Master Final Thesis

Master's Degree in Industrial Engineering

Usage of a mechatronics software for efficient machine development

REPORT

Author:	Berta Carbonell i Vilaplana
Director:	Pablo Murciego
Academical advisor:	Lluïsa Jordi i Nebot
Call:	April 2020

ETSEIB

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

ETSEIB

Abstract

Industrial machine manufactures face an increasing demand to deliver new high-quality products in a short time and at a reasonable price. Customers expect to receive machines that are productive, robust and flexible enough to adapt to their needs. Meeting these requirements while developing new technologies is a great business challenge. The development process is complex and with high degrees of uncertainty. Thus, the industry needs new development methods and tools that help them deal with complexity, be more efficient, and reduce costs.

These tools are software that emulates physical systems into virtual models. By using them, engineers can verify machine concepts, integrate designs or validate control programs to make informed decisions before any hardware is built.

In this context, this project shows a case study where a system is developed using a model-based approach. The system is a prototype meant to automatically pick 3D printed parts and place them in a packaging box. The machine model is developed to validate and optimize the pick and place algorithms before testing them in the physical equipment. The final simulation can understand the position, speed and acceleration values introduced by the user and provide a correct 3D visualization of the machine movement.

The model includes the physical properties and kinematics of all the moving machine components, the behaviour of the motor drives and the data exchange between the controller and the peripheries. The simulation is governed by the same control program that is executed in the real machine.

The virtual model is validated by a 3D visualization of the machine motion as well as an analysis of the motion graphs of several movement sequences. The project also includes an analysis of the costs and the environmental impact of this work.

This thesis also analyses the advantages and implications of using a virtual model to develop a machine. This is done through a literature review and observations made during the development of the case study.

ETSEIB

Table of Contents

1. Glossary

MCD:	Mechatronics Concept Designer
PLC:	Programmable Logic Controller
HP:	HP Inc.
VC:	Virtual Commissioning
HMI:	Human-Machine Interface
PC:	Personal Computer
TO:	Technology Object

2. Introduction

2.1. Objectives

Machine manufacturers are facing a higher demand to reduce the time-to-market of their new systems. At the same time, customers expect to receive high-quality, customizable products at a reasonable price. This creates a need to re-think the development cycle to continue being competitive in the market.

In the industrial machinery business, equipment must be productive, flexible, robust and inexpensive. Modularity and reusability also become very important to adapt to customers specific demands. Meeting these requirements while developing new technologies like additive manufacturing, is a great business challenge. The development process is complex and with high degrees of uncertainty. It needs a lot of innovation and multiple disciplines collaborating, agreeing on the requirements and integrating their work. It also requires ways to shorten or parallelize design activities to reduce lead time.

In this environment, the industry needs new development methods and tools that help them deal with complexity, be more efficient, and reduce costs. Many tools are already available in the market. These software tools enable building prototypes and testing machine control programs in virtual models that emulate physical systems. These models allow sharing system functionality and constraints, analysing design concepts and validate control programs to make informed decisions before any hardware is built. At the same time, they enable an earlier integration of the different disciplines and a parallelization of design activities.

Having said that, the use of these tools is not yet widespread across the industry, as it requires an initial investment on licenses and skilled professionals with the know-how to build and operate the machine models.

In this context, this project aims to develop a case study where a system is developed using a model-based approach. The system is a prototype built to automatically pick 3D printed parts and place them in a packaging box. The 3D printed parts can have any size and shape within the machine limits. To perform this operation, pick and place algorithms have to be developed, validated and optimized. In this scenario, it is very useful to have a tool to verify the correct operation of the code with a 3D visualization of the movement sequence of the machine in a virtual environment, limiting the risk of damaging physical equipment. Thus, in this work, a model of the system is going to be developed to validate the system control logic before testing it in the physical hardware.

This project also aims to analyse and discuss further the advantages and implications of model-based development from an organizational perspective. This will be done through a literature review and the observations made during the development of the case study.

2.2. Context

2.2.1. Usage of programmable logic controllers

HP Inc. develops 3D printing and digital manufacturing solutions for industrial applications. As most of the industrial machinery, some of these solutions use Programmable Logic Controllers (PLC) to manage the logic of the system.

PLC are off-the-shelf electronics. One of the reasons to use PLCs in industrial machines is that they are expected to be reliable in harsh electrical and physical environments. They also comply with industry standards and are certified by standard organizations, like CE or UL. Besides, it generally takes less time to program a PLC than an embedded system, and they are known to be modular and easily replaceable if a specific module fails. Although PLCs have a high cost compared to custom electronics, it usually makes sense to use them in low-volume industrial projects where time-to-market is very important.

The use of PLCs comes with a toolset that not only includes programs for software development but also tools that connect PLC code with mechatronic designs into a virtual model, known as the digital twin. Developing machines using this toolset can bring several benefits to machine manufacturers. Some of them are:

1. The machine can be commissioned before real hardware is assembled, enabling the parallelization of development activities and reducing the time to market.
2. Errors are more likely to be detected in an early stage of the development process and without damaging any physical equipment.
3. Software debugging and testing can be done without using hardware resources.
4. The software can be developed by multiple users from multiple sites.
5. It simplifies the outsourcing of some subsystems, in cases in which it is believed that other parties can develop those systems more quickly, or cheaper.
6. It makes easier to define and communicate the functionality of the system between engineering disciplines.
7. It assists in getting aligned with customers' requirements.

As the prototype in this project has a low-volume industrial application and a short development cycle, a PLC has been chosen to control it. The control program will be tested against a virtual model created with some of the PLC simulation tools.

2.2.2. HP Metal Jet technology

The present project has been developed within HP Inc. at the Research and Development department of the 3D Printing division. HP has developed two additive manufacturing technologies: HP Multi Jet Fusion for plastics and HP Metal Jet for metals. This project has been developed inside the HP Metal Jet program.

HP Metal Jet is an additive manufacturing technology for mass-production. Thus, it is important to automate certain parts of the manufacturing process to reduce time and labour costs. Automated processes also increase repeatability and yield by minimizing human errors. Pick and place operations, like the one performed by the machine in this project, are a good example of standard tasks that can be automatized.

HP Metal Jet technology falls into the binder jetting category inside additive manufacturing technologies [13] according to the classification in Figure 1. The principle of binder jetting is to build a part by glueing powder particles together employing a binding agent.

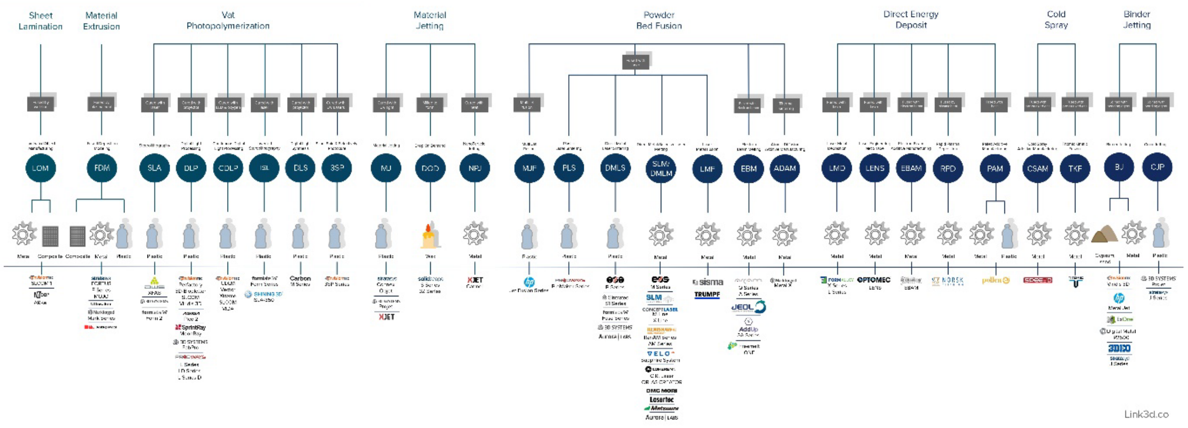


Figure 1: Additive Manufacturing Technologies [13].

The process to form a metal part using HP Metal Jet technology is shown in Figure 2 and described below.

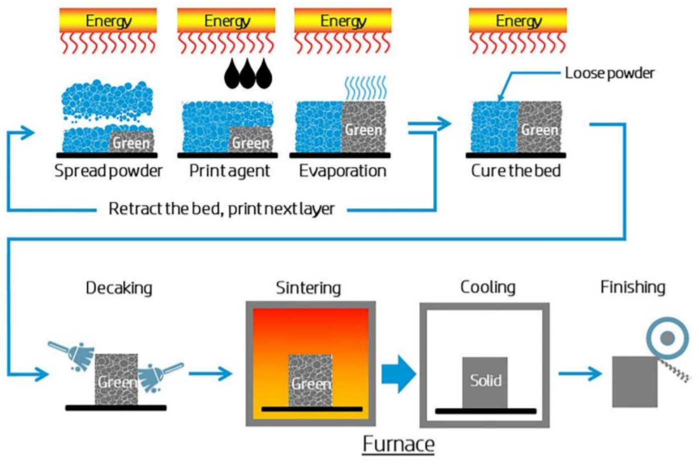


Figure 2: Schematic of HP Metal Jet additive manufacturing process [12].

Firstly, a build unit is loaded with powder and introduced to the printer. Then, the process continues as follows:

1. Spread powder: the process begins with a recoater spreading a layer of metal powder.
2. Print agent: printheads jet the binding agent at precise locations onto the powder bed to define the geometry of the parts.
3. Evaporation: the powder bed is heated throughout the printing process to evaporate the liquid components of the binding agent.
4. Retract the bed, print next layer: the powder bed is retracted the thickness of the printed powder layer, and the process repeats until the build is completed.
5. Cure the bed: the powder bed with its printed parts is heated to complete the evaporation of liquid components from the binding agent and to cure the polymers.
6. Decaking: the powder bed is cooled and then the parts can be extracted. Decaking is the process of removing loose powder from the surfaces of the parts. Once the parts have been decaked and extracted, the remaining powder can be processed and reused.
7. Sintering: the parts are moved into a furnace. At sintering temperatures, a diffusion process takes places at the surface of the metal particles. Also, the polymer from the binding agent decomposes. As a result, the metal particles are bonded together, and the final parts are formed.
8. Finishing: the parts may go through machining and polishing processes to meet dimensional and surface finish requirements.

Once the parts are finished, they are packed and shipped out to customers. The machine in this project is meant to be used in this final step to prepare the parts to be sent.

2.3. Scope

In this project, a system will be developed using a model-based approach. A virtual model of the system is going to be developed to validate the control program in a simulated environment. This will allow early detection of errors in the control software and thus reduced risk to cause any physical damage when testing the code in the real machine. It will also allow several developers to work on the control program simultaneously without having access to the prototype. In addition, building the simulation will provide a deeper understanding of the system.

The system is a prototype built to automatically pick 3D printed parts from a tray and place them in a packaging box. The 3D printed parts can have any size and shape within the machine limits. The mechanical solution of the system has already been designed. It consists of a gripper mounted on a 2-axis cartesian robot and a conveyor that transports the packaging box and the tray with the printed parts.

The machine model must be able to understand the position setpoints introduced by the user and provide a physics-based representation of the machine motion. The following machine characteristics will be simulated to achieve this goal:

- The physical properties and kinematics of all the machine components with motion. That is the rigid bodies mass and inertia, the joints, the sensors and the actuators.
- The behaviour of the drives (motor controllers).
- The data exchange between the controller and the peripheries (motors, sensors and drives).
- The controller governing the machine.

The simulation is going to be controlled by the same program that governs the real machine. However, the control program of the conveyor was not ready when starting this project. Thus, a program will be developed by the author of this work. This will be done to be able to test the simulation of the complete machine.

The virtual model will be validated by a 3D visualization of the machine motion as well as an analysis of the motion graphs of several given movement sequences. The project also includes an analysis of the costs and the environmental impact of this work. Lastly, some conclusions are derived from the experience of creating the virtual model of a machine during its development, determining if this can help to build new hardware more efficiently, with better quality and less cost.

3. State of the art

3.1. Machine development process

In the industry, all projects aim for achieving the maximum quality and the minimum cost within a tight schedule. While this is broadly accepted, it is very hard to achieve these three objectives simultaneously. Pretty often, project managers find themselves forced to sacrifice one or two of the goals to deliver a product that meets the specifications, the price, and the timeline required by the customers. It is quite common to encounter situations where companies go over the budget or are required to reduce the project scope to deliver a product on time. Other times they must delay the schedule to achieve the desired product value at a reasonable cost.

The root cause of these problems is linked to having high degrees of uncertainty. Uncertainty in content (for many projects the work has never been done before and the customer may not have fully determined what they want), in vendor performance and internal skill sets [4].

To reduce project timelines, machine manufacturers are introducing innovations in internal design processes. One of the approaches is the introduction of systems engineering and concurrent design.

3.1.1. Traditional sequential engineering

Machines are nearly always developed with a combination of disciplines: mechanical, fluids, thermal, electrical and software. In the traditional sequential engineering process, one discipline comes after another, as illustrated in Figure 3 and Figure 4.

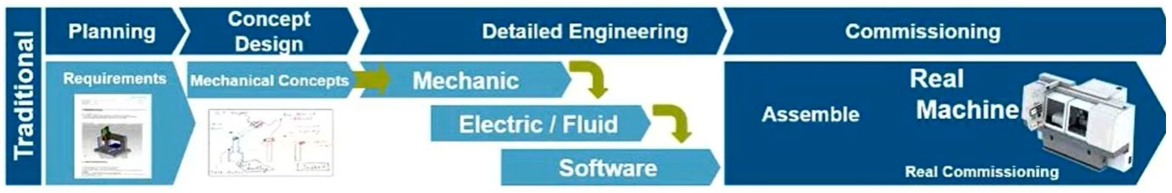


Figure 3: Traditional machine development process [8].

At the start of a project, external customers or internal project managers formulate the requirements, usually quite roughly. Requirements are often only considered during the scoping phase of the process and not revised afterwards. Then, conceptual designs are formulated, being mainly mechanically driven. Moving to the detailed design, mechanical engineers are the ones who start first. Their work results in the CAD of the machine, the drawings of the components and the manufacturing of all the parts. While the machine is being manufactured and build, mechanical engineers deliver the list of sensors and actuators to electrical engineers, who make the wiring diagrams and assign the inputs and outputs of the hardware. Only after the IO assignment is finished, the software developers begin to program the software that controls the machine. Therefore, software engineering tends to be the last step within the development process [3] and it is usually carried out during the commissioning phase.

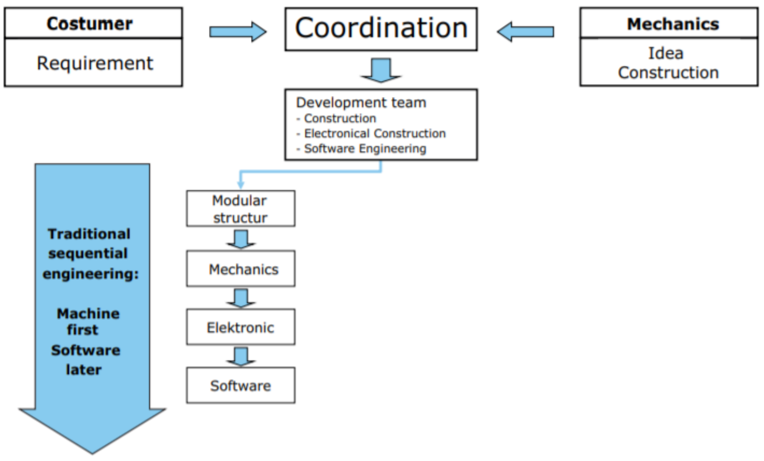


Figure 4: Traditional, sequential engineering process [7].

This sequential way of engineering comes with some drawbacks, especially in the age of industry 4.0, when the software content of a machine continuously increases [7], as shown in Figure 5.

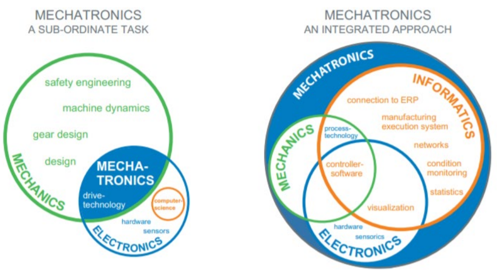


Figure 5: Evolution of software content of machines [7].

Some of these drawbacks are:

- Longer lead time: the commissioning of the assembled machines can take up to 15-25% of the total duration of the project according to [3]. If these activities are done sequentially and with the machine stopped, they can potentially cause a delay in the delivery date.
- Late design error identification: if control programs are debugged in the real machine, unexpected design errors and derived machine damage or people injury are likely to arise, as the software is not mature enough.

- Requirements mismatch: since customers' requirements are only revised at the beginning of the project, the focus on the detailed design can lead to a result that differs from the customers' expectations. Besides, when the functionality definition focuses only on the mechanical side of the machine, integration issues might appear in the later stages of development, affecting the quality of the final product or introducing unexpected outlays.

3.1.2. Systems engineering and concurrent design

New developing methods such as systems engineering and concurrent design have been spreading across the industry to improve the traditional process.

Systems engineering is a field that focuses on how to effectively design and manage complex systems with an interdisciplinary and holistic approach. The development process starts with a detailed definition of the requirements of the system. This can be done through a conceptual system model where the various engineering disciplines can work together to define the functions and processes in every given operating mode of the machine. As soon as these requirements are stated, each discipline can use its tools to perform the detailed design. Then again, simulation models are used to check the proper functioning of the software and integrate and validate the functionality of the machine. This allows the detection of errors long before any physical prototype is built and substantial time savings. This process is illustrated in Figure 6 and Figure 7.

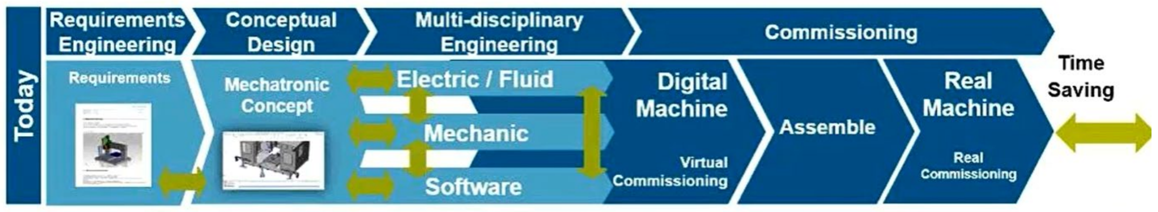


Figure 6: Today’s machine development process [8].

Systems engineering also involves customers in the design process and ensures that the system developed is viable throughout its life [8]. Requirements are tracked through the design process and not only in the beginning. The result is a higher quality product developed more efficiently.

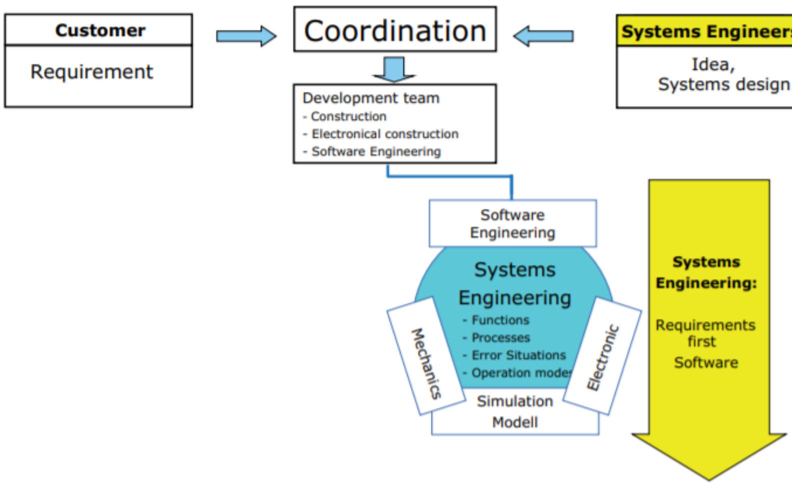


Figure 7: Engineered systems development process [7].

3.2. Virtual Commissioning

In traditional machine development, software programs are tested, in the first place, in the control software development tools themselves, without real feedback of the process behaviour. Then, with the system already assembled and connected, the commissioning is carried out. However, with the growth of the computational capacity of computer equipment and the irruption of tools for emulating machinery and processes, we are now able to test the software before the real commissioning. This process is called virtual commissioning (VC) [1].

Virtual commissioning consists of using simulation tools to emulate a physical system (mechanical components, sensors, actuators, motor drives, valves, pumps, controllers, HMI, etc.) in a virtual environment during the development phase. This allows coding of the system before the physical hardware is installed. It also gives the developer the option to confirm the correct operation of the code as well as the physical movement and feedback from the hardware before any equipment is built. Virtual commissioning also allows for easy reconfiguration of an existing system, where process, software or hardware changes can be made in the digital model [2].

The real-time digital representation of the system is known as the digital twin. Figure 8 provides a schematic of the elements that can be emulated through simulation tools (events, flow, behaviour, geometry and kinematics) and the hardware than can be tested against these simulations (RC, HMI, PLC, MES and ERP). In this project, a PLC program is validated against a simulation of events, behaviour, geometry and kinematics.

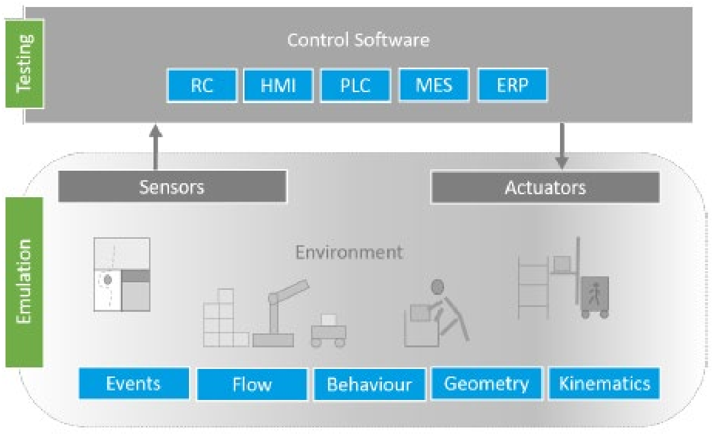


Figure 8: Virtual commissioning of control systems [5].

Virtual commissioning can be performed in two configurations: hardware-in-the-loop and software-in-the-loop. They are described in the following two chapters.

3.2.1. Hardware in the loop

Hardware-in-the-loop (HIL) configuration requires having a virtual model of the process or machine and a real controller. The real PLC or control hardware is connected to the model. It might involve some additional hardware and Fieldbus devices apart from the controller. This configuration is generally more expensive, but the testing environment is very similar to the final one, as the software is directly tested on the controller. Figure 9 provides a schematic of a HIL configuration.

