RobotStudio in Engineering Education

Ares Seuma Perat

Degree Thesis for Bachelor of Engineering

Double Degree Programme in Electronic Industrial and Automation Engineering and Energy Technology

Vaasa , 2020

Acknowledgements

I would like to mention all the people who I have shared this year abroad with, to my friends and my classmates for being with me the whole year helping in all the projects and tasks. To my teachers on Novia, for giving me the best of them, being patient, and prepare us for our future and our professional world. I would like also to thank my family for the accompaniment I have had from afar during this stage, and for the infinite support received.

Philip Hollins, thank you for your accompaniment throughout the course, the infinite feedback received and the desire that you show to improve in each of us, and not just in our tasks and on this thesis, also to improving professionally and learning effectively.

Roger Mäntylä, the customer of this project, thank you for guiding the project and thank you for being always there the whole course and for being free to talk and give me advice every time I have needed. Thank you for making me feel at home in Finland.

DEGREE'S THESIS

Author: Ares Seuma Perat

Degree Programme and place: Electronic Industrial and Automation Engineering and Energy Technology

Specialization Alternative: Robotics

Supervisor(s): Roger Mäntylä and Philip Hollins

Title: **RobotStudio in Engineering Education**

Date: 30th September 2020 Number of pages: 98 Appendices: 1

Abstract

The aim of this project was to learn how to use the software used for programme ABB robots, called RobotStudio, and to perform a course material with a range of simulation examples, step by step, which will be used in the course "Autonomous systems" at Novia University.

Firstly, this report contains some background about robots and the ABB company. Then, full research was carried out about the use of the software and how the examples could be implemented. For it, it was needed to acquire knowledge about the programme, how to program on the station and about the language code, called RAPID.

As a result, a manual was made to help students to learn how to use the software used on this project. On it, it is explained how to start putting elements on the simulation, creating a path, turning on the simulation, among other tools. The main part of the report is the simulation examples. An effort was made to work with different activities and increase the difficulty each time. Six different simulations have been developed, fully explained and they can be imitated easily to learn about the programme and the code.

Language: English Key words: Robotics, ABB, RobotStudio, simulation

Table of Contents

List of Figures

List of Tables

1. Introduction

In the past, robots and artificial intelligence were an idea created for fiction. However, the reality surpasses the fiction and the robots are becoming a true being. The technology is developing rapidly and the robots are taking every day more importance in our lives, routines and industry.

Humans are afraid because the robots might steal working places, especially in the industry, and it is true but they should think about all the advantages in the production which provide working with robots. The robots do not think and they just work continuously without matter the hours. They do not get tired, and that means that the precision and speed working will be always the same. Working with technology helps to get better production and quality.

The Swiss-Swedish multinational company ABB is one of the world leaders in industrial productivity, manufacturing processes and robot software and equipment. They have installed more than 400.000 robots around the world. The company has its robots with its own language code called RAPID to programme them. RobotStudio is a virtual controller copy of the software which is used to execute the different robots and it is is an excellent tool for creating realistic simulations.

The aim of this project is to perform a course material with some simulation examples, step by step, which will be used in the course “Autonomous systems” at Novia University. The course material will be about robots simulation, including programming knowledge. This project will be undertaken using the ABB Robot Studio 6.08 software.

The university course has three credits in the present and the students deal with the understanding of the principles of self-learning systems and control relevant concepts keeping in mind autonomous robots. Each group of students have a robot car with a programable processor and its sensors. They must programme it for complete autonomously a labyrinth. The course contents are robot hardware construction; implementation of autonomous software; selection of sensors; optimization of the implementation of the software and sensors; functionality tests; demonstration and final tests and documentation of the implementation and selected algorithms. The competences of the university course and the credits will increase with the simulation examples of this project.

The project presents background about the robotic evolution and then a simple manual has been written about the use of the software and its tools. The manual explains and gives examples about the common steps which the user will follow to reach a simulation. Subsequently, some examples of simulations are explained with different steps easy to follow by a student. An effort was made to create different simulations to study and understand different cases and situations.

2. Purpose

The aim of the thesis is to create different simulations using the software RobotStudio 6.08 and explain them step by step. The simulations created will be used in the course "Autonomous Systems" at Novia University.

To meet this aim, the following objectives have been established:

- Acquire knowledge of the program, get used to it and learn how to program with the new programming code.
- Develop a range of examples of robot simulation
- Establish a simple manual step by step of how to reach each simulation.

3. Robots

There are a lot of different types of robots and there is not an exact definition for the "robot" word. The most frequently used as a robot definition is "A robot is an actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks. Autonomy in this context means the ability to perform intended tasks based on current state and sensing, without human intervention. (European Engineering Industries Association, 2020) and (Robotics, 2015)" But we can also consider different opinions as:

> "I would say that a robot is a physically embodied artificially intelligent agent that can take actions that have effects on the physical world" says roboticist Anca Dragan of UC Berkeley (Simon, 2017).

> "My definition of a robot, given that there is no very good universal definition, would probably be a physical machine that's usually programmable by a computer than can execute tasks autonomously or automatically by itself. What a lot of people tend to follow is this sense, think, act paradigm" says Kate Darling, a roboticist at the MIT Media Lab (Simon, 2017).

> "It is a system that exhibits 'complex' behaviour and includes sensing and actuation," says the roboticist Hanumant Singh at Northeastern University (Simon, 2017).

> "A robot is some sort of device which has sensors, which sensors the world, does some sort of computation, decide on an action, and then does that action based on the sensory input which makes some change out on the world outside its body" says Rodney Brooks, an American roboticist (IEEE Spectrum, 2020).

A robot is actuated mechanism, a device, that manages to do different tasks and applications. Evidence suggests that are faster, more precise and more consistent than a human worker. The robot does not need to have rest and it can work continuously. Nowadays, the robot can be in every field (as in the medicine for example). Another benefit of robotics is that they eliminate dangerous jobs from human workers: they can manage better with heavy loads or toxic substances, for example (RobotWorx, 2020).

An industrial robot is a robot system used for manufacturing. It is automated, programmable and capable of movement on three or more axis. There are different types of industrial robots like Cartesian robots, gantry robots, SCARA (Selective Compliance Articulated Robot Arm) robots, articulated arm robot and human-assist robots. To observe the evolution of industrial robots a few examples are presented (Wallén, 2008).

The name "robot" was invented by the Czech writer Karel Capek and it was used in his science fiction stage play on 1920. Between 1956 and 1959 in Danbury (Connecticut, USA), George Charles Devol and Joseph F. Engelberger co-founded the first robotics manufacturing company: Unimation. In 1961, the first industrial robot was born. A few years later, in 1967 the first industrial robot in Europe was installed in Metallverken, Sweden (Robot Worx SCOTT company, 2020).

In 1973, at Cincinnati Milacron Corporation, Richard Hohn developed the robot called The Tomorrow Tool (T3). It was the first commercially available industrial robot controlled by a microcomputer (Internation Federation of Robotics, 2020). The Cincinnati Milacron T3 robot is an example of a robot arm which is similar to the human arm. The arm consists of several rigid members connected by rotary joints. T3 was controlled using a Hierarchical Control System; it is partitioned vertically into levels of control.

The Olivetti SIGMA, a cartesian-coordinate robot, is one of the first used in assembly applications. It was born in 1975 and it was used in Italy for assembly operation with two hands. In 1962, the company IBM (USA) created a powerful, easily programming language, specifically for robotic applications; engineers could quickly and easily create application programs. In 1994, Motoman introduced the first robot control system (MRC) which provided the synchronized control of two robots. MRC also made it possible to edit robot jobs from an ordinary PC; it could also synchronize the motions of two robots (Cox, 2020).

ABB developed the FlexPicker on 1998, the world's fastest picking robot based on the delta robot developed by Reymond Clavel. It was able to go to the velocity of 10 meters per second, using image technology. In 2006, Germany presented the first Light Weight Robot: KUKA. The structure of the KUKA lightweight robot is made of aluminium; it is highly sensitive and it only weighs 16 kg. The robot is energy-efficient and portable and can perform a wide range of different tasks.

On 2008, the world's first collaborative robot able to operate safely alongside people enters the market. A few years later, in 2009, ABB launched the smallest multipurpose industrial robot: IRB120. It weighs just 25kg.

3.1. Robots in the industry

Europe has a strong position in the robotics world, having 32% of the world market, but China is the largest industrial robot market (SPARC The Partnership for Robotics in Europe, 2020). According to the International Federation of Robotics[1], by the end of 2021, there will be around 3,788,000 industrial robots (Executive Summary World Robotics 2019 Industrial Robots, 2019).

Nowadays the robots have become remarkably important in our lives, as at home or work. In each field, there is the presence of the robotics; as is the health care or medicine,

[1] IFR is a international federation with twelve different countries as members founded on 1987. They are focus on the robotic industry worldwide.

agriculture, the food preparation, the manufacturing, the military (Ohio University, 2018). In these last five industries, they are using the new technology to improve the efficiency for the businesses and the consumers.

