Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

CREACIÓN DE DIGITAL TWINS PARA LAS ESTACIONES 1 Y 2 DE LA LÍNEA DE ENSAMBLAJE AUTOMÁTICO FMS-200

Autor:

Sánchez Álvarez, Tomás

Responsable de Intercambio en la Uva:

Eusebio de la Fuente López

Universidad de destino:

UCLL Leuven-Limburg

Valladolid, febrero de 2023.

TFG REALIZADO EN PROGRAMA DE INTERCAMBIO

TÍTULO:	CREATION OF DIGITAL TWINS FOR STATION 1 AND STATION 2 FROM SMC'S AUTOMATIC ASSEMBLY SYSTEM FMS-200
ALUMNO:	Tomás Sánchez Álvarez
FECHA:	27 de Enero de 2023
CENTRO:	UCLL Leuven-Limburg, campus de Diepenbeek
UNIVERSIDAD:	UCLL Leuven-Limburg
TUTOR:	Geert Leen

Resumen

La Industria 4.0 está siendo considerada como el cambio más moderno del paradigma industrial gracias a la introducción del Internet de las Cosas. El sector secundario actualmente invierte cuantiosas cantidades de tiempo y dinero en tratar de introducir estas nuevas tecnologías en sus procesos, una de estas tecnologías son los llamados Digital Twin o Gemelo Digital.

Esta memoria describe la confección paso a paso de un Digital Twin para una cadena de montaje modular y completamente automatizada llamada FMS-200. La construcción es descrita para el software de Siemens NX, y cada paso va acompañado de los problemas y soluciones necesarios para su creación. Posteriormente se muestra el proceso necesario para unirlo a otro Gemelo dedicado a otra parte de la misma máquina y su simulación simultánea, esto se logra haciendo uso de las herramientas proporcionadas por el software de TIA Portal.

Palabras clave: Industria 4.0, Internet de las Cosas, Digital Twin, Siemens NX, Simulación.

Abstract

Industry 4.0 is becoming well stablished as a completely new change of the industrial paradigm thanks to the introduction of the Internet of Things. Industries invest large amounts of time and money in the shake of introducing these new technologies within their processes, one of them is called Digital Twin.

This thesis guides step by step through the creation of a Digital Twin for a completely automatic modular assembly line called FMS-200. The steps are explained for the Siemens NX software accompanied with every problem and solution that came up during the construction. After that, the thesis describes the process required to link the newly created Digital Twin with another one intended for a different part of the same machine and how to simulate with them simultaneously, this is achieved by making use of the tools provided by the TIA Portal software.

Key Words: Industry 4.0, Internet of Things, Digital Twin, Siemens NX, Simulation.

Table of contents

List of figures

List of abbreviations

DT	**D**igital **T**win
UCLL	**U**niversity **C**ollege **L**euven **L**imburg
IoT	**I**nternet **of** **T**hings
MCD	**M**echatronics **C**oncept **D**esigner
TIA Portal	**T**otally **I**ntegrated **A**utomation **Portal**
PLC	**P**rogrammable **L**ogic **C**ontroller

1. Introduction

1.1. Objectives

UCLL University engineering program is focused on providing students with the most updated education system, teaching them a wide range of skills that are really useful in the todays industry as well as for the future of its scope. As industry is evolving towards a completely automated and interconnected paradigm, new engineers have to adapt to fit new era called “Industry 4.0”.

Nowadays the students are most familiarized with the automatization of machines making use of Programmable Logic Controllers (PLC). These units are linked to different machines so students can simulate their codes and receive a visual and tangible responses in return. One of these machines is an automated assembly line called FMS-200, this contraption designed for educational purposes provides students with a wide range of sensors and actuators as the one used on the real industry.

A new technology regarding “Industry 4.0” is Digital Twins, a DT is a digital replica of an existing machine that behaves as its real twin does. UCLL does it best to stay up to date with the real industry so the study of Digital Twins is the way to go to guide students into this new era of engineering. FMS-200 line is divided by modules, previously another student designed and created the DT for the first module, so the creation of the next station digital model was proposed for this project.

The first objective of this project was to create a Digital Twin of Station 2 of the FMS-200 automated assembly line and to prove the benefits of using this new technology in the learning program. The second objective was once this DT was created it would be linked with the previously done model of Station 1 to test if more than one independent DT can be connected and simulated together.

1.2. Thesis outline

The first chapter of this thesis called **Background** gives a broad definition of what is “Industry 4.0” as well as a for the concept “Digital Twin”. Then a more in-depth explanation of the FMS-200 assembly line is given to better understand how it works, this paragraph mostly focuses on Station 2 to better understand further explanations of its modelling.

The next chapter called **Digital Twin of Station 2** shows step by step the making of the DT, as well as its simulation and proving the resemblance with the real model. Various problems that came up during the process are shown next to their resolutions.

The fourth chapter, **Union with Digital Twin of Station 1**, shows all problems that needed to be fixed to be able to connect both stations. It ends up with a satisfactory simulation proving the possibility of simulating more than one PLC and DT together.

The las chapter is composed of all the **Conclusions** of the thesis and the justification of why the objectives were achieved.

2. Background

2.1. Industry 4.0

Industry has become one of the main pillars of the actual paradigm of economy, as a broad definition it can be described as the production of goods which are highly mechanized and automatized. Revolution is also a word that is commonly related with industry, these “Industrial Revolutions” occur when big technological improvements really change the behavior in which this economy branch works. First revolution occurred when major improvements in mechanization where made, in the second one electrical energy became the main source of power, and in the third one digitalization was introduced and quickly spread to all the industry processes. Nowadays, the combination of Internet technologies and the improvements on technologies regarding “smart” objects (internet of things) are leading people to think there is a new inflexion point in this field and so the concept “Industry 4.0” is being used to describe the fourth Industrial Revolution.

Industry 4.0 implies a complete change of the way industrial processes work, this change bets for a more automated and interconnected process making use of one the greatest inventions ever made by humankind, the Internet. This Revolution lays its foundation in some key technologies:

- **Smart Factory:** Machinery are equipped with a wide variety of sensors, actuators and PCs which are in charge of a complete automatized process.
- **Internet of Things (IoT):** Some sensors and processors of these newly named “smart machines” are equipped with IP addresses that allow them to connect with other web-enable devices. This means they can be controlled by operators from a PC, but also they can be interconnected with other machinery so they can communicate and work together as whole creating complex network based procedures.
- **Cloud Computing:** Now that industrial processes have been connected to the internet a wide range of data can be stored, as it can be production data, sales, supply chain numbers and so many more. This data can be stored in the Cloud,

so all information crosses in the same meeting point and allows a more efficient use of it to optimize manufacturing process from the first step to its end.

- **AI and machine learning:** The large amount of data that can be collected, from not only machines but the whole business process, can be processed by Artificial Intelligence. These technologies can create insights providing visibility, predicatively and automatization not only of machines but business processes.
- **Digital Twins:** A DT is a virtual replica of processes, production lines, factories and supply chains. They are creating by utilizing data collected from IoT sensors, PLCs and other devices connected to the internet. These models can be used to optimize processes and create new products.

2.2. Digital Twin

A Digital Twin is a virtual replica designed to accurately reflect a physical object. The DT is interconnected with the reference object or so called real twin via exchange of data in real time. On an industrial context, this technology is used to recreate machines and complex assembly lines in the digital world.

These digital models are designed to mimic the state of its physical twin. The contraption is endowed with various sensors and actuators that behave as those of the real object of study. Moreover, these software tools provide the user a wide range of data of the different parts of the model for its further study. The virtually identical behavior of the DT with the real machine leads to various advantages:

- **Speed prototyping:** As well as a DT can mimic an already existing object the process can be reversed, this means a new machine or contraption can be created and tested digitally before is actual construction. This new way of design means prototypes can be done quicker as there is no need to build the machine to be able to test its capabilities. It also improves re-designing for the same reasons.
- **Cost-effective:** Almost all resources use for the creation of a DT are digital. This means there needs to be trained operators with wide knowledge on this field, but these costs will be lower than those of usual prototyping with the time. The traditional prototyping and re-designs are time-consuming and expensive procedures because of the use of physicals materials and time-consuming labor

as well as the need of running expensive destructive test. On the other hand, DT can be run through the same test without the material loss and cutting on these expenses.

- **Optimization:** As simulations can be run efficiently with the use of DTs and their re-design is quicker that the one with traditional methods, this means machines or complex contraptions can be re arranged in order to find the most optimize layout for its use. It can also lead up to testing out different machines or technologies for the same task to find the most suitable one. As a conclusion, with the use of these models there can be found the most efficient solutions without the need of going through the same process in real life, this would mean designing, building and testing a solution that may end up not being approved meaning a loss of time, money and resources.

2.3. FMS-200

The FMS-200 is a mechatronics learning system designed for Industry 4.0 with educational purposes. This installation consists of ten different stations, each of them can work completely autonomously or connected with others in order to create a complex assembly. Its modular design allows to work with just one station by itself up to complex installations including all ten of them.

These stations consist of many components and technologies already in use in the industry. To name some of them, there are pneumatic and hydraulic actuators, electric motors, sensors, electrical connections and many more. Moreover, each station is controlled by using a Programmable Logic Controller (PLC) by Siemens, these controllers are widely used in industry and their understanding and controlling have become a main subject of focus for UCLL students. The resemblance of these technologies with those use in the actual industries provide students with a really useful set skills and knowledge for their professional future. The FMS-200 comes included with troubleshoot box, with this box system failures system can be simulated in real time for students to diagnose and fix.

Regarding UCLL, their installation consists of eight modules connected in a lineal transfer layout as shown in figure 1. The assembly takes place in a sequence where station 1 goes through its automated process, providing some part of the contraption to a transport belt located in the middle of all the module. Then the belt moves the contraption to the next station triggering its movements, once that station has provided the belt with its intended part or function the process repeats itself until the belt has gone by all stations and the contraption has been fully assembled. The assembly consist of a box shaped body where a bearing and axis are introduced and lately closed and screwed with a cover, the final result is a solid body with a rotating axis as the one found in an office chair for example.

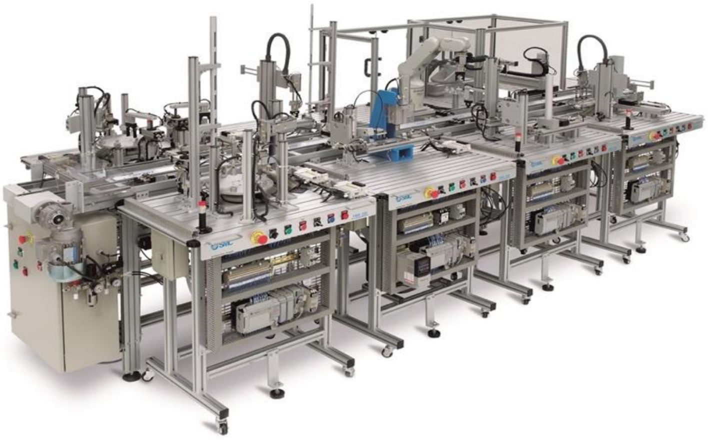


Figure 1. FMS-200 lineal transfer layout with 8 modules

Each station is provided of its own electrical box as well as their individual PLC from the 300 series. All these units are interconnected, including the belt one, with a master PLC via an industrial communication bus. Below there is a list of all modules with a brief explanation of their intention:

- **FMS-201 Body supply:** The box shaped body that will house is moved from a storage column all the way to the transport belt.
- **FMS-202 Bearing selection/supply:** System requires of one of the two different height bearings of the storage, and the one selected is placed on top of the body on a centrical hole intended for it.
- **FMS-203 Hydraulic press:** The bearing on top of the body is pressed down the hole which diameters are almost identical. This method ensures the bearing will not slide out of position.
- **FMS-204 Shaft selection/supply:** The system once again chooses between different length shafts and the one need is placed inside the bearing.

- **FMS-205 Cover selection/supply:** A cover is placed on top of the body to prevent the shaft sliding out.
- **FMS-206 Screw supply:** Four screws are placed through the for corners of the cover, these holes are aligned with those in the body intended for this use.
- **FMS-207 Robotized screwing:** A robotic arm is in charge of screwing down all screws to finalize the assembly process.
- **FMS-208 Automatic warehouse:** The final product is picked up from the transport belt and stored on top of a table.

To this day UCLL is missing the robotic arm and some parts of the hydraulic press. The absence of those machine do not affect this project as the first part of the thesis focuses on Station 2, for that reason its movements and behavior is explained in more detail down below. The final part of the thesis also focuses on Station 1.

2.3.1. Station 2 (FMS-202)

The Digital Twin created for this thesis is based on the second station of the assembly, the bearing supply. Once the body of the contraption arrives to the station on the transport belt, this triggers the automated process to place the bearing on top of the body.

The process has been overcomplicated to provide students with more variables, movements and sensors, this way their training can result in larger knowledge of programming PLCs. Every different movement of the station is described below to give a more specific look to understand the intent of this build, the left figures are pictures of the real station and the ones on the right are their respective 3D models:

- **Supply:** Bearings are stacked in the cylindrical hollow tube. When the program requires of a new one, the pneumatic cylinder pushes it out to position it for its transfer. Once the piston retracts a new bearing will slide down into position for future use when required.

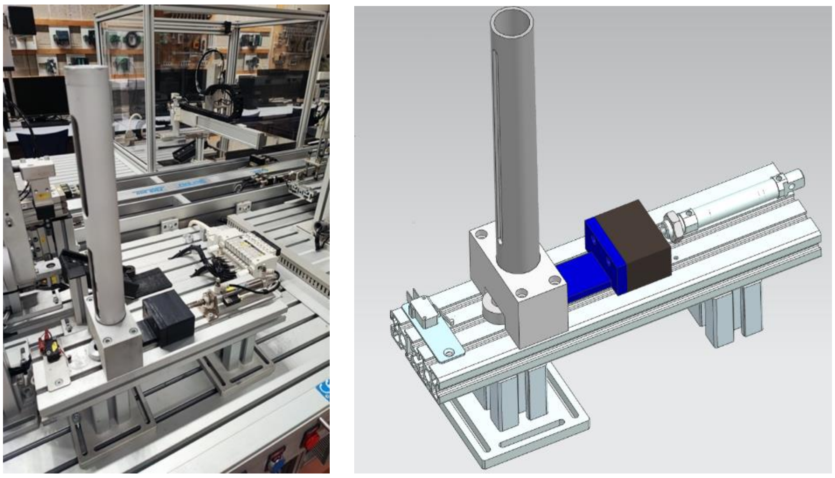


Figure 2. Bearing supply and its 3D model

- **Transfer:** The arm on figure 3 moves in a circular motion, first moving itself to the position where the bearing was placed beforehand and then moving it to the platform of the lift. The body in front of the arm is provided with a pneumatic claw, when it opens because of the air pressure it provides enough friction to hold the bearing by its internal face while it is being moved. The body of the claw keeps itself on the same vertical position during the circular motion thanks to a belt built inside the arm. Whichever degree the arm moves, it also rotates the claw to counter the rotation.

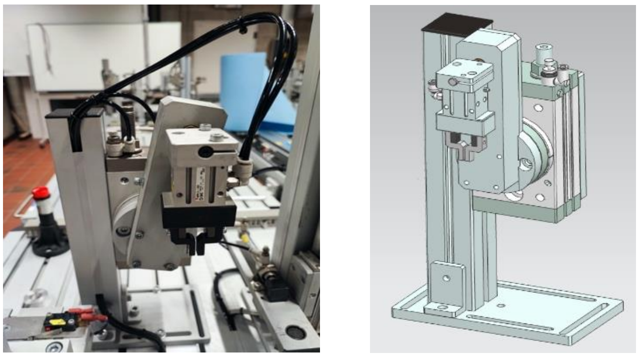


Figure 3. Transfer arm and its 3D model

- **Lift and potentiometer:** The platform moves vertically to reach the potentiometer on top. This potentiometer reads the height of the bearing to check if it is the one needed. When the lift comes back to its original position two things can happen, if the measure is correct, the program proceeds to the next step, if it is not, the pneumatical cylinder on the back ejects the bearing and the program should ask for a new one.