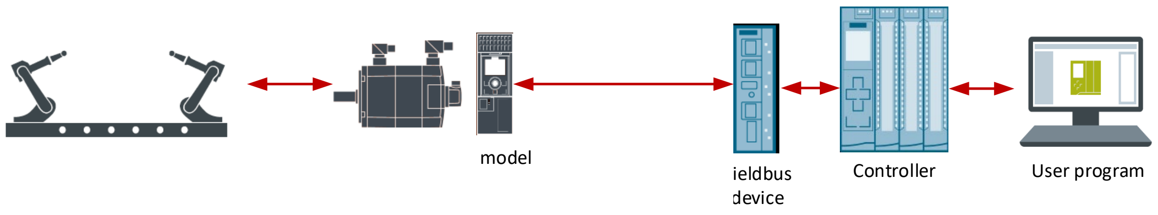


Figure 9: Hardware-in-the-loop (HIL) configuration.

3.2.2. Software in the loop

In a software-in-the-loop (SIL) configuration, the model of the system is controlled by a virtual controller. This provides a very flexible solution as the only hardware needed is a personal computer. The software can be developed simultaneously by different engineers based around the world against the same system model. However, every simulation software brings a license cost associated with it, which must be considered when evaluating emulation benefits. Figure 10 provides a schematic of a SIL configuration.

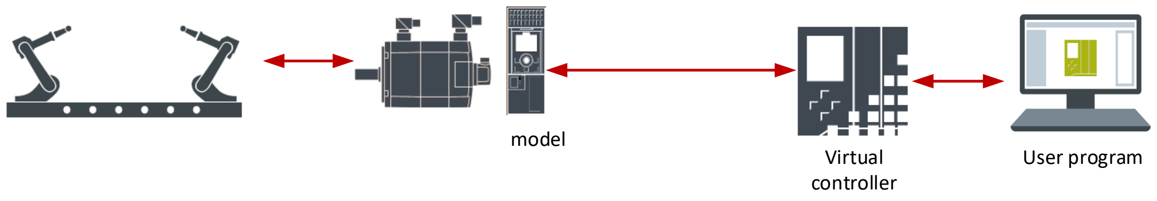


Figure 10: Software-in-the-loop (SIL) configuration.

3.2.3. Virtual commissioning software

There are numerous tools for emulating industrial systems and processes. ABB Robot Studio, Excelgo Xperior, Simumatik3D or Siemens NX Mechatronics Concept Designer are tools to model the geometry and kinematics of the system [1].

Other tools such as Siemens SIMIT or Simulink can be used for modelling machines and processes in applications such as water treatment plants. SIMIT also simulates the behaviour of devices like sensors or drives for motors, valves, and pumps. However, it does not model the geometry or kinematics of the system. Similar tools like Simscape and Simcenter Amesim enable to create simulations with libraries of ready-to-use models of physical systems (electric motors, refrigerator systems, hydraulic actuators, etc.).

Tecnomatix Process Simulate is a tool that provides the possibility of simulating the motion of a robotic cell within a production line and detecting collisions between a robot and the production machine or between two robots.

Figure 11 provides an overview of the simulation levels of Siemens virtual commissioning software. On the automation level, PLCSIM Advanced and WinCC are the Siemens tools that simulate the PLC and the human-machine interface (HMI), where the user can interact with the system and introduce commands to control it.

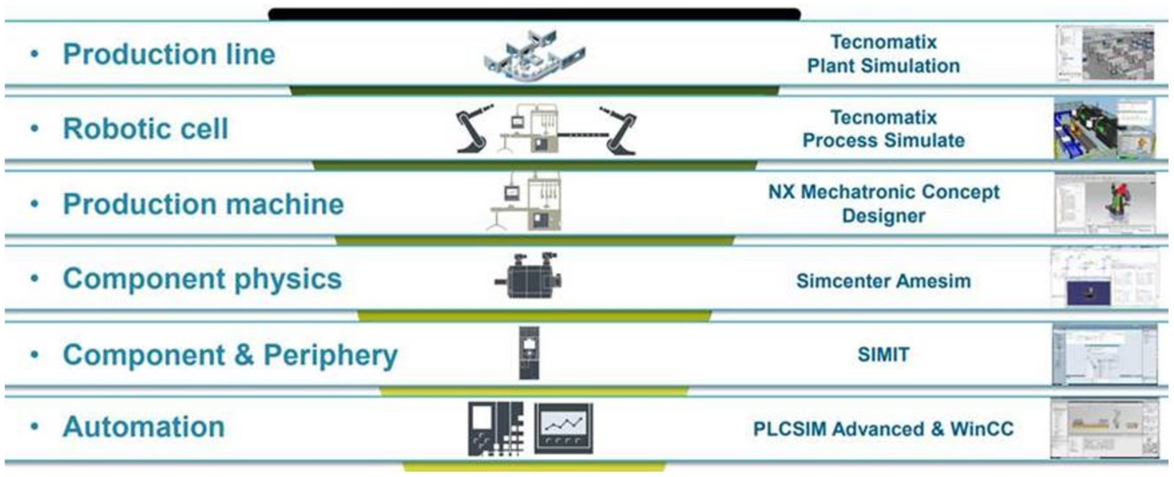


Figure 11: Simulation levels of Siemens virtual commissioning software [8].

All these tools make the creation and adjustment of the model much easier and faster than if it had to be built from scratch without employing reusable components.

Siemens virtual commissioning programs provide a fast, easy and reliable integration with Siemens PLCs or their simulation in PLCSIM Advanced. As our machine is controlled by a Siemens PLC, it makes sense to use the Siemens toolkit. Therefore, NX Mechatronics Concept Designer, SIMIT and PLCSIM Advanced will be used to build the digital twin of our system and perform the virtual commissioning. The purpose and characteristics of these tools are explained with more detail in chapter 4.3.

4. System description

4.1. Functionality

Virtual commissioning can be performed in a wide variety of systems. This project shows an example of one of many applications.

The purpose of the machine in this work is to perform a pick and place operation to transport 3D parts manufactured using the HP Metal Jet technology, explained in chapter 2.2.2. HP Metal Jet parts can have any size and geometry within the limits of the printing bucket. Figure 12 shows sample parts produced with this technology.

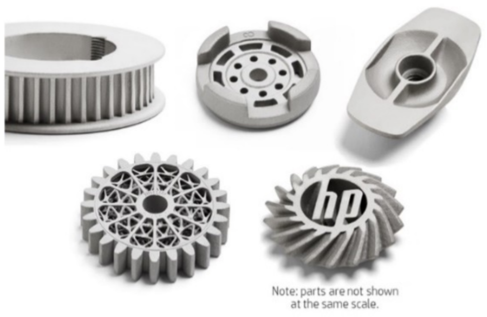


Figure 12: Sample 316L stainless steel parts created using HP Metal Jet technology [12]

Throughout the production process, the 3D printed parts are moved from one process to another on a tray. When they reach the end of the manufacturing line, the parts are placed in packaging boxes before shipping them out to customers. In a mass-production environment, it is key to automate this task to reduce time, labour costs and human errors. That is why a system is being developed in HP to pick and place the final 3D printed parts automatically.

The system must be able to identify the location and geometry of the parts within the tray that carries them. At the same time, it must be able to position a gripper at any height in the range of 0 to 200 mm. Finally, it must be capable of reaching any point in the XY dimensions of the tray and the packaging box where parts will be placed. Figure 13 shows a schematic of the system architecture.

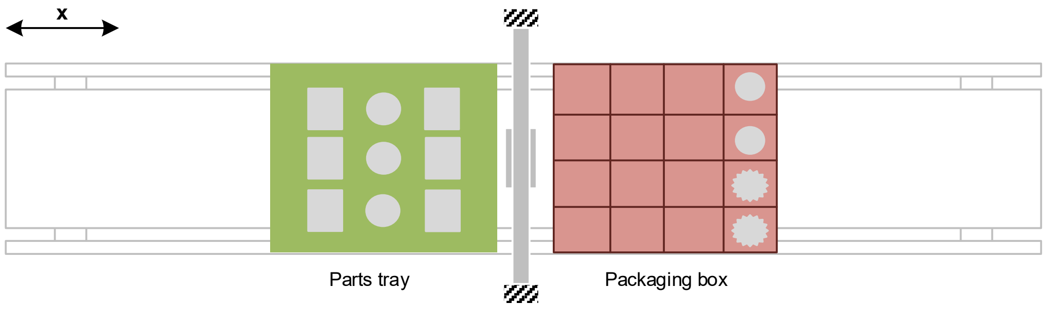


Figure 13: Top view of a schematic representation of the system architecture.

The gripper is positioned using a 3-axes actuator, which is described in chapter 4.2. The geometry and location of the parts are acquired with a 3D camera at the beginning of the process. Having this setup, pick and place algorithms have to be developed to move any set of 3D printed parts to the respective packaging box. Those algorithms process the images coming from the 3D camera, get the coordinates of every printed part of the tray, define the movement sequence and send it to the 3-axis actuator. This is done in a PC. The PC then communicates with a PLC that controls the operation of the 3 axes of the actuator. Figure 14 provides a schematic of the interaction between the hardware elements.

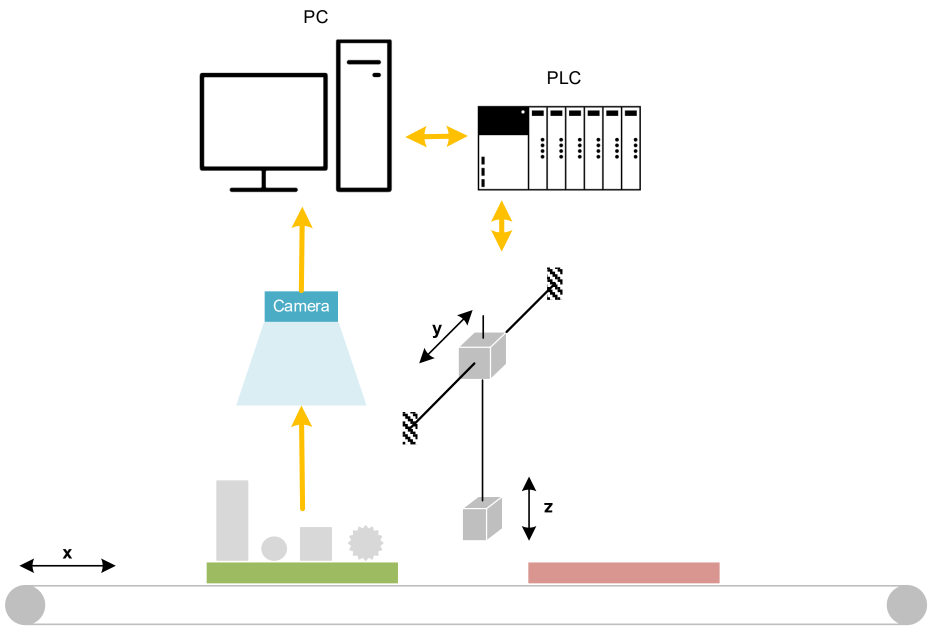


Figure 14: Front view of a schematic representation of the system architecture.

The program that controls the simulation is the same that governs the physical machine. Before starting any movement sequence, it requires homing the 3 axes. Then, it starts reading cyclically the position setpoints sent by the program running in the PC. In this project, all the hardware of the system has been simulated except for the camera and the PC. Thus, the position setpoints of the 3 axes are introduced manually. The virtual model provides a visualization of the movement sequence of the system. The simulation tools used are described in chapter 4.3.

4.2. Hardware architecture

The mechanical solution for the pick and place process is a 3 axes actuator system composed by a 2-axis cartesian robot and a conveyor belt. The cartesian robot moves in the y and z axes, as shown in Figure 14. The z-axis carries the gripper to pick and place the 3D printed parts.

At the beginning of the process, the 3D camera captures the location and geometry of the printed parts, which are transported on a tray. Then, the pick and place algorithms are calculated in a PC and the position setpoints are sent to the PLC that controls the machine. The tray is then moved under the cartesian robot by the conveyor belt and the cartesian robot is displaced to pick one part from the tray. The conveyor belt also transports the box where the 3D printer parts will be packed. Then, the conveyor belt moves until a line of slots of the packaging box is under the cartesian mechanism. After that, the z and y axes move again to place the carried part inside the box. Finally, the conveyor belt moves the tray under the cartesian robot to start the process again for the next part.

The characteristics of each piece of equipment used in this solution are described in the following chapters. The datasheets of the simulated devices are also available in Appendix B. Some of this information will be later used as an input for the simulation.

4.2.1. Cartesian robot

The cartesian robot consists of two axes. The y-axis provides horizontal movement to the carriage that transports the z-axis. The z-axis provides vertical movement to the gripper, which is not included in this simulation as it is not selected yet. Figure 15 shows the cartesian robot in the home position.

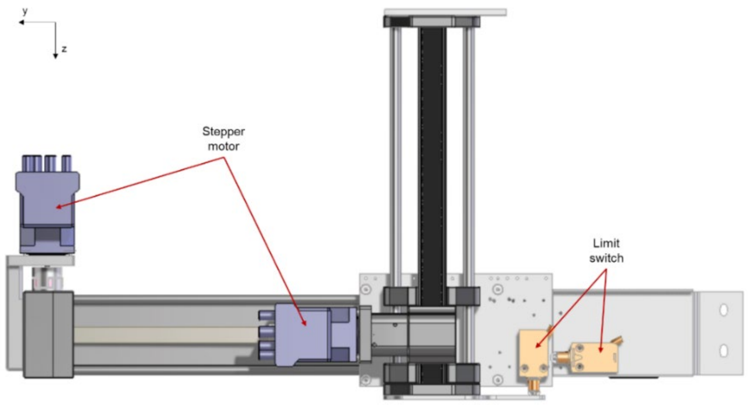


Figure 15: 2-axis cartesian robot at home position

The mechanical characteristics of the two axes are the following:

- Y-axis: the stroke of the mechanism is 297,33 mm, although it is limited by software to 280 mm. The transmission is done through a timing belt and pulley mechanism, which is shown in Figure 16. The linear travel per motor revolution is 70 mm. The gear ratio is i=1. The pulley shafts are mounted on ball bearings and an elastic coupling transmits the torque from the motor shaft to the pulley shaft.

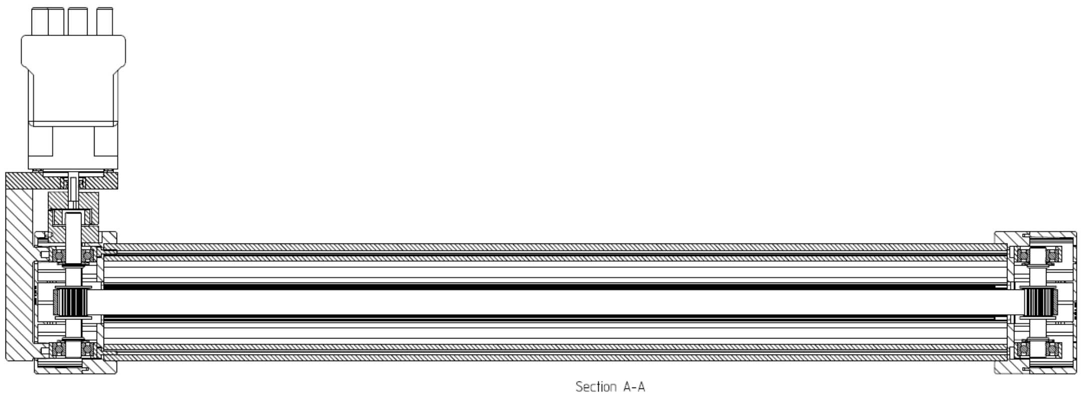


Figure 16: Section of the Y-axis mechanism

- Z-axis: the stroke of the mechanism is 350 mm, although it is limited by software to 240 mm to avoid a collision between the gripper and the tray carrying the parts. The transmission is done directly from the motor axis via a rack and pinion mechanism. The linear travel per motor revolution is 72,25 mm. An elastic coupling transmits the torque from the motor shaft to the gear shaft. Figure 17 provides a close view of the z-axis mechanism.

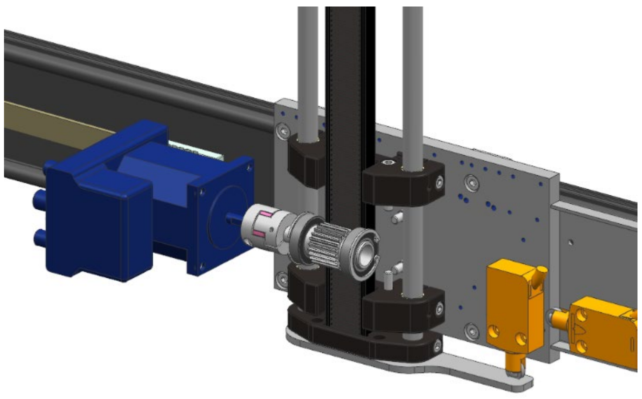


Figure 17: Detail of the Z-axis mechanism.

The y and z axes of the cartesian mechanism are an Igus S.L. module with reference DLE-LG-0158. The power in both axes is provided by an integrated stepper motor from Schneider Electric, the Lexium MDrive with reference LMHCE517C. It has a built-in controller and an incremental magnetic encoder. The drive is integrated with the motor. Each motor turn is equivalent to 51200 micro-steps.

The home positions are detected by two limit switches from Telemecanique (reference XCMD4102L5EX).

4.2.2. Conveyor

The x-axis of the mechanism is a conveyor belt, shown in Figure 18. The belt carries the tray with the 3D printed parts and the packaging box. The tray and the box are attached to the belt by friction. Thus, the belt has been selected to prevent slip.

The effective length of the conveyor is 1650 mm. It is powered by a brushed DC motor form Dunkermotoren, reference GR 63X55 40V. It has a nominal speed of 3450 min^{-1} and a no-load speed of 3600 min^{-1}. The motor comes with a brake and an incremental encoder of the same provider, with reference E-90 and RE-30 respectively.

The drive used to control this motor is the SIMATIC Micro-Drive PDC from Siemens.

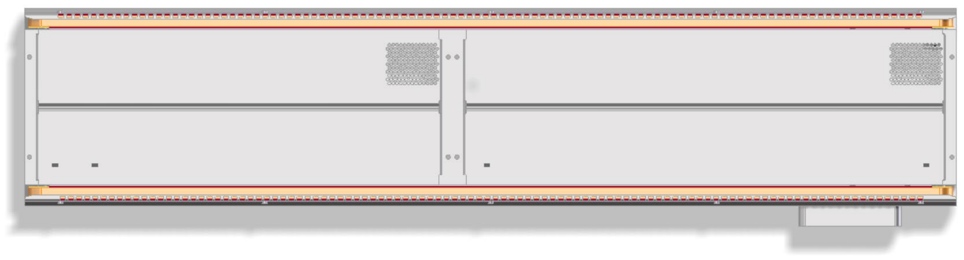


Figure 18: Top view of the conveyor.

The motor rotation is converted into the linear movement of the belt in three steps, illustrated in Figure 19. The first step of the transmission is a planetary gearbox (Dunkermotoren reference PLG52) with i=1/288 gear ratio. The second stage is a timing belt and pulley with i=1 gear ratio that transfers the power to a second axis. The third step is another mechanism consisting of a timing belt, a timing pulley and several idle pulleys as belt tensioners on each side of the conveyor. This system converts the rotary movement of the second axis to the linear movement of the conveyor belt. The linear travel per revolution is 150 mm. Figure 20 shows a lateral view of the conveyor transmission system.

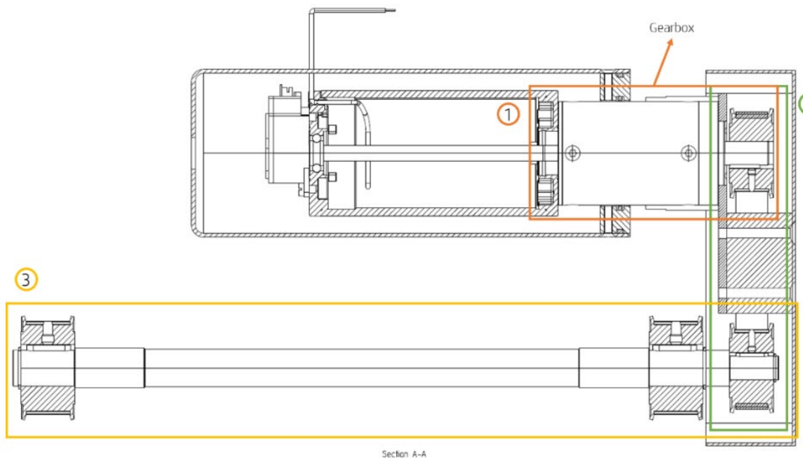


Figure 19: Transmission steps of the conveyor belt actuation.

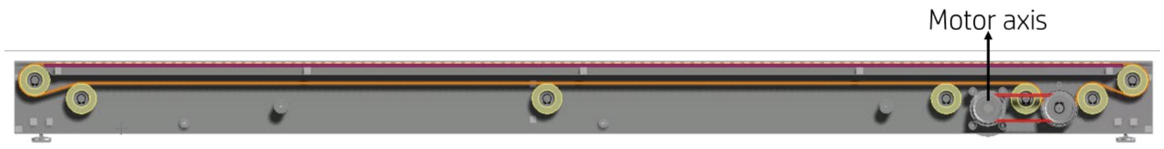


Figure 20: Lateral view of the conveyor transmission mechanism.

The home position is detected by a limit switch from Telemecanique (reference XCMD4102L5EX) that is activated by the tray that carries the 3D printed parts. Figure 21 shows the location of the X-axis home sensor.

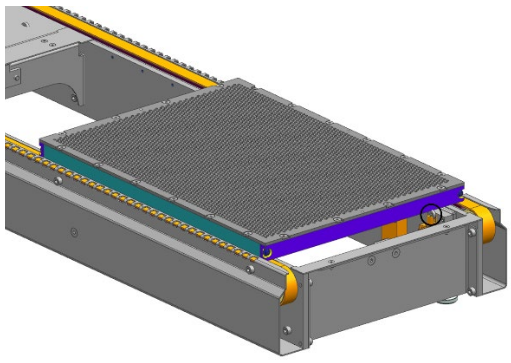


Figure 21: Location of the X-axis home sensor.

4.2.3. Camera

The camera used to capture the height and location of the printed parts is the Intel® RealSense™ D415 depth camera. It is a low-cost camera designed for easy set-up and portability, that allows building applications using sample code. Despite its small form factor, it integrates two depth sensors (IR cameras), an RGB sensor, an infrared projector and a low power vision processor. It has a standard depth field of view (FOV) of 65°±2°x 40°±1° x 72°±2°

that is well suited for accurate computer vision. It supports indoor and outdoor lighting conditions. The distance to objects is obtained through triangulation by comparing the distances measured by the two depth sensors.

