wang and A. Nee, “Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0:,” *Engineering*, vol. 5, p. 653–661, 2019.
- [10] F. Tao, H. Zhang, A. Liu and A. Y. C. Nee, “Digital Twin in Industry: State-of-the-Art,” *IEEE*, vol. 15, no. 4, 2019.
- [11] C. Wu, Y. Zhou, M. V. P. Pessôa, Q. Peng and R. Tan, “Conceptual digital twin modeling based on an integrated five-dimensional framework and TRIZ function model,” *Journal of Manufacturing Systems*, 2020.
- [12] L. Sun, A. Pei, X. Qi, S. Cao, R. Yang and X. Liu, “Dynamic Analysis of Digital Twin System Based on Five-Dimensional Model,” *Journal of Physics: Conference Series*, 2020.
- [13] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang and A. Y. C. Nee, “Enabling technologies and tools for digital twin,” *Journal of Manufacturing Systems*, 2019.
- [14] K. P. White and R. G. Ingalls, “Introduction to simulation,” in *2015 Winter Simulation Conference (WSC)*, Huntington Beach, CA, 2015.
- [15] M. L. Loper, *Modeling and simulation in the systems engineering life cycle*, London: Springer, 2015.
- [16] V. P. Singh, *System modeling and simulation*, New Delhi: New Age International (P) Ltd, 2009.

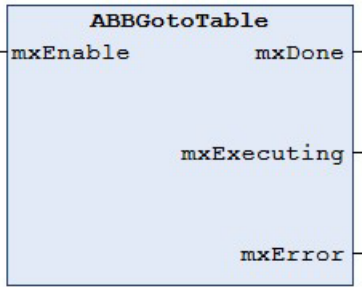
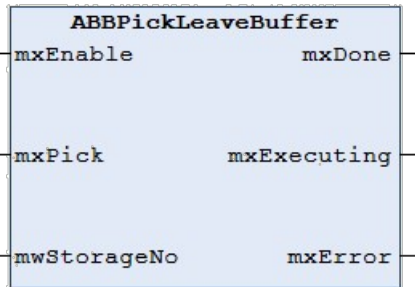
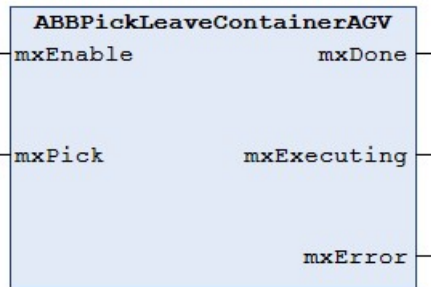
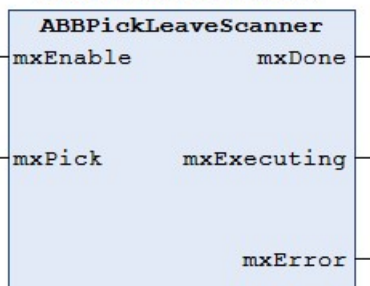
- [17] C. L. Dym, Principles of mathematical modeling, 2nd ed. ed., Amsterdam; Boston: Elsevier Academic Press, 2004.
- [18] A. C. Fowler, Mathematical models in the applied sciences, Cambridge: Cambridge University Press, 1997.
- [19] J. A. Sokolowski and C. M. Banks, Handbook of Real-World Applications in Modeling and Simulation, Hoboken N.J.: Wiley, 2012.
- [20] J. Norrman and J. Persson, *Virtual Production Line - Virtual Commissioning*, Lund: Lund University, 2018.
- [21] I. McGregor, "The relationship between simulation and emulation," in *The relationship between simulation and emulation*, San Diego, CA, USA, 2002.
- [22] M. Ayani, M. Ganebäck and A. H. Ng, "Digital Twin: Applying emulation for machine reconditioning," *Procedia CIRP*, vol. 72, p. 243–248, 2018.
- [23] B. Brooks, A. Davidson and I. McGregor, "THE EVOLVING RELATIONSHIP BETWEEN SIMULATION AND EMULATION: FASTER THAN REAL-TIME CONTROLS TESTING," in *Proceedings of the 2014 Winter Simulation Conference*, 2014.
- [24] B. R. Mehta and Y. J. Reddy, Industrial process automation systems, Oxford, UK: Elsevier Inc., 2015.
- [25] J. Machado and E. Seabra, *HiL simulation workbench for testing and validating PLC programs*, IEEE, 2013.
- [26] W. M. Hawkins, Automating manufacturing operations, New York: Momentum; London, 2013.
- [27] L. Durkop, L. Wisniewski, S. Heymann, B. Lucke and J. Jasperneite, "Analyzing the engineering effort for the commissioning of industrial automation systems," *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1-4, 2015.
- [28] M. Foehr, J. Vollmar, A. Calà, P. Leitão, S. Karnouskos and A. W. Colombo, "Engineering of Next Generation Cyber-Physical Automation System Architectures," *Multi-Disciplinary Engineering for Cyber-Physical Production*, 2017.
- [29] T. Coito, J. L. Viegas, M. S. Martins, M. M. Cunha, J. Figueiredo, S. M. Vieira och J. M. Sousa, "A Novel Framework for Intelligent Automation," *IFAC-PapersOnLine*, vol. 52, nr 13, p. 1825–1830, 2019.
- [30] M. P. Groover, Automation, production systems, and computer-integrated manufacturing, Fourth edition, Global edition ed., Pearson Education Limited, 20116.
- [31] F. Lamb, Industrial Automation Hands-on, New York: McGraw-Hill Publishing, 2013.
- [32] W. Bolton, Mechatronics - A multidisciplinary approach, 5th ed. ed., Harlow: Pearson, 2011.
- [33] W. Bolton, Programmable logic controllers, 5th ed. ed., Amsterdam; Boston: Newnes, 2009.
- [34] D. H. Hanssen, Programmable logic controllers - A practical approach to IEC 61131-3 using CODESYS, Chichester West Sussex United Kingdom: Wiley, 2015.