Robots are becoming more and more involved in the medicine and care of humans (Michelini, 2014). Today, robots are already being used in everything from caring for an elderly person alone in his home to robotically intervening in a surgery. On the surgery, the precision is better, and usually, the robot can operate in areas that the human hand cannot.

The uses of the robotics in the agricultural improve the efficiency, the precision and it takes less time for the worker. For example, an autonomous robotic vehicle can scan the field, by GPS, and organize the most effective path. While the farmer is going with his tractor through the field, the vehicle will know exactly the field left (Sorensen, 2015).

In the field of the food preparation, the robots are used for seeding, spraying water and harvesting to cutting, processing and packaging the products. Other functions can also be for automatic quality detection. After including the robots on the industry, the food preparation industry has grown 25% their productivity (IQBAL, et al., 2017).

The United States military, for example, wants more agility and faster movement on their missions (School of Electrical and Electronic Engineering Nanyang Technological University, 2020). The planning to prepare for modern war includes the robotic use on the military. The military forces want to start with the human-controlled system and ending with autonomous, armed and cooperative robots.

3.2. Latest innovations

Since the beginning of the new millennium, technology has developed rapidly at a rate often faster than the people have adapted. The technology has already impacted our lives and it will change it even more. In this chapter, it will be explained some of the latest innovations in technology.

Firstly, 3D printing needs to be mentioned. Recent developments and wider applications of 3D additive printing have resulted in a revolution in the economic field, the culture, and the industry in the actual century. It will be a big change in the production: for use a

3D printer just a few workers are needed; the jobs' offer will decrease. In addition, 3D printing produces less waste than conventional manufacturing, using less energy and recycling (the material used on the printing can be recycled material). An important use of this innovation is in the medical field. They have already started to use the 3D printing for the surgery, the prosthetics and even more medication (3D printed pills are real).

Focusing on the robots, the latest innovations in the robotic field will be explained. The first robot that will be mentioned is the robot dog (Spot) (BostonDynamics, 2020), the robot created by Boston Dynamics. It was introduced in 2015 but it was not available on the market until 2019. It can run at 1.6 meters per second, it has 360-degree cameras for a full range of vision, it is water-proof and it can operate in temperatures from -20 to 45 degrees Celsius. It can be used on the construction, entertainment and even for public safety. The robot can record and repeat autonomous actions and nowadays it is used for the police.

The first living robot has been created in January 2020, its name is "Xenobot" and it is created with stem cells from frog embryos and left them to incubate (Yeung, 2020). The cells are cut and reformed into specific body forms and then, they can work on their own (repairing itself for example). The robot is smaller than a millimetre and it can travel through the human body. It can walk, swim and survive for weeks without food. It can be used for medicine, to reduce radioactive waste or to clean the microplastics on the ocean. According to Joshua Bongard (ScienceDaily, 2020), a computer scientist and robotics expert at the University of Vermont, the Xenobot is "a new class of artefact: a living, programmable organism".

The new generation of flying robots has arrived also in January 2020 with the innovation called PigeonBot (Temming, 2020). It is a drone which has a pair of biohybrid morphing wings (the shape can be changed). The robot is now considered the most advanced bird-like robot ever made and it has been used feathers from real pigeon cadavers. Once the team at Stanford's Bird lab has discovered how birds could transform their shape: controlling the feathers by wrist and finger movements; they replicate that motion in a robot (Linder, 2020). A single piece of laser-cut foam board is the body of the flying robot. The robot includes an electric propulsion system to fly, some sensors, a GPS, barometers, an autopilot system, etc.

In the last years, some industrial cleaning robots have been created, especially robots used as a vacuum, for example, Roomba (Linder, 2020). On January 2020 it arrived from Zhejiang University (China) a new cleaning vacuum robot, but this time a wall-climbing one capable of work in any kind of surface. The robot has a suction cups used to grid to walls but if the environment, the one that the robot touch, has some roughness, the machine fails.

One of the latest innovations on the technology and robotic field that should be mention is the self-driving vehicle. There are some companies that they already have created their own models of an autonomous vehicle, but there are two companies that stand out: Google and Tesla. The first autonomous vehicle was from Google, in 2016, and nowadays it is a separate vehicles company: Waymo (WAYMO, 2020). The company has already driven more than 12 millions of kilometres with their autonomous-vehicles in different parts on the USA, every day in different conditions and situations.

Tesla has already different models of autopilot cars and the hardware needed in the future for full self-driving capabilities in almost all circumstances (TESLA, 2020). The autonomous car has eight cameras that provide a vision of 360 degrees around the car with a range of 250 meters. The car has twelve ultrasonic sensors to complement visibility. Autopilot enables the car to steer, accelerate and brake automatically within its lane. Nowadays, legislation obliges active driver supervision.

The National Highway Traffic Safety Administration said that 94% of serious crashes are due to dangerous choices or errors people make behind the wheel (Ayers, et al., 2020). The autonomous car will give the safety that the people need for travel and the deaths will decrease. The autonomous car is already real but it is not on the street yet because of three problems. The first one is the legislation: the rules have to change for being able to drive an autonomous vehicle and coexist with the existing vehicles. The second one is society: the people have to adapt to the new technology world and all the innovations that have to come.

The last problem is the hardest one: the ethic. In an extreme situation, how can a robot decide which action take in each circumstance? There is a typical hypothetical situation: the driver has to kill someone with the car because they can not stop, who would the driver choose. Most of the statistics and studies show that the people agree with three priorities: the human before an animal; the legality before the illegality; and less quantity of people as possible. Another hypothetical considered situation is the following one: if there is an accident between a driver and a pedestrian, which action the autonomous vehicle would take to save one of them. All the autonomous vehicles will try to safe as long as possible the driver because no one will buy a vehicle which can kill themselves.

In France, the autonomous bus is a reality. There are already nine buses working on different regions of the country. There are also some autonomous buses working on Las Vegas and Canada. Experts believe that in the future, the property vehicle will not exist: everything will be shared and no ones will need parking. This is not a revolution, it is an evolution.

Experts on the mobility field believe on a future with flying-vehicles, and the reality is that it already exists the flying-taxi. On Singapore, for instance, they already have created legislation for drones and they are creating mobility more clean, smart and electric. According to the United Nations, in 2050, three-quarters of the world residents will live in the cities. It is already a lot of problems for traffic excess and it will increase, so a solution will be to have the roads on the air.

3.3. Robots and the environment

The world population has already more than seven thousand millions of humans (Worldmeters, 2020), and it is still growing. There are a lot of people that need to be fed, kept warm, educated... Each individual on the planet means consuming resources and producing waste. In addition, all the people need a place to live and that means growing the urbanization and in fact, growing the consumption of energy and the pollution of the air. This is just some of the basic needs that a person needs for living, but there is more as the car that the individual can need to go to work every day.

Experts on the field are looking for ways of modifying the Earth's environment and try to control climate change. For reducing the climate-related risks it is needed technology in the moment of building or upgrading infrastructure, and also in different industry sectors. The key to reducing pollution is automation and robotics in different fields. For example, preventing pollution and emissions through monitoring the release of harmful greenhouse gasses or optimizing the manufacturing processes, or using new software to scale down the operating cost and reduce impacts of the waste on the environment (Kudlak, 2019). There are also a lot of advantages to using robots in waste treatment and recycling.

As a consequence that the population is growing, it is also growing the food demand and agriculture is one of the responsible for the increase of greenhouse gas emissions (Anon., 2019). Agriculture is already changing with the automation and new technologies, for example, the robots are already substituting some of the tasks on this field. From robots to autonomous tractors to robotic arms, the technology is growing in creative and innovative applications as harvesting and picking (one of the most popular due the accuracy and velocity needed), weeds control, sorting and packing, utility platforms... (Team, 2017).

In the present, less than 1 per cent of the freshwater is truly accessible for humans (NRDC, 2020). The unsafe water is one of the main causes of death and as the population will grow as estimated, the water demand will also increase. As a consequence, some devices have been created. A swimming machine that eats the pollution on the waterways, called Row-bot, digests the rubbish and converts it into electricity for fuel (ThePermacultureResearchInstitute, 2020) and (Philamore, 2015). The robot can operate per months because it can generate more energy than it consumes. The prototype robot has two subsystems: the first shows the power generation capability of the robot and the second subsystem moves the robot by feeding energetically (less energy than created on the first subsystem). This device is still in research because it is not biodegradable, but it would be the future to clean the pollution on the water. Another example of cleaning the water is the WasteShark (Gabbatiss, 2019), it is able to swim autonomously and trap plastic, oil and other pollutants with its huge mouth. The robot can capture 60 kg of waste, which will be recycled, at a time.

It is obvious that recycling is an important action to reduce produced waste. Recycling robots can do dangerous and laborious manual work for humans. Robots have the potential to improve waste management and be able to efficiently recover raw materials without damaging their quality. For example, Liam created by Apple in 2016 (McSweeney, 2017), can melt the iPhone 6 plastic and use the plastic to create new products. The robot is capable of melt each iPhone in just eleven seconds and it is used tools for separate the different devices from the phone. On 2018, another recycling robot was created by Apple: its name is Daisy (Deahl, 2018). The new robot can work with nine different versions of the iPhone and it also separates the different parts. The last example of a recycling robot that will be mentioned is Dorabot (Mohammed Bin Rashid Initiative for Global Prosperity, 2020). This robot is capable to identify, pick and classify recyclable items thank its learning-based computer vision and dynamic planning.