4.2.4. Controller

The controller used in this system is a programmable logic controller (PLC). A PLC is an industrial computer that is programmed to monitor input signals, perform logical operations, and trigger specific output signals. In most cases, PLCs are modular, which means you can install I/O modules in a plug-and-play manner. The PLC used for this application is the CPU 1512SP-1 PN provided by Siemens. Apart from the CPU, it also includes a power supply, a rack to plug in the modules and the input and output cards.

Since a PLC is a dedicated controller, it will only process one program repeatedly. One cycle through the program is called a scan time. It involves reading the inputs from the other modules, executing the logic based on these inputs, and then update the outputs accordingly. The scan time happens in the range of 1ms [14].

Siemens PLCs mainly use the Profinet communication protocol to communicate with other devices like drives or HMIs. Profinet is an Ethernet-based industrial communications protocol that works at 100 megabits per second. Due to its high-speed operation and the response time of less than 1ms, Profinet is suitable for high-speed applications. Profinet cables have robust shielding to function well in harsh environments, which make them appropriate for PLC applications.

4.3. Simulation environment

As explained in chapter 3.2.3, this project uses the Siemens virtual commissioning toolset to build the model of the mechatronic system. Figure 22 shows the functions and interactions between the four programs composing the toolkit:

- TIA Portal
- PLCSIM Advanced
- SIMIT
- NX Mechatronics Concept Designer

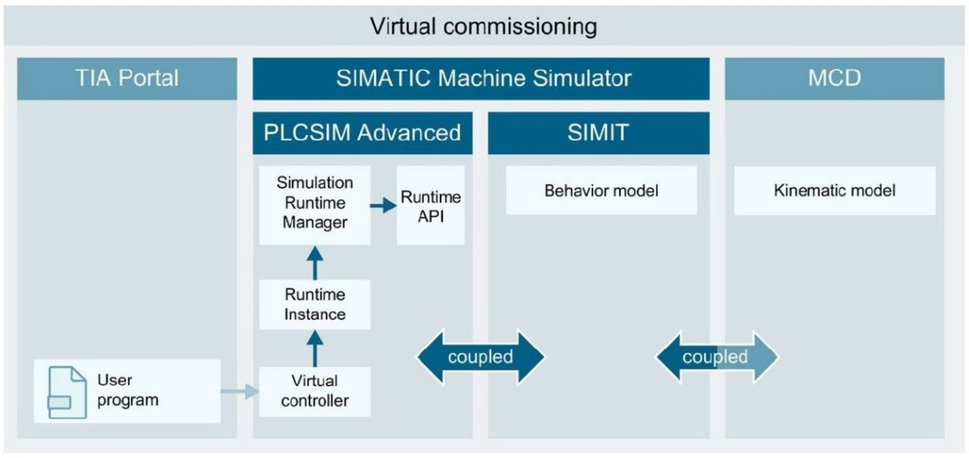


Figure 22: Interactions between the software of the Siemens virtual commissioning toolkit [9].

TIA (Total Integrated Automation) Portal is the software used to program the code of the PLC, the so-called user program. A TIA Portal project also includes the hardware configuration: the controllers, screens, periphery, actuators and communication.

Once the TIA Portal project is finished, it can be tested and validated using PLCSIM Advanced, which simulates the controller (the PLC). PLCSIM Advanced also takes care of the communication and synchronisation with external applications, like MCD or SIMIT.

NX Mechatronics Concept Designer (MCD) is the tool used to build the physical and kinematic model. The user can introduce the physics and kinematics of the machine components over a previously existing 3D model (CAD) of the machine. That includes defining the mass and inertia of the rigid bodies, the collision bodies, the kinematic joints and constraints, the sensors and the actuators. With this model, the engineer can generate and validate the operation sequence and motion of the machine or the load curves of the actuators. It can also be used to detect collisions and mechanical interferences happening during the machine operation. The aim of this simulation is not to conduct a detailed analysis and validation of the specific characteristics of individual components (vibration, thermal, frequency response, or stress), but to simulate the overall machine behaviour [6].

MCD is also used by mechanical engineers to create and validate machine concepts in the early stages of development. Likewise, the data of their models can be used by electrical designers to select sensors and actuators. At the same time, automation engineers can use the cams and operation sequence information from the same model for software development [11].

SIMIT emulates the behaviour of devices like sensors or drives for motors, valves, and pumps. It can also simulate process variables like temperature, pressure or flow. The models are built using standard libraries or user-made components. SIMIT is also used to communicate and synchronize the mechatronics model in MCD with the controller model in PLCSIM Advanced. It automatically detects the signals coming from both programs, which are transformed and adapted to be understood by both tools.

SIMIT also provides a user interface with graphics and controls that can be used to train operators, monitor or control signals of the machine and force errors to test the software.

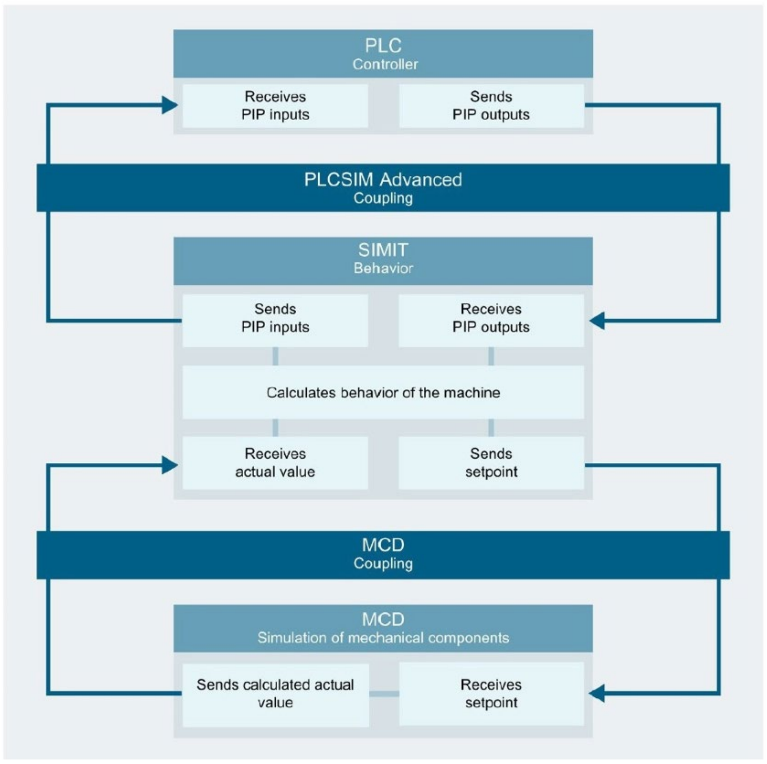


Figure 23: Data exchange between PLC, SIMIT and MCD [9].

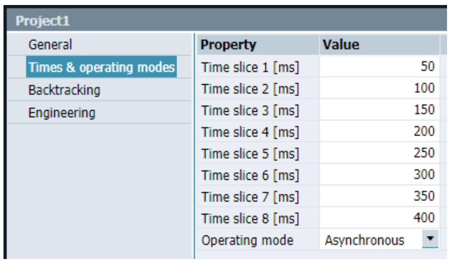
Figure 23 explains the interaction and signal exchange between PLCSIM Advanced, SIMIT and MCD:

- SIMIT receives the outputs from the PLC.
- SIMIT sends setpoints (e.g. speed setpoints, position setpoints, binary signals) to MCD.
- MCD simulates the mechanical components movements based on the defined degrees of freedom and received setpoints.
- MCD sends the calculated actual values (e.g. speed actual values, position actual values, binary signals) to SIMIT.
- SIMIT calculates the behaviour of the machine using the drive logic defined.
- SIMIT sends the inputs to the PLC.

4.3.1. Time management

4.3.1.1. Time slices

In a simulation, the cycle time is the time frame in which the calculations of the models have to be executed and the data has to be exchanged. In SIMIT, the cycle times are set using one of the eight available time slices in the project, as shown in Figure 24. The minimum cycle time that can be set is 1ms.



Property	Value
Time slice 1 [ms]	50
Time slice 2 [ms]	100
Time slice 3 [ms]	150
Time slice 4 [ms]	200
Time slice 5 [ms]	250
Time slice 6 [ms]	300
Time slice 7 [ms]	350
Time slice 8 [ms]	400
Operating mode	Asynchronous

Figure 24: Project times and operating modes definition in SIMIT.

Then, each coupling, chart or individual component can be assigned to a different time slice. For couplings, the time slice is the time rate at which they exchange data. For charts and individual components, it corresponds to the assigned calculation time.

4.3.1.2. Operation modes

SIMIT has 3 modes of operation when it comes to time management: asynchronous, synchronous and bus synchronous. The operating mode is selected in SIMIT in the window shown in Figure 24.

Asynchronous operating mode

In asynchronous mode, the calculation of the model and the exchange of signals in the coupling are triggered cyclically. That means that it is time controlled. The model calculation continues even if the signal exchange with one coupling is blocked by a communication problem or not calculated within the allocated time. However, the processing stops if one or more machine models in SIMIT are not calculated within its allocated time. In that case, it will only continue when all SIMIT models meet their cycles again. There is a time offset between the data communication of couplings and the model calculation, as shown in Figure 25.

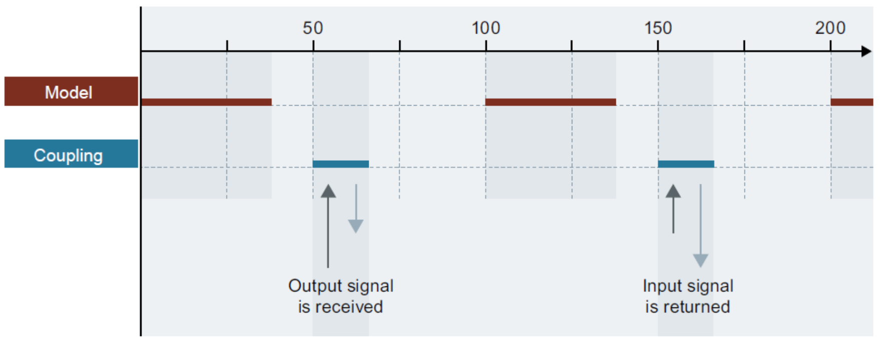


Figure 25: Time offset between the signal exchange and the model calculation in asynchronous operation mode [10].

Figure 26 illustrates an example of a simulation performed in asynchronous mode. The system has a coupling with a simulated controller in PLCSIM Advanced. The simulation model is calculated in SIMIT. Both the controller and the simulation model have a cycle time of 100ms and they are triggered with a time offset between them.

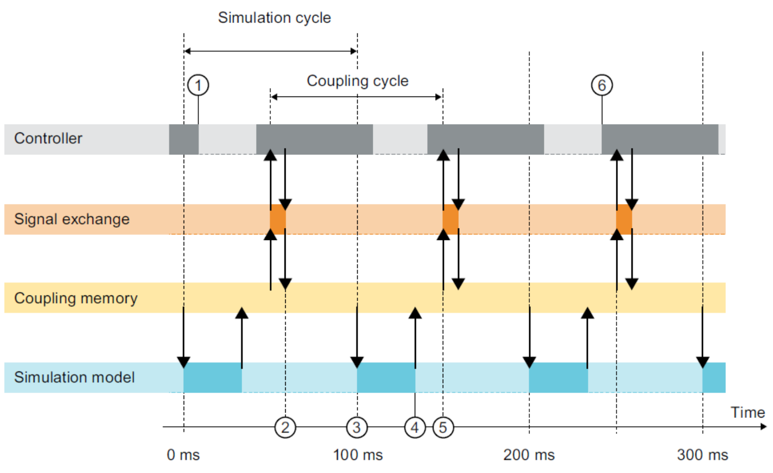


Figure 26: Example of the operation of an asynchronous simulation [10].

The numbers in Figure 26 represent the steps of the simulation:

1. The controller has finished its calculation cycle. Signals are available at the outputs of the controller.
2. The output signals of the controller are applied to the SIMIT coupling memory. Here they are available for the calculation of the model.
3. The calculation of the model is triggered.
4. End of the model calculation cycle. The calculated input signals for the controller are now available in the coupling memory.
5. With the next clock pulse of the coupling, the input signals are transferred to the controller.

6. If the controller only evaluates its inputs at the start of a cycle, the transferred input signals are only used in the control program now. This means the reaction of the simulation does not take place until two control cycles later.

Synchronous operating mode

In synchronous operating mode, the simulation calculation and the signal exchange are done in a specified sequence. One action only starts after completing the previous one, as shown in Figure 27. This results in better response times for short cycle times. However, each module and each coupling can block the entire simulation execution.

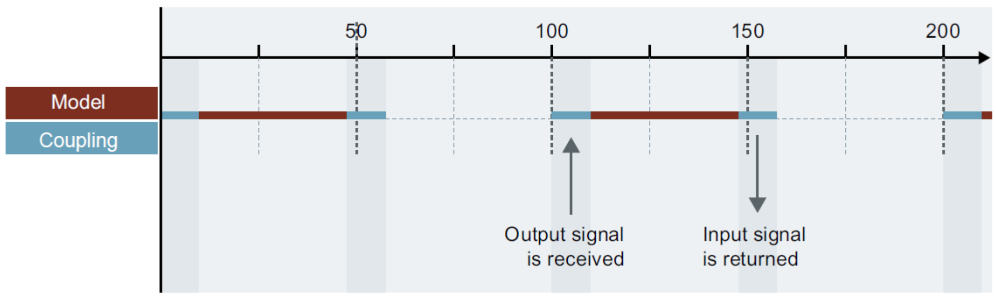


Figure 27: Simulation execution in synchronous mode [10].

Bus synchronous operating mode

In bus synchronous operating mode, all the components involved in the simulation have the same execution cycle. As shown in Figure 28, the real plant cycle time always matches the simulation cycle time, no matter how long it takes to calculate a simulation step.

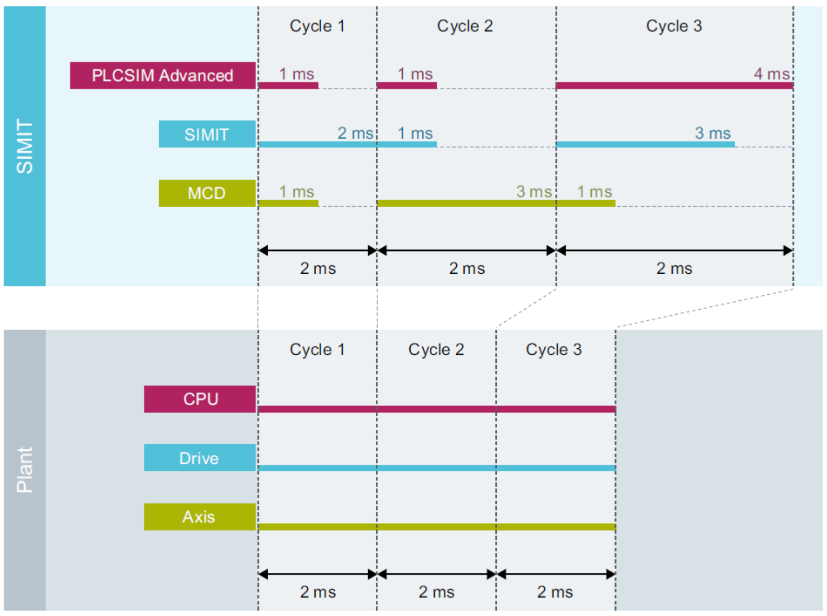


Figure 28: Correspondence of the simulation cycle times with the real-plant cycle times in bus synchronous operation mode [10].

The top part of Figure 28 shows how long PLCSIM Advanced, SIMIT and MCD take for their calculations. Each cycle starts only when all the calculations are done, and the results have been exchanged through the couplings. The bottom part of the figure represents the timing of the CPU, drive and axis in the real plant.

Bus synchronous operating mode is suitable for applications with real-time requirements and it is possible with PLCSIM Advanced and MCD. That is why it has been chosen for this project.

In bus synchronous operating mode, a time slice of the project must correspond to the cycle time.

5. Development of the digital twin

In this chapter, the models created to simulate each component of the system and the configuration of the simulation are explained.

5.1. Mechatronic simulation

The mechatronic simulation is done by giving physical properties and kinematic constraints to the 3D model of our machine. This is done in the NX Mechatronics Concept Designer (MCD).

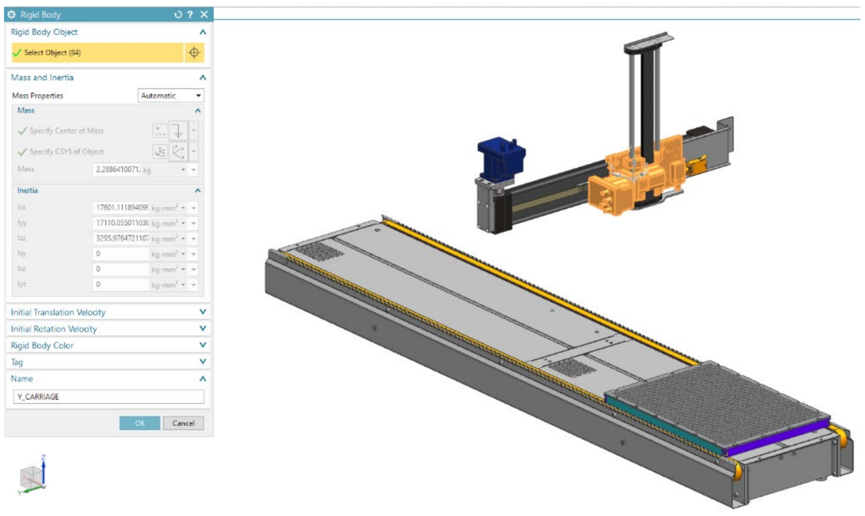
5.1.1. Basic physics

The basic physics is the definition of the rigid bodies and the collision bodies of the system.

5.1.1.1. Rigid bodies

In MCD, rigid bodies are components that can move. They have mass, inertia, translational and angular velocity, position and orientation. They respond to forces like gravity. In general, all the objects that move together are part of the same rigid body. Stationary objects do not have to be defined as rigid bodies and are part of the background. This prevents the need to simulate the structure of the machine, for example.

To create a rigid body, the user has to select all the objects to be included. Then, the mass and inertia of the body are automatically calculated based on the material properties of each component, which have to be previously available in the CAD model. Otherwise, the user can also introduce the mass and inertia properties manually. Figure 29 shows the dialogue box in MCD to create a rigid body.



Mass Properties: Automatic

Mass

Specify Center of Mass

Specify CSYS of Object

Mass: 2.2886410071. kg

Inertia

Ixx: 17601.11189409 kg·mm²
Iyy: 17110.05501103 kg·mm²
Izz: 3295.976472110 kg·mm²
Ixy: 0 kg·mm²
Ixz: 0 kg·mm²
Iyz: 0 kg·mm²

Name: Y_CARRIAGE

Figure 29: Dialogue box to create the Y carriage rigid body.

The rigid bodies that have been defined in MCD are:

a) The tray carrying the printed parts
b) The Z carriage
c) The Y carriage
d) The X motor shaft
e) The X gearbox output shaft
f) The X driving shaft

The 3D model of the gearbox is a single component. Hence, the rigid body of X gearbox output shaft has been represented by the timing pulley linked to it. Figure 30 illustrates the six rigid bodies of the mechatronic simulation.

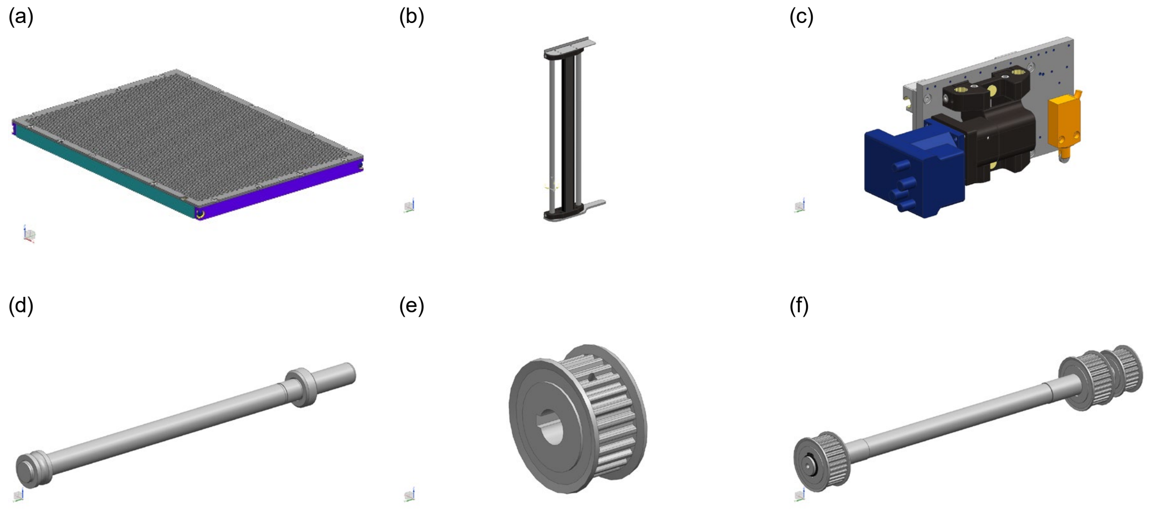


Figure 30: Rigid bodies in MCD. (a) Tray, (b) Z carriage, (c) Y carriage, (d) X motor shaft, (e) X gearbox output shaft, (f) X driving shaft.

Note that the movement of the z and y axes will be applied directly to the carriages, while in the x-axis, the whole transmission is simulated in MCD. The transmission ratios of the y and z axes are introduced in SIMIT as part of the drives' behaviour model. Those are two alternatives to perform the kinematics simulation. The first one reduces the computational needs in MCD, while the second one provides more accurate results, as it takes into account the physical properties of the transmission components. Furthermore, when trying to simulate the z and y axes transmission in MCD, the simulation has an unexpected behaviour. This behaviour has been reported to Siemens and it has been classified as a bug in the software. That is why the option to simulate the transmission of the z and y axes in SIMIT has been chosen.

5.1.1.2. Collision bodies

Collision bodies define how elements collide with other elements that also have a collision body. Elements without a collision body pass through other objects. Collision bodies are not rigid bodies, but they can be part of one. MCD calculates collisions using simplified collision shapes that encapsulate elements [15]. Increasing the geometric accuracy of the collision shapes may reduce simulation performance and cause simulation failures like objects passing through, sticking together or jittering. Table 1 shows the collision shapes available in MCD and their geometric accuracy, reliability and simulation performance.

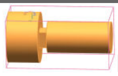
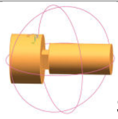
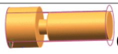
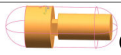
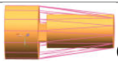
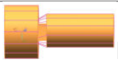
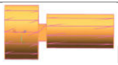
Type	Geometric accuracy	Reliability	Simulation performance
 **Box**	Low	High	High
 **Sphere**	Low	High	High
 **Cylinder**	Low	High	High
 **Capsule**	Low	High	High
 **Convex**	Medium	High	Medium
 **Multi Convex**	Medium	High	Medium
 **Mesh**	High	Low	Low

Table 1: Collision shapes in MCD and their geometric accuracy, reliability and simulation performance [15].