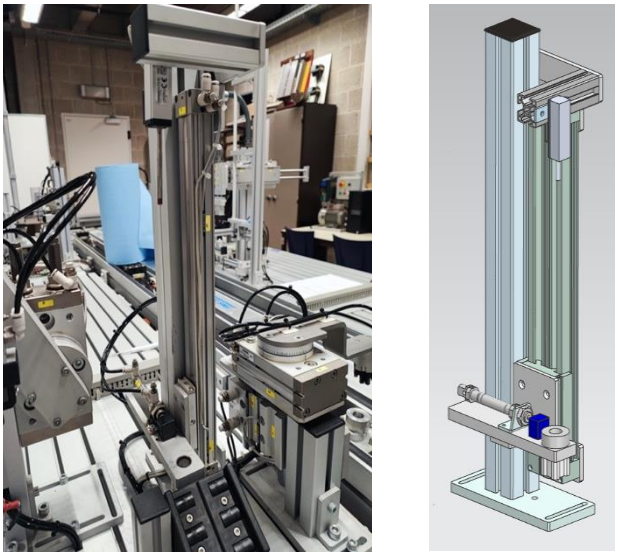


Figure 4. Lift with potentiometer and its 3D model

- **Transfer to the body:** When the measure is correct, the moving arm of the picture below spins on its axis to position itself on top of the bearing, then the block below lets out air pressure, this results in the arm and claw moving downwards into the inside of the bearing. Then the claw opens up holding the bearing, the block goes up and the moving arm rotates until it is on top of the body of the contraption. Finally, everything slides down once again and the bearing is left put inside its designated hole in the body.

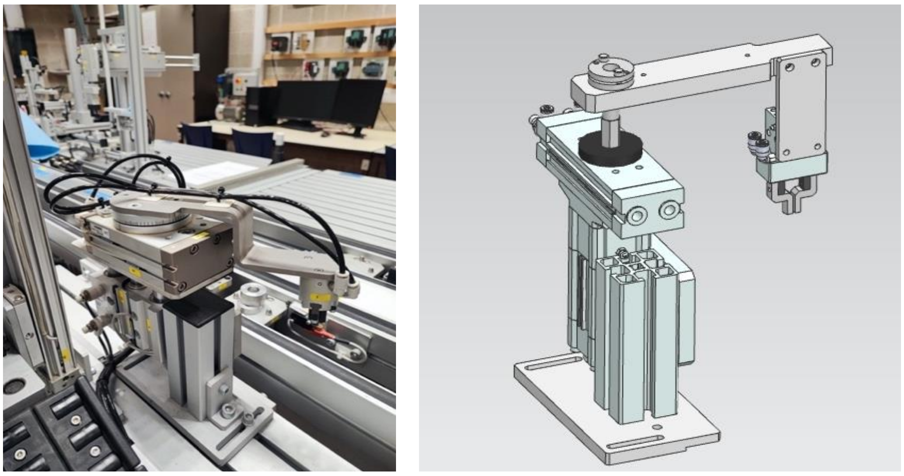


Figure 5. Transfer to the body arm and its 3D model

3. Digital Twin of Station 2

3.1. Siemens NX

Siemens NX is a CAD program designed to give the user a broad range of tools to model, sketch and simulate real life situations. For this project sketching tools are needed, but there is a tool that will be crucial to create the Digital Twin called MCD. Mechatronics Concept Designer (MCD) is a set of tools used for giving basic physics to a model, constrains, movements and allow the user to simulate real life situations of their designs. This simulation aspect is extremely important for Industry 4.0 and it will be used to prove the veracity of the DT.

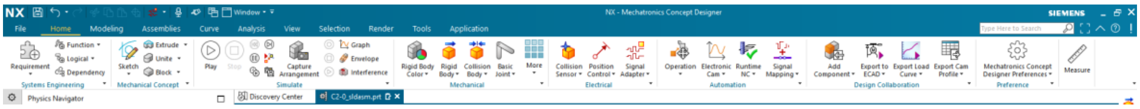


Figure 6. MCD toolbar

3.1.1. Static model

First thing needed is a 3D model of the station. To complete this task, NX provides the user with sketching tools, to make two dimensional bodies, modelling tools, to give those sketches volume and assembly tools, to merge different models to create the final 3D file.

The main model for this project was provided by SMC, the modelling of every part can be a tedious task and this way a lot of time was saved. Nevertheless, there were some details that did not correlate with the real machine. The distances between components were not the real ones, to solve this problem measures had to be taken and parts relocated so when simulated they would not collide or misalign each other. On the other hand, some modelled components do not resemble the real ones, for example the three-point claw of the real machine is modelled as a two-point one.

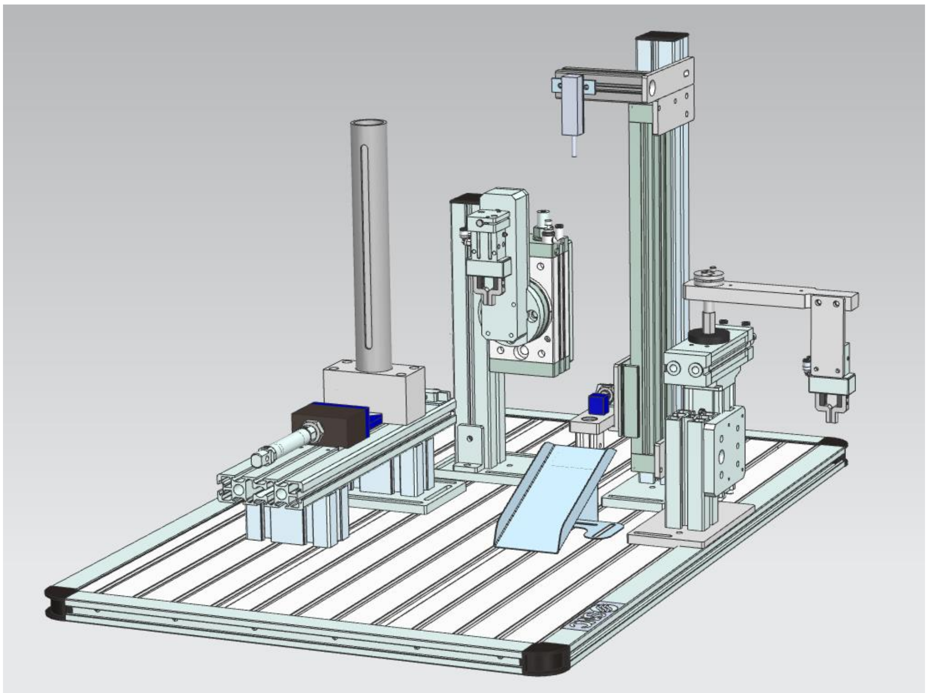
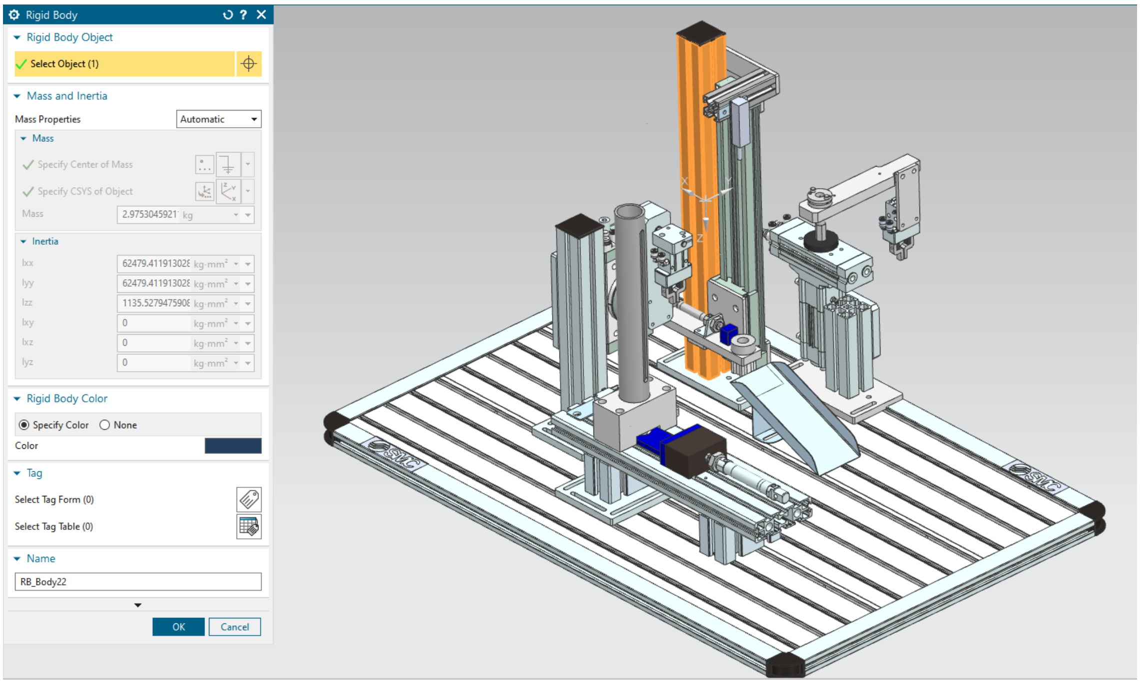


Figure 7. Static 3D model

When this is step is completed, we are left with a 3D model as close to reality as possible. But now it needs to be given physics and movements.

3.1.2. Rigid body

The next steps take place inside the Mechatronics Concept Designer. The rigid body tool provides components with weight and inertia calculated regarding their dimensions. For this project, as seen on figure 8, these characteristics are left on automatic because of lack of more specific information. If given more details, they could be changed manually to have better resemblance to the real model. For this case, by changing the material to aluminum, whit its usual density, it is enough.



Rigid Body

Rigid Body Object

✓ Select Object (1)

Mass and Inertia

Mass Properties: Automatic

Mass

✓ Specify Center of Mass

✓ Specify CSYS of Object

Mass: 2.9753045921 kg

Inertia

Ixx	62479.41191302	kg·mm²
Iyy	62479.41191302	kg·mm²
Izz	1135.527947590	kg·mm²
Ixy	0	kg·mm²
Ixz	0	kg·mm²
Iyz	0	kg·mm²

Rigid Body Color

● Specify Color ○ None

Color

Tag

Select Tag Form (0)

Select Tag Table (0)

Name

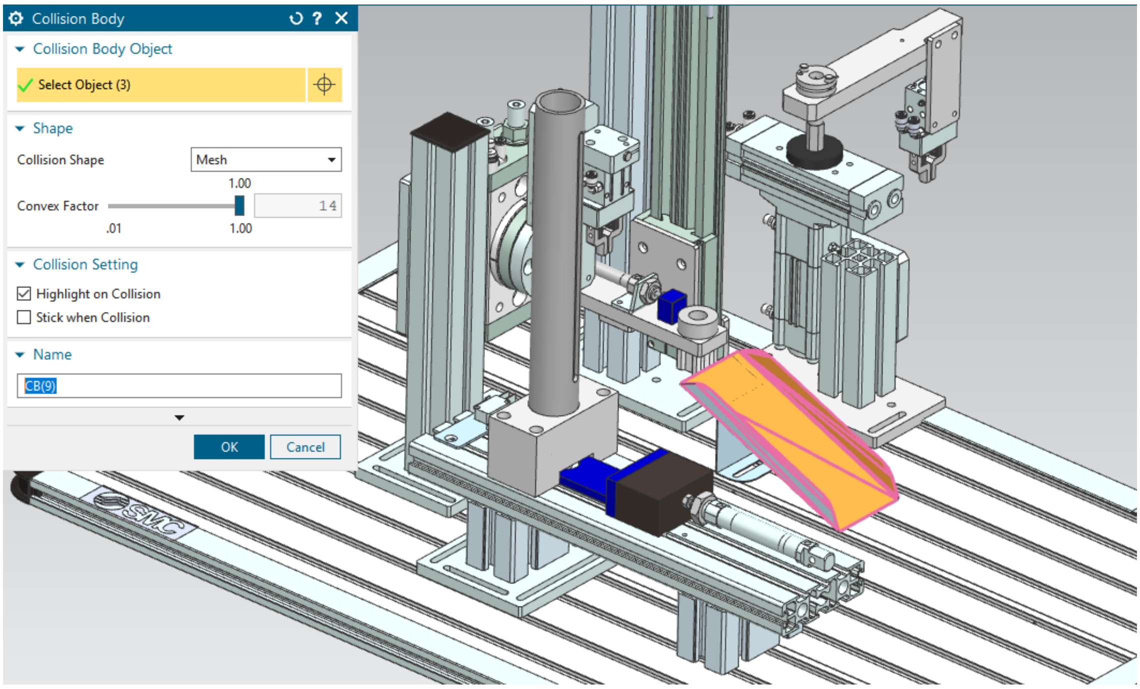
RB_Body22

Figure 8. Rigid body tool

Now basic physics as gravity affect the model, but if a simulation was to be started, the whole model would fall to infinity as there is nothing stopping it.

3.1.3. Collision body

Before using this tool, if components where to be simulated they would behave under the laws of physics independently. This means the software cannot identify that components can collide between each other. This tool gives any surface a collision feature, so if two components with this feature were to move into each other they would not go through each. By using the mesh as the collision shape, the surface will adapt itself to the surface evenly.



Select Object (3)

Collision Shape: Mesh

Convex Factor: 1.00 (slider .01 – 1.00), 14

☑ Highlight on Collision

☐ Stick when Collision

Name: CB(9)

Figure 9. Collision body tool

This is most notably on the bearing behavior, now it cannot fall through the table, it can be pushed around, and it has friction so the claw can actually grab it. Friction factor, static and dynamic, can be changed depending on the material used.

3.1.4. Constrains

By constraining the components, some of their degrees of freedom are taken away so they behave as intended in reality. This tool, called basic joint, provides the user with many different constrains, the ones used for this DT are explained below.

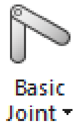
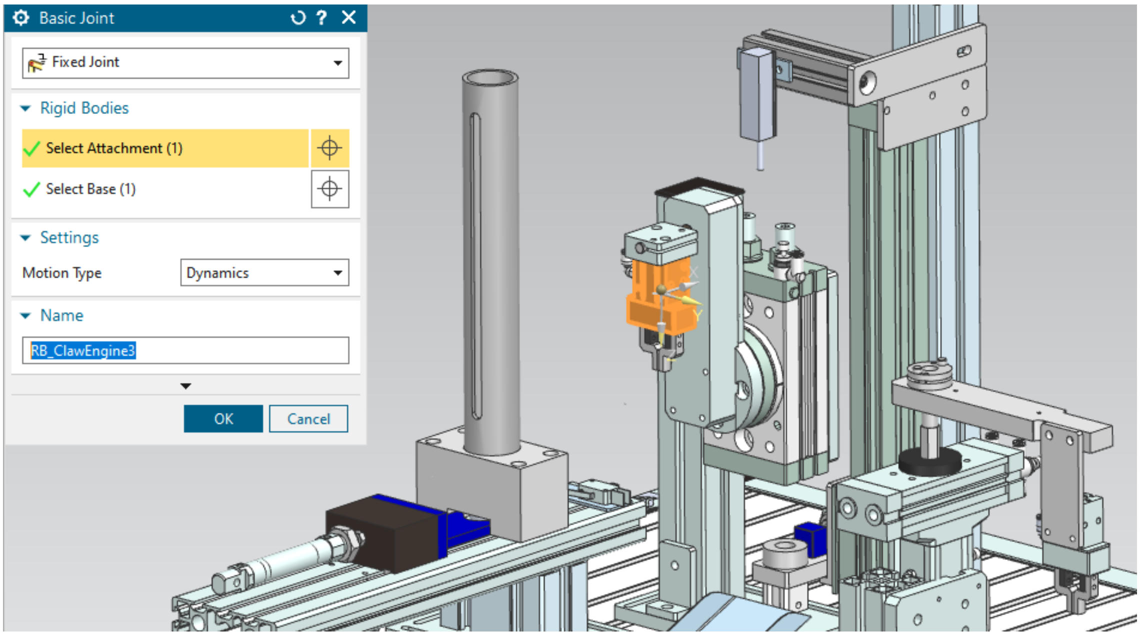


Figure 10. Basic joint icon

3.1.3.1. Fixed joint

Constrains all degrees of freedom of the part defined as the attachment relative to a base. Figure 11 shows how the attachment is highlighted in orange and it is being fixed to a base, which is the body directly on top of it. An attachment can also be fixed directly to the coordinate system if there is no other option or to make a starting point of constrains, which is used for fixing the table where everything is attached. This tool is used in every attachment that does not move in the real machine as well as for different pieces that are supposed to be welded together, the orange body below is fixed relative to the body on top but that body will be able to rotate and so it will the attachment too.