Since the industrial revolution, carbon dioxide emissions have increased and it affects air pollution (Rogers, 2018). As a consequence, the amount of contamination in the air contributes to the greenhouse effect, the climate change, the acid rain and some health problems for the citizens. Some researchers are already trying to reduce the carbon dioxide on the air using robots and automation. For example, it has been created as a tool that allows separating the carbon dioxide from the air. The device, electro-swing reactive adsorption for CO2 capture (a demonstration of how it works can be seen here: https://vimeo.com/368583616), is basically a battery that absorbs carbon dioxide from the air that is transported through its electrodes while charging, and then releases the gas while discharging (Massachusetts Institutes of Technology, 2020). The robot would simply alternate between charging and discharging, blowing air for power through the system during the charging cycle; and then expelling the carbon dioxide captured and concentrated during discharge.

One-third part of the carbon dioxide emissions ends on the rivers, lakes, oceans... It seems that the effects of the greenhouse gas emissions are not just for the humans and the terrestrial life, there are harmful effects for life underwater. Some of the carbon concentrated on the water become carbonic acid and causes ocean acidification that threat important creatures. Engineers at the University of California-San Diego propose sending out micrometre-length tubes that rotate around in the water, converting the CO_2 into a harmless subproduct (Schiller, 2015). The robots have a layer of an enzyme that helps convert CO2 into calcium carbonate. On the laboratory, the devices have converted until 90% of the CO2 provided.

4. Simulation

A simulation is an imitation or a representation of a real operation over time. The fictional copy must have all the characteristics, behaviours and functions as the real system. The simulation affords the experimentation on a valid digital representation of the system because the model simulation is usually programmed with the same algorithm as the real-world model.

A simulation can be also used for study or intervene on an inaccessible or danger system for example. The three main uses of the simulation are the analysis, the training and the research. Creating a simulation of a product, an activity or a process could have some advantages as testing and exploring before the creation or making changes in the real-world. Save money and time; usually, the creation of a simulation is faster than creating a real-world system. The visualization of the operation can be seen easily and more understandable. A simulation provides more accuracy and more precision; it also contributes to having a better visualization and idea of the system behaviour over time.

Like any algorithm, the simulation has also a few steps to follow to approach the final result: Problem definition, project planning, system definition, model formulation, input data collection and analysis, model translation, verification and validation, experimentation, analysis and implementation (Anon., 2020).

The type of measurement is reflected by the type of model or language of the simulation: discrete (variables affected by instantaneous changes of the system state), continuous (variables which represent state changes in a continuous way) or combined (Thesen, et al., 2020).

5. Artificial intelligence

It can be defined as a group of algorithms that are implanted in a robot for being able to perform tasks or make decisions, thinking and acting humanly and rationally. As the definition of a robot, the artificial intelligence or AI has given rise to many questions and debates and there is no singular definition accepted for everyone.

There are considered three types of artificial intelligence according to their abilities (Builtin, 2020) and (O'Carroll, 2020). The first one is the narrow or weak AI; it operates in a limited context and it is focused to execute a single task, for instance, the voice recognition. It does not have any memory or data storage. The next type of AI is artificial general intelligence (AGI) or strong AI; it has the ability to learn and apply its intelligence to solve any problem. The last intelligence is the one called Artificial Superintelligence (ASI) which the machines become self-aware and they pass the human capacity, ability and intelligence.

Nowadays, scientists have the tools to create AI systems with high levels of understanding but they have not archived even strong AI (Kedia, 2020). The three areas where the AI is at a higher level are natural language processing which is an analysis and understanding of the human language; autonomous vehicles and reinforcement learning which is an area of machine learning. This last field is a system concerned about how software agents would take actions.

6. Industry 4.0

The industrial revolution has four different stages. The first major change was in the 18th century when mechanization arrived along with steam power and water power. This is known as Industry 1.0. On the 19th century, Industry 2.0 was born which holds the start of mass production and assembly lines. A clear example is Ford starting to use it for cars production. Next, there is the automatization of different processes, otherwise known as Industry 3.0. This industry made production processes safer and more accurate in the 20th century. Finally, Industry 4.0 has been developed in the last years and it will develop even more during the 21st century.

Smartphones are a good example of this revolution: everything is faster and safer and the devices are connected to each other. The actual revolution enables a person to have control of each device from everywhere. That means that the production could be controlled from everywhere and the quality is higher for the reason that is easier to detect the anomaly. In other words, the new revolution is focused on the interconnectivity, automation, machine learning and real-time data. In addition, the fourth industrial revolution is also related to technologies which can create virtual versions of real-world installations, processes and applications as the use of the simulation (TWI, 2020).

7. ABB

ABB is an industrial company of electrical equipment founded in 1988. It is the result of many mergers, but firstly ASEA (translated from Swedish: General Swedish Electrical Limited Company) and BBC (Brown, Boveri and Compagnie) working together (ABB, 2020). On the beginning, ABB's operations included power generation, transmission and distribution, electric transportation and industrial automation and robotics. But in the last years, ABB has worked on the innovation in technologies to harvest energy, improve productivity, take care of the environment, increase their profits, etc. In the present ABB is working in more than ninety countries (Anon., 2020).

ABB is one of the technology leaders nowadays and the company is managing the digital transformation of industries, it wants to write the future of industrial digitalization and realize value. The company has five focused globally leading businesses: electrification, industrial automation, motion, power grids; and robotics and discrete automation (ABB, 2020).

ABB is today supported by its common ABB Ability digital platform (ABB, 2020). This platform was launched in 2017 and it offers more than two hundred digital solutions to increase productivity and safety at lower costs. Thanks to that platform, the company ABB grew fast into the automation market for industrial companies. This action put ABB on the top of the leadership in digital solutions and artificial intelligence.

There a few technologies from ABB which were important for the world (ABB, 2020). The first example is the smart sensor created in 2016 which connects low-voltage electric motors with internet and it allows to be controlled via a smartphone or an app. ABB has created in 2008 a software which allows people to work close and safely with the robots. ABB has introduced the first collaborative industrial robot in 2015. The company also launched a system which ables to include all plant automation functions in a single operation.

One of the latest innovations in the company is focused on sustainable mobility creating intelligent free-emissions solutions as the new infrastructure in Singapore port (ABB, 2020). It is about a new battery charging technology as a consequence of the increase in demand for electric vehicles. It will be completed in 2040 and it will be the world’s largest fully automated container terminal but the first container will work at 2021. Automation is the key to build an installation of this size and complexity.

ABB is also implementing in the traditional motors and pumps a connected system with smart sensors (ABB, 2020). For instance, they are using algorithms to provide information about the operation as the vibration or the temperature and share it via Bluetooth or a smartphone. The new smart system helps to reduce the risks and maintenance costs or prevent unexpected breakdowns. These sensors are implemented for example in oil refineries in Australia to avoid overheating.

The company has different industrial robots with different weight, payload and reachability to adjust to each application. Most of them are industrial robots and collaborative robots and they have more than 400.000 robots already installed over the world.

8. ABB RobotStudio

ABB RobotStudio is a software of simulation. The program gives the possibility of robot programming providing the tools and functions needed. RobotStudio is built on the ABB Virtual Controller which is a copy of the real software that runs the ABB robots in production (ABB, 2020). The program allows doing realistic simulations as in the laboratory with the ABB robots.

RobotStudio first document which can be found on the internet is the presentation of the product in 2002. The program needs licence but there is some free version of the software. It is important to have a suitable licence for work with the program, if not the user might have problems with the configuration of the virtual controller and some functions or converters. Right now, since 27th March and until the end of 2020 the software is free as a consequence of the pandemic (infoPLC, 2020). A lot of people in the robotic and automation industry is working from home and ABB wants to keep the production lines.

An ABB robot can be programmed in different ways: with the FlexPendant, with the RAPID language on the RobotStudio or with the RobotStudio on the three-dimensions space (there are different options for select and program). Does not matter which way is used to programme the robot because when you modify instruction in one place, the user can synchronize both spaces.

RAPID is considered a high-level programming language and it is just used for the control of the ABB industrial robots. Programming with this language can be complex, each line of the code belong especially to each target which constitutes the path of the robot. Additionally, there are some functions like any other programming language as to assign a variable, do an iteration, change the value of output or an input.

9. Manual RobotStudio

9.1. Movement on the three-dimensional space.

Table 1. Movement on the station – Manual RobotStudio

Keyboard	**Mouse bottom**	**Mouse movement**	**Movement on the three-dimension space**
Ctrl	Left	Yes	Movement camera lineal in the same orientation
Ctrl + Shift	Left	Yes	Movement camera circular changing the orientation. There is a reference central point.
-	PUSH wheel bottom	Yes	Zoom in/out
-	SPIN wheel bottom	No	Zoom in/out

If it is pressed the right bottom of the mouse, it will appear a menu where you can choose the orientation where you want to see the robot.