The user can define the collision material and set its dynamic friction, static friction, rolling friction and restitution coefficients.

In this project, the surfaces of the rigid bodies that collide with the home sensors are defined as collision bodies with a box shape. The home sensors themselves are defined as collision sensors, which are explained in chapter 5.1.4.

5.1.2. Joints and constrains

Joints and constraints set the degrees of freedom of the rigid bodies. In this project, only hinge and sliding joints are needed. Hinge joints allow one rotational degree of freedom along an axis between two rigid bodies. Slide joints allow one translational degree of freedom between two rigid bodies along a vector.

The joints that have been set in MCD are:

1. A hinge joint between the X motor shaft and the background.
2. A hinge joint between the X gearbox output shaft and the background.
3. A hinge joint between the X driving shaft and the background.
4. A sliding joint between the tray and the background.
5. A sliding joint between the Y carriage and the background.
6. A sliding joint between the Z carriage and the Y carriage.

Figure 31 shows the coordinate system of the machine and the joints vectors numbered as the list above. In the hinge joints, the rotation axis of the joint coincides with the axis of each shaft in the Y positive direction. The joint rotation anchor point is the centre of each shaft. In the sliding joints, the sliding vectors correspond to the directions of the coordinate system.

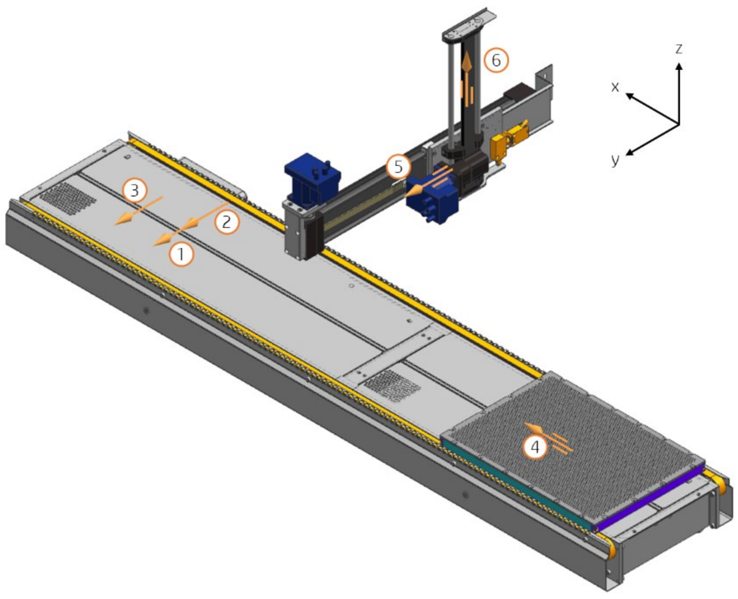


Figure 31: Machine coordinate system and joints vectors.

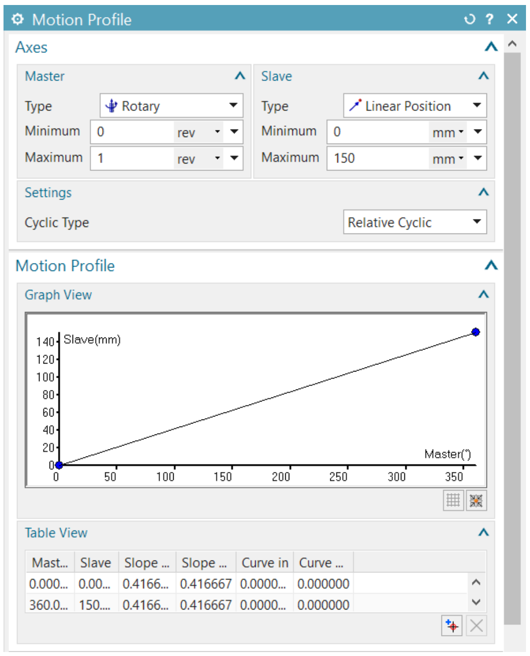
5.1.3. Couplers

Couplers are used to simulate the coordinated motion of multiple axes. In this project, two gears and one mechanical cam coupler are used to simulate the transmission steps of the x-axis. A mechanical cam is a coupler that connects the motion of two axes and forces them to maintain a relationship determined by a motion profile. The motion's reaction force is transferred back to the master axis. A gear coupler forces two rotating axes to maintain a constant turn ratio [15].

The first gear coupler represents the gearbox transmission. The hinge joint of the X motor shaft (1) is set as the master and the hinge joint of the X gearbox output shaft (2) as the slave. The gear ratio is 288:1.

The second gear coupler simulates the belt transmission between the gearbox output shaft and the second shaft. The master is the hinge joint of the X gearbox output shaft (2) and the slave is the hinge joint of the X driving shaft (3). The gear ratio is 1:1. As this transmission uses a toothed belt, the option to allow some slip in the coupler is not selected.

The mechanical cam simulates the transmission of the rotary movement of the driving shaft to the linear movement of the tray. The master is the hinge joint of the X driving shaft (3) and the slave is the sliding joint of the tray (4). The linear travel per revolution ratio is set using a motion profile shown in Figure 32. The ratio is 150 mm/rev. As this transmission uses a toothed belt, the option to allow some slip in the coupler is not selected.



Master	
Type	Rotary
Minimum	0 rev
Maximum	1 rev

Slave	
Type	Linear Position
Minimum	0 mm
Maximum	150 mm

Settings	
Cyclic Type	Relative Cyclic

Mast...	Slave	Slope ...	Slope ...	Curve in	Curve ...
0.000...	0.00...	0.4166...	0.416667	0.0000...	0.000000
360.0...	150....	0.4166...	0.416667	0.0000...	0.000000

Figure 32: Motion profile defining the motion of the slave axis relative to the motion of the master axis.

5.1.4. Sensors

In this simulation, there are three collision sensors. They correspond to the roller plungers of each limit switch used as a home sensor. Collision sensors are very similar to collision bodies, but they also provide feedback about the interaction of rigid bodies during the simulation. The status of the collision sensor can be used to trigger mechatronic operations like starting or stopping operations, changing the speed of an actuator, creating new objects or detecting their position, among others. In this case, the status of the collision sensors is sent to SIMIT so that the motor drives and the PLC can detect when the axes are at the home position.

5.1.5. Actuators

As there are three motors in the machine, three actuators are needed. The actuators can be electrical, pneumatic or hydraulic. In the present case, all the actuators are electrical. The y and z motors have been defined as position control actuators and the x motor as a speed control actuator.

Position control actuators use the position as the driving parameter, while speed actuators use the speed and its feedback to control the motion. Position control is used for the precise positioning of axes or when a suspended (gravity loaded) axis has to be controlled. Speed control is used, for example, to control a conveyor belt without position detection or to operate pumps and fans. In both kinds of actuators, the acceleration, the force or torque and the jerk can be limited for more realistic behaviour.

Although the parameter that must be controlled in the 3 axes is the position, the drive that controls the x-axis motor communicates with the PLC using the PROFIdrive standard. In this case, the position control loop is executed by the PLC and the drive receives speed setpoints. Therefore, the control of this axis in MCD is modelled as a speed control actuator. The operation of this drive is explained further in chapter 5.5.1.

In the three axes, the acceleration of the actuators has been limited to have a more realistic simulation.

5.2. User program

The program that controls the motion of the machine had mostly been developed before the start of this simulation project. However, the program to control the conveyor motor was not ready. Hence, a PLC program has been developed and added to the main program by the author of this report, which can be found in Appendix A. Like this, the whole system can be simulated to test the machine program once it is ready.

In the physical version of the machine, the position setpoints of each axis are sent to the PLC by a PC that processes the images of the 3D printed parts captured by the camera and calculates the movement sequences. To establish the communication between the PC and PLC, some read and write variables have been defined:

Write variables

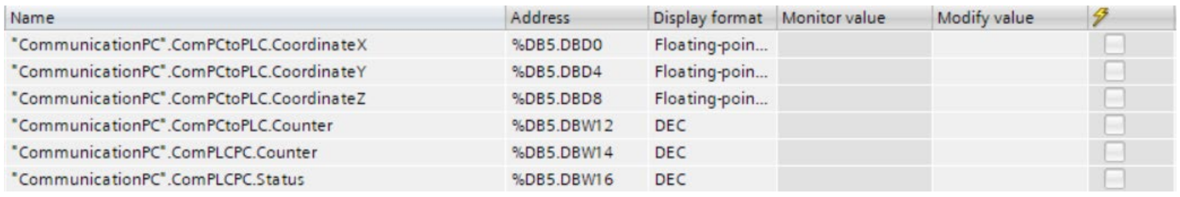
- Coordinate X (float): indicates the X-axis setpoint in mm.
- Coordinate Y (float): indicates the Y-axis setpoint in mm.
- Coordinate Z (float): indicates the Z-axis setpoint in mm.

- Counter (unsigned int): verifies that the PC command has been correctly executed by the PLC.

Read Variables

- Status (unsigned int): indicates the different states of the system (e.g. ready for new command or homing pending)
- Counter (unsigned int): indicates the completion of an action by the PLC by matching the value of the Counter variable from the PC.

As explained in chapter 4.1, the camera and the PC are not simulated in this project. Hence, the position setpoints will be introduced manually through a watch table in TIA Portal shown in Figure 33. Watch tables allow the user to monitor and modify variables of the program during its execution.



Name	Address	Display format	Monitor value	Modify value	⚡
"CommunicationPC".ComPCtoPLC.CoordinateX	%DB5.DBD0	Floating-poin...			☐
"CommunicationPC".ComPCtoPLC.CoordinateY	%DB5.DBD4	Floating-poin...			☐
"CommunicationPC".ComPCtoPLC.CoordinateZ	%DB5.DBD8	Floating-poin...			☐
"CommunicationPC".ComPCtoPLC.Counter	%DB5.DBW12	DEC			☐
"CommunicationPC".ComPLCPC.Counter	%DB5.DBW14	DEC			☐
"CommunicationPC".ComPLCPC.Status	%DB5.DBW16	DEC			☐

Figure 33: Watch table in TIA Portal with the variables involved in the PC-PLC communication.

In TIA Portal, apart from programming the PLC, the connection between the drives and the PLC is defined. The 3 drives are assigned to the PLC network as shown in Figure 34. By doing so, the input and output addresses of each drive are determined:

- Drive X:
  - I address: %I124...141
  - Q address: %Q126…135
- Drive Y:
  - I address: %I200...327
  - Q address: %Q200...327
- Drive Z:
  - I address: %I328...455
  - Q address: %Q328...455

These addresses allocate the PLC tags, which are the variables that contain the input and output signals of the PLC. Those tags are the signals read in SIMIT.

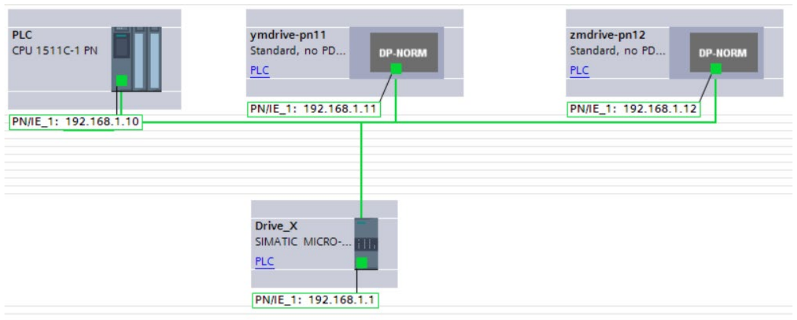
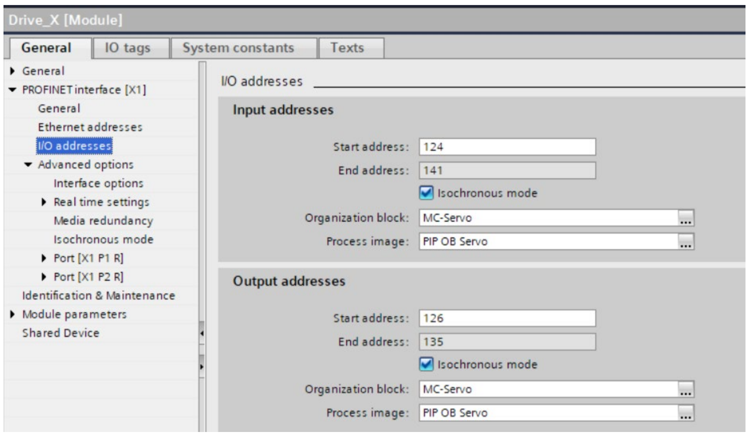


Figure 34: Network configuration in TIA Portal.

Although the PLC code tested in the simulation will be the same as the one used in the physical machine, some adjustments have to be made in TIA Portal (the software used to program the PLC) to support the simulation. Firstly, in the properties of the project, the "Support simulation during block compilation" checkbox must be enabled.

Secondly, the communication of the drives has to be set to isochronous mode to be able to use the bus synchronous operation mode described in chapter 4.3.1.2. The cycle time of the bus is set to 1 ms, which is the minimum. The isochronous mode must also be activated in the SIMATIC Micro-Drive device configuration and in the program block that controls it (MC-Servo) as shown in Figure 35 and Figure 36. To complete the isochronous mode configuration, the topology of the connection between the devices must be set, as shown in Figure 37. In this way, the PLC can calculate the delays in data transmission.



I/O addresses

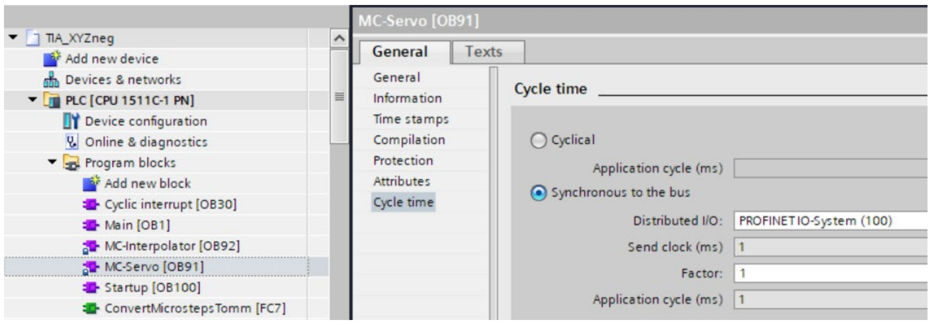
Input addresses

Start address: 124
End address: 141
☑ Isochronous mode
Organization block: MC-Servo
Process image: PIP OB Servo

Output addresses

Start address: 126
End address: 135
☑ Isochronous mode
Organization block: MC-Servo
Process image: PIP OB Servo

Figure 35: Tab of the SIMATIC Micro-Drive configuration where the isochronous mode is set.



- TIA_XYZneg
  - Add new device
  - Devices & networks
  - PLC [CPU 1511C-1 PN]
    - Device configuration
    - Online & diagnostics
    - Program blocks
      - Add new block
      - Cyclic interrupt [OB30]
      - Main [OB1]
      - MC-Interpolator [OB92]
      - MC-Servo [OB91]
      - Startup [OB100]
      - ConvertMicrostepsTomm [FC7]

MC-Servo [OB91]

General | Texts

General
Information
Time stamps
Compilation
Protection
Attributes
Cycle time

Cycle time

( ) Cyclical

Application cycle (ms):

[✓ Synchronous to the bus]

Distributed I/O:	PROFINET IO-System (100)
Send clock (ms):	1
Factor:	1
Application cycle (ms):	1

Figure 36: Settings of the operation mode in the MC-Servo program block.

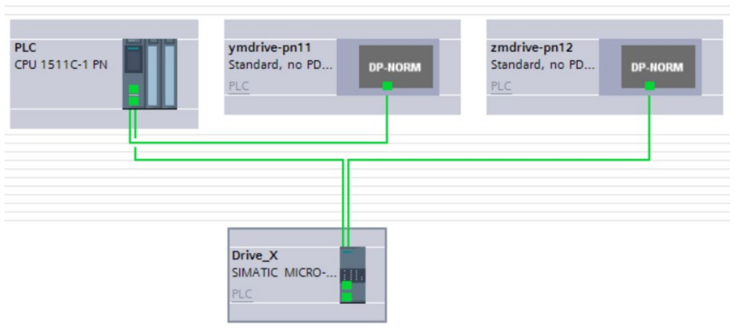


Figure 37: Topology of the connection between the PLC and the drives.

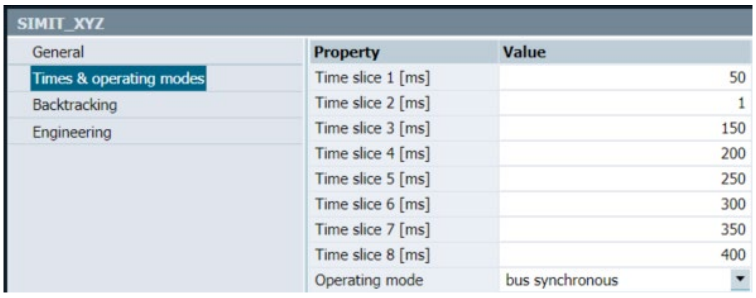
5.3. Controller simulation

The PLC is simulated in PLCSIM Advanced. When the simulation is started in SIMIT, an instance of a controller is automatically created in PLCSIM Advanced and set to the "Run" operating state. Then, the user program in TIA Portal has to be compiled and loaded into the created virtual controller. After that, the virtual PLC is ready to communicate and synchronise with SIMIT and MCD.

5.4. Data exchange

SIMIT is used to synchronize the mechatronics model in MCD with the virtual controller in PLCSIM Advanced. SIMIT automatically detects the input and output signals coming from both programs. Then, the user can transform and adapt them so that they are understood by the two software. To enable this communication, two couplings must be configured.

The first coupling is with PLCSIM Advanced. When creating it, it has to be configured as bus synchronous. The SIMIT project operating mode is also set as bus synchronous. The time slice used as the simulation cycle time must match the cycle time configured previously in TIA Portal (1 ms), as shown in Figure 38. In this case, the time slice 2 is used.

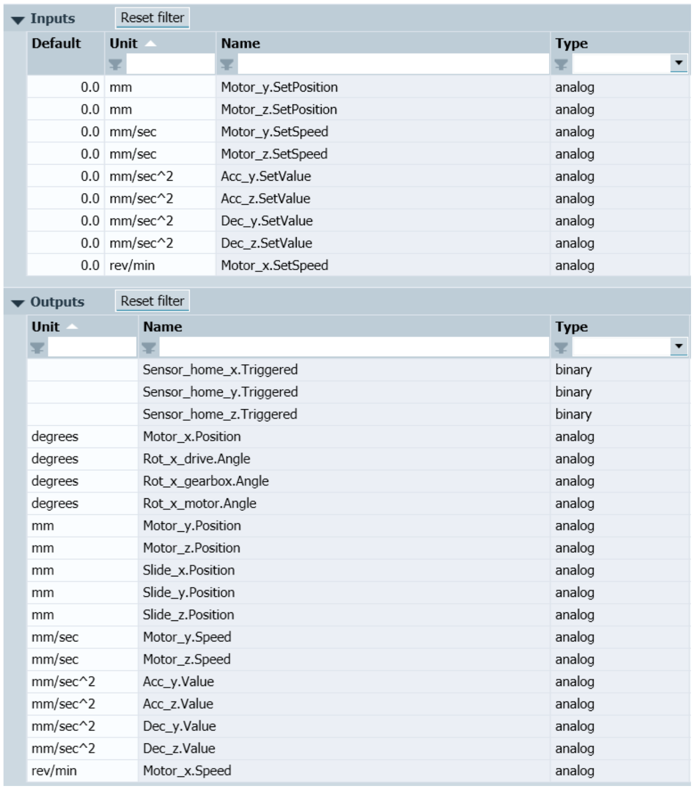

SIMIT_XYZ

General
Times & operating modes
Backtracking
Engineering

Property	Value
Time slice 1 [ms]	50
Time slice 2 [ms]	1
Time slice 3 [ms]	150
Time slice 4 [ms]	200
Time slice 5 [ms]	250
Time slice 6 [ms]	300
Time slice 7 [ms]	350
Time slice 8 [ms]	400
Operating mode	bus synchronous

Figure 38: Operating mode and cycle time configuration in the SIMIT project.

The second coupling is with the MCD project. The units of the MCD signals must be set as they are used in the drive simulation and the PLC, as shown in Figure 39. The MCD coupling is also set as bus synchronous and the time slice is set to 2.


Inputs

Default	Unit	Name	Type
0.0	mm	Motor_y.SetPosition	analog
0.0	mm	Motor_z.SetPosition	analog
0.0	mm/sec	Motor_y.SetSpeed	analog
0.0	mm/sec	Motor_z.SetSpeed	analog
0.0	mm/sec^2	Acc_y.SetValue	analog
0.0	mm/sec^2	Acc_z.SetValue	analog
0.0	mm/sec^2	Dec_y.SetValue	analog
0.0	mm/sec^2	Dec_z.SetValue	analog
0.0	rev/min	Motor_x.SetSpeed	analog

Outputs

Unit	Name	Type
	Sensor_home_x.Triggered	binary
	Sensor_home_y.Triggered	binary
	Sensor_home_z.Triggered	binary
degrees	Motor_x.Position	analog
degrees	Rot_x_drive.Angle	analog
degrees	Rot_x_gearbox.Angle	analog
degrees	Rot_x_motor.Angle	analog
mm	Motor_y.Position	analog
mm	Motor_z.Position	analog
mm	Slide_x.Position	analog
mm	Slide_y.Position	analog
mm	Slide_z.Position	analog
mm/sec	Motor_y.Speed	analog
mm/sec	Motor_z.Speed	analog
mm/sec^2	Acc_y.Value	analog
mm/sec^2	Acc_z.Value	analog
mm/sec^2	Dec_y.Value	analog
mm/sec^2	Dec_z.Value	analog
rev/min	Motor_x.Speed	analog

Figure 39: Signal units in the MCD coupling.

5.5. Drivers behaviour simulation

SIMIT emulates the behaviour of devices like sensors or drives for motors, valves, and pumps. The models are built using standard libraries or user-made components. The model of each device is defined using a chart. In this project, three charts have been created to model the logic of the motor drives.

5.5.1. Drive X

The drive used to control the motor of the conveyor belt is the SIMATIC Micro-Drive PDC from Siemens.