Basic Joint

Fixed Joint

Rigid Bodies

✓ Select Attachment (1)

✓ Select Base (1)

Settings

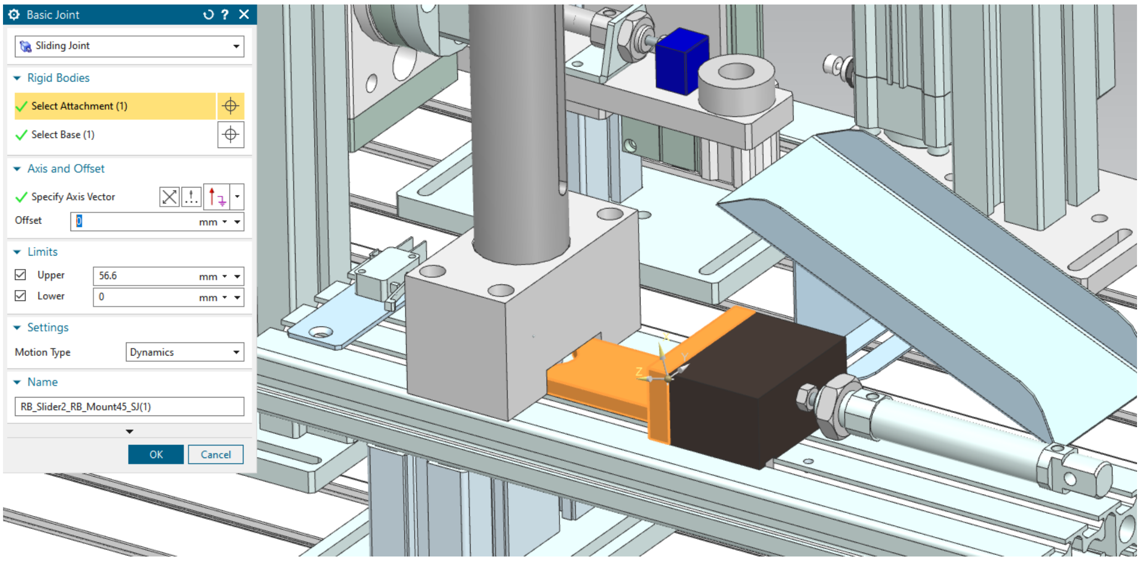
Motion Type: Dynamics

Name: RB_ClawEngine3

Figure 11. Fixed joint tool

3.1.3.2. Sliding joint

These ones are used for all the pneumatics piston previously mentioned as well as for the lift and transfer to the body on its vertical movement. In figure 12 the sliding joint is being created on the piece that pushes the bearings out of the storage cylinder (highlighted in orange), and the table below is being used as a base where it will slide. Then a direction has to be chosen, when selected a very intuitive cross of arrows referring to the three main axis of movement appears on the screen for the user to choose from. Moreover, limits of the movement can be set, in this case as the slider is moving outwards so there is an upper limit at 56.6 mm which matches with the final position of the bearing for its transfer.



Sliding Joint

✓ Select Attachment (1)

✓ Select Base (1)

✓ Specify Axis Vector

Offset 0 mm

☑ Upper 56.6 mm

☑ Lower 0 mm

Motion Type Dynamics

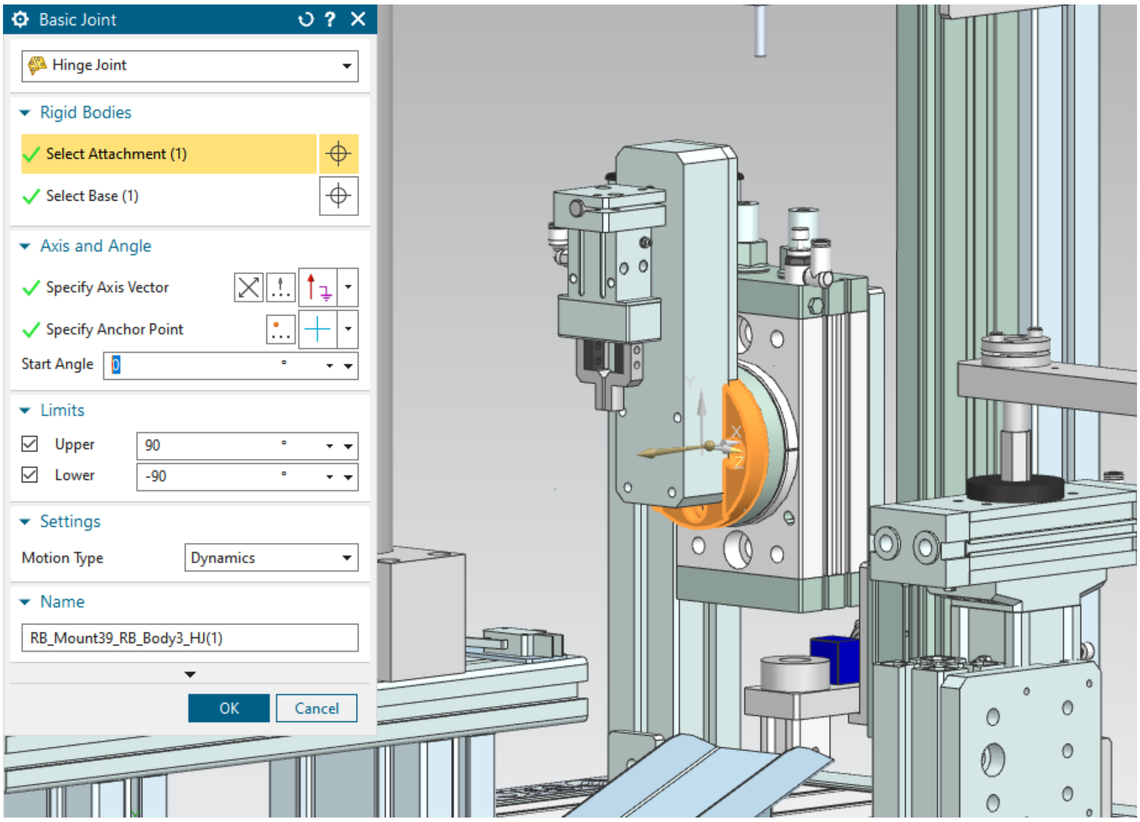
RB_Slider2_RB_Mount45_SJ(1)

Figure 12. Sliding joint tool

The black piece has to be fixed to the sliding part, and the actual piston rod has to be fixed referred to the black piece. This way all pieces behave as they are welded together, and they can all slide as the real contraption.

3.1.3.3. Hinge joint

This is the one used for the rotating arms of both transfer movements. It constrains the attachments as if they were a hinge from a door, in can turn around but fixed in one point and axis. There are more characteristics to be specified than on previous constrains. In figure 13 the rotating attachment is the orange disk and the base is the big body behind it. To specify the axis vector the user can once again choose it from the cross of arrows that will pop up when chosen, in this case the arrow will act as the axis of rotation and not the direction of movement. Finally, there has to be an anchor point which the attachment will rotate around, in the example below it can simply be the center of the circumference of the disk.



Basic Joint

Hinge Joint

Rigid Bodies

✓ Select Attachment (1)

✓ Select Base (1)

Axis and Angle

✓ Specify Axis Vector

✓ Specify Anchor Point

Start Angle 0 °

Limits

☑ Upper 90 °

☑ Lower -90 °

Settings

Motion Type Dynamics

Name

RB_Mount39_RB_Body3_HJ(1)

OK Cancel

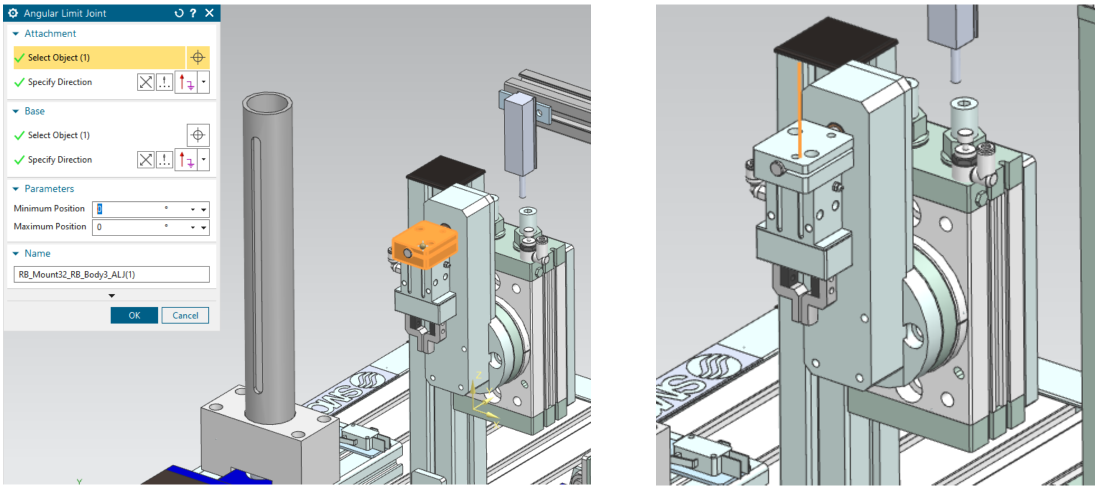
Figure 13. Sliding joint tool

Once again, the vertical arm has to be fixed to the rotating disk in order to be able to do the same movement. In the example there has also been set limits of movement of 90 degrees in both directions, as those are the ones intended to transfer the bearing properly.

3.1.3.4. Angular limit joint

This is the last joint used for this DT and it is specific for one not so usual movement regarding the transfer arm. As mentioned previously, the real arm has belt inside of it that helps counterrotate the claw body when the arm moves. The 3D model is not provided with this belt, so to create it, a hinge and angular limit joints are needed.

The hinge joint has been created on the orange attachment from figure 14, which is meant to rotate around the axis going through it. On the same image the menu from the angular limit joint can be seen, the attachment is once again the orange piece, and the base is the big body behind it. For this joint there are two directions to be specified, on for each part involved in the process. As shown on figure 14 on the right, both directions of the bodies have to be looking the same way. By choosing these directions, the user can now set the parameters of the menu to 0, this means there cannot be any angle between those two parts and when the arm rotates the claw will stay upright as shown on figure 15.



Angular Limit Joint

Attachment

✓ Select Object (1)

✓ Specify Direction

Base

✓ Select Object (1)

✓ Specify Direction

Parameters

Minimum Position 0 °

Maximum Position 0 °

Name

RB_Mount32_RB_Body3_ALJ(1)

Figure 14. Angular limit joint

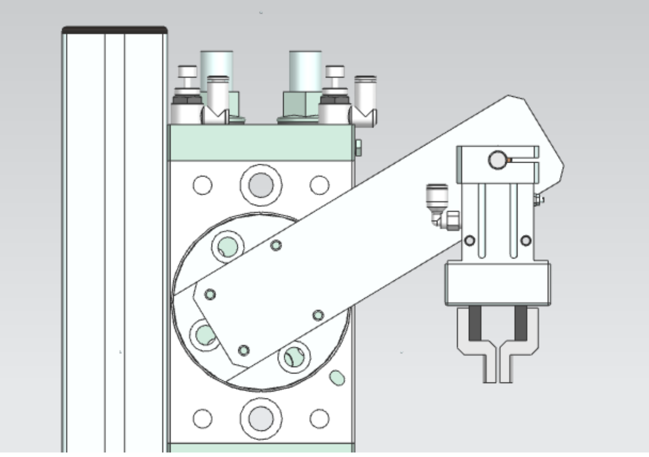


Figure 15. Rotation arm with angular limit joint implemented

3.1.5. Movements

Now that the model is completely constrained, it is time to assign movements regarding the degrees of freedom left for the different parts. Figure 16 shows all movements of the station, the arrows with parallel lines to their sides are linear movements, while the ones without them define rotational movements.

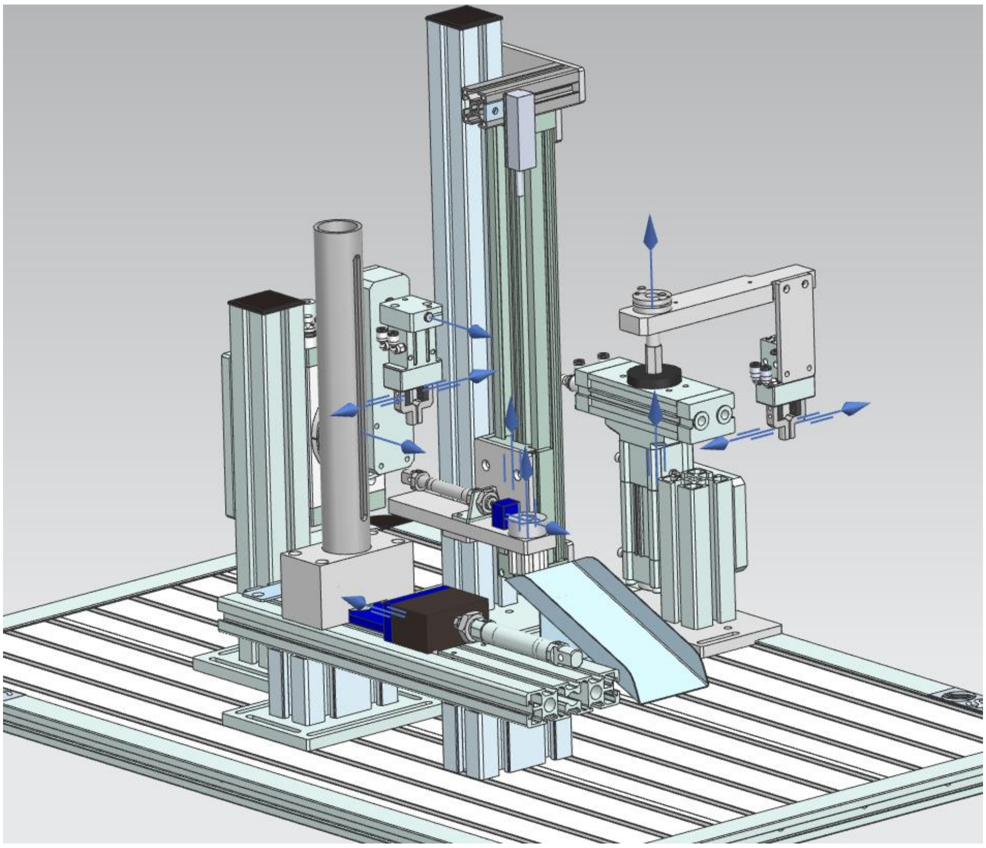
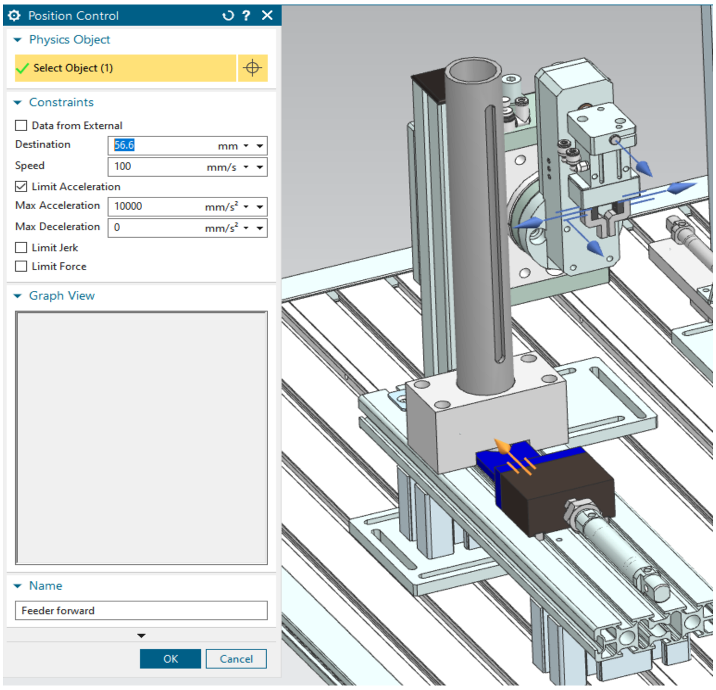


Figure 16. Movements of the 3D model

The software detects the constrains that leave some degrees of freedom open for the user to select them, if wanted, to manipulate their state. To control these movements there are two different options depending on what parameter you want to control:

- **Position control:** Figure 17 shows the control of the sliding joint of the feeder, with this tool you can choose the speed the feeder will move and its final destination. When the PLC is programmed, if the input of this pneumatic piston is triggered the piston will move outwards until it reaches its maximum or distance or something tells it to stop.