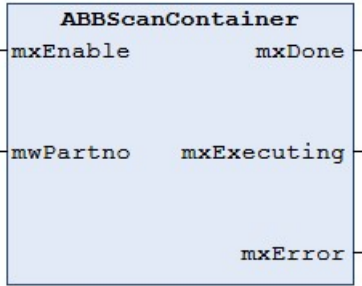
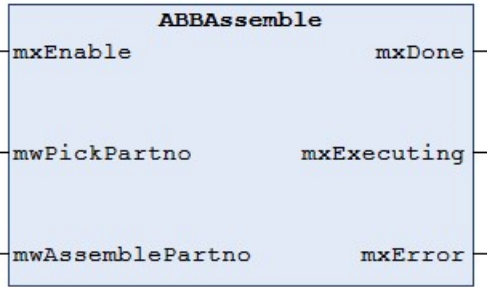
- [35] S. Vitturi, C. Zunino and T. Sauter, "Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G," in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, Bochum, Germany, 2013.
- [36] T. Imanto and A. Adriansyah, "Performance Analysis of Profinet Network in PLC-Based Automation System," in *2020 2nd International Conference on Broadband Communications, Wireless Sensors and Powering (BCWSP)*, Yogyakarta, Indonesia, 2020.
- [37] S. MacKay, *Practical industrial data networks: Design, installation and troubleshooting*, Oxford; Burlington, MA: Newnes, 2004.
- [38] M. Bennulf, F. Danielsson and B. Svensson, "Identification of resources and parts in a Plug and Produce system using OPC UA," *Procedia Manufacturing*, vol. 38, p. 858–865, 2019.
- [39] *The Commissioning Process; ASHRAE Guideline; 0-2005*, American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., 2005.
- [40] M. G. Tribe and R. R. Johnson, "Effective Capital Project Commissioning," in *2008 54th IEEE Pulp and Paper Industry Technical Conference - PPIC*, Seattle, WA, USA, 2008.
- [41] A. Almasi, "Pre-commissioning, commissioning and start-up of industrial plants and machineries," *Australian Journal of Mechanical Engineering*, vol. 12, no. 2, p. 257–263, 2014.
- [42] P. Hoffmann, R. Schumann, T. M. Maksoud and G. C. Premier, "Virtual commissioning of manufacturing systems: A review and new approaches for simplification," in *European Conference on Modelling and Simulation; ECMS*, Kuala Lumpur, Malaysia, 2010.
- [43] T. Lechler, E. Fischer, M. Metzner, A. Mayr and J. Franke, "Virtual Commissioning – Scientific review and exploratory use cases in advanced production systems," *Procedia CIRP*, vol. 81, p. 1125–1130, 2019.
- [44] H. Vermaak and J. Niemann, "Virtual commissioning: A tool to ensure effective system integration," in *2017 IEEE International Workshop 52017*, Donostia, San Sebastian, Spain, 2017.
- [45] J. Nibert, M. E. Herniter and Z. Chambers, "Model-Based System Design for MIL, SIL, and HIL," *World Electric Vehicle Journal*, vol. 5, p. 1121, 2012.
- [46] M. Ponchant, "D1.3 – Report on virtual test benches (MiL, SiL, HiL)," 2019.
- [47] M. Cech, A.-J. Beltman and K. Ozols, "I-MECH – Smart System Integration for Mechatronic Applications," in *2019 24th IEEE International Conference 92019*, Zaragoza, Spain, 2019.
- [48] C. G. Lee and S. C. Park, "Survey on the virtual commissioning of manufacturing systems," *Journal of Computational Design and Engineering*, vol. 1, no. 3, p. 213–222, 2014.
- [49] S. T. Mortensen, *Identify - Quantify - Obtain: Qualifications for virtual commissioning*, Aalborg, Denmark: Aalborg University Press, 2019.
- [50] C. R. Kothari, *Research methodology*, New Delhi: New Age International (P) Ltd., Publishers, 2004.
- [51] A. Øvern, *Industry 4.0 - Digital Twins and OPC UA*, Norwegian University of Science and Technology, 2018.