Siemens drives use the standard PROFIdrive for their communication. PROFIdrive is a standard for drives that are connected to the controller via PROFInet (explained in chapter 4.2.4). It defines how a drive behaves on the field bus. PROFIdrive devices are activated by control words (STW) and provide feedback in status words (ZSW) [10]. All the PROFIdrive drives present a common functionality, shown in Figure 40:

- A state machine
- A ramp generator to transfer the speed setpoint received from the controller
- Adapted feedback of the actual speed and the actual position

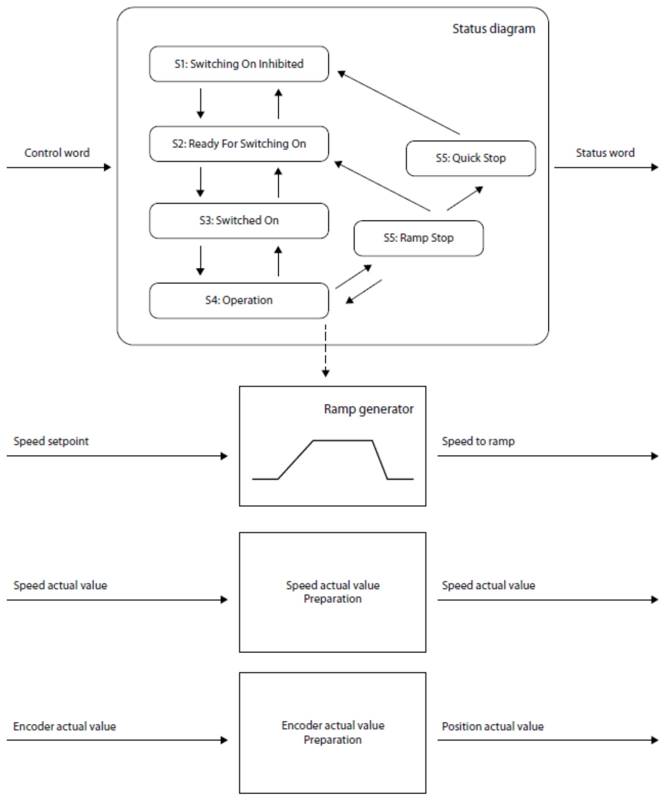
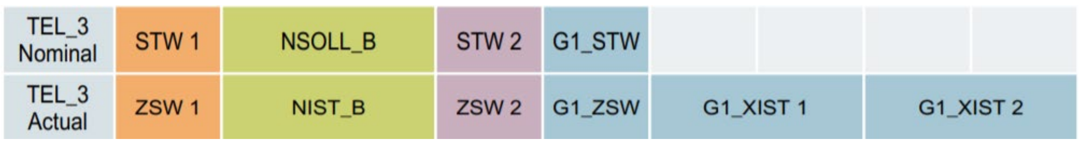


Figure 40: Basic functionality of PROFIdrive drives.

The state machine defines the transitions from one status to another. The ramp generator converts a jump in the speed setpoint into linear ramp increases with an adjustable gradient [10]. Depending on the telegram used for communication, PROFIdrive devices can also include a position control or a torque limiting. The telegram used in this project is the telegram 3, which includes a position control. It uses 5 words to send information from the PLC to the drive and 9 words to transmit data from the drive to the PLC, which are shown in Figure 41 and explained in Table 2.



TEL_3 Nominal	STW 1	NSOLL_B	STW 2	G1_STW				
TEL_3 Actual	ZSW 1	NIST_B	ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2	

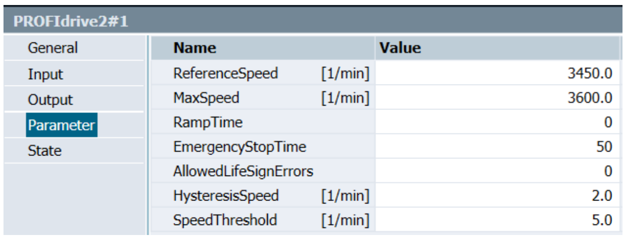
Figure 41: Structure of the PROFIdrive telegram 3 [10].

TEL_3 Nominal		***TEL_3 Actual***	
STW1	*Control word 1*	*ZSW1*	*Status word 1*
NSOLL_B	*Standardized speed setpoint*	*NIST_B*	*Standardized actual speed*
STW 2	*Control word 2*	*ZSW 2*	*Status word 2*
G1_STW	*Encoder control word*	*G1_ZSW*	*Encoder status word*
		G1_XIST 1	*Encoder cyclical value*
		G1_XIST 2	*Encoder absolute value*

Table 2: PROFIdrive telegram 3 words meaning.

PROFIdrive drives are simulated in SIMIT using standard blocks. Figure 43 shows the model of the x-axis drive in this project. The blocks used are:

1. *PROFIdrive2* component: it simulates the state machine and the communication according to the PROFIdrive standard. It also sends the speed setpoint to MCD and reads the actual speed coming back. The reference and maximum speed of this component must be coherent with the motor characteristics and the configuration of the motor in TIA Portal. The ramp time is set to 0 because, in this case, the PLC is managing the acceleration and deceleration ramps. Figure 42 shows the PROFIdrive2 block parameters.



PROFIdrive2#1

General
Input
Output
Parameter
State

Name		Value
ReferenceSpeed	[1/min]	3450.0
MaxSpeed	[1/min]	3600.0
RampTime		0
EmergencyStopTime		50
AllowedLifeSignErrors		0
HysteresisSpeed	[1/min]	2.0
SpeedThreshold	[1/min]	5.0

Figure 42: PROFIdrive2 block parameters.

2. *Sensor* component: it is the connection to an encoder. The resolution parameters must be set according to the encoder used.

3. *SensorProcessRotatory* component: it simulates a rotary encoder, either absolute or incremental. It converts a signal in degrees coming from MCD to an encoder signal. In this case, the encoder is incremental, and it has 2048 steps per revolution. These are the same parameters set in TIA Portal.

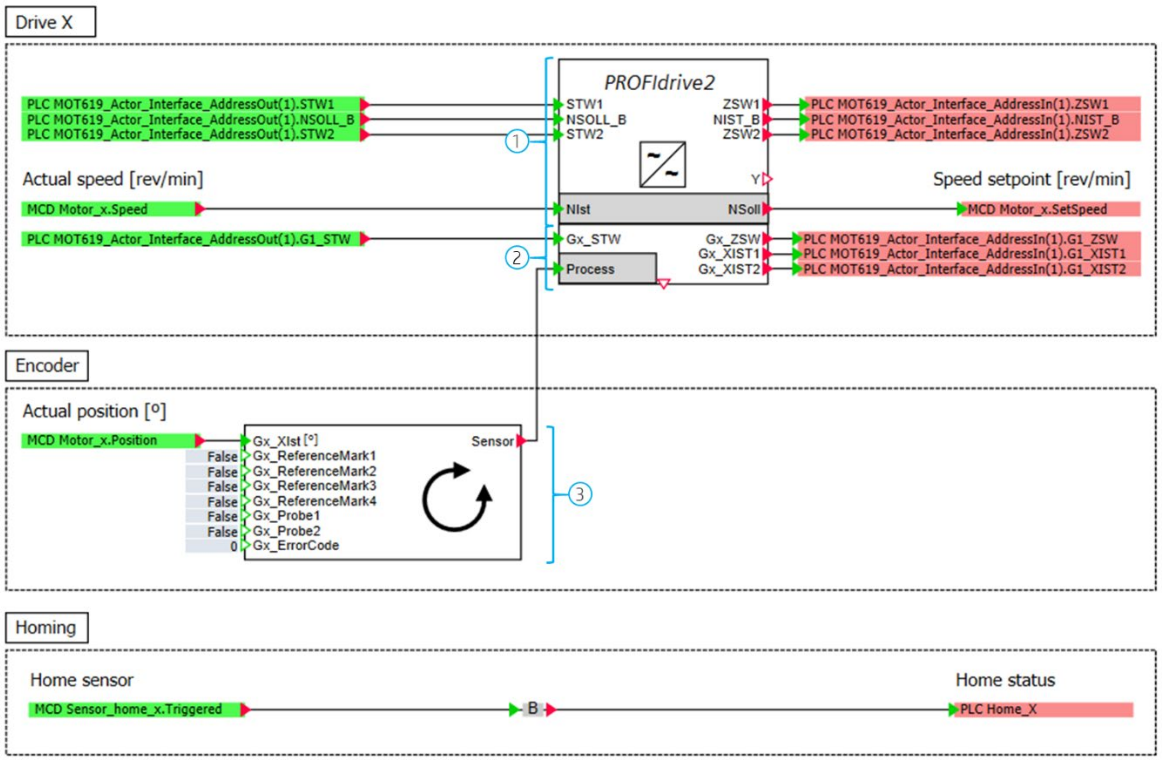

Drive X
PROFIdrive2
PLC MOT619_Actor_Interface_AddressOut(1).STW1
PLC MOT619_Actor_Interface_AddressOut(1).NSOLL_B
PLC MOT619_Actor_Interface_AddressOut(1).STW2
STW1
NSOLL_B
STW2
ZSW1
NIST_B
ZSW2
PLC MOT619_Actor_Interface_AddressIn(1).ZSW1
PLC MOT619_Actor_Interface_AddressIn(1).NIST_B
PLC MOT619_Actor_Interface_AddressIn(1).ZSW2
1
Y
Actual speed [rev/min]
Speed setpoint [rev/min]
MCD Motor_x.Speed
NIst
NSoll
MCD Motor_x.SetSpeed
PLC MOT619_Actor_Interface_AddressOut(1).G1_STW
Gx_STW
Process
2
Gx_ZSW
Gx_XIST1
Gx_XIST2
PLC MOT619_Actor_Interface_AddressIn(1).G1_ZSW
PLC MOT619_Actor_Interface_AddressIn(1).G1_XIST1
PLC MOT619_Actor_Interface_AddressIn(1).G1_XIST2
Encoder
Actual position [°]
MCD Motor_x.Position
Gx_XIst [°]
False Gx_ReferenceMark1
False Gx_ReferenceMark2
False Gx_ReferenceMark3
False Gx_ReferenceMark4
False Gx_Probe1
False Gx_Probe2
0 Gx_ErrorCode
Sensor
3
Homing
Home sensor
Home status
MCD Sensor_home_x.Triggered
B
PLC Home_X

Figure 43: Model of the x-axis drive in SIMIT.

These PROFIdrive blocks exchange information with MCD and the virtual controller using the words in Table 2 plus NIst, NSoll and Gx_XIst. NIst is the actual speed in min^{-1}, NSoll is the speed setpoint in min^{-1} and Gx_XIst is the motor position feedback in degrees.

As the encoder used is incremental, the axis has to be referenced at the beginning of the operation. To do so, the binary signal of the collision sensor in MCD is sent to the corresponding input of the PLC, as seen at the bottom of Figure 43.

5.5.2. Drive Y and Z

The drive used to control the Y and Z motors is the Lexium MDrive integrated motor by Schneider Electric. Although it is connected to the controller via PROFInet, it does not comply with the PROFIdrive standard. Thus, a specific model has to be developed to simulate its behaviour. As there is no documentation available explaining the logic of this drive (state machine, control and status words, etc.), a simplified model has been built by a reverse engineering process. The outcome of understanding the signals and the necessary transformations to get a correct functioning of the simulation can be seen in Figure 44 for the y-axis drive, and in Figure 45 for the z-axis drive.

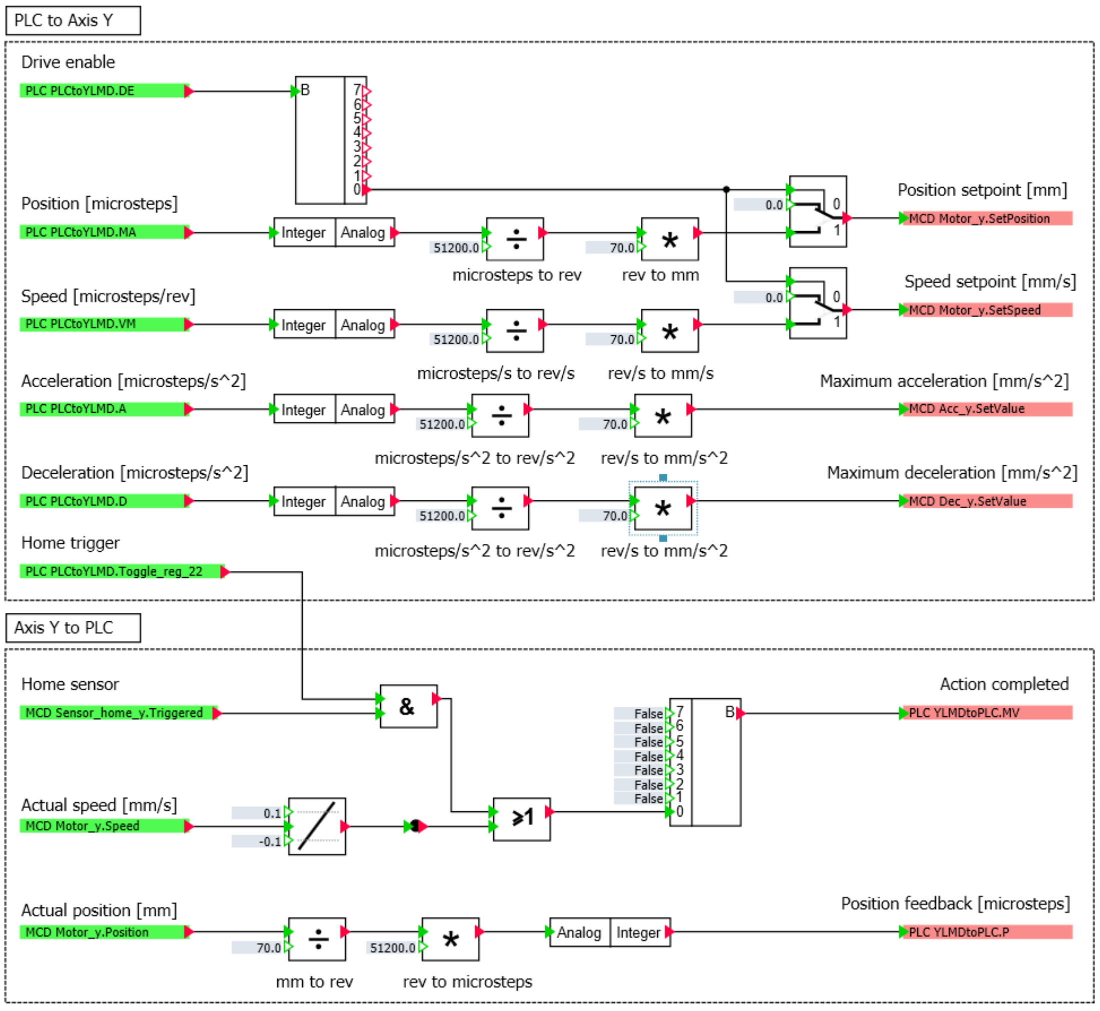


Figure 44: Model of the y-axis drive in SIMIT.

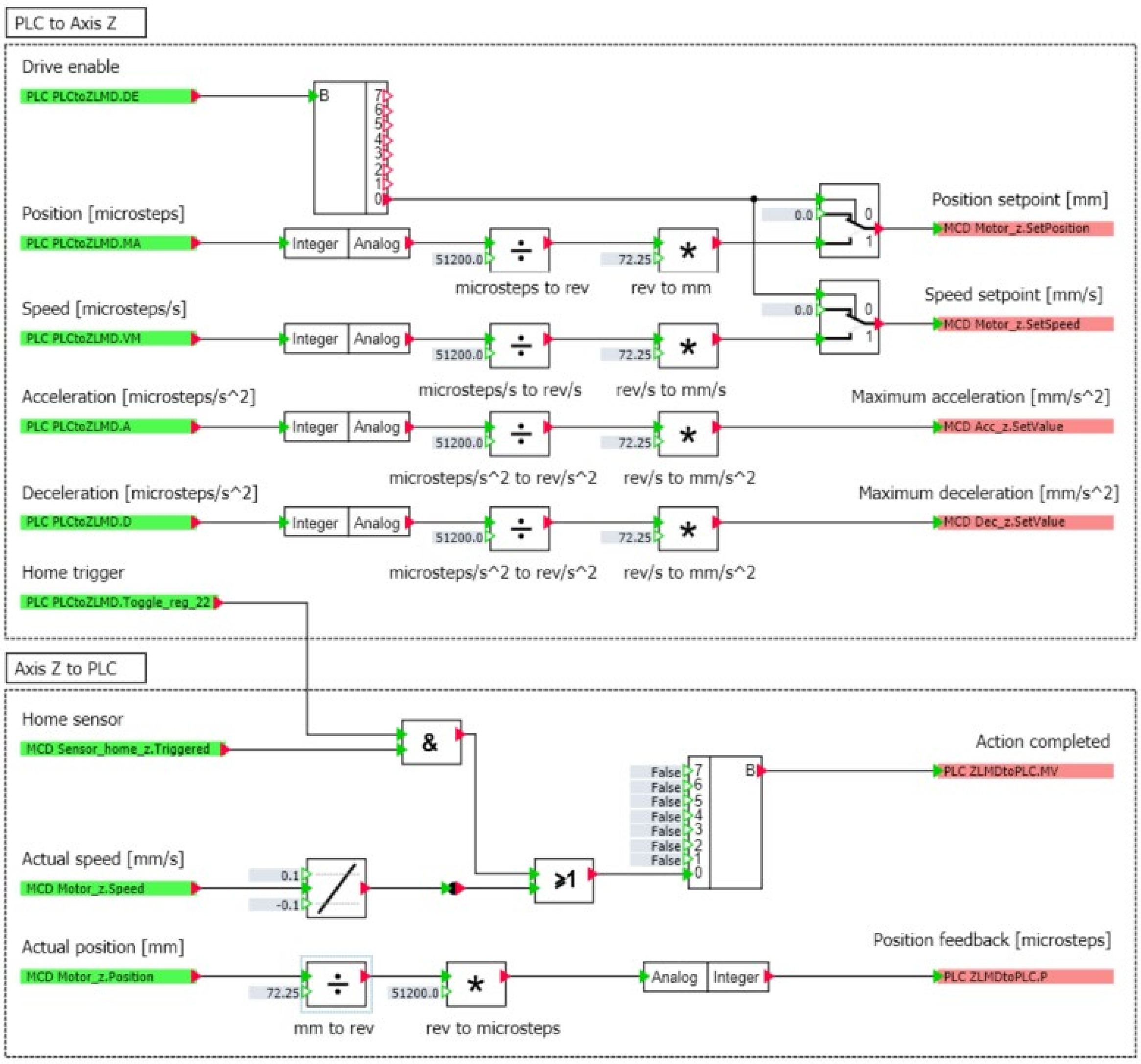
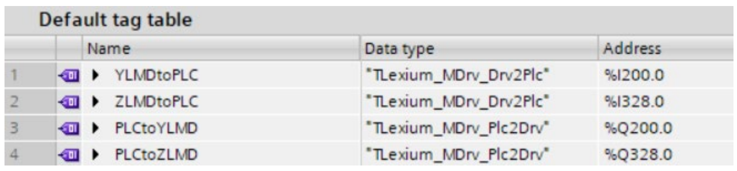


Figure 45: Model of the z-axis drive in SIMIT.

The PLC tags used in the data exchange between the Lexium MDrive and the PLC have a specific data structure, called “TLexium_MDrv_Drv2Plc” for the PLC input signals and “TLexium_MDrv_Plc2Drv2” for the PLC output signals, as seen in Figure 46. Each of these tags is a folder with a list of other tags that are used for the operation of the motor and the signal exchange with the PLC.



Default tag table

	Name	Data type	Address
1	YLMDtoPLC	"TLexium_MDrv_Drv2Plc"	%I200.0
2	ZLMDtoPLC	"TLexium_MDrv_Drv2Plc"	%I328.0
3	PLCtoYLMD	"TLexium_MDrv_Plc2Drv"	%Q200.0
4	PLCtoZLMD	"TLexium_MDrv_Plc2Drv"	%Q328.0

Figure 46: PLC tags for data exchange with the Lexium MDrive.

The tag *DE* carries a byte signal that enables the drive. If the lower bit of this signal is set to one, the position and speed setpoints sent by the PLC are transferred to the MCD model. The position setpoint is set by the tag *MA (DInt)*. It is an absolute position. The speed setpoint is set by the tag *VM (DWord)*. The maximum acceleration and deceleration in the MCD movements are set by the tags *A (DWord)* and *D (DWord)*.

The position, speed, acceleration and deceleration signals sent by the PLC are integer values in the units understood by the real drive and motor (micro-steps). Those signals must be converted to analogue values and change the units to the ones understood by the mechatronic model in MCD. Each motor revolution equals to 51200 micro-steps.

The drive model also applies the transmission ratio. The linear travel per motor revolution is 70 mm/rev in the y-axis and 72,25 mm/rev in the z-axis.

The position feedback from the mechatronic model is transformed and linked to the PLC signal *P (DInt)*, simulating the incremental encoder.

Finally, the tag *MV (Byte)* monitors the movement of the motor. In the real drive, it is set to 1 if the motor is moving to a position or 0 if it is stopped. When the PLC sees a change of this tag from 1 to 0, it sets the action that was being performed by the drive as completed. To simulate this behaviour in SIMIT, the signal MV is set to 1 when one of the following conditions is true:

- The speed of the motor in the mechatronic model is out of the range of -0,1 and 0,1 mm/s.
- The PLC program has requested a referencing of the axis and the home sensor is activated. When a homing operation is needed, the PLC triggers the tag Toggle_reg_22.

As the simulation will start with the three axes at the home position, the movements of the homing operation are not simulated.

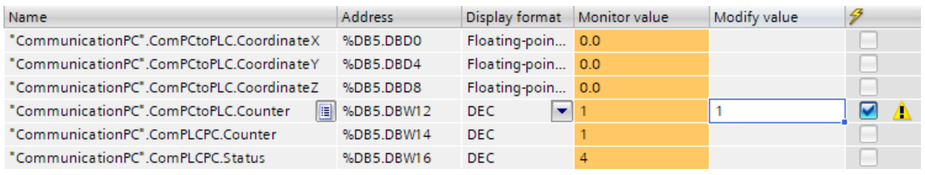
6. Validation of the simulation

6.1. Operation of the simulation

Once the simulation is ready, it can be launched from SIMIT. MCD starts automatically and shows the machine. At the same time, a virtual controller is created and started in PLCSIM Advanced. Then, the user program has to be compiled and loaded to the virtual PLC. After that, the simulation can be operated using the watch table in TIA Portal explained in chapter 5.2, which contains the variables for the communication between the PLC and the PC in the real machine.

The control code for the x-axis has been added during this project and with the sole purpose to test the simulation. As this portion of code will not be part of the final machine control program, it has been decided not to dedicate further efforts to integrate it with the PC-PLC communication. Therefore, while the PC-PLC communication variables trigger the y and z movements and provide feedback on their completion, they do not reflect the x-axis motion status.