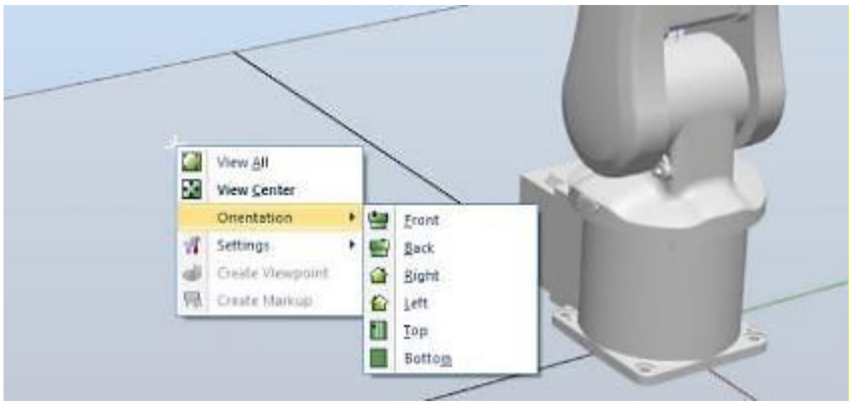


Figure 1. Manual RobotStudio - screenshot 1

9.2. Robot's movement

There is a possibility of setting the position of the robot on the three-dimensional space. On the left side on the screen, there is the layout list. In this list, it appears all the robots/tools/elements that are being used on the program at that moment. You can select the robot from there, and then press the right bottom of the mouse.

A menu will appear with different options which one of them is the "Position", "set position". It will appear the tab of Set position and there you can choose from which reference you want to change the position, and it can be typed the position in the three-axis and also the orientation in degrees.

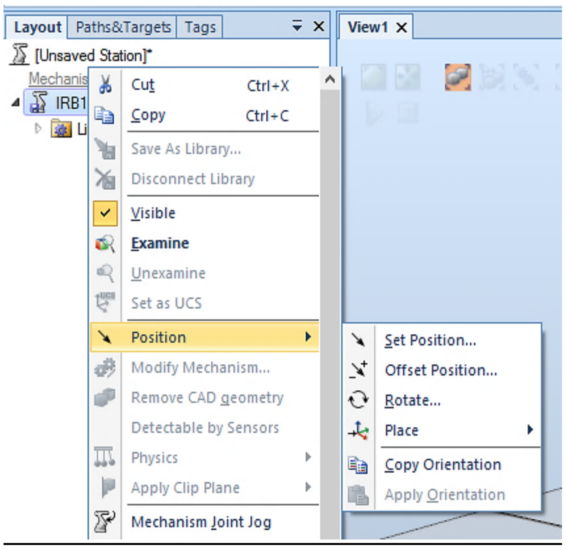
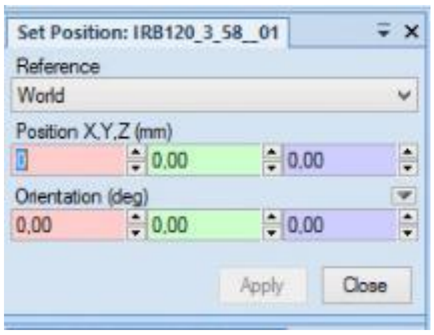


Figure 2. Manual RobotStudio - screenshot 2



Reference: World
Position X,Y,Z (mm): 0 | 0,00 | 0,00
Orientation (deg): 0,00 | 0,00 | 0,00

Figure 3. Manual RobotStudio - screenshot 3

There is another choice to move the robot. The option mentioned is the freehand movement located on the Home menu. While there are selected the robot on the layout list located on the left, you can also choose one of the options on the menu freehand (see Figure 4), and move the robot with the mouse.

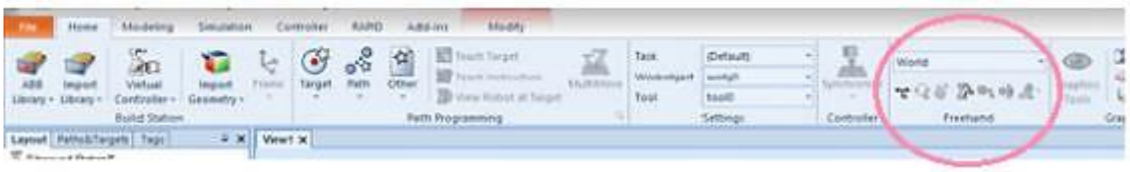
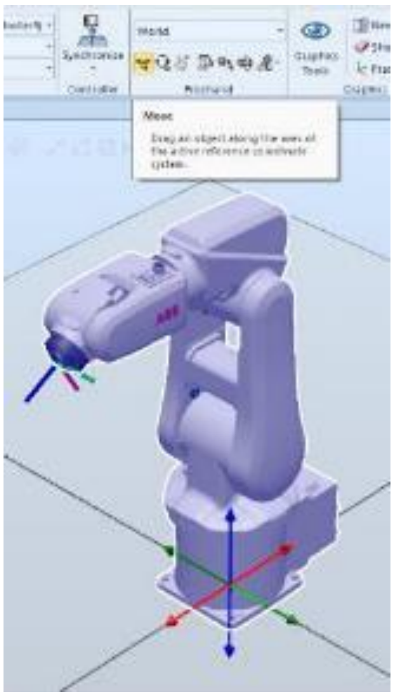


Figure 4. Manual RobotStudio - screenshot 4

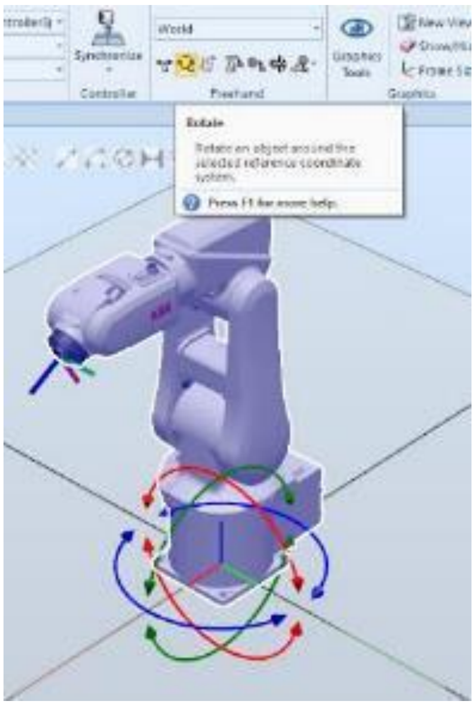
The first two options are the "move" (movement lineal in the three axes) and "rotate". As shown in the following Figure 5, it appears some axis on the robot's base which can be used to move the robot in the three different directions.



Move

Drag an object along the axes of the active reference coordinate system.

Figure 5. Manual RobotStudio - screenshot 5



Rotate

Rotate an object around the selected reference coordinate system.

Press F1 for more help.

Figure 6. Manual RobotStudio - screenshot 6

Other choices on the freehand menu are the "Jog Joint", "Jog Linear" and "Jog Reorient" options. Pressing the first option mentioned it is possible to move the different parts (from a joint to the next one) of the robot in its axes. The last two Jog's options are the linear and the reorient; this two work the same as the options "move" and "rotate" explained before, but the difference is that the axes for moving the robot do not appear on the robot's bases, they appear on the sharp point (in the same point that the tool can be connected).

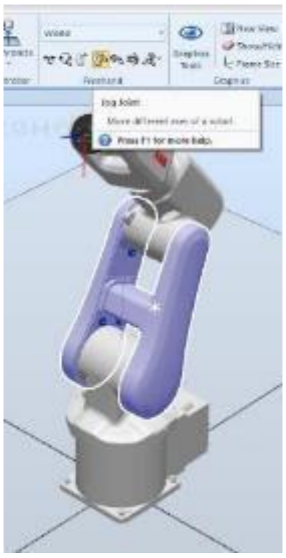


Figure 7. Manual RobotStudio - screenshot 7

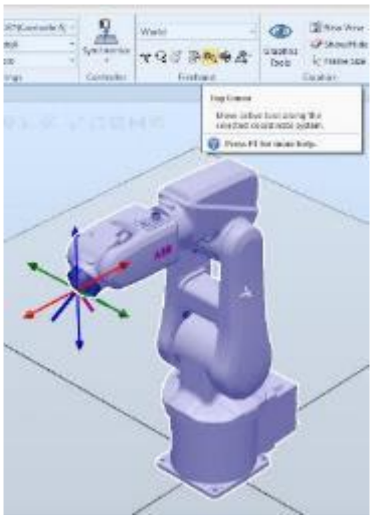


Figure 8. Manual RobotStudio - screenshot 8

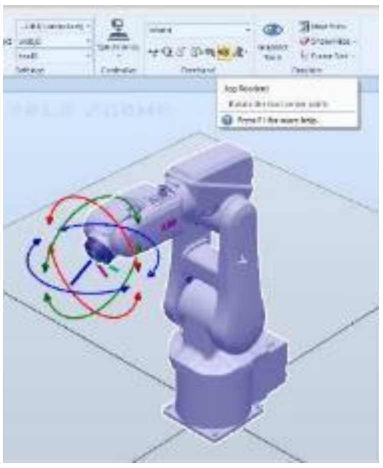


Figure 9. Manual RobotStudio - screenshot 9

There is another option to have this type of movements but more accuracy, without the freehand choice. On the layout list, if it is selected the robots' name and it is pressed the right bottom of the mouse, the same menu as when we were programming the position of the robot, will appear. On it, it can find the options "Mechanism Joint Jog" and "Mechanism Linear Jog". Once the user presses one of these options, the movement can be controlled without freehand, using values on its axes.