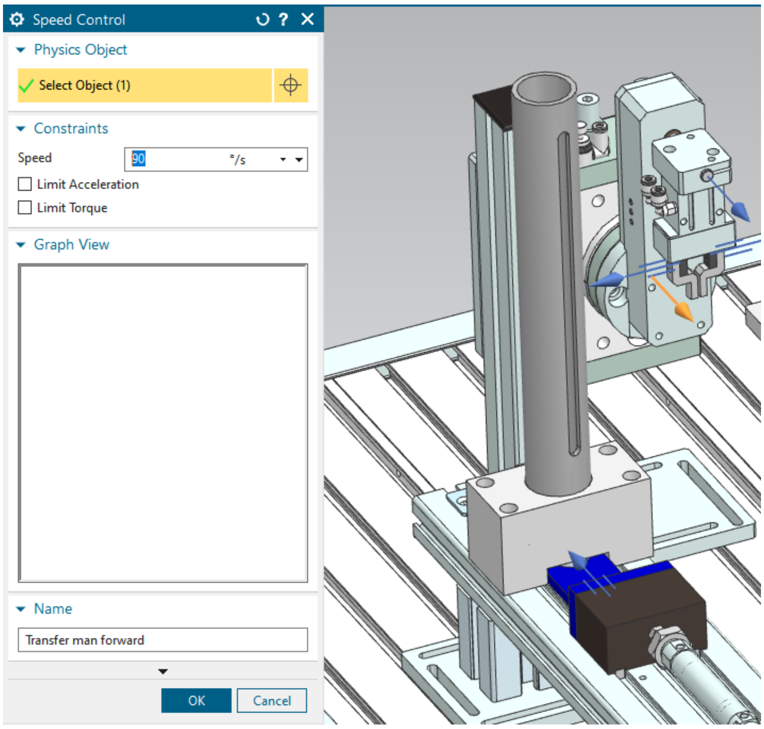


□ Data from External
Destination 56.6 mm
Speed 100 mm/s
☑ Limit Acceleration
Max Acceleration 10000 mm/s²
Max Deceleration 0 mm/s²
□ Limit Jerk
□ Limit Force

Name: Feeder forward

Figure 17. Position control tool

- **Speed control:** Now figure 18 is showing how to control the rotation of the moving arm. In this case only the speed is being set, so if the PLC was to trigger the input related to the movement, in this case, the arm would rotate at speed of 90º/s counterclockwise until some output or sensor tells it to stop.



Speed 90 °/s
□ Limit Acceleration
□ Limit Torque

Name: Transfer man forward

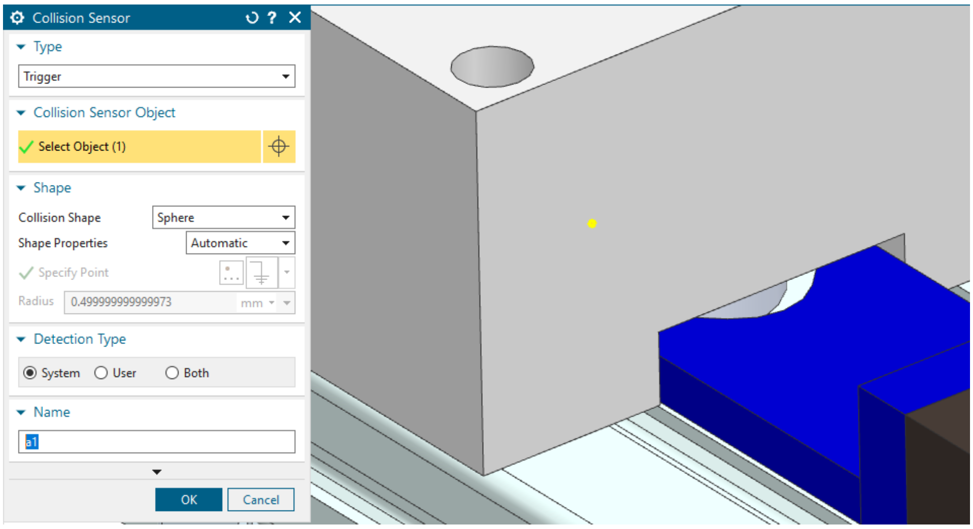
Figure 18. Speed control tool

All real pistons as well as the claws work with air pressure, meaning by just using one input they can define two positions. For example, when the input of a piston is triggered one of its chambers fills up with air while the other evacuates the air inside of it, if the input is switched off the process goes backwards, and it resets it to its initial position. To fix this problem in the DT there has to be two movements designed for all pneumatic actuators, one going in the desired direction and the other backwards so we can simulate the same effect as in reality.

3.1.6. Sensors

The movements or outputs of the PLC program are completed, but the inputs are not yet. The inputs for any program are triggered by sensors, to create these sensors in Siemens NX we can use the tool of collision sensor.

First step is to create a body of any shape in 3D, in this case they are spheres of a really small radius. Then they can be selected making use of the Collision Sensor tool. Figure 19 show the parameters that can be configured for these sensors, on the white rectangle there are options for how the sensor should behave, by selecting trigger the output will be active as long as there is something colliding with it.



Trigger

✓ Select Object (1)

Collision Shape: Sphere

Shape Properties: Automatic

✓ Specify Point

Radius: 0.499999999999973 mm

● System ○ User ○ Both

a1

Figure 19. Collision sensor tool

Now the spheres can be placed in the coordinates required for them to be triggered when an actuator reach a certain position. The example shown on the left of figure 20 collides with the slider of the feeder when it reaches its fully extended position, meanwhile the sensor in the right gets triggered when a bearing is pushed out by the glider. This is the simplest solution to recreate basic sensors in MCD, there are many more complicated options and with plenty of parameters to recreate any real sensor, but they were not needed for this DT.

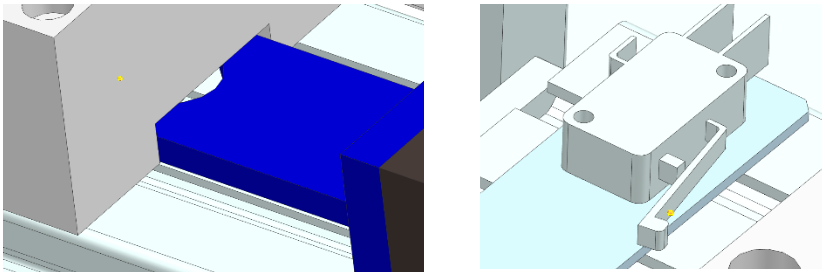
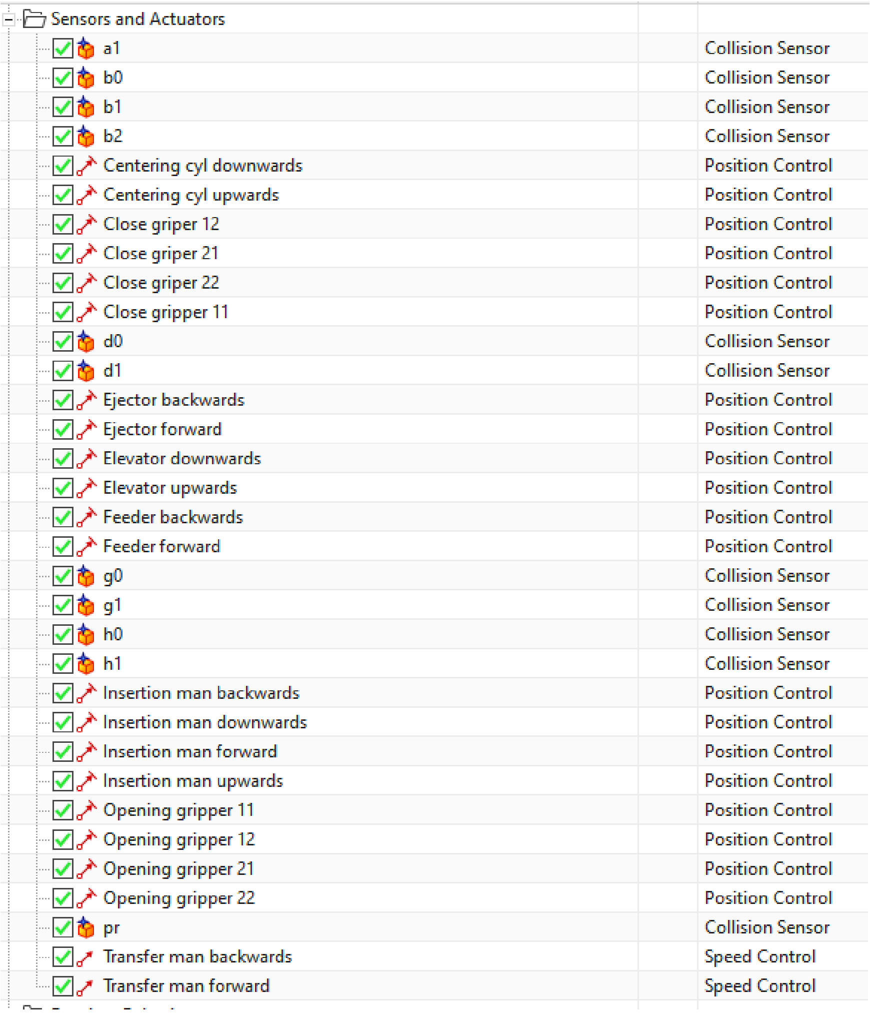


Figure 20. Collision sensor examples

Figure 21 shows an index of all the movements (red arrow) and sensors (orange boxes) needed for this Digital Twin.



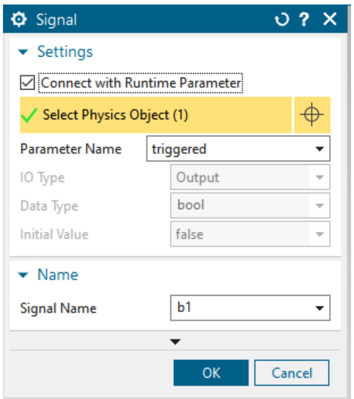
Sensors and Actuators	
☑ a1	Collision Sensor
☑ b0	Collision Sensor
☑ b1	Collision Sensor
☑ b2	Collision Sensor
☑ Centering cyl downwards	Position Control
☑ Centering cyl upwards	Position Control
☑ Close griper 12	Position Control
☑ Close griper 21	Position Control
☑ Close griper 22	Position Control
☑ Close gripper 11	Position Control
☑ d0	Collision Sensor
☑ d1	Collision Sensor
☑ Ejector backwards	Position Control
☑ Ejector forward	Position Control
☑ Elevator downwards	Position Control
☑ Elevator upwards	Position Control
☑ Feeder backwards	Position Control
☑ Feeder forward	Position Control
☑ g0	Collision Sensor
☑ g1	Collision Sensor
☑ h0	Collision Sensor
☑ h1	Collision Sensor
☑ Insertion man backwards	Position Control
☑ Insertion man downwards	Position Control
☑ Insertion man forward	Position Control
☑ Insertion man upwards	Position Control
☑ Opening gripper 11	Position Control
☑ Opening gripper 12	Position Control
☑ Opening gripper 21	Position Control
☑ Opening gripper 22	Position Control
☑ pr	Collision Sensor
☑ Transfer man backwards	Speed Control
☑ Transfer man forward	Speed Control

Figure 21. Table of al sensors and movements

3.1.7. Signaling

The final step to be able to connect the newly made Digital Twin to a PLC program is to give signals to every movement and sensor of the model. A PLC works by sending and receiving signals through its installed input and output ports, but for the PLC to detect a sensor as an input signal it has to be modelled as an output signal of the 3D model. On the other hand, the outputs of the PLC, which will dictate the movements of the DT, are connected to input signals in the NX file so they can actually move the actuators.

Signals for sensors are fairly easy to create, as figure 22 shows, when opening the signal tool, the user must tick the box to connect the signal to a running parameter. Then the physics object selected is each of the collision sensors, as they are already designed as a trigger, the rest of the parameters will autofill themselves.



☑ Connect with Runtime Parameter

✓ Select Physics Object (1)

Parameter Name: triggered

IO Type: Output

Data Type: bool

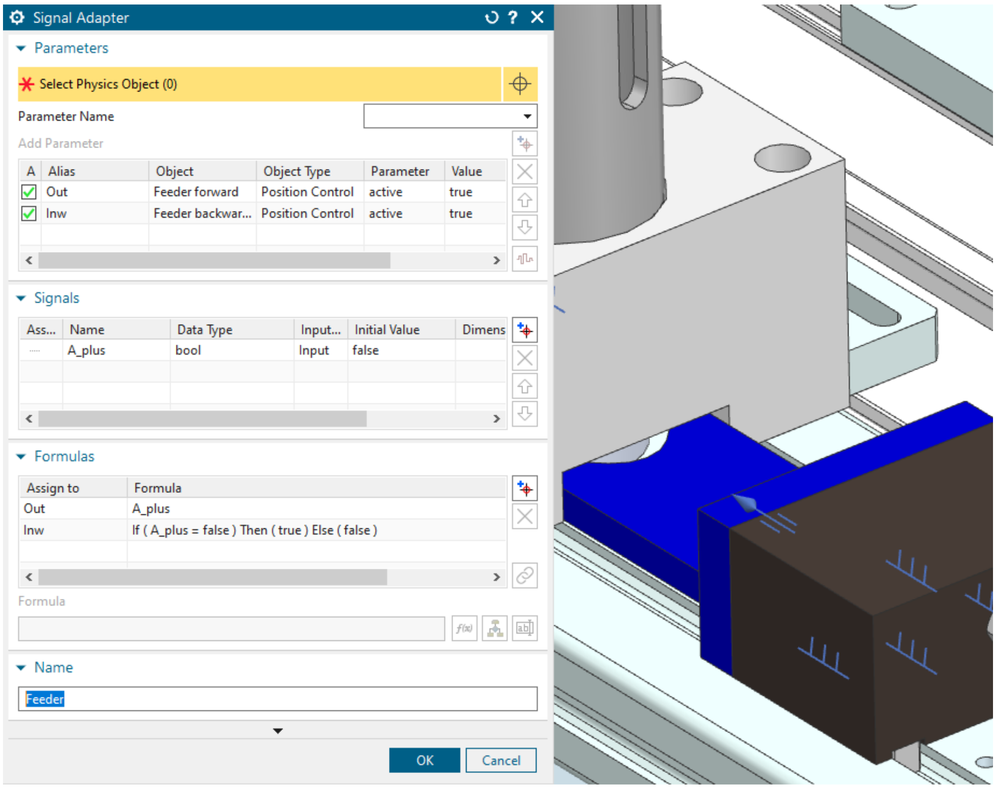
Initial Value: false

Signal Name: b1

Figure 22. Signal tool

On the other hand, the real model works mainly with pneumatics, this means by using just one output of the program there can be two movements involved. For example, if an output of a piston is triggered, the cylinder will move outwards, but when it is turned off, the piston is able to retract itself to its initial position. This cannot be achieved with just one position control in the 3D model but two. To fix this problem there has to be used a Signal Adapter to control both movements whit just one signal.

Figure 23 shows the design of the Signal Adapter for the movement of the feeder. On the top box, the user has to include as many movements as wanted to be modeled. The parameter that needs to be controlled is “active”, this way when the output is triggered it will change the state of the movement to active, as other as its speed have already been set. They have to be given an alias to simplify further steps, in this case “Out” is linked with the outwards movement of the piston and “Inw” with the one that retracts it. On the second box a signal has been created, it has to be modelled as an input as it is connected to movements as explained previously. The last box defines the formulas that dictate the behavior of the movements regarding the signal. In this case, if the signal “A_plus” is to be triggered the “Out” will be set as active and the piston will move outwards. But if the signal is not active, “Out” is off and “Inw” is on, thanks to the if else condition created, this means it will retract the piston once it is out and it will not let the rod slide out by itself while inactive as the inwards movement will be on. This condition is used in the feeder, both grippers, ejector and transfer to the body movements of the DT.