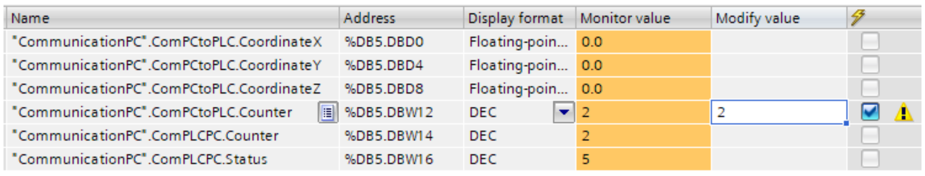
At the start of the simulation, the value of the *ComPLCPC.Status* variable is 3. The user program requires referencing the z and y axes, in this order, before accepting any position setpoint. The homing of the z-axis is triggered by setting the *ComPCPLC.Counter* to 1. Once the z-axis is homed, the PLC sets *ComPLCPC.Counter* to 1 and the *ComPLCPC.Status* to 4, as in Figure 47.



Name	Address	Display format	Monitor value	Modify value	⚡
"CommunicationPC".ComPCtoPLC.CoordinateX	%DB5.DBD0	Floating-poin...	0.0		☐
"CommunicationPC".ComPCtoPLC.CoordinateY	%DB5.DBD4	Floating-poin...	0.0		☐
"CommunicationPC".ComPCtoPLC.CoordinateZ	%DB5.DBD8	Floating-poin...	0.0		☐
"CommunicationPC".ComPCtoPLC.Counter	%DB5.DBW12	DEC	1	1	☑ ⚠
"CommunicationPC".ComPLCPC.Counter	%DB5.DBW14	DEC	1		☐
"CommunicationPC".ComPLCPC.Status	%DB5.DBW16	DEC	4		☐

Figure 47: Values of the communication variables after referencing the z-axis.

To reference the y-axis, the user has to set the *ComPCPLC.Counter* to 2. After the home sensor is triggered and the actual position is set to 0, the communication variables have the values shown in Figure 48.



Name	Address	Display format	Monitor value	Modify value	⚡
"CommunicationPC".ComPCtoPLC.CoordinateX	%DB5.DBD0	Floating-poin...	0.0		☐
"CommunicationPC".ComPCtoPLC.CoordinateY	%DB5.DBD4	Floating-poin...	0.0		☐
"CommunicationPC".ComPCtoPLC.CoordinateZ	%DB5.DBD8	Floating-poin...	0.0		☐
"CommunicationPC".ComPCtoPLC.Counter	%DB5.DBW12	DEC	2	2	☑ ⚠
"CommunicationPC".ComPLCPC.Counter	%DB5.DBW14	DEC	2		☐
"CommunicationPC".ComPLCPC.Status	%DB5.DBW16	DEC	5		☐

Figure 48: Values of the communication variables after referencing the y-axis.

The x-axis is enabled and homed at the start of the simulation. However, it can not start moving until the other two axes are homed.

Once the 3 axes are referenced, the user can start introducing position setpoints in the *ComPCtoPLC.CoordinateX*, *ComPCtoPLC.CoordinateY* and *ComPCtoPLC.CoordinateZ* variables and observe the machine motion in MCD. Each time a new position command is introduced, the user has to change the value of *ComPCPLC.Counter* to an integer bigger than 2. After the y and z axes reach their target positions, the *ComPLCPC.Counter* matches the *ComPCPLC.Counter*.

[17] provides a video of the operation of the simulation and the resulting visualization of the motion of the machine. It can be observed that the 3 axes respond to the position commands as expected.

The default values for the maximum linear speed, acceleration and deceleration used in the machine program are shown in Table 3. The y and z acceleration and deceleration values are equivalent to 100000 micro-steps/s^2.

Axis	Speed [mm/s]	Acceleration [mm/s^2]	Deceleration [mm/s^2]
X	30	600	600
Y	10	137	137
Z	10	141	141

Table 3: Default values for linear speed, acceleration and deceleration.

For a deeper analysis of the accuracy of the simulation, the position, speed and acceleration values versus time of each axis are exposed and discussed in the following chapter.

6.2. X-axis motion

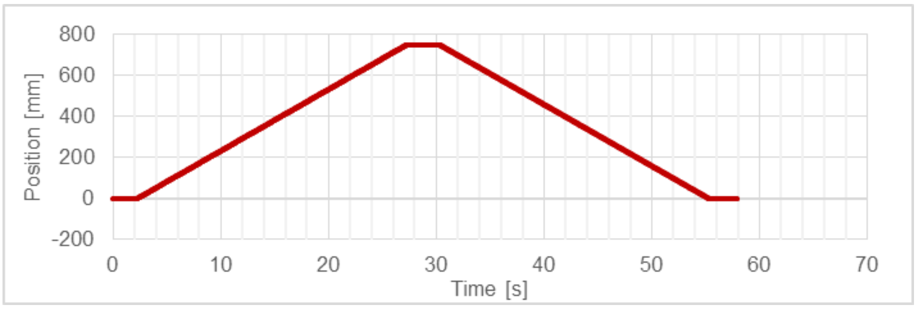
To test x-axis motion, four tests have been conducted with different values for the position, speed, acceleration and deceleration. In all the tests, the movement starts from the 0 mm position. Then, the position command is introduced. Once it is reached, the position setpoint is again changed to 0. The characteristics of each movement can be seen in Table 4.

Test	Target position [mm]	Maximum speed [mm/s]	Maximum acceleration [mm/s^2]	Maximum deceleration [mm/s^2]	Comment
1	750	30	600	600	Default values
2	5	30	600	600	Close position
3	750	55	600	600	High speed
4	750	30	60	30	Low acceleration and deceleration

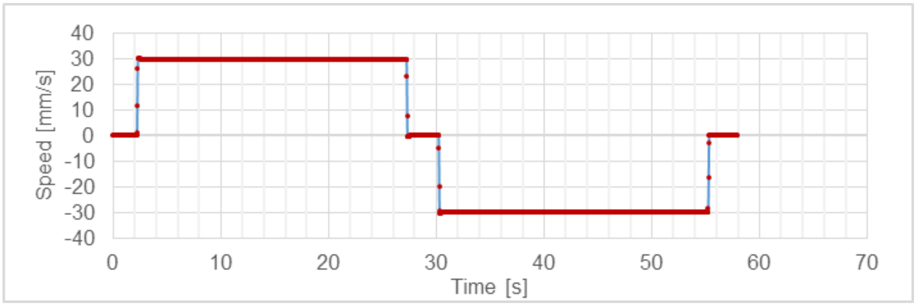
Table 4: Characteristics of the x-axis motion tests.

During the performance of the tests, it has been detected that, with a simulation cycle time of 4 ms, the x-axis model has an unexpected behaviour when it is set to speeds higher than 1875 min^{-1}. This error has been communicated to Siemens technical support. Their suggestion is to lower the simulation cycle time to 1 ms, which is the solution adopted in this project. The cause of the faulty behaviour is related to the software and it is currently under investigation by Siemens.

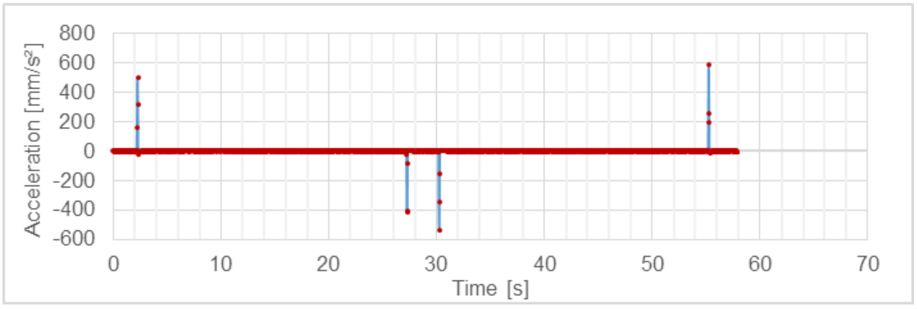
Figure 49 shows the motion graph obtained using the default settings of the x-axis motor. The 30 mm/s speed corresponds approximately to the motor nominal speed of 3450 min^{-1}. In Figure 49 (c) it seems like the maximum acceleration values (600 mm/s^2) are not reached. However, this is due to the discrete data collection. With a sample time of 0,03 s, the moments where the maximum acceleration is reached are not recorded.



(a)



(b)



(c)

Figure 49: X-axis motion with a target position of 750 mm, a maximum speed of 30 mm/s and maximum acceleration and deceleration of 600 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

Figure 50 shows the motion profile of the second test, where the target position is 5 mm. As seen in Figure 50, the target position is reached in less than 0,05 s. This does not cause any instability in the simulation.

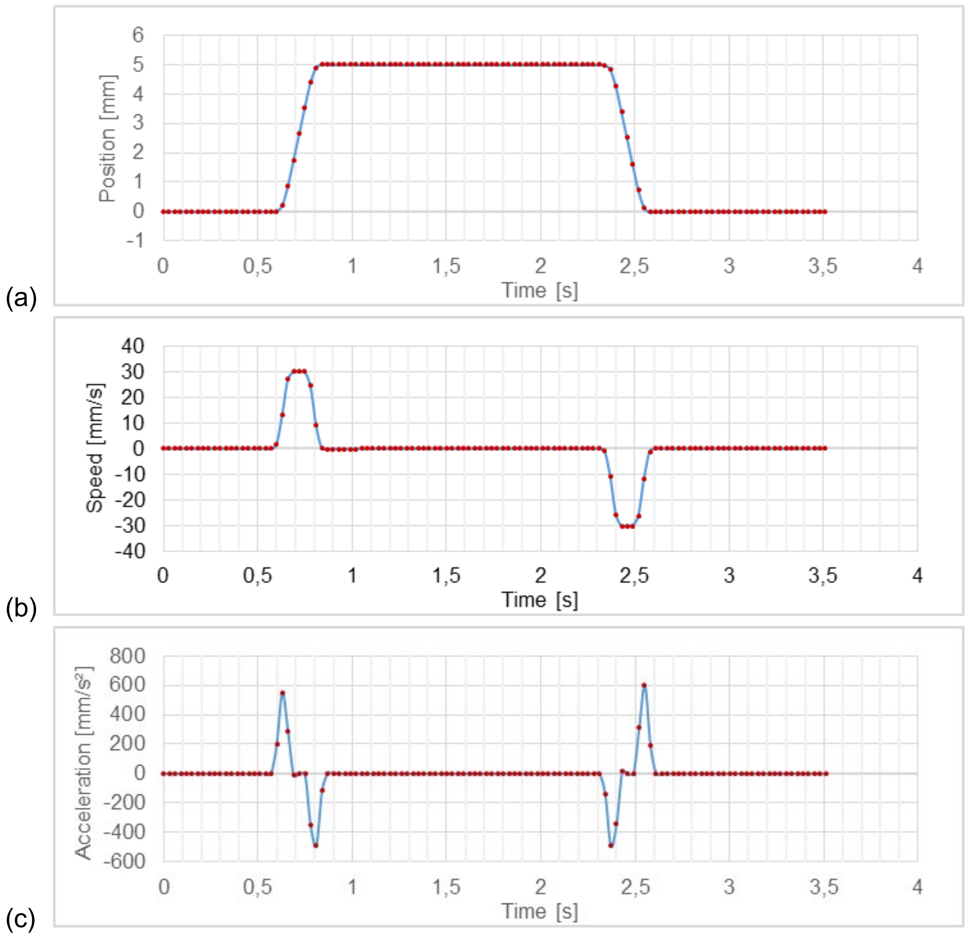


Figure 50: X-axis motion with a target position of 5 mm, a maximum speed of 30 mm/s and maximum acceleration and deceleration of 600 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

The speed used in test 3 corresponds to a motor speed of approximately 6500 min^{-1}. This speed can not be reached by this motor. However, the test was performed to check if the model presented unexpected behaviours at high speeds. As seen in Figure 51, the motion profile obtained is correct.

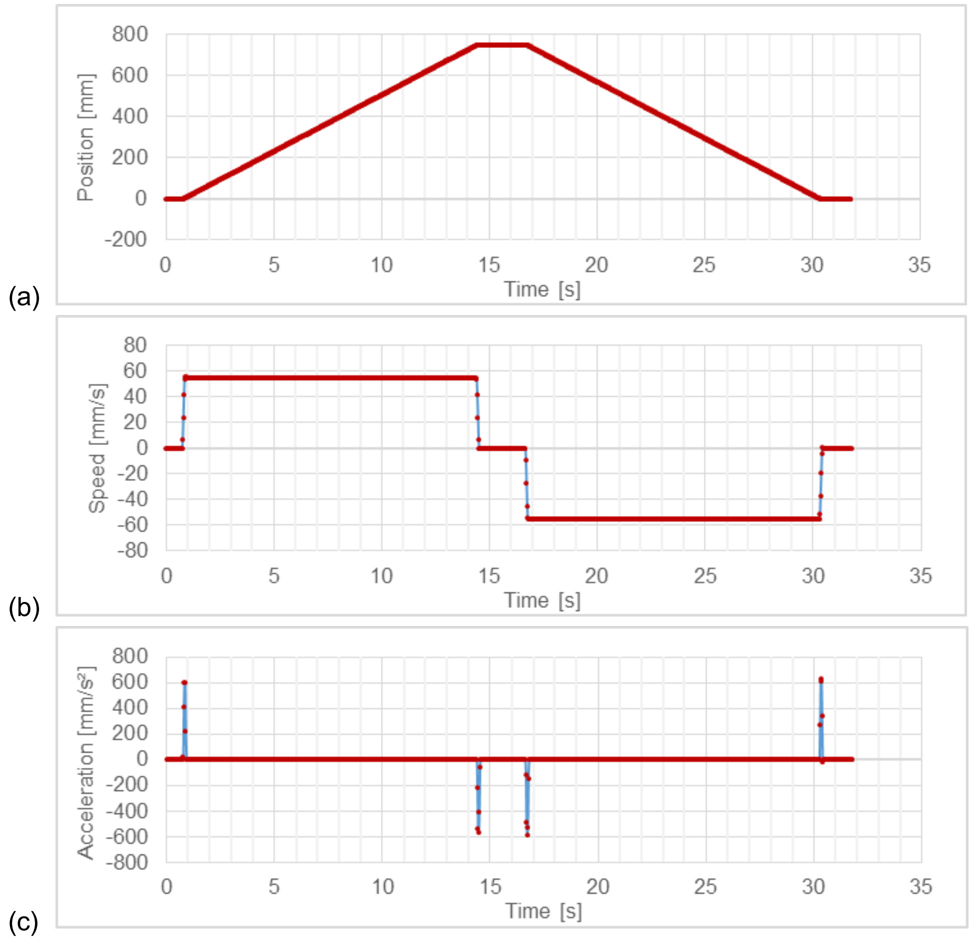


Figure 51: X-axis motion with a target position of 750 mm, a maximum speed of 55 mm/s and maximum acceleration and deceleration of 600 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

Figure 52 shows how the simulation responds to getting different values for acceleration and deceleration. The speed profile obtained in the four tests is trapezoidal, as the target position is far enough to give time to reach the maximum speed set in each case.

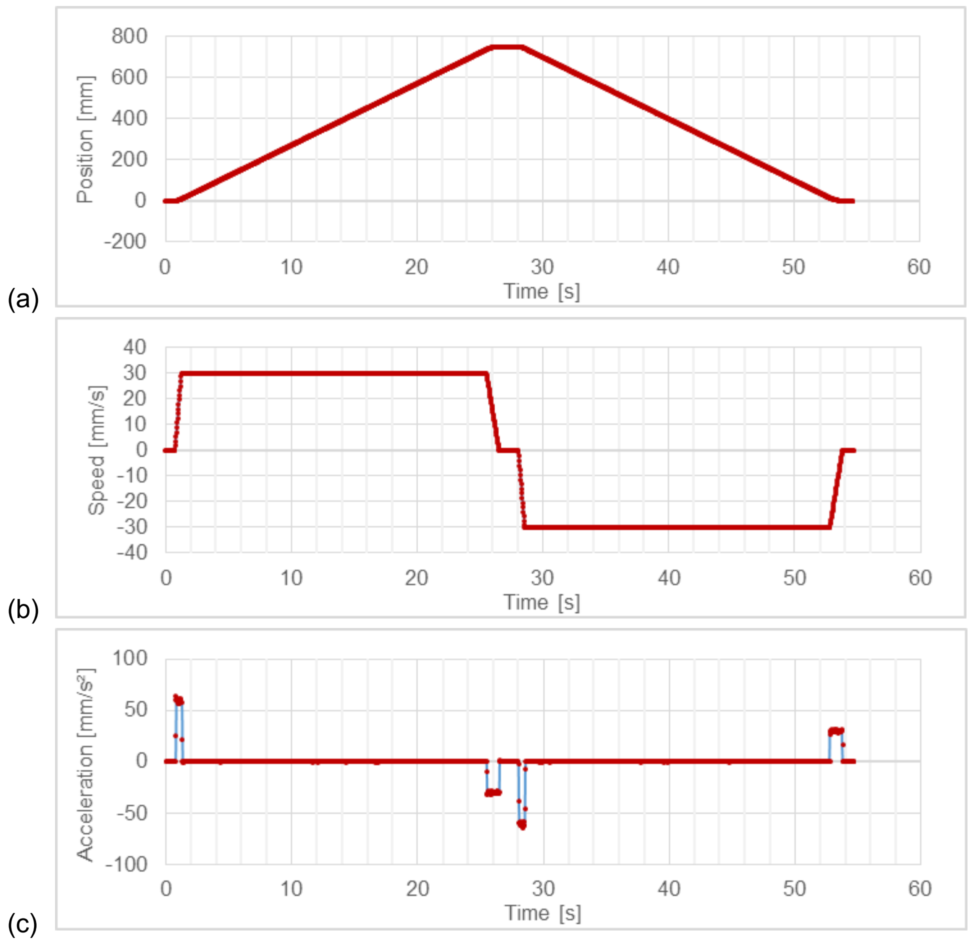


Figure 52: X-axis motion with a target position of 750 mm, a maximum speed of 30 mm/s, a maximum acceleration of 60 mm/s^2 and a maximum deceleration of 30 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

6.3. Y-axis motion

To test y-axis motion, four tests have been conducted with different values for the position, speed, acceleration and deceleration. The movement sequence is the same as for the x-axis. The characteristics of each movement can be seen in Table 5.

Test	**Target position [mm]**	**Maximum speed [mm/s]**	**Maximum acceleration [mm/s^2]**	**Maximum deceleration [mm/s^2]**	**Comment**
1	280	10	137	137	Default values
2	2	10	137	137	Low position
3	280	1075	137	137	High speed
4	280	10	14	7	Low acceleration and deceleration

Table 5: Characteristics of the y-axis motion tests.

Figure 53 shows the motion graph obtained using the default settings of the y-axis motor. With an acceleration of 137 mm/s^2, a speed of 10 mm/s is reached in 0,07 s. Since the data recording sample time is 0,03 s, the speed change is only represented by two points where the acceleration is not 0, as shown in Figure 53 (c).

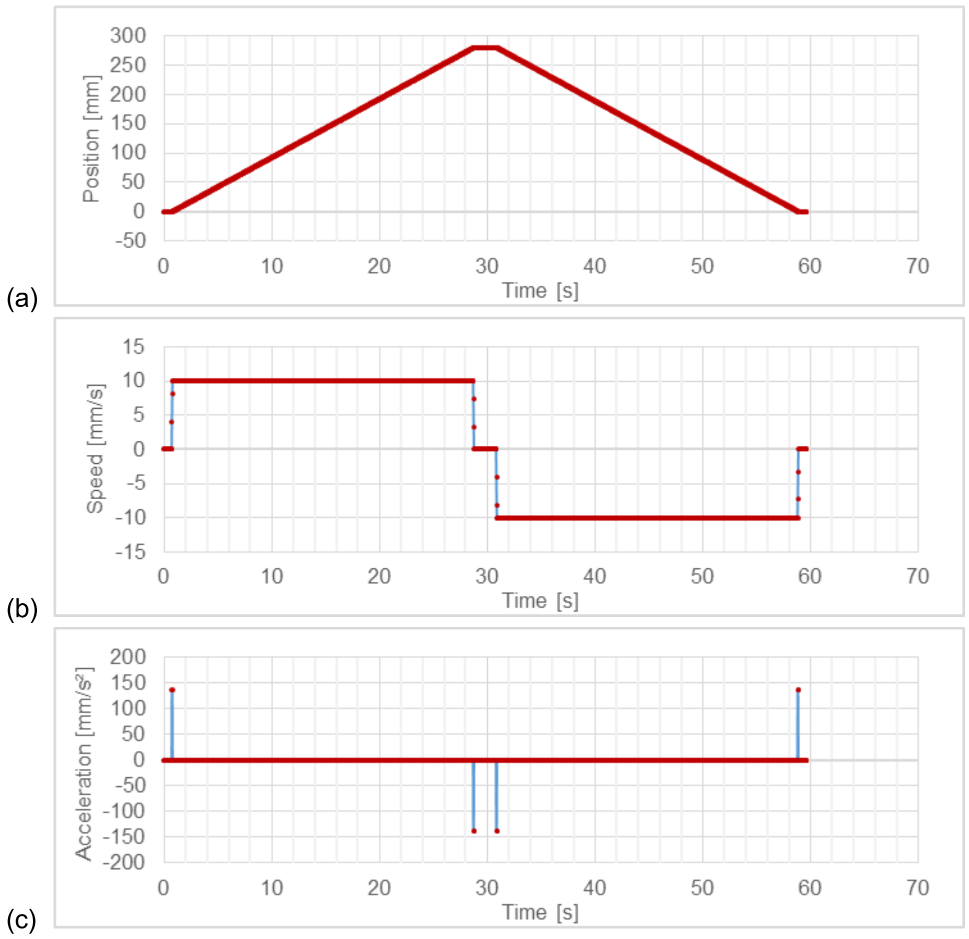


Figure 53: Y-axis motion with a target position of 280 mm, a maximum speed of 10 mm/s and maximum acceleration and deceleration of 137 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

In Figure 54 (c), the acceleration phase is represented by two or three points like in the previous case. They are more visible because, as the movement is shorter, the amount of data is lower, and the points are more spaced.