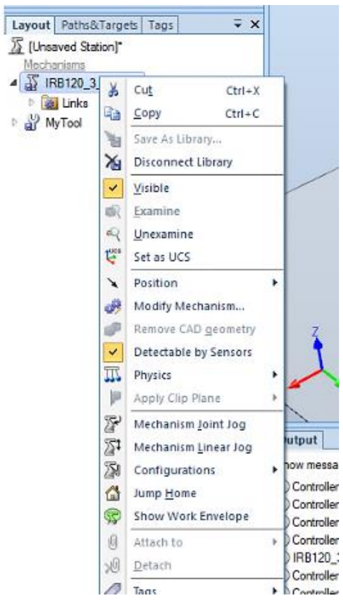
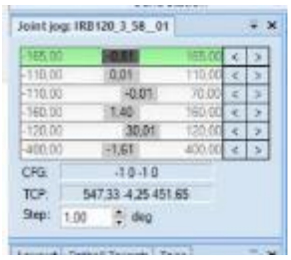


Figure 10. Manual RobotStudio - screenshot 10

"Mechanism Joint Jog"

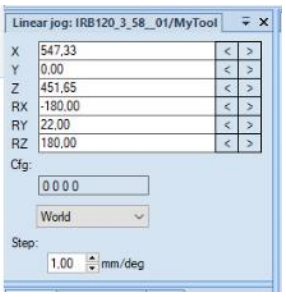

Joint jog: IRB120_3_58__01

-165,00	-0,01	165,00
-110,00	0,01	110,00
-110,00	-0,01	70,00
-160,00	1,40	160,00
-120,00	30,01	120,00
-400,00	-1,61	400,00

CFG: -1 0 -1 0
TCP: 547,33 -4,25 451,65
Step: 1,00 deg

Figure 11. Manual RobotStudio - screenshot 11

"Mechanism Linear Jog"


Linear jog: IRB120_3_58__01/MyTool

X	547,33
Y	0,00
Z	451,65
RX	-180,00
RY	22,00
RZ	180,00

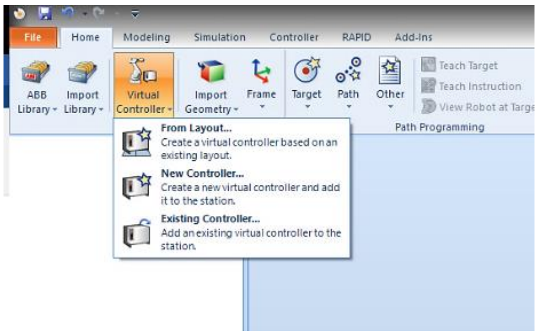
Cfg: 0 0 0 0
World
Step: 1,00 mm/deg

Figure 12. Manual RobotStudio - screenshot 12

9.3. Configuration of a virtual controller

It can be compared to a virtual controller of the robot with the brain of a human. It is important to know that without the virtual controller the robot could not work and either be programmed.

For creating a virtual controller on the ABB software, on the menu home, there is the option to do the configuration of it. It will be chosen the first one (from layout...) and it will appear the following window in Figure 14. On this pop-up window, it will be inserted the name of the new controller, the location and the RobotWare version to use. Then, pressing "next" it can be selected the mechanism for the controller (the robot). After the second and last "next", there it will be a change option for editing some characteristics of the controller as the language, the security, the industrial networks, different motion tools...

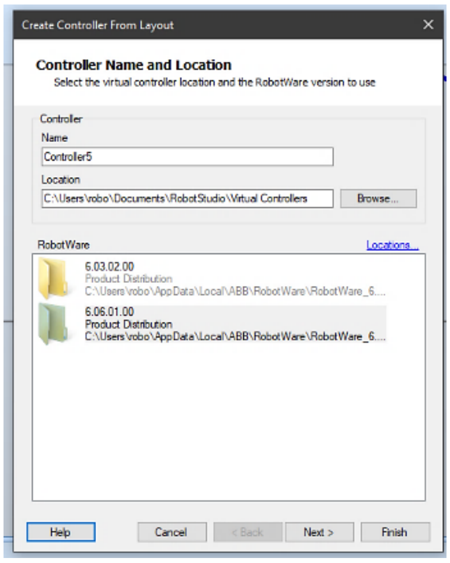


From Layout...
Create a virtual controller based on an existing layout.

New Controller...
Create a new virtual controller and add it to the station.

Existing Controller...
Add an existing virtual controller to the station.

Figure 13. Manual RobotStudio - screenshot 13


Create Controller From Layout

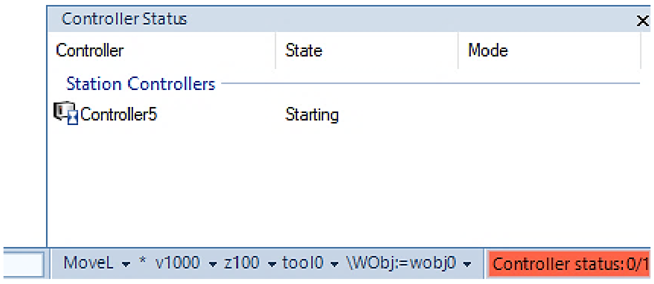
Controller Name and Location
Select the virtual controller location and the RobotWare version to use

Controller
Name
Controller5
Location
C:\Users\robo\Documents\RobotStudio\Virtual Controllers

RobotWare
6.03.02.00
Product Distribution
C:\Users\robo\AppData\Local\ABB\RobotWare\RobotWare_6....
6.06.01.00
Product Distribution
C:\Users\robo\AppData\Local\ABB\RobotWare\RobotWare_6....

Figure 14. Manual RobotStudio - screenshot 14

After it has been configurated the virtual controller the window on Figure 15 will appear. It will appear as long as the virtual controller is starting. Once it is ready, the controller status will change to a green colour and that means that the controller is configured and ready for work.



Controller Status

Controller	State	Mode
Station Controllers		
Controller5	Starting	

MoveL * v1000 z100 tool0 \WObj:=wobj0 Controller status: 0/1

Figure 15. Manual RobotStudio - screenshot 15

9.4. Configuration of the tools

For use the robot with a tool on its sharp point, there are two options: the first one is to insert one of the tools that are already created by RobotStudio. The second option is creating a tool, using a solid for example, and configured it as a tool with a reference point. Subsequently, the program will let connect the tool to the robot.

On the home menu, there is an option for import equipment; as cabinets, Welding equipment, tools or training objects; from the library. The training objects, as the tool called myTool, are useful when the person who is using the program is a beginner. On the menu it can be selected one of the tools and automatically the tool will be inserted on the coordinate origin and its name will appear on the Layout list.

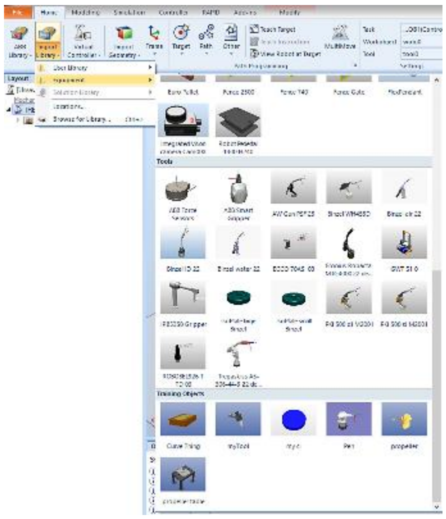


Figure 16. Manual RobotStudio - screenshot 16

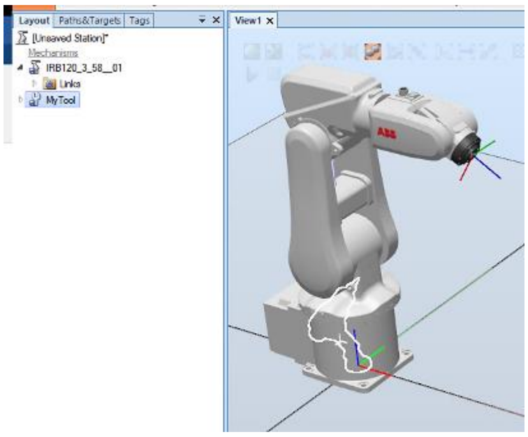


Figure 17. Manual RobotStudio - screenshot 17

Once the tools' name is on the Layout list, it can be dragged to the robots' name and the tool will take its place on the robots' sharp point.