A	Alias	Object	Object Type	Parameter	Value
☑	Out	Feeder forward	Position Control	active	true
☑	Inw	Feeder backwar...	Position Control	active	true

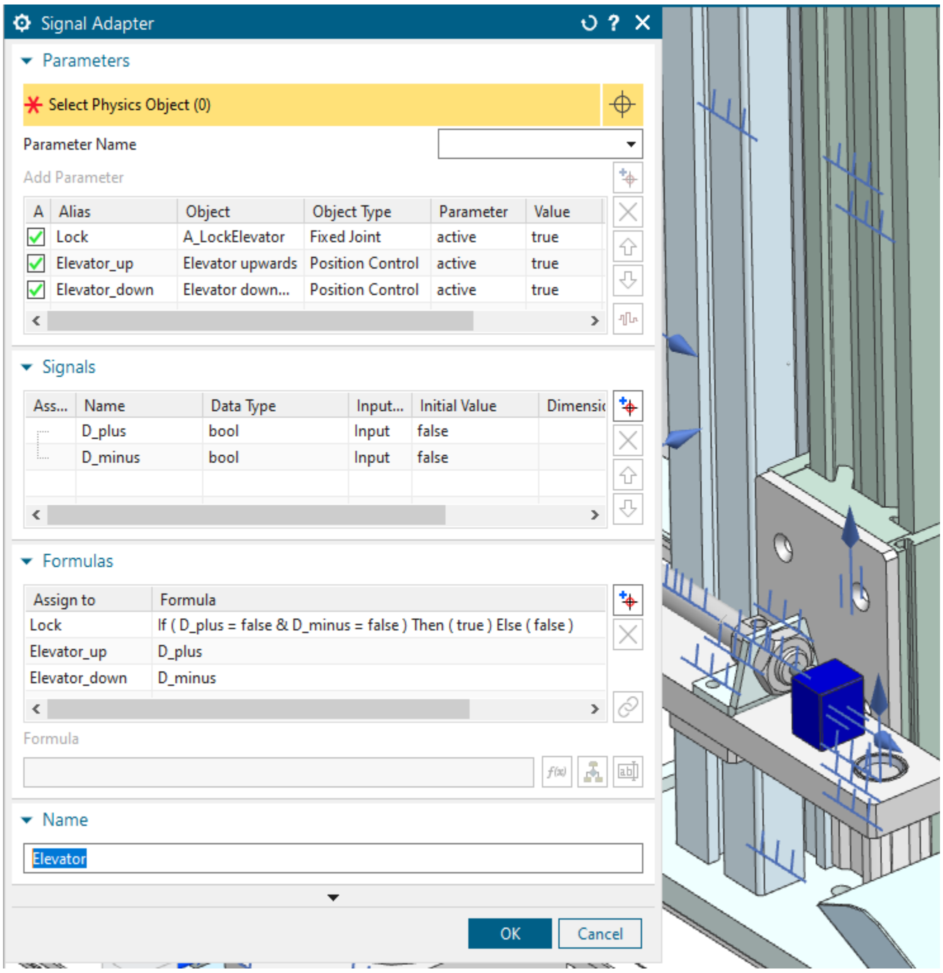
Ass...	Name	Data Type	Input...	Initial Value	Dimens
	A_plus	bool	Input	false	

Assign to	Formula
Out	A_plus
Inw	If (A_plus = false) Then (true) Else (false)

Name: Feeder

Figure 23. Signal adapter tool for the feeder

There is another particular case, for example in the elevator there is another parameter to have in mind. If this case there are two signals, one to move it up and one to move it down, but on figure 24 it can be seen a third parameter, a fixed joint for the elevator relative to the whole system. The if else formula on the picture means this joint becomes active when neither signal is on, this means the lift will stay put in place and not fall back to its initial position. In the fixed joint, the attachment is the platform of the elevator and it is attached to the reference axis. This condition is also used for the moving arm of the “transfer” action, so when it is not carrying any bearings it will stay put in the initial position as intended in the real machine.



A	Alias	Object	Object Type	Parameter	Value
☑	Lock	A_LockElevator	Fixed Joint	active	true
☑	Elevator_up	Elevator upwards	Position Control	active	true
☑	Elevator_down	Elevator down...	Position Control	active	true

Ass...	Name	Data Type	Input...	Initial Value	Dimensic
	D_plus	bool	Input	false	
	D_minus	bool	Input	false	

Assign to	Formula
Lock	If (D_plus = false & D_minus = false) Then (true) Else (false)
Elevator_up	D_plus
Elevator_down	D_minus

Name: Elevator

Figure 24. Signal adapter for the elevator

3.2. TIA Portal

To prove that the Digital Twin works, a PLC program needs to be loaded on the computer hosting it. The software used in UCLL for programming is TIA Portal, a powerful engine from Siemens designed for its use on the automated industry. This program embraces programming, simulation and real time diagnosis of fully built automated installations as well as DTs.

The shake of this project is to prove the digital model works the same way as the real one. To prove this point, a working program designed by UCLL students is needed. These programs have already been tested in the FMS-200 by coding them on a PLC_315-2. For this reason, some changes need to be done as for the DT only works with PLC_1500 as it will be explained on the next chapter of this paper. This can be done by right clicking on the folder which contains all the information of the program and the click on “Migrate to S7-1500” as shown in figure 25.

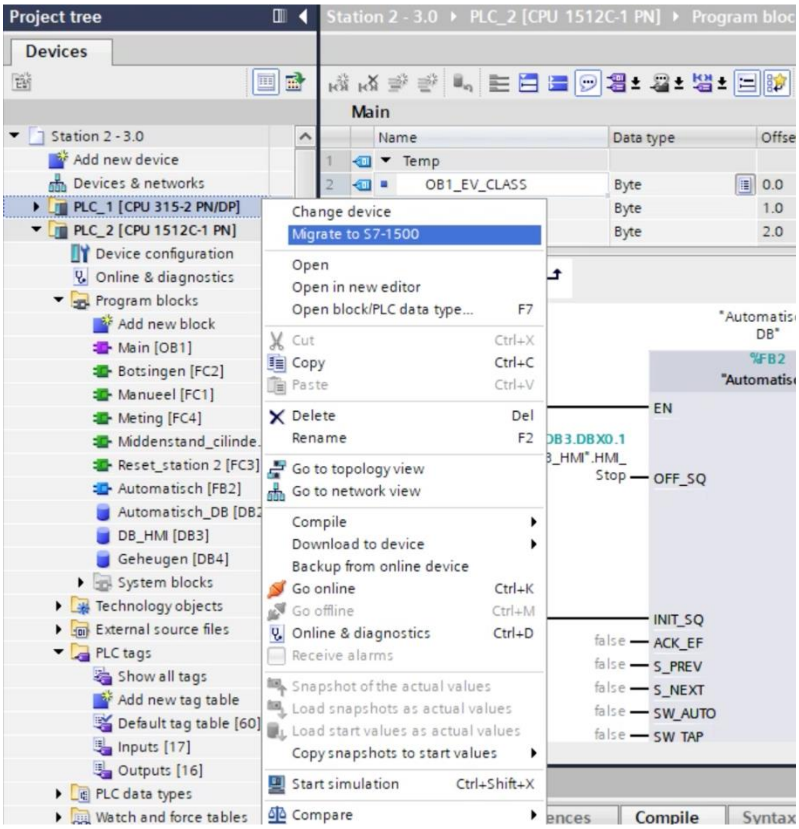


Figure 25. Migrate to S7-1500 option

When the software recompiles the program, there will be some errors with the migration. These problems are caused by the communication blocks designed, the different stations from the FMS_200 are meant to be connected so they can work as a whole. This communication means one station will initialize its program when the previous one has finished, and the contraption being created arrives to the next one through the conveyor for example. There is no quick fix for this problem, as PLCs from the 1500 series communicate differently from the 300 series. The best solution for this is to program the blocks directly in the new PLC architecture, as there is no way of implementing a PLC_300 on the DT. For the simulation of this project, the problematic code needs to be disconnected or deleted as it is not the focus of it.

The newly edited program can be loaded on the computer, but first a simple HMI is needed. An HMI is a user interface to connect the designer with the program with no need of interacting with the real PLC, this is done by simulating switches, buttons, and graphics. For this simulation, there only needs to be created the buttons that belong in the real machine which are “Start”, “Stop”, “Reset” and a switch for the

“Automatic/Manual” modes. The rest of buttons shown in figure 26 are used to control outputs independently, so the designer can move the pistons and other actuators without the need of running the full program.

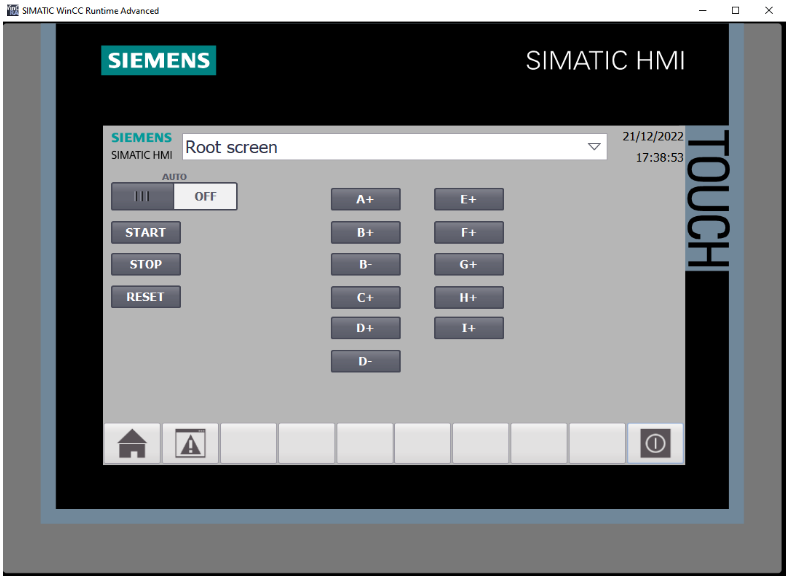


Figure 26. Simulated HMI screen

The main buttons from above are linked to different positions of a dedicated memory of the program. By pressing the buttons, the directions change their value between 0 and 1. These variables need to be implemented in the code blocks so the user can activate the automatic process by just using the HDMI. In figure 27 the HMI inputs can be seen placed parallel to those inputs of the real buttons so the program can be triggered any way.

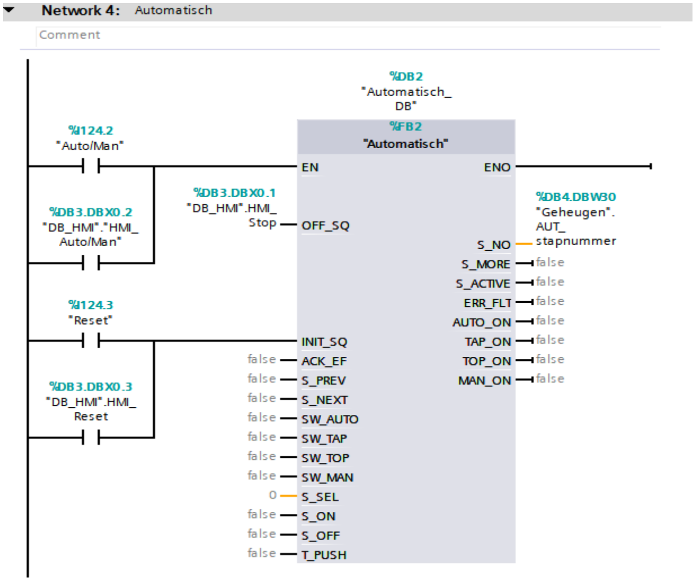
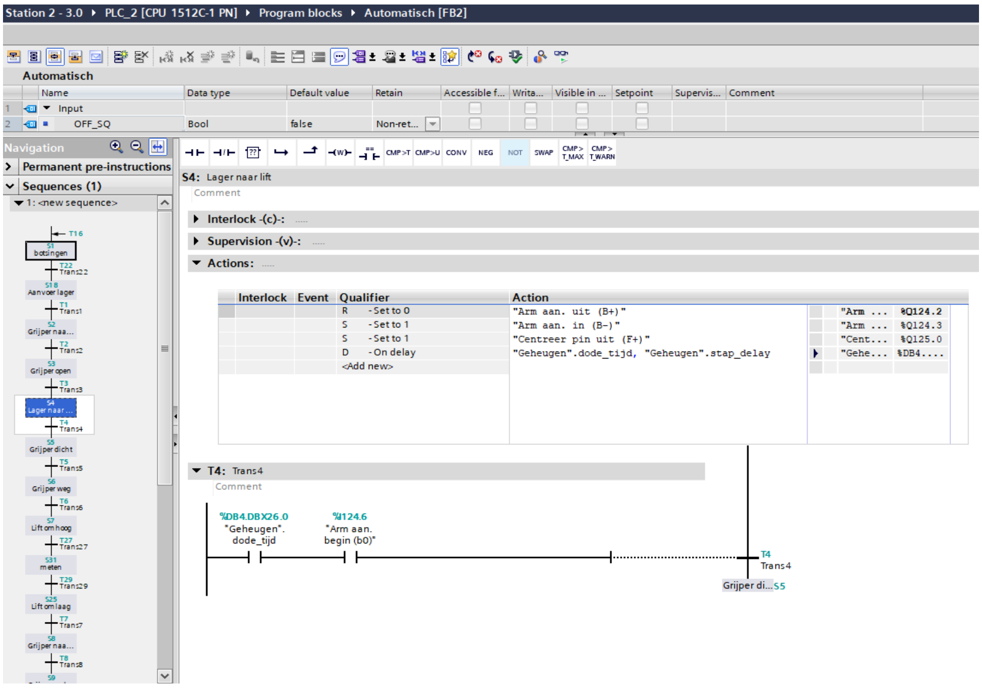


Figure 27. Modified inputs implemented in the program

To briefly explain the function of the program, “Automatisch_DB” block is programmed in a linear sequence of events. Figure 28 shows the architecture of one those blocks, the table on top shows the outputs (movements) that are activated or deactivated. The movements that may result in the activation of those outputs will then trigger the inputs (sensors) in the line shown below. In this case, when the inputs are active, the circuit will close, and it will activate the transition to the next block. This code language resembles am electrical circuit, where the circuit must be closed to be able to flow the electricity and proceed to the next step.



Station 2 - 3.0 ▸ PLC_2 [CPU 1512C-1 PN] ▸ Program blocks ▸ Automatisch [FB2]

Automatisch

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervis...	Comment
1	▼ Input				□	□	□	□		
2	OFF_SQ	Bool	false	Non-ret...	□	□	□	□		

S4: Lager naar lift

Comment

▸ Interlock -(c)-: ......

▸ Supervision -(v)-: ......

▼ Actions: ......

Interlock	Event	Qualifier	Action		
		R - Set to 0	"Arm aan. uit (B+)"	"Arm ...	%Q124.2
		S - Set to 1	"Arm aan. in (B-)"	"Arm ...	%Q124.3
		S - Set to 1	"Centreer pin uit (F+)"	"Cent...	%Q125.0
		D - On delay	"Geheugen".dode_tijd, "Geheugen".stap_delay	"Gehe...	%DB4....
		<Add new>			

▼ T4: Trans4

Comment

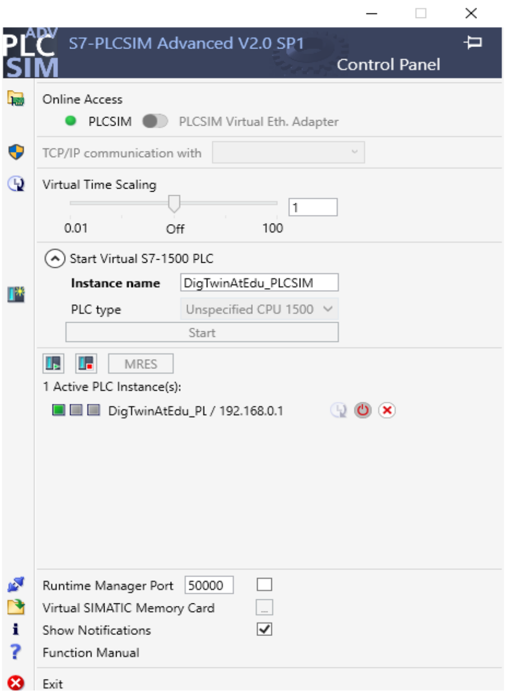
Figure 28. Automatic program architecture

3.3. PLCSIM Advanced

This is the last piece of software needed for the project. PLCSIM Advanced allows the user to simulate a PLC directly in their computer without the need of a physical unit. The downside of using this tool is it can only simulate units from the 1500 series, the 300 series is not supported. This is the reason why the TIA program must be migrated, to this day there is no way around this problem besides programming having in mind the different communication characteristics of this version.

To simulate a new PLC, the user has to choose a new instance name and PLC type, then it shall be started, and the new unit will show below as the one in figure 29.

“DigTwinAtEdu_PLCSIM” is the name of the unit where the modified TIA program for station 2 needs to be loaded. The program will stay loaded even if this software is closed or the computer shuts down, there is a folder called “Virtual SIMATIC Memory Card” where every program and PLC created will be stored as if it was any other usual folder of the PC.