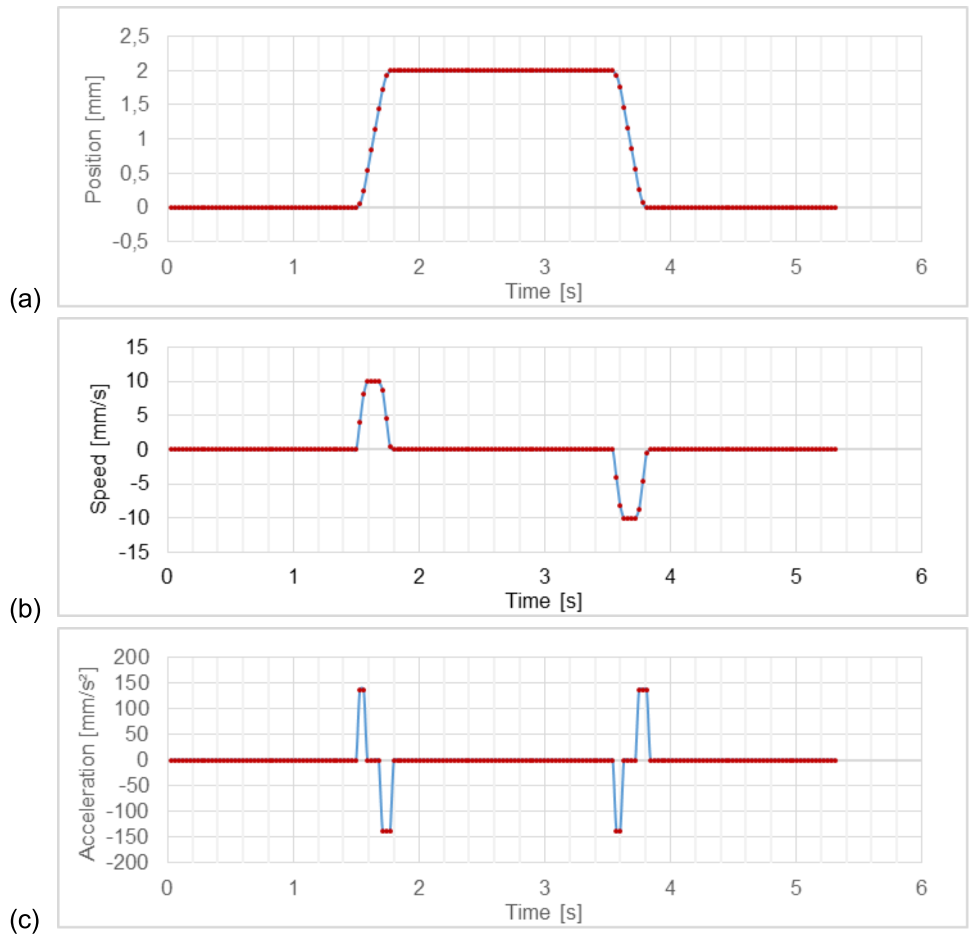


Figure 54: Y-axis motion with a target position of 2 mm, a maximum speed of 10 mm/s and maximum acceleration and deceleration of 137 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

With a higher speed or a lower acceleration and deceleration, the motion profile is smother, as in Figure 55 and Figure 56. The speed of 1075 mm/s in test 3 is approximately 786000 micro-steps/s or 921,1 min^{-1}, which is the default maximum speed value for the Lexium MDrive motor drive. As seen in Figure 55, the actuator responds to this parameter with a triangular speed profile that has its vertex at the speed of 194,71 mm/s. Therefore, the maximum speed is not reached in this movement, although it could be reached if the target position was further. However, 280 mm is the limit position of the y-axis.

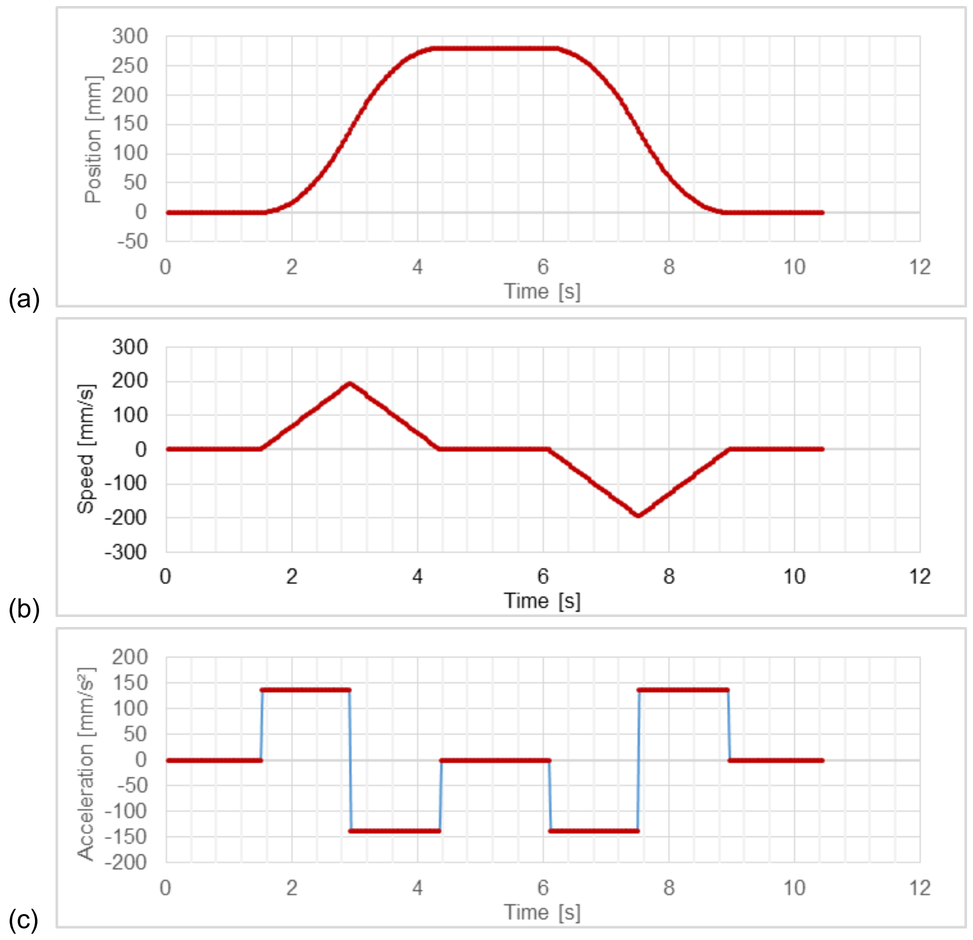
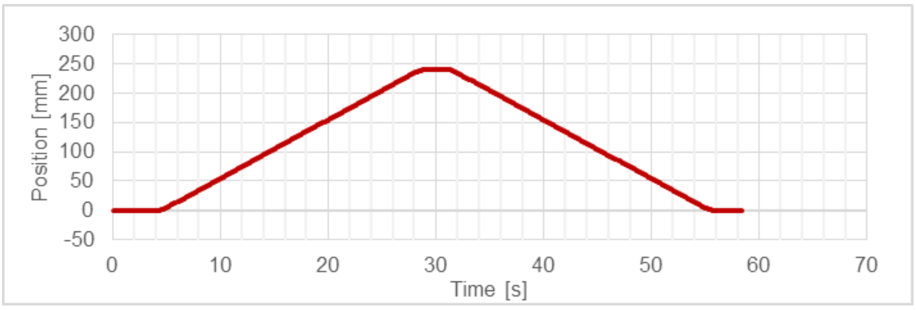
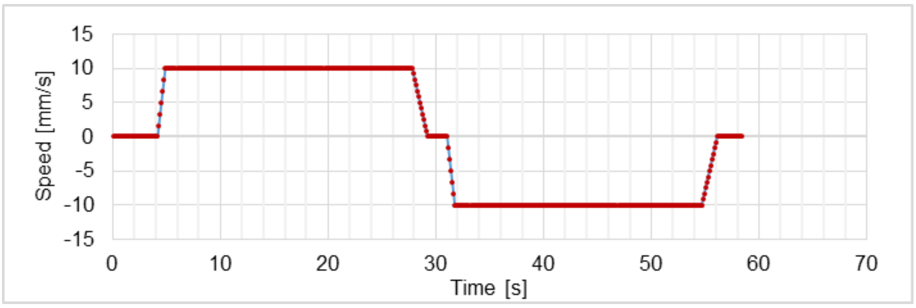


Figure 55: Y-axis motion with a target position of 280 mm, a maximum speed of 1095 mm/s and maximum acceleration and deceleration of 137 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

Figure 56 shows how the simulation responds to getting different values for acceleration and deceleration. Figure 56 (c) shows that the model processes them correctly. In test 1, 2 and 4, the speed profile is trapezoidal, as the target position is far enough to give time to reach the maximum speed of 10 mm/s.

(a)


(b)


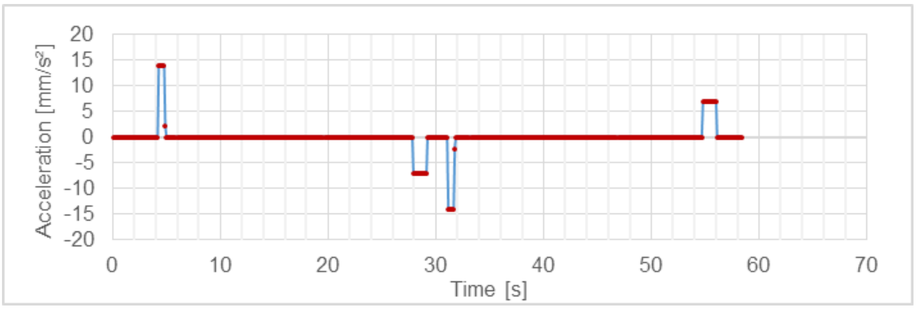
(c)


Figure 56: Y-axis motion with a target position of 280 mm, a maximum speed of 10 mm/s, a maximum acceleration of 14 mm/s^2 and a maximum deceleration of 7 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

6.4. Z-axis motion

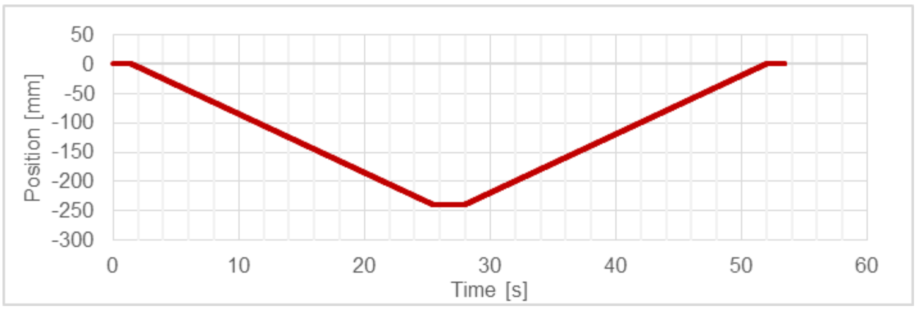
To test z-axis motion, four tests have been conducted with different values for the position, speed, acceleration and deceleration. The movement sequence is the same as for x and y axes. The characteristics of each movement can be seen in Table 6.

Test	Target position [mm]	Maximum speed [mm/s]	Maximum acceleration [mm/s^2]	Maximum deceleration [mm/s^2]	Comment
1	-240	10	141	141	Default values
2	-2	10	141	141	Close position
3	-240	1009	141	141	High speed
4	-240	10	14	7	Low acceleration and deceleration

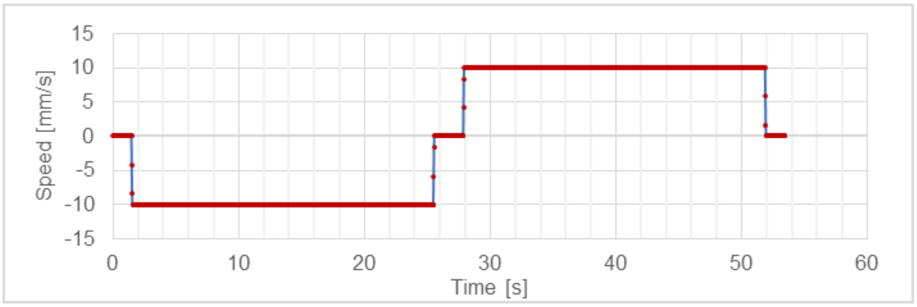
Table 6: Characteristics of the z-axis motion tests.

Figure 57, Figure 58, Figure 59 and Figure 60 show the resulting motion graphs of the tests. The same observations done in the y-axis are applicable for the z-axis. Note that the position of the z-axis is negative.

(a)



(b)



(c)

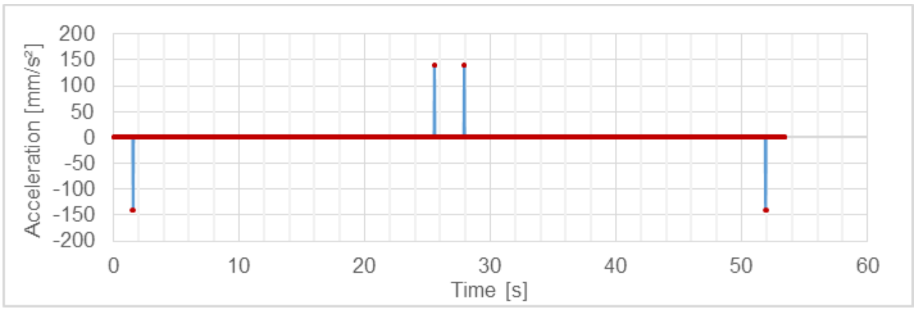
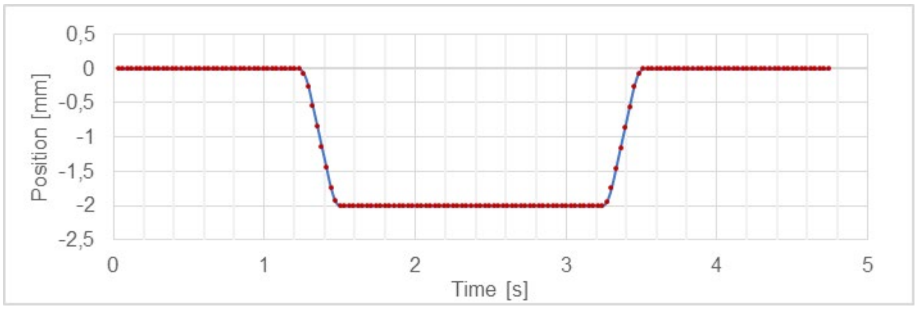
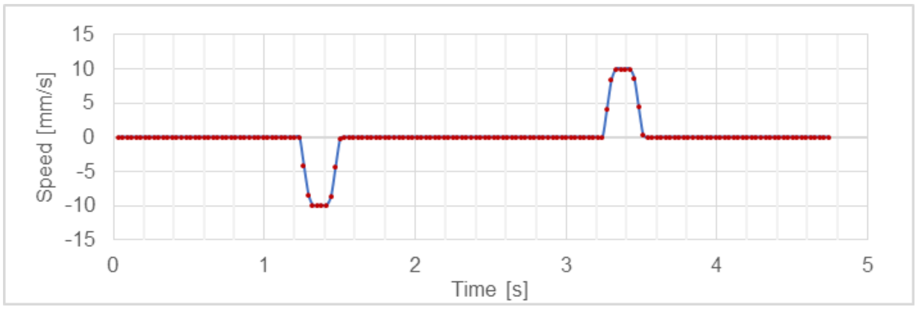


Figure 57: Z-axis motion with a target position of 240 mm, a maximum speed of 10 mm/s and maximum acceleration and deceleration of 141 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

(a)


(b)


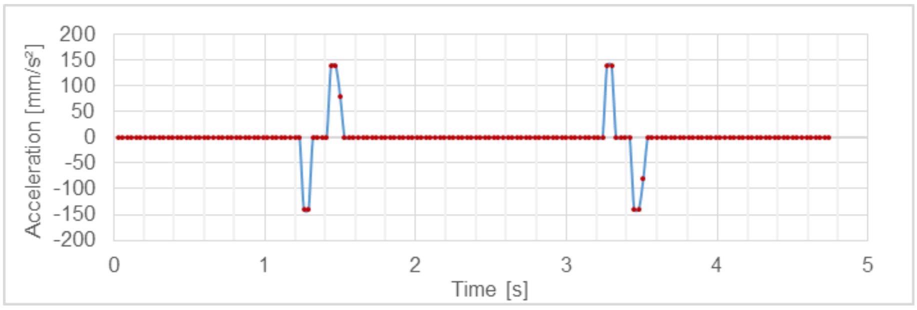
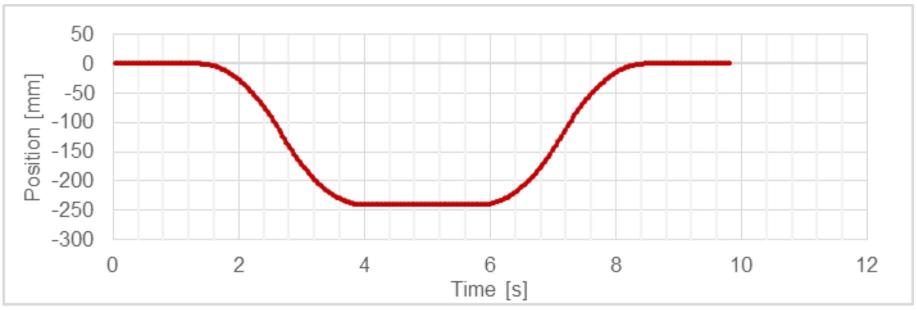
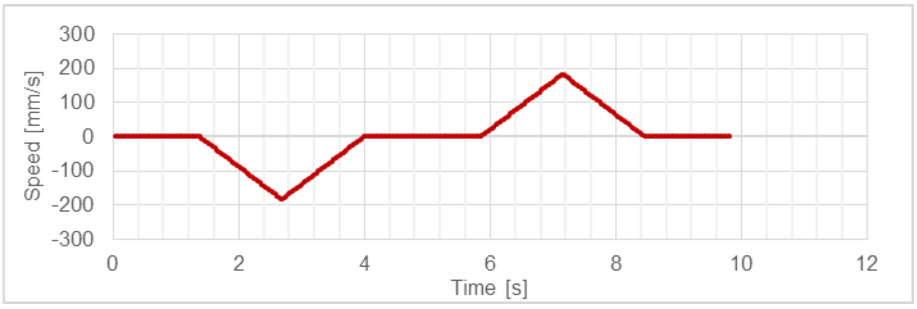
(c)


Figure 58: Z-axis motion with a target position of 2 mm, a maximum speed of 10 mm/s and maximum acceleration and deceleration of 141 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

(a)


(b)


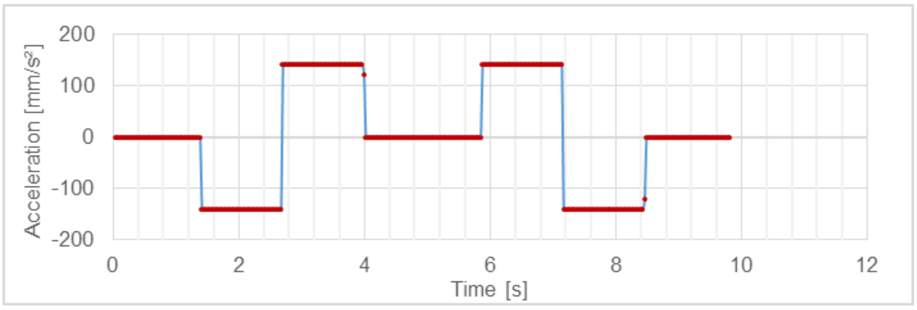
(c)


Figure 59: Z-axis motion with a target position of 240 mm, a maximum speed of 1009 mm/s and maximum acceleration and deceleration of 141 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

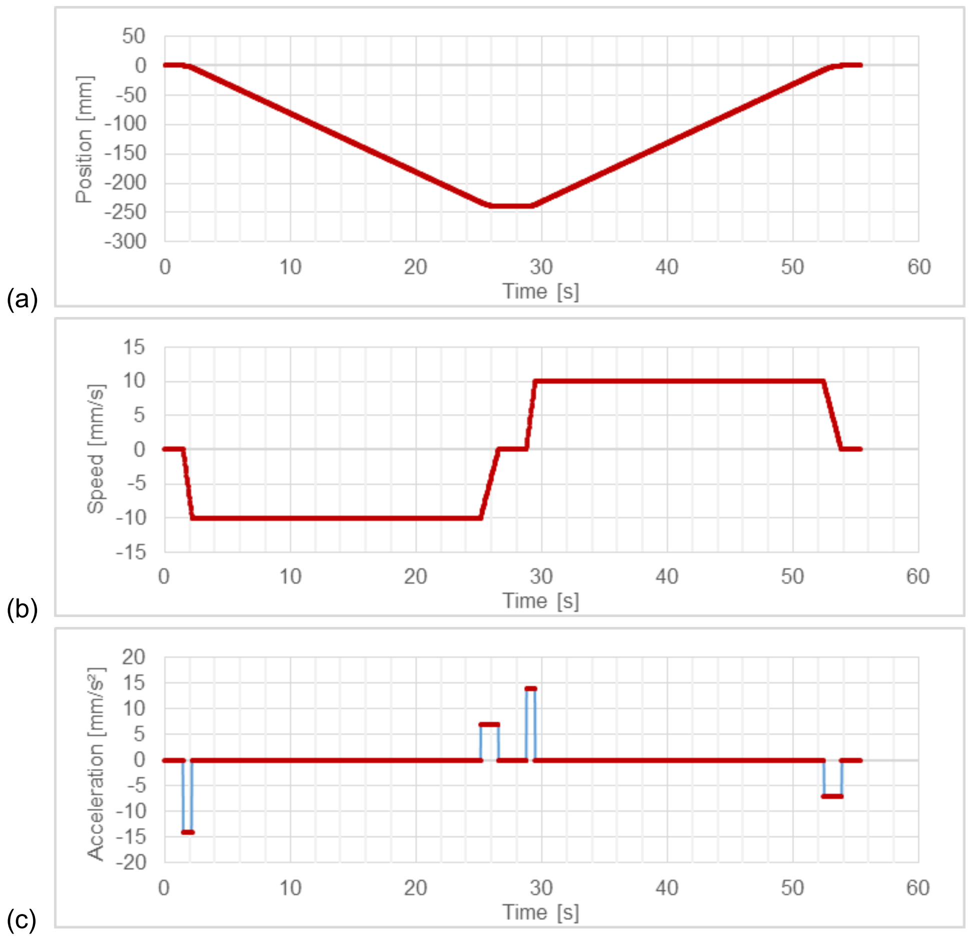


Figure 60: Z-axis motion with a target position of 240 mm, a maximum speed of 10 mm/s, a maximum acceleration of 14 mm/s^2 and a maximum deceleration of 7 mm/s^2. (a) Position, (b) Speed and (c) Acceleration versus time.

6.5. Discussion

After analysing the results, it can be concluded that the motion graphs are coherent and match with the expected physical behaviour of the machine.

At this point, the simulation can be used to evaluate the performance of different speed and acceleration values and optimize the pick and place algorithms. The default speed and acceleration values used in the z and y axes are quite low because they are selected to reduce the risks of unexpected collisions when doing the tests in the real machine. However, they are not optimum to pick and place a big amount of parts, as the operation cycle time is very high.