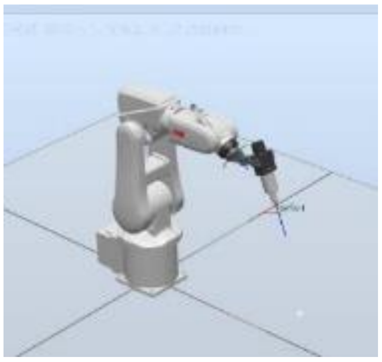


Figure 18. Manual RobotStudio - screenshot 18

In addition, alluding to the movements "Jog Linear" and "Jog Reorient" mentioned before, the axes were on the sharp point of the robot. Once the tool is connected to the robot, this reference point of the movements will change as shown in Figures 19 and 20.

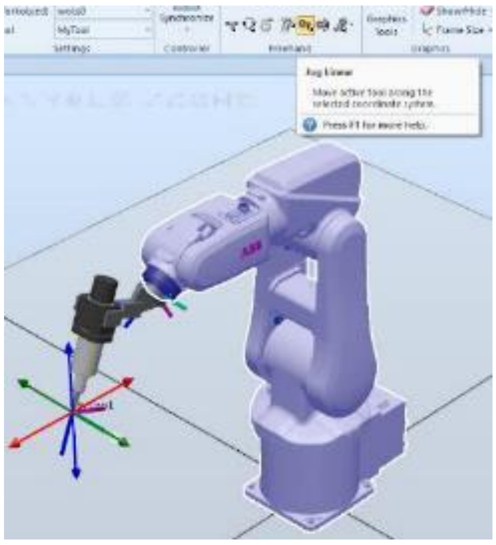


Figure 19. Manual RobotStudio - screenshot 19

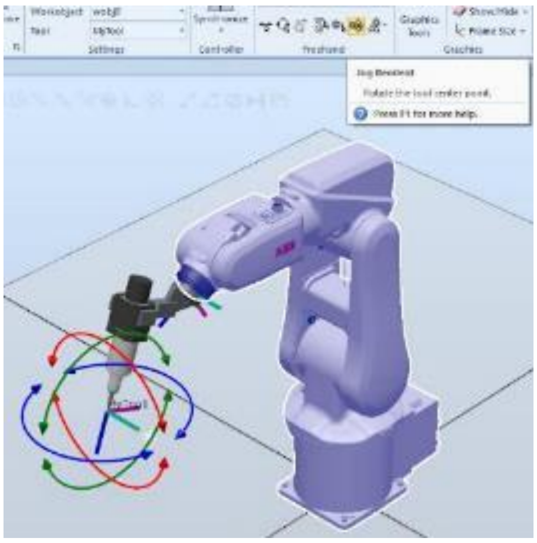


Figure 20. Manual RobotStudio - screenshot 20

As mentioned before, there is an option for creating a tool by the users. An option is using the option "solid" found on the Modeling menu to create an object that we can use as a tool. As shown in Figure 21, it can be chosen the solid that the user prefers and a pop-up window to choose all the solid's parameters like the height. Once the solid is created, the object will appear on the coordinate origin and its name on the Layout list. The solid can be moved as the robot, using the same motion options on the freehand or pressing the right bottom of the mouse. On the layout list, pressing the right bottom of the mouse over the name of the new object, it can be changed some characteristics as the name, the position, set the UCS (User Coordinate System)[2]...

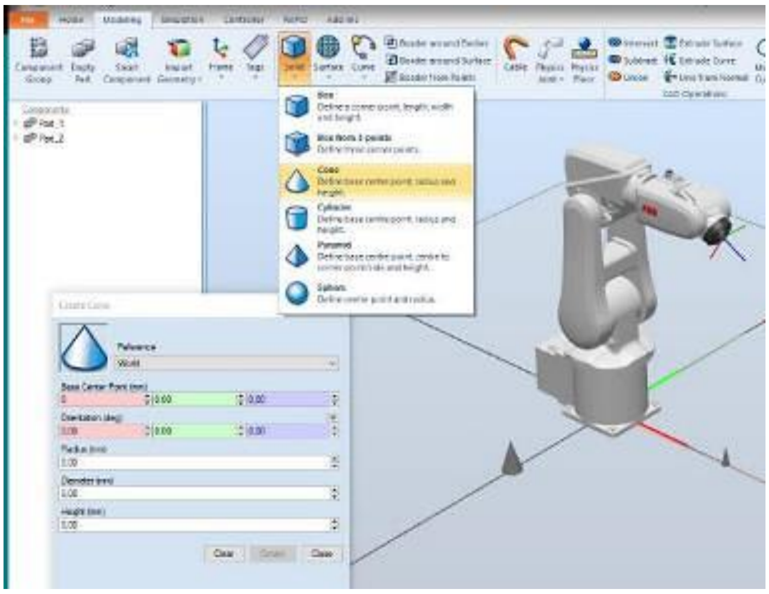
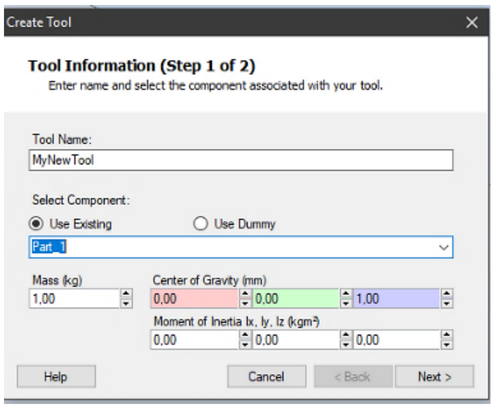


Figure 21. Manual RobotStudio - screenshot 21

It can be found the option "create tool" on the modelling menu and a pop-up window, on Figure 22, will be shown. In the first page of the window, it can be selected the name and the component that will be the new tool (choosing the name that the new solid has). It can also choose the centre of gravity or the mass of the object. The second page is for configured the TCP position of the new tool. The TCP (Tool Center Point coordinate system) is the point of the tool that it will be connected with the sharp point of the robot.



Create Tool

Tool Information (Step 1 of 2)

Enter name and select the component associated with your tool.

Tool Name: MyNewTool

Select Component: (●) Use Existing ( ) Use Dummy

Part_1

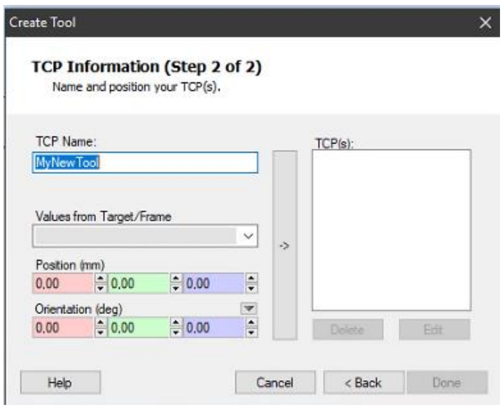
Mass (kg): 1,00

Center of Gravity (mm): 0,00 0,00 1,00

Moment of Inertia Ix, Iy, Iz (kgm²): 0,00 0,00 0,00

Help | Cancel | < Back | Next >

Figure 22. Manual RobotStudio - screenshot 22



Create Tool

TCP Information (Step 2 of 2)

Name and position your TCP(s).

TCP Name: MyNewTool

Values from Target/Frame

Position (mm): 0,00 0,00 0,00

Orientation (deg): 0,00 0,00 0,00

TCP(s):

-> | Delete | Edit

Help | Cancel | < Back | Done

Figure 23. Manual RobotStudio - screenshot 23

[2] Creating reference point on users' choice.

Once the new tool is created, it can be dragged to the robots' name and the tool will take its place on the robots' sharp point.

9.5. Creation targets and path

A path is the combination of different positions, targets, programmed. RobotStudio has two options for creating targets on the station. The first choice is on the menu “Home” on the submenu “Target”; there is the option to create a new position. The window shown in Figure 25 will appear and it will be able to insert the new target coordinates and add it on the program.