- [52] V. Potkonjak, M. Gardner, V. Callaghan, P. Mattila, C. Guetl, V. M. Petrović and K. Jovanović, “Virtual laboratories for education in science, technology, and engineering: A review,” *Computers & Education*, vol. 95, p. 309–327, 2016.

Appendix A: Robot-level function blocks

Function block	Description
	The function block is used to move the robot arm to the AGV position before picking from or placing plates on the AGV.
	The function block is used to move the robot arm to the buffer position at the working table before picking from or placing plates on the storages places at the buffer.
	The function block is used to move the robot arm to the home position.
	The function block is used to move the robot arm to the scanner position at the working table.

Function block	Description
<p style="text-align: center;">ABBGotoTable</p> 	<p>The function block is used to move the robot arm to the assembly position at the working table.</p>
<p style="text-align: center;">ABBPickLeaveBuffer</p> 	<p>The function block is used to pick/place pallets from/on a specific storage place on the table.</p>
<p style="text-align: center;">ABBPickLeaveContainerAGV</p> 	<p>The function block is used to pick/place pallets from/on the AGV surface.</p>
<p style="text-align: center;">ABBPickLeaveContaineronTable</p> 	<p>The function block is used to pick/place pallets from/on the AGV surface.</p>
<p style="text-align: center;">ABBPickLeaveScanner</p> 	<p>The function block is used to pick/place pallets from/on the scanner.</p>

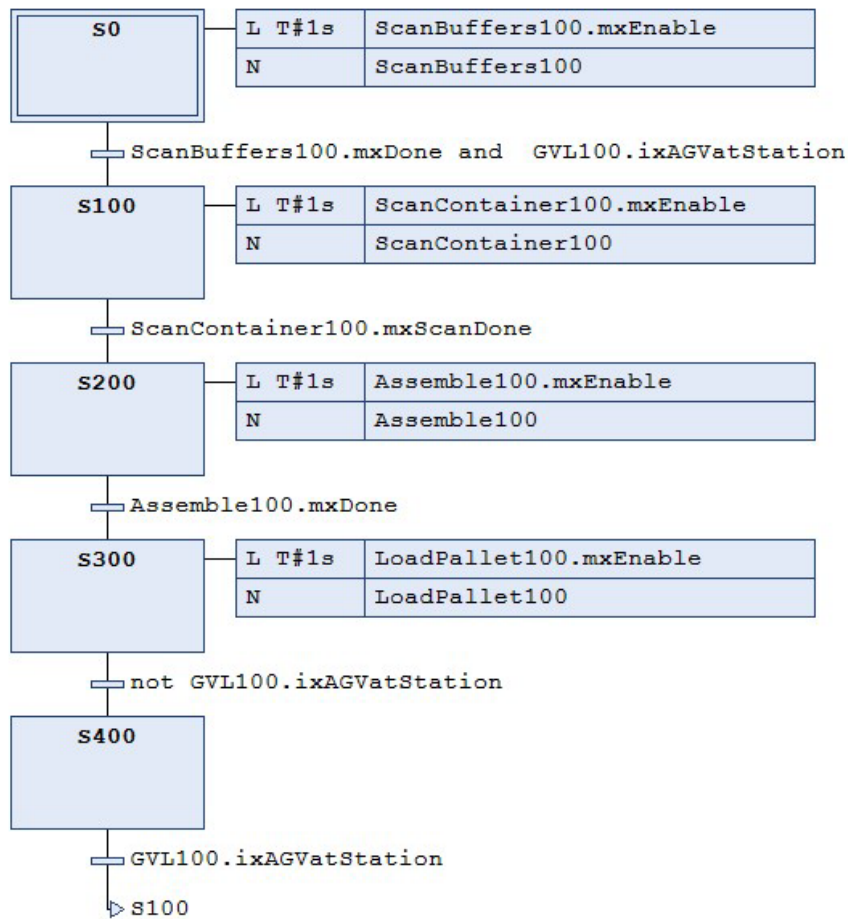
Function block	Description
<p style="text-align: center;">ABBScanContainer</p> 	<p>The function block is used to scan assembly pallets to detect the assembled part at a specific assembly position.</p>
<p style="text-align: center;">ABBAssemble</p> 	<p>The function block is used to pick a production part from a specific storage place and leave it in a specific position on the assembly pallet.</p>

Description of the inputs/outputs used in the function blocks.