As pick and place applications do not require an area with constant speed, like machining operations, they usually use triangular speed profiles. Triangular speed profiles also provide the fastest movement between two points. They are achieved when using high maximum speeds, like in tests number 3 of the y and z axes. Therefore, the virtual model can be used to validate high-speed movement sequences and optimize cycle times before running them in the physical machine. This can be done remotely, without requiring expensive and scarce hardware to run the experiments. By doing so, the real commissioning phase can be reached with higher quality and more mature code. Likewise, the commissioning time can be shortened as most of the debugging will be already completed.

The benefits pointed out for this project apply to many other industry cases where movement sequences have to be verified and optimized. To name a few, machining, sorting, packaging or cleaning operations.

7. Economic analysis

The costs of this project are gathered in Table 7. They are associated with labour, software licenses and the hardware to run the software.

Concept	**Quantity [h]**	**Yearly cost [€/year]**	**Hourly cost [€/h]**	**Total cost [€]**
Junior engineer	360	28000	13,83	4980,24
NX Mechatronics Concept Designer license	360	25572	12,63	4548,38
SIMIT 10.1 M license	360	6125	3,03	1089,43
S7-PLCSIM Advanced V2.0 SP1 + TIA Portal V15.1 license	360	5000	2,47	889,33
Microsoft Office 365 license	360	130	0,06	23,12
Personal computer	360	667	0,33	118,58
Total				**11649,07**

Table 7: Costs of the project.

The length of this project is calculated considering that it is comparable to the 12 ECTS of the Master Final Thesis. Assuming that a 1 ECTS is approximately 30 h working hours, this project is equivalent to 360 h. The hourly cost of all the concepts is calculated assuming 8-hours workdays in a calendar with 253 workdays per year. The total cost comes from multiplying the hourly cost per the number of hours of this project.

The labour costs are calculated assuming an annual gross salary for a junior industrial engineer of 28000 €.

The software licenses used to conduct the simulation project are:

- NX Mechatronics Concept Designer
- SIMIT 10.1 M
- S7-PLCSIM Advanced V2.0 SP1
- TIA Portal V15.1

The yearly prices in Table 7 correspond to the cost of a floating license with a yearly subscription. Those prices are an approximation.

To write this report, multiple software of the Microsoft Office 365 professional package has been used. The price in Table 7 corresponds to the cost of a yearly subscription for one user.

Finally, this thesis has been conducted and written in a 2000 € personal computer. Assuming a computer life of 3 years, the cost per year is 667 €.

The resulting total cost of this project is 11649,07 €. This cost can be understood as the necessary initial investment to acquire the tools and the knowledge to develop a system with a model-based methodology. It costs 30,83% more than the cost of building a prototype, estimated in Table 8.

Concept	Quantity [h]	Hourly cost [€/h]	Cost [€]
Conveyor	-	-	714,00
Cartesian robot	-	-	3950,00
Tray	-	-	1000,00
Structure	-	-	840,00
Mechanic technician	80	15	1200,00
Electrical technician	80	15	1200,00
Total			**8904,00**

Table 8: Costs of building the prototype.

When it comes to the time, the number of hours devoted to this project are 360 h, which are equivalent to 9 weeks working 40 hours per week. However, it is estimated that after gaining the knowledge to build the first model, the time required to create a new one would be between 2 and 4 weeks. Therefore, the cost of the project would be also divided by 0,44 in the worst case, resulting in 3957,33 €.

The assembly of the prototypes usually takes 3 weeks, while the lead time between the order and the reception of the manufactured components can vary between 2 and 8 weeks. During the manufacturing, electric engineers prepare the wiring diagrams and the I/O assignment, which takes 3 weeks. After that, the simulation can be developed. Therefore, there is more than enough time and human resources to parallelize the development of the model while the machine components are being manufactured and assembled. If the commissioning is also started before the assembly is finished, which typically takes 3 or 4 more weeks, the total project length can be reduced. Figure 61 shows a possible Gantt chart of the project introducing virtual commissioning.

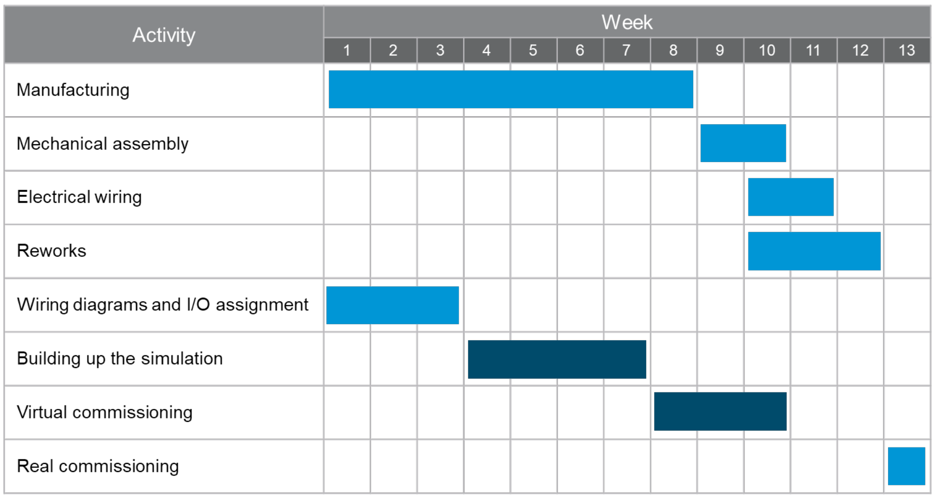


Figure 61: Gantt chart of the machine-building process using virtual commissioning.

On the other hand, if hardware errors are detected once the machine is built, reworks or new parts might be needed, and thus an additional cost and delay in the machine start-up. Validating the machine concepts in a virtual prototype before the machine is approved for manufacturing can help reduce these unexpected expenses and time.

All in all, despite the initial investment required, adopting a model-based development approach shortens the project timeline and potentially brings cost-savings in the long term.

8. Environmental impact

The environmental impact associated with this project comes from the energy consumption of the personal computer and the CO_2 emissions generated to produce this energy.

Figure 62 and Figure 63 show the evolution of the electric energy generation and the CO_2 emissions in the Spanish Peninsula from the 11th of April 2020 at 3 am till the 12th of April 2020 at 3 am [16].

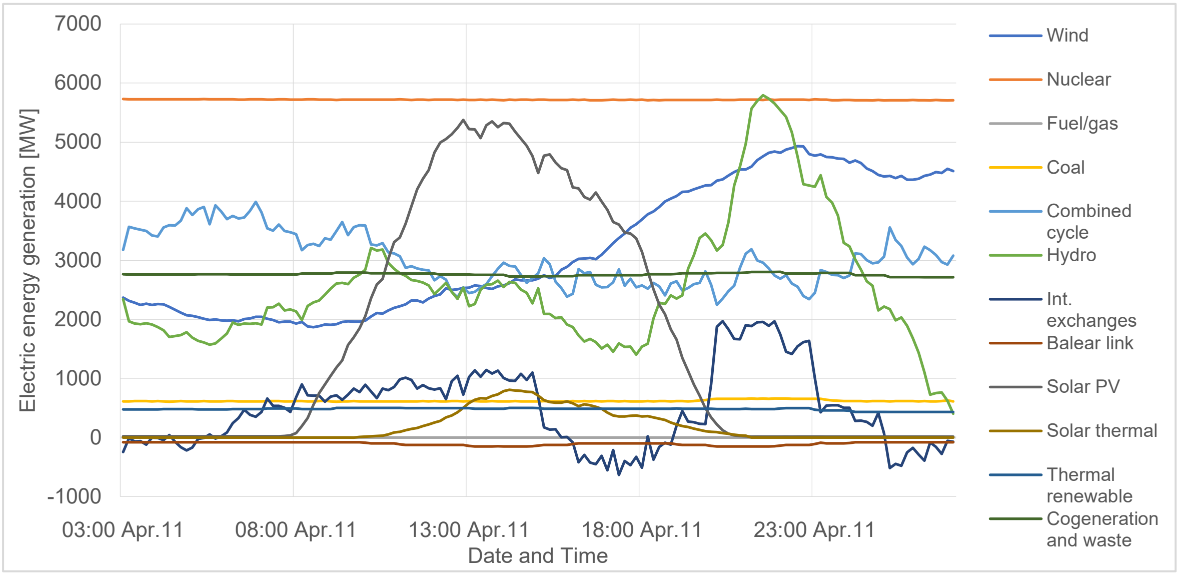


Figure 62: Evolution of the electric energy generation by central type along one day in the Spanish Peninsula.

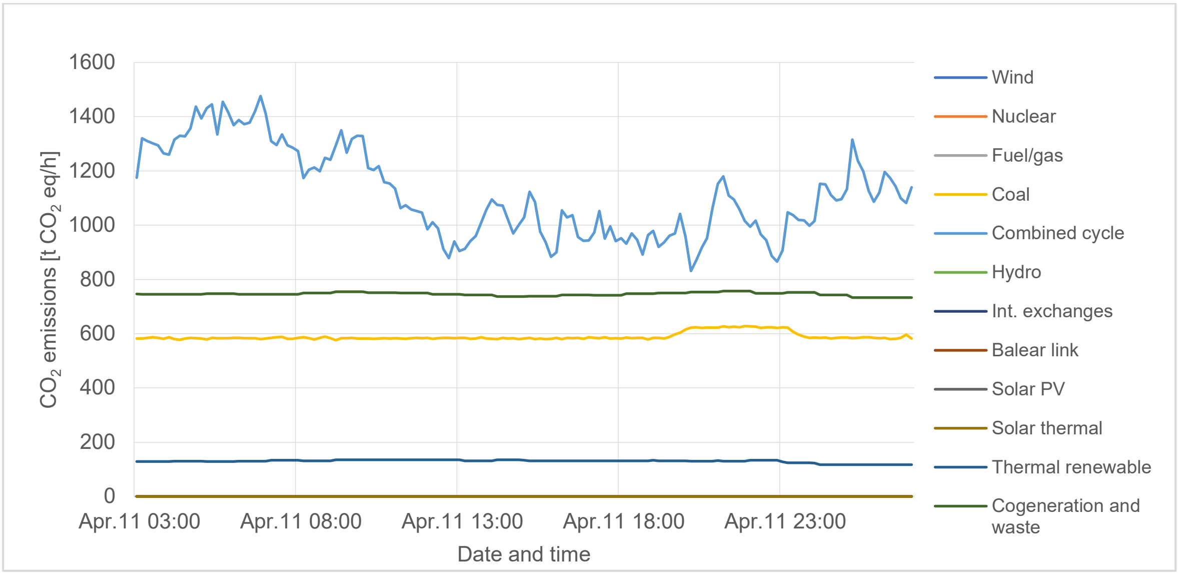


Figure 63: Evolution of the CO_2 emissions by central type along one day in the Spanish Peninsula.

Figure 64 shows the resulting CO_2 emissions caused by the generation of electric energy along one day in the Spanish Peninsula. As it varies a lot, the average of 35,62 gCO_2 eq/MJ is taken for the calculations.

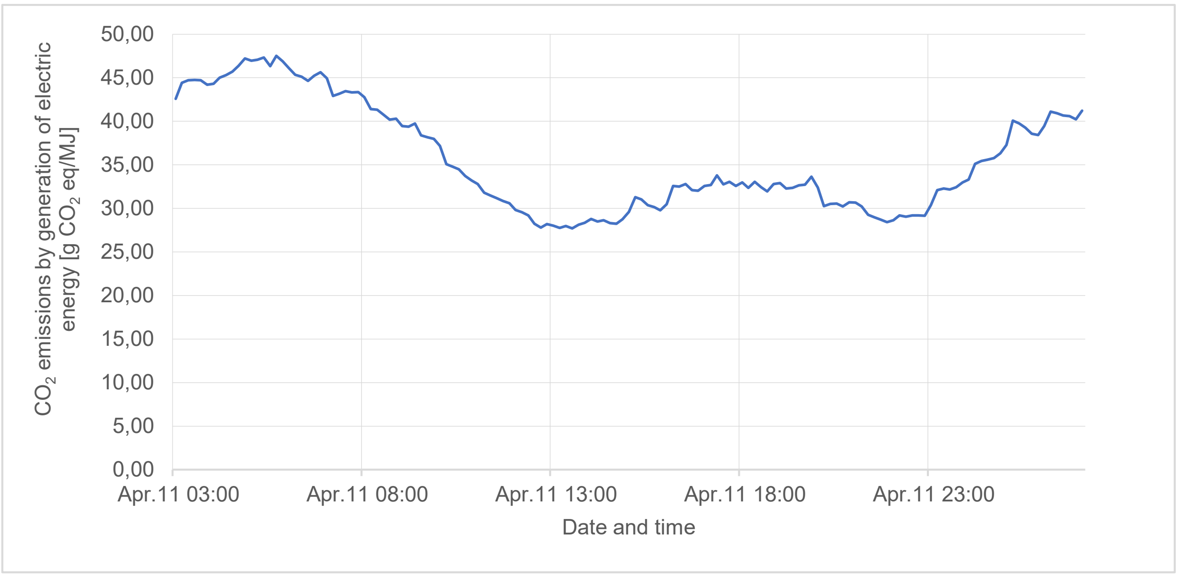


Figure 64: Evolution of the CO_2 emissions by the generation of electric energy along one day in the Spanish Peninsula.

As the power of the personal computer is 150W and it is used for 360h, the resulting CO_2 emissions are shown in Table 9.

Power [kW]	**Usage time [h]**	**Energy [kWh]**	**CO2 Emissions [g CO_2 eq]**
0,15	360	54	6924,96

Table 9: CO_2 emissions associated with the use of the personal computer.

Conclusion

The results obtained from the validation of the simulation proof that the model provides a good representation of the physical machine. The simulated physics, behaviour of the motor drives and data exchange perform as expected when the control program is executed in the virtual controller.

The use case presented in this project shows the advantages of using a model-based development approach. Firstly, the machine can be commissioned before the real hardware is assembled. This enables parallelization of tasks. For example, automation engineers can develop the control programs and test them while the components are being manufactured and the machine is being built. As they arrive at the real commissioning phase with an already validated system, it becomes be much faster to get the machine up and running to do further testing or deliver it to customers.

Testing in a virtual model also enables early error identification and reduces the risk of damaging real equipment while the machine operation is validated. In the application of this project, for instance, multiple grippers could be tested without the danger of breaking 3D printer parts. Likewise, it can be used to validate high-speed movement sequences, which could cause unexpected collisions if they are tested in the real hardware first.

By having a digital twin of the system, the control software can be developed from multiple locations. This is particularly useful in a multinational company like HP, where most of the projects are co-developed by international teams. It also decreases the investment needed in hardware for testing. Likewise, it provides the opportunity to outsource some of the development to third parties. In the use case of this project, the virtual model can be used to develop and validate pick and place algorithms for customers without having access to their facilities or 3D printed parts. It must be pointed out that having multiple users working with the simulation requires having a version tracking system to prevent developers from working with faulty or outdated models.

Using virtual system models can also help in the early stages of development. Machine concepts and operation sequences can be validated in tools like NX Mechatronics Concept Designer without having to develop the electronics or the software. If the concept does not work, the equipment is not built. However, this does not remove the need to build prototypes to identify performance characteristics that can not be simulated in the virtual model.

These tools also make it easier to agree upon the functionality of the system between various disciplines. The integration issues are reduced as does the documentation of the project, as the simulation serves as the documentation.

When it comes to customers, the virtual model can be a tool to demonstrate the proposed system to the client. Then, the client can suggest changes and the machine manufacturer can implement them at a minor cost before the system is built. In this way, the system builder also makes sure the final product is aligned with customers' requirements.

Finally, model-based development opens the door to test automation and automatic generation of code, which would make the development process even more efficient.

Despite the multiple advantages of model-based development and virtual commissioning, some stoppers still prevent companies from adopting this methodology. In the beginning, it needs an investment in the simulation tools and the know-how to create the models, which is something not paid by the customer. Furthermore, the amount of time and effort needed to develop and fine-tune the first models is significant. In this project, a big part of the time was invested in understanding and modelling the Lexium MDrive, as no standard simulation block was available. However, once the model of a component is created, it can be reused as many times as desired, reducing the time required to perform future simulations.

This project ends with the delivery of a simulation ready to understand the position, speed and acceleration values introduced by the user and provide a correct visualization of the machine operation. With this, the scope of this work is covered. However, to simulate the complete system described in chapter 4, the following future work would be needed:

- Establish the communication between the virtual controller and the PC that calculates the pick and place algorithms. This would allow obtaining the target coordinates automatically instead of introducing them manually.
- Add the packaging box and the 3D printer parts detected by the camera in the physics simulation. This would enable detecting collisions between these objects and the machine.
- Develop the program to control the motion of the x-axis, the gripper to pick and place the parts and the machine door sensors that prevent access to the machine when it is running. This would be done by automation engineers.
- Model the physics of the gripper and the behaviour of the driver that controls it.
- Model the behaviour of the machine door sensors.

Once this work is completed, automation engineers will be able to test in a virtual model the pick and place movement sequences for each part, as well as the operation of the whole machine.

Acknowledgements

I would like to thank Pablo Murciego for introducing me to model-based development processes and mindset and guiding me through the election of the topic of this project. Also, for supporting the development of this work by providing all the licenses and resources needed.

I would also like to thank Lluïsa Jordi for her support and advice on the format and language of this report and the academical suitability of this project.

I would also like to acknowledge Álvaro Hernández for his technical support regarding the Siemens simulation toolset.

Likewise, I would like to thank Blanca de Prada, Jose Juan Hidalgo and Carles Centeno for helping me understand the control program and operation of the machine simulated in this project.

Finally, I would like to thank my family and friends for their continuous and unconditional support.

Bibliography

Citations

[1] A. FERNÁNDEZ, M. A. EGUÍA AND L. E. ECHEVERRÍA, *Virtual commissioning of a robotic cell: an educational case study,* 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, pp. 820-825. [https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/document/8869373/references#references]

[2] H. VERMAAK AND J. NIEMANN, "Virtual commissioning: A tool to ensure effective system integration," 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), Donostia-San Sebastian, 2017, pp. 1-6. [https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/document/7945899/references#references]

[3] REINHART, G., WÜNSCH, G. *Economic application of virtual commissioning to mechatronic production systems*, Prod. Eng. Res. Devel. 1, 371–379, 2007. [https://doi-org.recursos.biblioteca.upc.edu/10.1007/s11740-007-0066-0]

[4] SHAHIM, N., MØLLER, C. *Economic justification of Virtual Commissioning in automation industry*, *2016 Winter Simulation Conference (WSC)*, Washington, DC, 2016, pp. 2430-2441. [https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/document/7822282]

[5] BORSTELL, H., REGGELIN, T. *Towards Virtual Commissioning of Image-based Information Systems for State Detection in Logistics*. 9th IFAC Conference Manufacturing Modelling, Management and Control MIM 2019, Berlin, 2019. [https://www.researchgate.net/publication/336369277_Towards_Virtual_Commissioning_of_Image-based_Information_Systems_for_State_Detection_in_Logistics#pf3]

[6] ARC ADVISORY GROUP*. Siemens PLM Software's Advanced Machine Engineering Streamlines Machine Development*, ARC White Paper, 2015. [https://www.plm.automation.siemens.com/media/global/en/AME_-_ARC_White_Paper%20-%20Advanced_Machine_Engineering_Streamlines_Machine_Development_tcm27-53754.pdf, 29th of February 2020]

[7] ITQ, *ITQ Corporate Brochure*, 2018. [https://www.itq.de/wp-content/uploads/2019/06/ITQ_corporate_brochure.pdf, 29th of February 2020]

[8] SIEMENS AG, *Address critical design, engineering and manufacturing challenges with Advanced Machine Engineering*, Webinar. [https://www.plm.automation.siemens.com/global/en/webinar/advanced-machine-engineering/25892, 29th of February 2020]

[9] SIEMENS AG, *SIMATIC Machine Simulator V2.0. Getting started*. [https://support.industry.siemens.com/cs/attachments/109758943/s71500_simatic_machine_simulator_getting_started_v2.0_en-US.pdf, 21st of March 2020]

[10] SIEMENS AG, *SIMIT Simulation Platform (V10.1). Operation Manual.* [https://support.industry.siemens.com/cs/attachments/109759317/SIMIT_enUS_en-US.pdf, 21st of March 2020]

[11] SIEMENS AG, *Mechatronic Concept Designer*. [https://www.plm.automation.siemens.com/global/en/products/mechanical-design/mechatronic-concept-design.html, 21st of March 2020]

[12] HP INC., *HP Metal Jet technology*, Technical white paper. [https://h20195.www2.hp.com/v2/GetDocument.aspx?docname=4AA7-3333ENW, 8th of March 2020]

[13] LINK3D, *Link3D Additive Manufacturing Technologies Overview.* [https://solution.link3d.co/3d-printing-resources.html, 7th of March 2020]

[14] PLC DEV, *How PLCs Work.* [http://www.plcdev.com/how_plcs_work, 7th of March 2020]

[15] SIEMENS PRODUCT LIFECYCLE MANAGEMENT SOFTWARE INC., *Siemens Documentation: Mechatronics Concept Designer*, 2019. [https://docs.plm.automation.siemens.com/tdoc/nx/1872/nx_help/#uid:index_mechatronics, 10th of April 2020]

[16] RED ELECTRICA DE ESPAÑA, *Spanish Peninsula – Electricity demand tracking in real-time*. [https://demanda.ree.es/visiona/peninsula/demanda/total/2020-04-10, 13th of April 2020]

[17] CARBONELL, B. *Simulation with MCD, SIMIT and TIA Portal.* [https://youtu.be/Ulq8Oe2rYSI, 18th of April 2020]

Complementary bibliography

[18] C. HENKE, J. MICHAEL, C. LANKEIT AND A. TRÄCHTLER, *A holistic approach for virtual commissioning of intelligent systems: Model-based systems engineering for the* *development of a turn-milling center*, 2017 Annual IEEE International Systems Conference (SysCon), Montreal, QC, 2017, pp. 1-6. [https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/document/7934735]

[19] GUERRERO, L., LÓPEZ, V., MEJÑIA, J., *Virtual Commissioning with Process Simulation (Tecnomatix)*, Computer-Aided Design and Applications, 11:sup1, S11-S19, 2014. [https://www.tandfonline.com/doi/full/10.1080/16864360.2014.914400]

[20] DEFENSE ACQUISITION UNIVERSITY PRESS, *Systems Engineering Fundamentals*, 2001. [https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-885j-aircraft-systems-engineering-fall-2005/readings/sefguide_01_01.pdf, 29th of February 2020]

[21] SIEMENS AG*, SIMATIC S7-1500T - Virtual commissioning for kinematics in NX MCD with Software in the Loop*, January 2019. [https://support.industry.siemens.com/cs/document/109760078/simatic-s7-1500t-virtual-commissioning-for-kinematics-in-nx-mcd-with-software-in-the-loop-?dti=0&lc=en-US, 21st of March 2020]