Virtual Time Scaling: 1

Start Virtual S7-1500 PLC

Instance name: DigTwinAtEdu_PLCSIM

PLC type: Unspecified CPU 1500

1 Active PLC Instance(s):

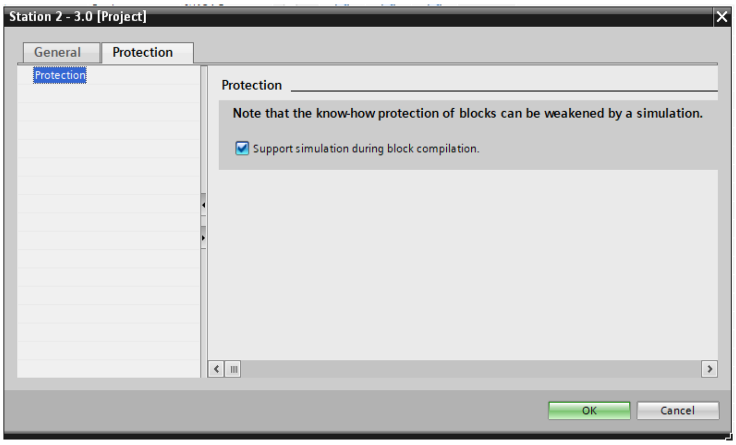
DigTwinAtEdu_PL / 192.168.0.1

Runtime Manager Port: 50000 □

Show Notifications ☑

Figure 29. PLCSIM Advanced interface

Now if the TIA program is to be compiled there will only be a PLC to load it, the IP direction of the simulated PLC is 192.168.0.1. But if tried to load it, there will be an error as the code is not supported to be simulated with SIMATIC S7-PLCSIM. To fix this issue, on properties of the project on the protection tab, “support simulation during block compilation” needs to be activated.



Protection

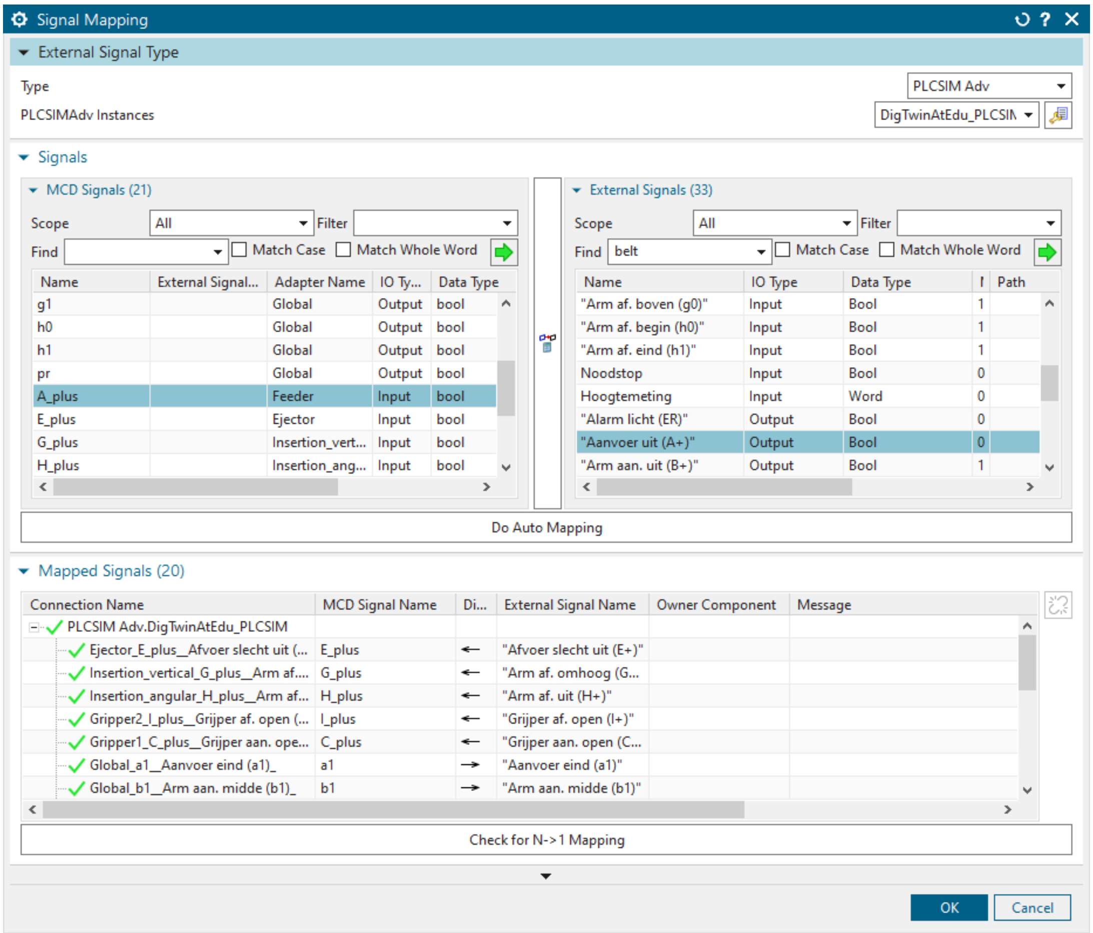
Note that the know-how protection of blocks can be weakened by a simulation.

☑ Support simulation during block compilation.

Figure 30. Project protection options

3.4. Interconnection and mapping

The program is loaded on the virtual unit and running, now the inputs and outputs must be connected with the signals of the 3D model, this is called mapping. On the MCD software, the Signal Mapping tool has to be opened. The dialog box of figure 31 shows the PLCSIM Advanced instance, if none was to be selected, the user should click on the box on the right with the yellow wrench. This action would open a menu as the one shown in figure 32, on top of it appears all the instances or PLCs that are being simulated. By clicking on "DigTwinAtEdu_PLCSIM" all the tags of the program loaded into it will show on the bottom. The user must select "IOM" as the area and tick off the "HMI Visible Only" box, this way when updated the table down below will show all the inputs and outputs of the program. The tags can be selected manually, or they can all be selected to load them in the "Signal Mapping". When selected they can be loaded by clicking the green arrow on the right.



Type: PLCSIM Adv
PLCSIMAdv Instances: DigTwinAtEdu_PLCSIN

MCD Signals (21)

Scope: All — Filter:
Find: □ Match Case □ Match Whole Word

Name	External Signal...	Adapter Name	IO Ty...	Data Type
g1		Global	Output	bool
h0		Global	Output	bool
h1		Global	Output	bool
pr		Global	Output	bool
A_plus		Feeder	Input	bool
E_plus		Ejector	Input	bool
G_plus		Insertion_vert...	Input	bool
H_plus		Insertion_ang...	Input	bool

External Signals (33)

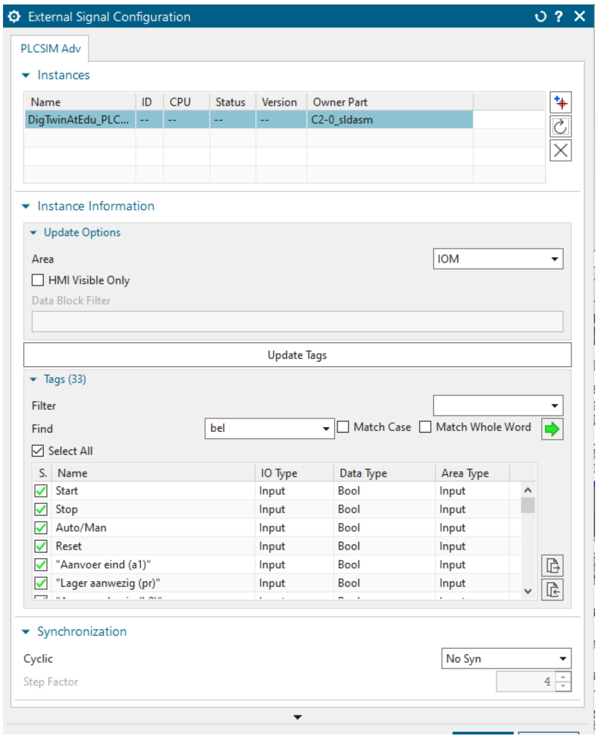
Scope: All — Filter:
Find: belt □ Match Case □ Match Whole Word

Name	IO Type	Data Type	I	Path
"Arm af. boven (g0)"	Input	Bool	1	
"Arm af. begin (h0)"	Input	Bool	1	
"Arm af. eind (h1)"	Input	Bool	1	
Noodstop	Input	Bool	0	
Hoogtemeting	Input	Word	0	
"Alarm licht (ER)"	Output	Bool	0	
"Aanvoer uit (A+)"	Output	Bool	0	
"Arm aan. uit (B+)"	Output	Bool	1	

Mapped Signals (20)

Connection Name	MCD Signal Name	Di...	External Signal Name	Owner Component	Message
✓ PLCSIM Adv.DigTwinAtEdu_PLCSIM					
✓ Ejector_E_plus__Afvoer slecht uit (...	E_plus	←	"Afvoer slecht uit (E+)"		
✓ Insertion_vertical_G_plus__Arm af....	G_plus	←	"Arm af. omhoog (G...		
✓ Insertion_angular_H_plus__Arm af...	H_plus	←	"Arm af. uit (H+)"		
✓ Gripper2_I_plus__Grijper af. open (...	I_plus	←	"Grijper af. open (I+)"		
✓ Gripper1_C_plus__Grijper aan. ope...	C_plus	←	"Grijper aan. open (C...		
✓ Global_a1__Aanvoer eind (a1)_	a1	→	"Aanvoer eind (a1)"		
✓ Global_b1__Arm aan. midde (b1)_	b1	→	"Arm aan. midde (b1)"		

Figure 31. Signal Mapping tool



Name	ID	CPU	Status	Version	Owner Part
DigTwinAtEdu_PLC...	--	--	--	--	C2-0_sldasm

Area: IOM

☐ HMI Visible Only

Data Block Filter

Filter

Find: bel ☐ Match Case ☐ Match Whole Word

☑ Select All

S.	Name	IO Type	Data Type	Area Type
☑	Start	Input	Bool	Input
☑	Stop	Input	Bool	Input
☑	Auto/Man	Input	Bool	Input
☑	Reset	Input	Bool	Input
☑	"Aanvoer eind (a1)"	Input	Bool	Input
☑	"Lager aanwezig (pr)"	Input	Bool	Input

Cyclic: No Syn

Step Factor: 4

Figure 32. Inputs and outputs tags of the PLC

Going back to figure 31, the top left table shows all signals that have been created for the 3D model and the top right table shows all inputs and outputs of the TIA program. They way to link them is by clicking one signal and its corresponding PLC tag, then by clicking the button between both tables they will be linked, and it will appear on the bottom table. This is why the signals have been given names as similar as possible to those of the TIA tags, this way it is easier to find the names to be linked. As discussed previously, the signals from the sensors are outputs connected to inputs of the system, for this reason the arrow of their connections goes from the MCD signal to the PLC input, because the sensor is sending information to the PLC. On the other hand, outputs for the program will be instructions for the DT, that is why the arrow goes from the output of the PLC to the actual signal linked with the DT.

This mapping method is intuitive and easy to use, but it is not optimal. When a TIA program is loaded on a real PLC, the name of the tags is used to simplify the coding, but it is linked to the same input/output port even if the name is changed. Meanwhile when the DT is linked with the simulated device, the signals are linked by using the name of the tag and not their port direction. This means if the name of the tag was to be changed, the link with the DT would be cut as it will not be able to find the initial name of the tag it was connected to. This results in a major problem, if another program designed for the same exact station was to be loaded, if the tags names are different the mapping would need to be fully redesigned. This is the way MCD is programmed so there is no room for improvement right now, in future versions of the software Siemens could redesign it so signals could be linked directly to the ports installed in the real PLC.

3.5. Simulation

With every step completed, the Digital Twin of Station 2 of the automatic assembly system can be simulated. First the NX filed called “DigitalTwin_FMS202” must be opened, as this is the file that contains the 3D model. Secondly, PLCSIM Advanced must be booted up and the instance “DigTwinAtEdu_PLCSIM” has to be activated. Then, the TIA program must be opened so the virtual HMI can be simulated. By clicking the button play inside the toolbar of MCD the system becomes active, the parts highlighted in pink in figure 33 means those parts are colliding with something, either another part or a collision sensor. During the simulation, the collision of different components can be tracked by the featured mentioned, if the stop interacting with others it will stop being highlighted. Now by switching on the automatic switch and by clicking the start button in the HMI previously done, the Digital Twin can do the full process from the feed of a new bearing to its position on the contraption being created as it does the real station proving the veracity of this model.

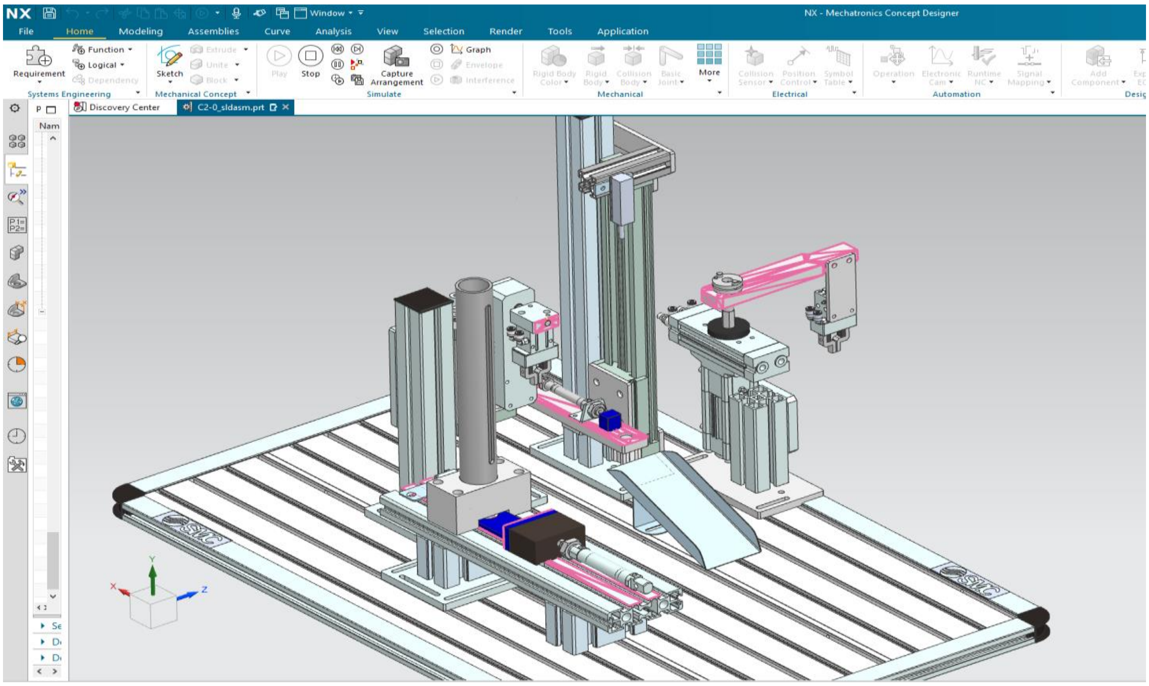
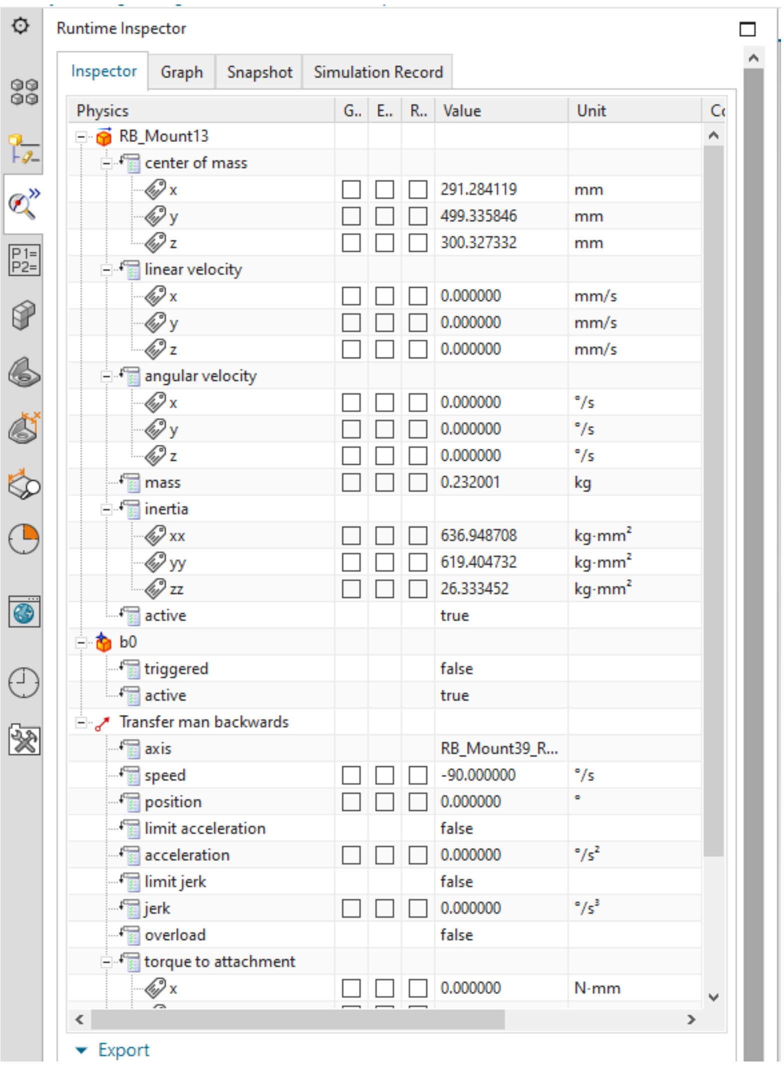


Figure 33. NX model during simulation

There is a really useful tool called “Inspector”, here a lot of characteristics of movements, bodies and so much more can be studied while the simulation is running. In the example below, with the help of the inspector it can be seen if the movement of the rotating arm is active, the speed that is creating, but it can also show if a collision sensor is triggered and the forces that are being applied to a certain part of the model. Moreover, various characteristics can be graphed out during the simulation, for example the user can see the force that is being applied in a moving arm through the process and see if it is optimal for its use or if it needs to be re-dimensioned.