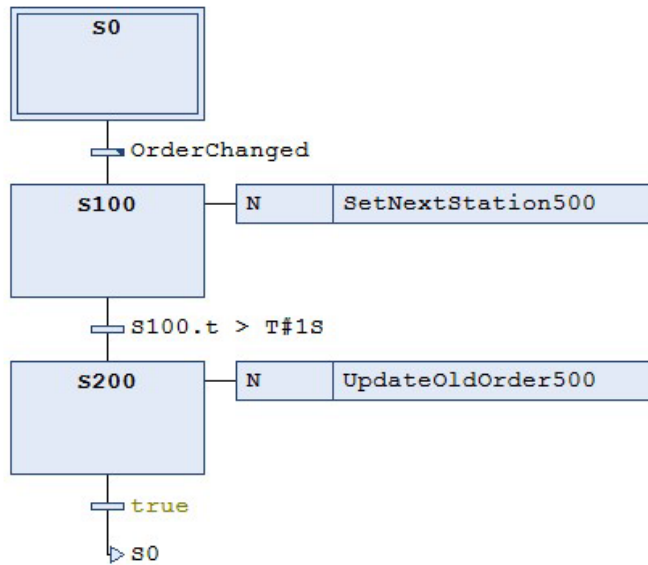
Pin	Type	Data type	Description
mxEnable	Input	Boolean	Enable the function block
mxPick	Input	Boolean	Picking mode (true: pick, false: place)
msStorageNo	Input	Integer	Define the storage ID
mwPartno	Input	Integer	The part ID on the assembly pallet
mwPickPartno	Input	Integer	The position of the part in the storage
mxDone	Output	Boolean	A flag indicates completing of FB execution
mxExecuting	Output	Boolean	True while FB is running
mxEError	Output	Boolean	A flag indicates FB execution error

Appendix B: Main control programs

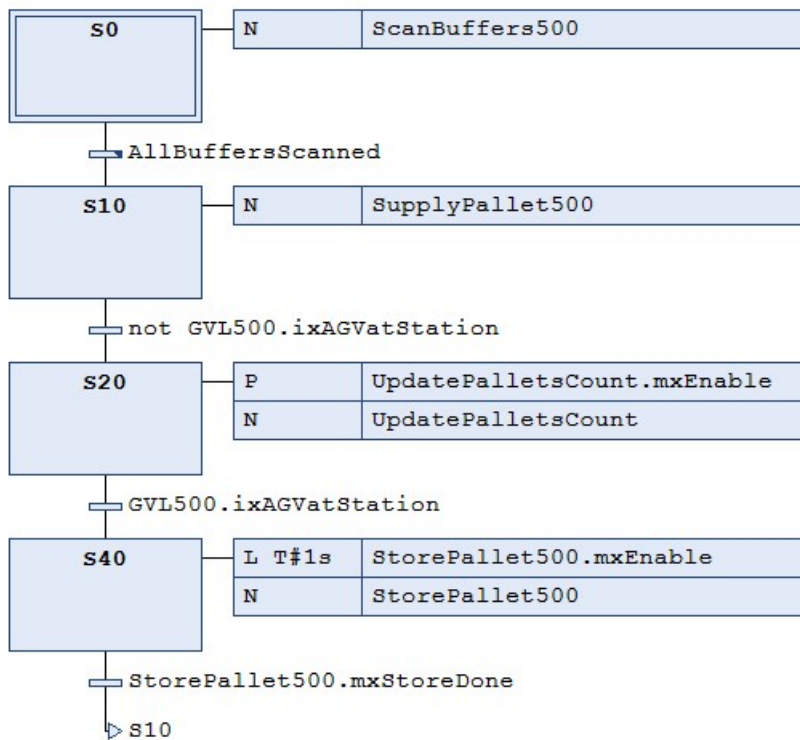
B1. STN100 control program:



B2. AGV control program:



B3. STN500 control program:



Appendix C: Simulation video link

<https://youtu.be/q2IQgSsj6mU>