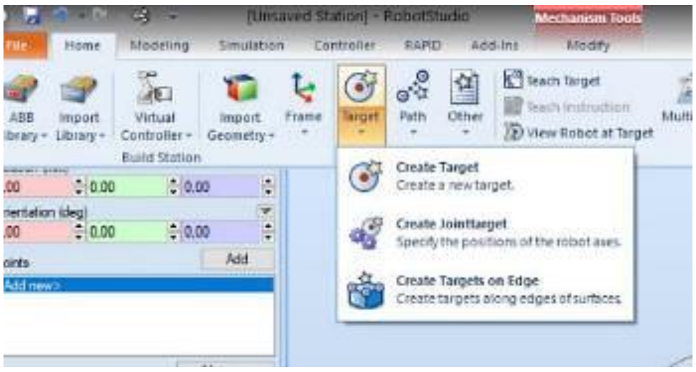
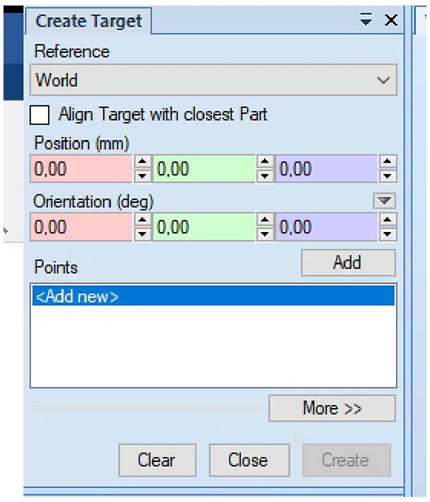


Figure 24. Manual RobotStudio - screenshot 24


Create Target

Reference

World

Align Target with closest Part

Position (mm)

0,00 | 0,00 | 0,00

Orientation (deg)

0,00 | 0,00 | 0,00

Points

<Add new>

Figure 25. Manual RobotStudio - screenshot 25

The second choice to create a new target is using the function “freehand” previously mentioned. The robot will be moved and then the new position of the sharp point will be saved as a new target. Once the robot is shifted, on the menu “Home” there is the option of “Teach Target”. Automatically, pressing this option the new position will be saved.

All the targets will be saved and mentioned on the list “Paths&Targets” found on the left of the interface. The names of each are inserted by the program but it is possible to change it with just clicking with the right bottom of the mouse over the target’s name. The different positions are represented on the station by a small coordinate system. Meanwhile, one target is selected on the list, on the station the same coordinate system related will be highlighted with white colour.

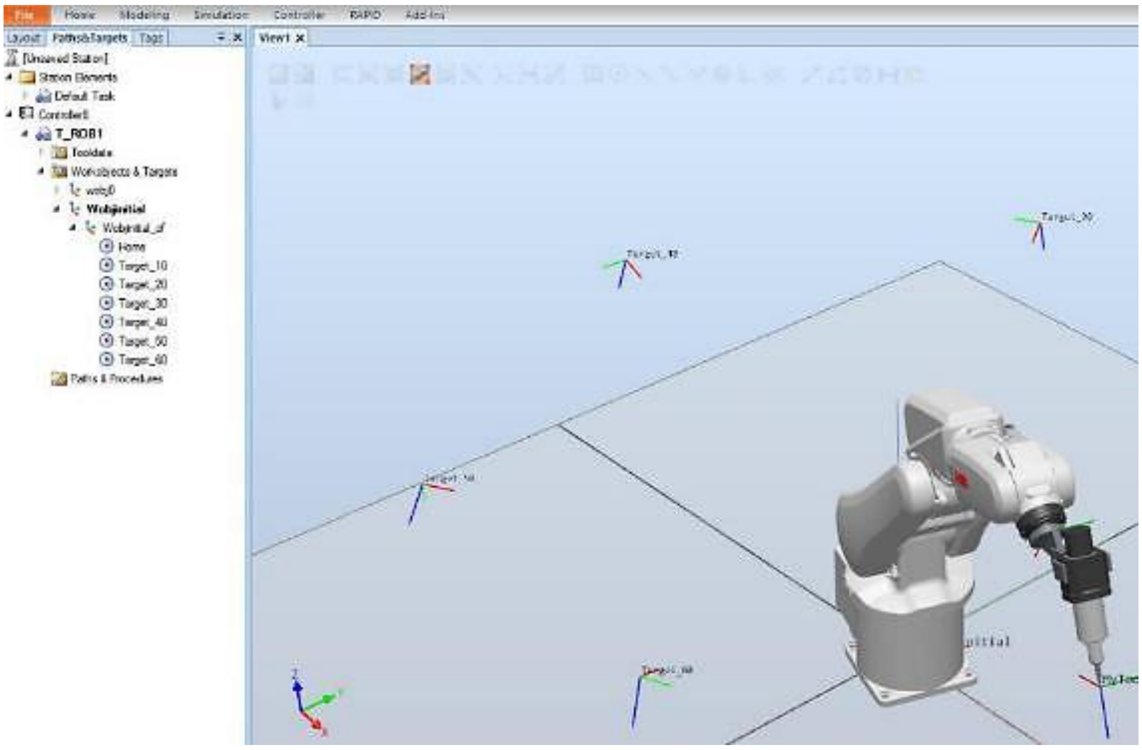


Figure 26. Manual RobotStudio - screenshot 26

The targets can be edited on any time with the list options pressing the right bottom of the mouse over the position’s name. It can be changed the position, the work object which has the reference, the name, etc. There is also the option for seeing the robot on the position of the target. Sometimes, with the creation of a new target, on its name appears a warning. That means that the configuration of the position has been changed manually and it can be that the robot does not have access. To solve this warning, on the same list, there is the option of “configurations”, pressing that different coordinates will be shown and the user can select their choice. The options are new robot’s positions and sometimes the difference between them is minimum.

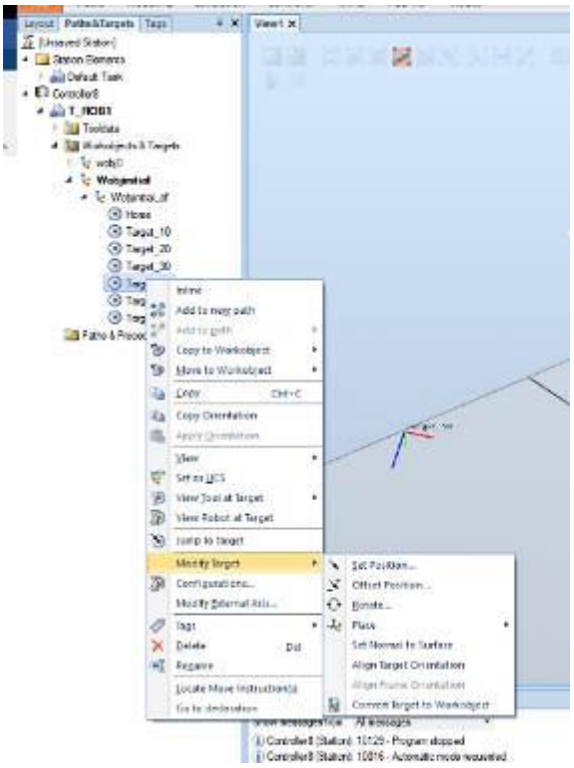


Figure 27. Manual RobotStudio - screenshot 27

Once the targets are created and configurated, a new path can be created. On the list of the previous Figure 27, there is the option of “Add to new path”. The user can select all the targets and then press this option or they can press this option with just one target and add the others later. There is also the option to create the path from the menu “Home”.

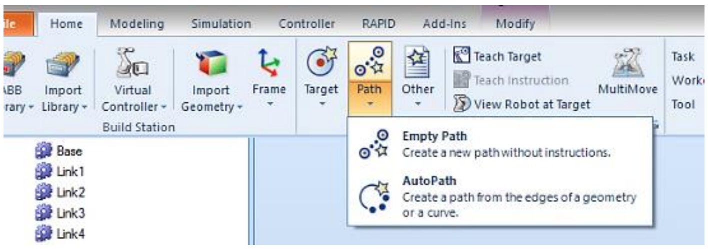


Figure 28. Manual RobotStudio - screenshot 28

The path will be created and the targets can be added just dragging the mouse, from the targets list, or deleted using the list shown pressing the right bottom of the mouse. The user can decide the order of the targets’ execution, there is even the possibility to repeat the same target in two different positions on the paths’ list. For example, if the user wants to start and finish the path on the same target, they can put twice the position on the list. Besides, on the station, all the targets will be joined in the correct order by a line which symbolizes the path. This line can be invisible on the menu pressing over the name of the path with the right bottom of the mouse.

Once the path is created, the user must prove that the robot can attach all the targets with its tool. Pressing on the name of each target with the right bottom of the mouse and going to the option “View tool at target”, the tool will be shown on each target. It is important to change the orientation of the different targets for a better attachment of the robot. If the different targets have the same orientation, the user can copy the orientation and apply it for all the targets.