Physics	G..	E..	R..	Value	Unit	Co
RB_Mount13						
center of mass						
x	□	□	□	291.284119	mm	
y	□	□	□	499.335846	mm	
z	□	□	□	300.327332	mm	
linear velocity						
x	□	□	□	0.000000	mm/s	
y	□	□	□	0.000000	mm/s	
z	□	□	□	0.000000	mm/s	
angular velocity						
x	□	□	□	0.000000	°/s	
y	□	□	□	0.000000	°/s	
z	□	□	□	0.000000	°/s	
mass	□	□	□	0.232001	kg	
inertia						
xx	□	□	□	636.948708	kg·mm²	
yy	□	□	□	619.404732	kg·mm²	
zz	□	□	□	26.333452	kg·mm²	
active				true		
b0						
triggered				false		
active				true		
Transfer man backwards						
axis				RB_Mount39_R...		
speed	□	□	□	-90.000000	°/s	
position	□	□	□	0.000000	°	
limit acceleration				false		
acceleration	□	□	□	0.000000	°/s²	
limit jerk				false		
jerk	□	□	□	0.000000	°/s³	
overload				false		
torque to attachment						
x	□	□	□	0.000000	N·mm	

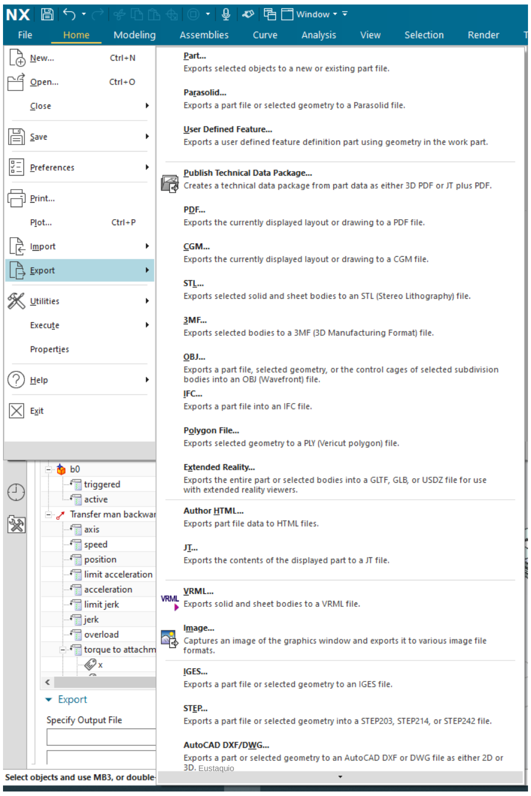
Figure 34. Inspector window during simulation

4. Union with Digital Twin of Station 1

The second part of this thesis consists in merging together the newly created Digital Twin with an existing one of the first station done by another student previously to prove the benefits of using Digital Twins. This process was really problematic, so the next pages are focused on displaying these troubles and their solutions.

4.1. MCD import problems

When trying to import the 3D model of a DT in the file where the other one is created there was a major issue, the static model could be imported but all MCD work was lost. A lot of time was spent trying to find a solution for this, this includes trying all the different export and import tools shown in the figure below. None of them worked and there was no information found on the internet about this issue neither by looking at tutorials or going through countless old and outdated forums. Only solution found was the least optimal of them all, the DT of the first needed to be redone from the beginning.



Part...
Exports selected objects to a new or existing part file.

Parasolid...
Exports a part file or selected geometry to a Parasolid file.

User Defined Feature...
Exports a user defined feature definition part using geometry in the work part.

Publish Technical Data Package...
Creates a technical data package from part data as either 3D PDF or JT plus PDF.

PDF...
Exports the currently displayed layout or drawing to a PDF file.

CGM...
Exports the currently displayed layout or drawing to a CGM file.

STL...
Exports selected solid and sheet bodies to an STL (Stereo Lithography) file.

3MF...
Exports selected bodies to a 3MF (3D Manufacturing Format) file.

OBJ...
Exports a part file, selected geometry, or the control cages of selected subdivision bodies into an OBJ (Wavefront) file.

IFC...
Exports a part file into an IFC file.

Polygon File...
Exports selected geometry to a PLY (Vericut polygon) file.

Extended Reality...
Exports the entire part or selected bodies into a GLTF, GLB, or USDZ file for use with extended reality viewers.

Author HTML...
Exports part file data to HTML files.

JT...
Exports the contents of the displayed part to a JT file.

VRML...
Exports solid and sheet bodies to a VRML file.

Image...
Captures an image of the graphics window and exports it to various image file formats.

IGES...
Exports a part file or selected geometry to an IGES file.

STEP...
Exports a part file or selected geometry into a STEP203, STEP214, or STEP242 file.

AutoCAD DXF/DWG...
Exports a part or selected geometry to an AutoCAD DXF or DWG file as either 2D or 3D. Eustaquio

Figure 35. NX export options

There has to be another way of solving this issue, but lack of time and the need of proving that two DTs can be linked together resulted in choosing this method.

4.2. Creation of MCD model of Station1

SMC provided UCLL with Solidworks files of all the stations, so once again these files can be loaded in NX to save time modelling the static structure. But all the MCD process needed to be done again, this means relocating parts, creating rigid bodies, collisions bodies, constrains, speed controls, sensors and signals for the whole process. The end result is shown below, the rectangular platform on front behaves as a belt that will transport the body supplied by station 1 to the station 2 so the bearing can be placed on top. To model that belt a simple sliding joint was used, this time there is no base and the direction of movement is the same as the X axis, which can be seen in red color at the bottom of the picture.

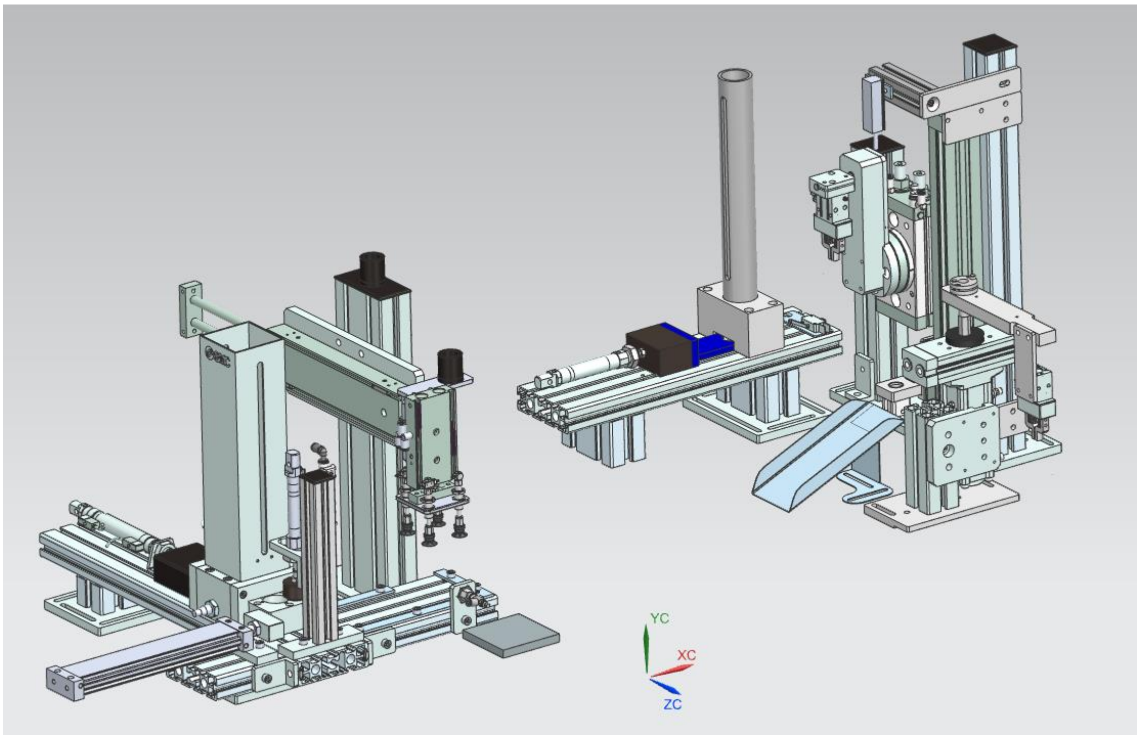


Figure 36. Both stations 3D models together

The movements of this model are simpler than those on the second one as they all are linear. The movements are shown by the blue arrows on figure 37. The movements work in a sequence to move the body from its supply all the way to the transport belt:

- **Feeder:** The piston with the dark block on the back slides out moving the body of the contraption out of the storage column until it is positioned under the vertical piston.
- **Centering:** The piston with the arrow pointing downwards on the left extends, and the cylindric back piece on its end slides inside the centric hole of the body of the contraption. This action ensures the body is positioned correctly for its future transfers. Once aligned the rod retracts until it stops doing contact with the body.
- **Transfer on the table:** Then the piston on the far right extends until the body reaches the end of the table and it is positioned underneath the suction piece.
- **Suction:** There is another vertical piston facing downwards on the far right, attached to it there is a suction device. First the piston slides out until the suction cups are on top of the body, then the suction is turned on so the body gets stuck to the device. Finally, the piston retracts to its original position allowing the transfer of the body to the belt.
- **Transfer to the belt:** The horizontal movement on top of the suction device allows the body to be moved outwards until it is positioned on top of the belt, then the suction piston will expand once again and when the body is placed on the belt the suction will be turned off.
- **Belt:** Once the body has been placed on top of it, the slider movement is activated until the platform reaches the second station where its automated process begins.

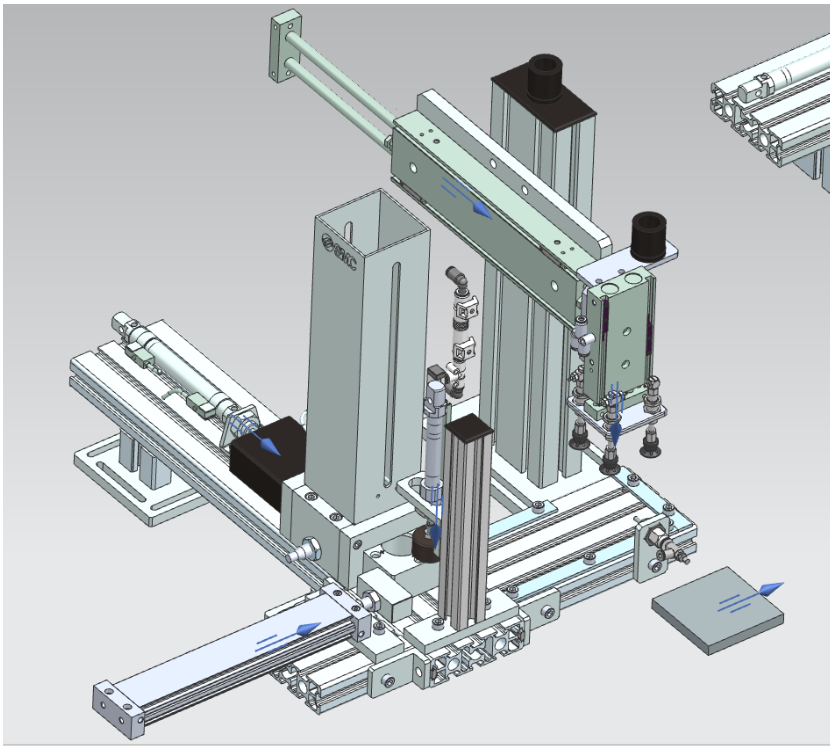
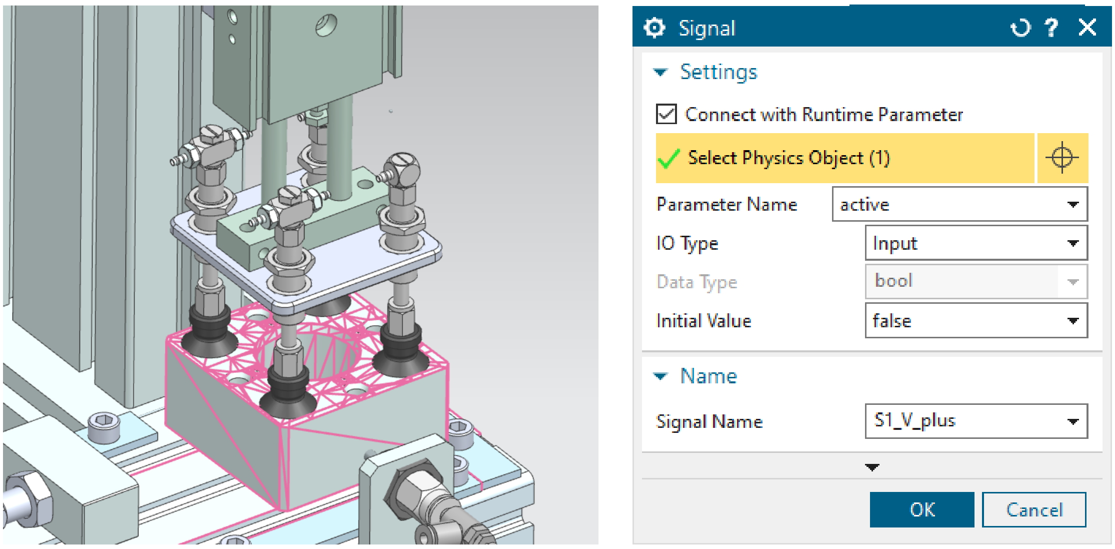


Figure 37. Movements of Station 1

As explained, most of signals are similar as those from Station 2 but there is one new feature compared with the last DT, the suction option. Once again MCD does not have a tool for this task so there needs to be found a way around this problem. In this case, the real model possesses an output dedicated to triggering a pneumatical vacuum that results in suction power on the device. Students use this output for their programs, so the DT can use its value to do a different action. In this case, the solution given is to link the signal to a fixed joint between the body and the suction device. The program triggers this output once the device is on top of the body and it switches it back to cero when the piece reaches the belt.

To create this, a signal has to be linked to the fixed joint. The signal is called "S1_V_plus", this name is given to be as similar as possible as the one from the PLC output which is "V+", the "S1" helps distinguish signals of the first DT from the second one as they all are together in the same index. As seen below, the initial value of the signal is false, this way while the body is moving around the station, its position is not set referring to the suction device, this would lead into a conflict of movements and position and would end up breaking the DT while being simulated. In figure 38 there can be seen the collision shape of the body, but on top there can also be seen the collision shape of the suction cups. The shape can barely be seen but it resembles a bidimensional square, this is a collision body option that englobes the surface of contact of all four cups but as it will contact with a box shaped body it can be used instead of a mesh to simplify the simulation.



☑ Connect with Runtime Parameter

✓ Select Physics Object (1)

Parameter Name: active

IO Type: Input

Data Type: bool

Initial Value: false

Signal Name: S1_V_plus

Figure 38. Creation of the suction signal

The other signals do not require of a detailed explanation as they follow the same steps as those described for the pneumatical pistons of Station 2. The long bar that is in charge of transferring the body to the belt behaves as the elevator on station 2, making use of two signals for its forward and backward movement and an if else condition that triggers a fixed joint to stop any sliding when no signal is active. The vertical pistons do not fall down while simulating because of the if else condition created, if the piston is not set to move out, an inwards movements is active impeding it falling, and the limit parameters of the sliding joint prevents the piston from sliding out of bounds in any direction.

Collision sensors behave the same way as in the first model but there are two that need special treatment. These sensors detect if the suction unit is up or down, but they have to be able to move with the unit as they have to read the movements in two different positions. For this matter a fixed joint need to be created using the spherical collision sensor as the attachment and the housing of the vertical piston that moves the suction unit will be set as the base. This way they can move forwards and backwards with the whole contraption but will not move vertically, this means they can read if the suctions cups are in their top or down position in every position.

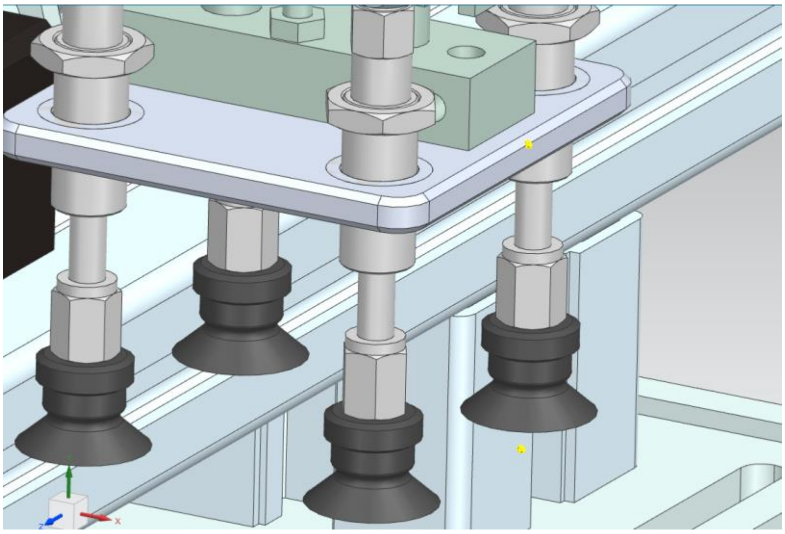


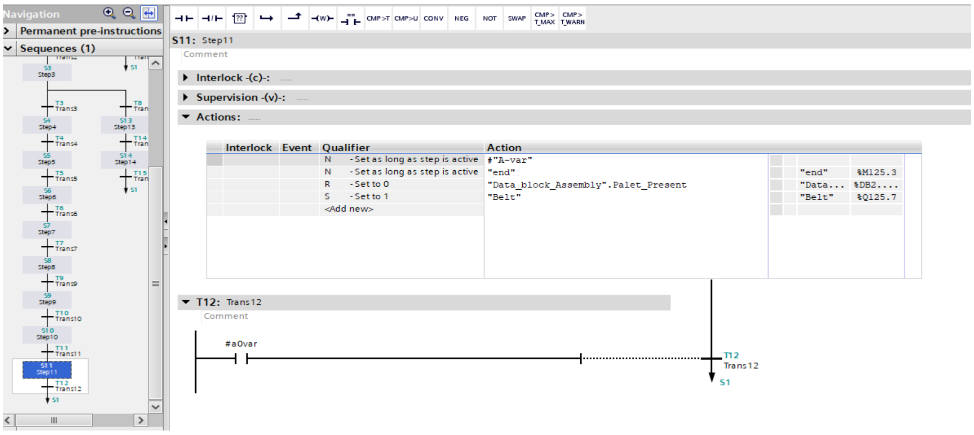
Figure 39. Sensors from Station 1 with special behavior

4.3. Creation of two linked TIA programs

As mentioned previously, students at UCLL program the FMS-200 making use of PLCs from the 300 series. The communication features of these devices do not match with the ones of the 1500 series, which are the only ones that can be simulated. The optimal fix for this problem would be to program directly in the 1500 architecture, but to prove that more than one PLC program can be loaded in the same MCD file there is a more rudimentary solution.

Now there are two stations, which means two separate TIA programs and the need to load them in two different PLC units. For the second station, the program modified for the first part of this thesis can be reused with some minor changes. On the other hand, a new program for the first station needed to be provided by another UCLL student. The programs cannot communicate between each other, but sensors from MCD can be triggered by a movement of the station one and then linked be linked with the PLC for station two. This means we can use an input modified by the belt to start the automatic process of the second DT. The next paragraphs describe the changes that must implemented in each program in order to make them work together.

The first station to be simulated is Station 1, for this reason there needs to be a way to start the automatic code of the machine. A simple HMI has been created following the same criteria as for the DT done previously, by linking HMI dedicated memory positions to buttons and switches and implementing these inputs at the beginning of the automatic program block. This program is responsible of moving the belt from station 1 to station 2, for this action a new output must be created. This output called “Belt” occupies an available port of the PLC module “%Q125.7”. This output will be set to active on the last block of the program, which is when the block sits on top of the belt ready for its transport.



S11: Step11

Interlock -(c)-:

Supervision -(v)-:

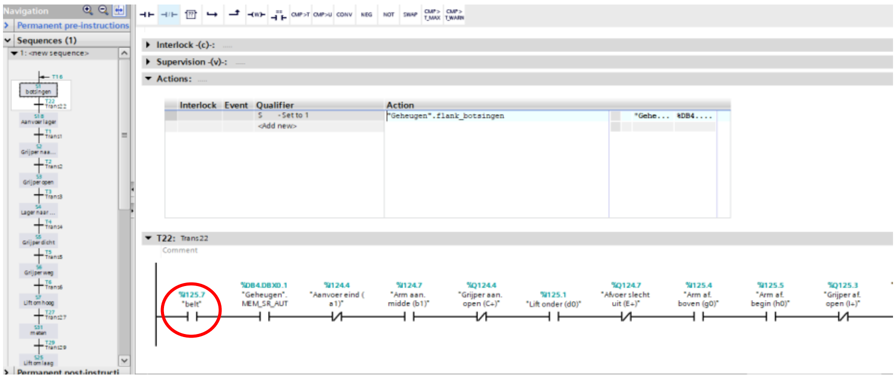
Actions:

Interlock	Event	Qualifier	Action		
		N - Set as long as step is active	#"A-var"		
		N - Set as long as step is active	"end"	"end"	%M125.3
		R - Set to 0	"Data_block_Assembly".Palet_Present	"Data...	%DB2....
		S - Set to 1	"Belt"	"Belt"	%Q125.7
		<Add new>			

T12: Trans12

Figure 40. Belt output implemented in the program blocks

When the belt arrives to the second station, the program in the second PLC responsible of the automatic movement of the system needs to be triggered. This is possible by implementing a new input in the program called “belt”. This input is connected to a collision sensor that gets triggered once the belt reaches its final position, these coordinates are measured to be aligned where the bearing will be dropped. The automatic blocks are set to always active by deleting the open input readers that controlled the program on the last simulation, this is accomplished by deleting everything related with the start buttons. Then, a new input condition is implemented in the first block of the program that reads the state of the newly created input “belt”. When the Belt arrives, the input is set to true closing the circuit and allowing the automatic process to begin as intended until it deposits the bearing on the center hole of the body.



Interlock -(c)-:

Supervision -(v)-:

Actions:

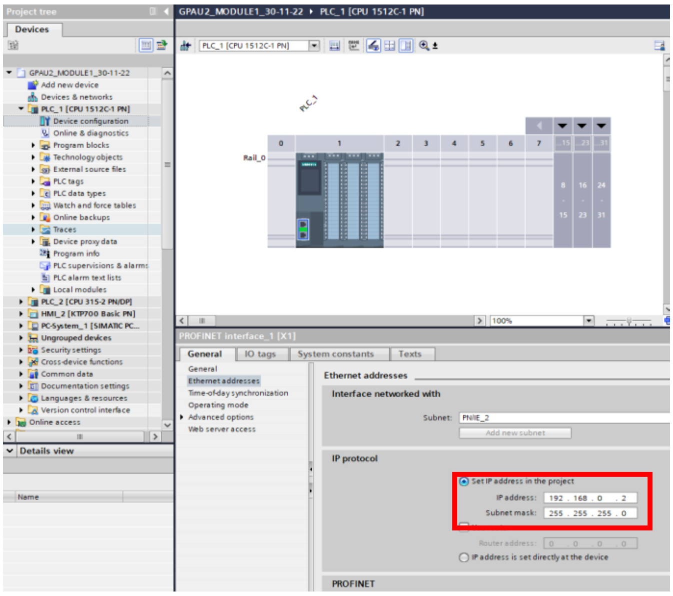
Interlock	Event	Qualifier	Action		
		S - Set to 1	"Geheugen".flank_botsingen	"Gebe...	%DB4....
		<Add new>			

T22: Trans22

Figure 41. Belt input condition on the program of Station 2

4.4. Simulation

As mentioned previously, two PLCs are needed to load the two modified programs. First an instance called “Station1” is created. When simulated alone the IP address will be the same as for the already saved PLC created for the first part of the thesis “DigTwinAtEdu_PLCSIM”. If both units were to be loaded, the TIA program would not be able to distinguish between both of them as their addresses would be “192.168.0.1” for both. To solve this problem only the “Station1” is started, then on TIA Portal when opening the program for station 1, the software can connect to the only available unit, so it is fair to confirm is the one intended. Once connected, by going to the “Device configuration” of the “CPU 1512C-1PN” and clicking on the Ethernet ports highlighted in green on figure 42 the IP address can be changed in the properties of the device. By changing the address to “192.168.0.2” it will also change the address of the simulated PLC “Station1” and will stay like this even if the PLCSIM Advanced software is turned off. This procedure ensures the TIA program can connect to the same simulated instance even if there are more units loaded at the same time.



Ethernet addresses

Interface networked with

Subnet: PN/IE_2

IP protocol

● Set IP address in the project

IP address: 192 . 168 . 0 . 2

Subnet mask: 255 . 255 . 255 . 0

Router address: 0 . 0 . 0 . 0

○ IP address is set directly at the device

Figure 42. IP Address being changed

The same process is applied for the creation of a PLC for the second station, this one is called is “Station2” and its IP Address is “192.168.0.3”. As all units have different addresses, they can be run at the same time with no issues.

Now the Digital Twins of both stations is done and ready to be simulated. When the simulation starts, the automatic sequence of the first station needs to be activated with the start button of the HMI, the process will result in the positioning of the body of the contraption on top of the belt. The belt will then carry the body to the second station triggering the second PLC and its automatic sequence consequently, this will result in the feed and positioning of a bearing on top of the body. This simulation proves that two DTs can be linked together with some room of improvement and avoidable problems.

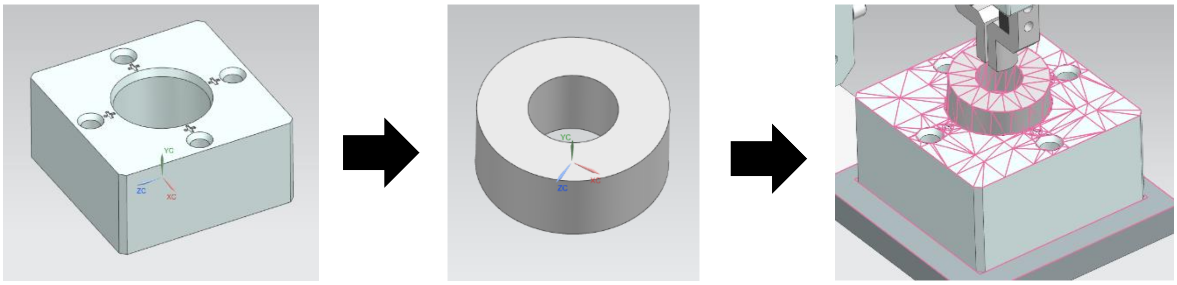


Figure 43. Body, bearing and their union

5.Conclusions

5.1. Results

The first objective was to create a Digital Twin of the station 2 of the automatic assembly line FMS-200 from SMC. The first part of this thesis has shown the steps followed for its creation, then it has been proved that is a functional replica of the real model by simulating it.

The second objective was to link the newly created DT with an already existing one. As a direct linking between stations was not viable, the Digital Twin of the first had to be redone. After that minor step back, both stations were modelled, and their results were satisfactory. The results proved that more than one PLC can be simulated simultaneously leading to a more complex automated process which resembles a more exact model of the real assembly line.

These simulations have proved the advantages of using a Digital Twin. If the model is designed properly, any user can run their programs intended for the real machine through the DT first. This means the software designer does not need to be in contact with the real assembly lines at all times, as a result code for the intended machine can be simulated and diagnosed from anywhere at any time. Nowadays this is a really useful feature as industry is evolving towards telecommuting.

On the educational aspect, teachers do not require to purchase expensive machinery as they can provide students with Digital Twins instead. This leads to a decrease in costs and can enlarge the number of institutions that use these kinds of technologies as they are more affordable.

Moreover, on its industrial use, MCD Inspector tool has already been proved to be a powerful software when it comes to diagnosing simulations. Being able to graph forces, movements, maximum speeds and so many other characteristics allows industries to redesign and optimize machinery without the need of building expensive prototypes. There is a clear example of this in FMS-201, the modeled two-point gripper in charge of placing the bearing on top of the body in reality possesses three points of contact instead of two. This means SMC employees, after designing and building a prototype of the station, decided through real life simulation that a three-point clamp would provide a better alignment with the hole of the body. If they had had a Digital Twin of the full assembly line, they could have detected this problem beforehand, this would have resulted in a reduced cost of prototyping and testing.

5.2. Future improvements

There is a lot of room for improvement both for UCLL students as well as for Siemens Software.

Regarding UCLL, they already have two Digitals Twins of the FMS-200. Now that is has been proved that they can be connected together, students should keep on creating DTs for the other 6 stations they have acquired. This way students can learn about Siemens NX software as well as PLCSIM Advanced. Nevertheless, code should be written taking in mind the software can only be simulated in the 1500 series PLCs and these have different requirements as those units from the real machine.

On the other hand, Siemens has an innovative software but there are many features to improve. First of all, it should be easier to import/export MCD files as no quick solution was found during this project. Secondly, to optimize the connectivity between MCD and PLCSIM Advanced, the mapping tool should link signals with the input/output port direction and not with their tag names. This means it will not matter to switch between different codes with different tag names, as long as they are intended for the same machine the links will stay the same as port directions will not change. At last, PLCSIM Advanced software could be improved to support PLC units from other series as for example the 300 series.

References

Lasi, H., Fettke, P., Kemper, HG. *et al.* Industry 4.0. *Bus Inf Syst Eng* 6, 239–242 (2014). https://doi.org/10.1007/s12599-014-0334-4

Singh, M.; Fuenmayor, E.; Hinchy, E.P.; Qiao, Y.; Murray, N.; Devine, D. Digital Twin: Origin to Future. *Appl. Syst. Innov.* (2021), *4*, 36. https://doi.org/10.3390/asi4020036

Learn-/Training Document | DigitalTwin@Education Module 150-001 | Edition 02/2020 | Digital Industries, FA

Learn-/Training Document | DigitalTwin@Education Module 150-002, Edition 02/2020 | Digital Industries, FA

Learn-/Training Document | DigitalTwin@Education Module 150-003, Edition 02/2020| Digital Industries, FA

Learn-/Training Document | DigitalTwin@Education Module 150-004, Edition 04/2020 | Digital Industries, FA

Learn-/Training Document | DigitalTwin@Education Module 150-005, Edition 04/2020 | Digital Industries, FA

Learn-/Training Document | DigitalTwin@Education Module 150-006, Edition 04/2020 | Digital Industries, FA

What is a digital twin: https://www.ibm.com/topics/what-is-a-digital-twin

Industry 4.0: https://www.ibm.com/topics/industry-4-0

Digital Twin: https://new.siemens.com/global/en/markets/pharmaceutical-life-science-industries/pharma-industry/focus-topics/digital-twin.html?gclid=CjwKCAiAoL6eBhA3EiwAXDom5lN5GOPTW5a3gjBfDsZ3nDFLu7wOJGL_3A3KvK7VcBzk6vFfqeugUxoCHdEQAvD_BwE&acz=1

NX help: https://docs.plm.automation.siemens.com/tdoc/nx/1899/nx_help/#uid:index