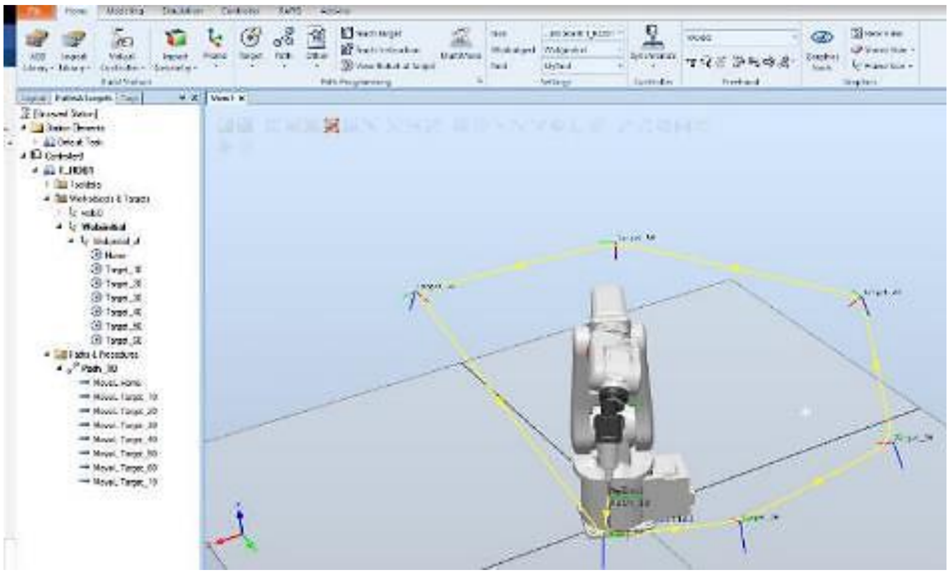
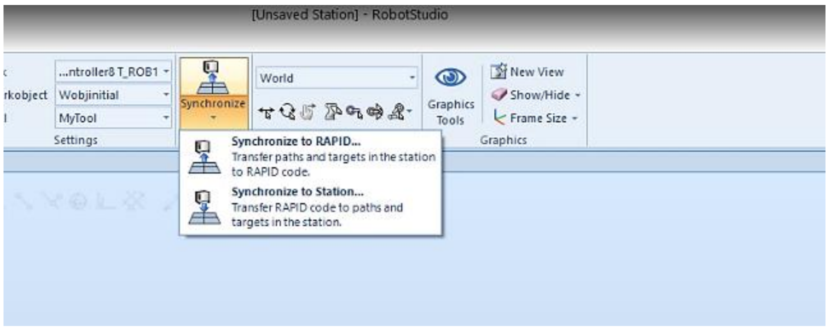


Figure 29. Manual RobotStudio - screenshot 29

On the same menu with the option creation a path there is also a creation of an autopath (see Figure 28). Pressing this function, the software will draw a path automatically around an object chosen by the user.

9.6. Simulation

Once the path is created, it can simulate by the software. The first step to start a simulation is to synchronize the station with the virtual controller pressing the option “Synchronize” on the menu “Home”. On the drop-down menu, it will be chosen the first option from it.



Synchronize to RAPID...
Transfer paths and targets in the station to RAPID code.

Synchronize to Station...
Transfer RAPID code to paths and targets in the station.

Figure 30. Manual RobotStudio - screenshot 30

Pressing this option, a new window will appear with the simulation settings. The user will choose what they want to synchronize from a list. Once that, from the menu “Simulation” it can run and configure different settings as the TCP Trace. If the user wants to paint the path while the robot is running simultaneous, the TCP tool can be used. It can be found on the simulation menu on the monitor section. A new window will appear to configure it choosing the colour, for example.

When the software has to simulate, it will look at the RAPID code on the instructions inside “main” which is the starting point. Without editing the code and if it is just one path in the whole simulation, the user can define the path as the starting point pressing over its name with the right bottom of the mouse and selecting the option on the menu. The other option is to write the name of the path on the main process as shown on the code below. If the simulation has more than one path, then the user will write all the paths’ name in the correct order, even repeating them, on the main process on the code.

```
 PROC main()
     Path_10;
   ENDPROC
PROC Path_10()
   MoveL Target_10,v1000,z100,MyTool\WObj:=Workobject_1;
   MoveL Target_20,v1000,z100,MyTool\WObj:=Workobject_1;
   MoveL Target_30,v1000,z100,MyTool\WObj:=Workobject_1;
   MoveL Target_40,v1000,z100,MyTool\WObj:=Workobject_1;
   MoveL Target_10,v1000,z100,MyTool\WObj:=Workobject_1;
ENDPROC
```

As shown in the previous square, each target has its line of code with its properties. The first one is the type of move, in this case, it is linear (MoveL). RobotStudio has three different movements: MoveL, MoveC (circular movement) and MoveJ (joint movement). The second property is the name of the position. Following, it can be found the speed of movement and accuracy on the target. The next property is the name of the tool that the robot is carrying and on the end of the code line, there is the name of the work object where the target belongs to.

These properties can be managed before creating the target or after. If the user wants to configure a property before creating the target (or more than one with the same property), on the bottom of the interface it can be changed (see Figure 31).

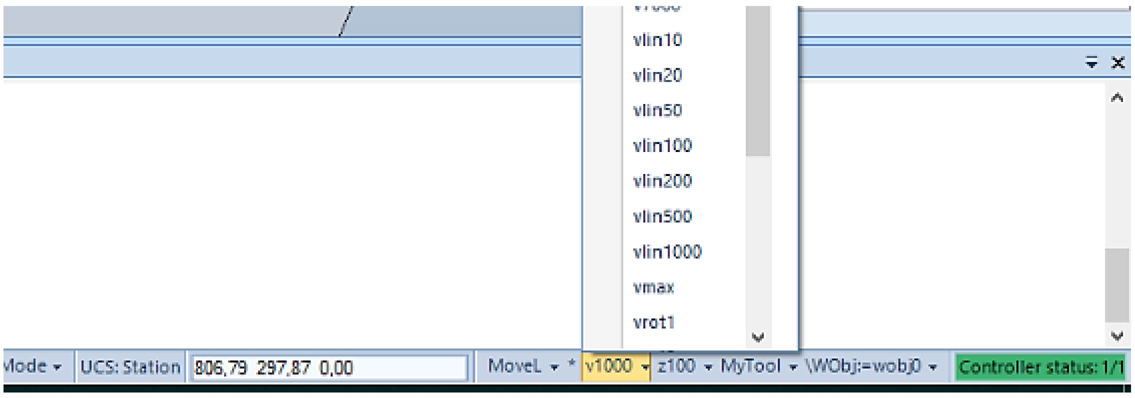
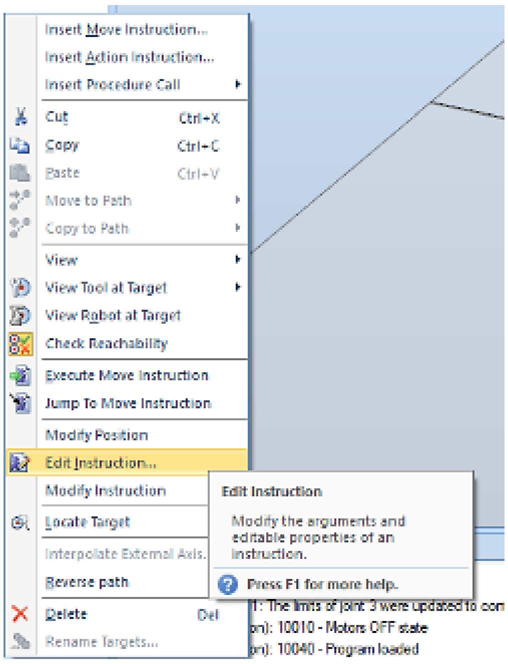


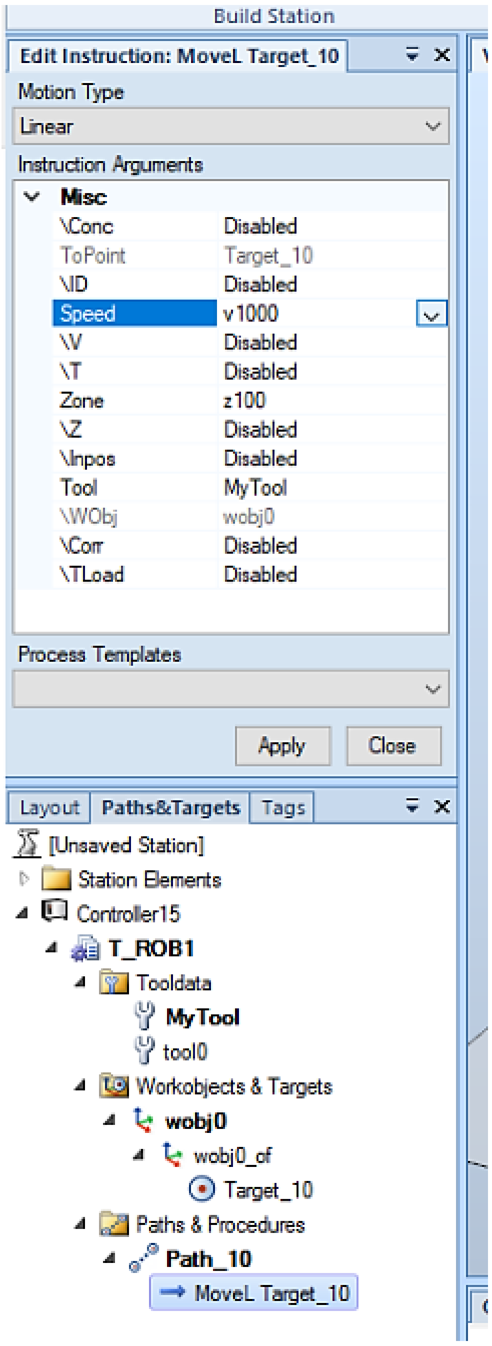
Figure 31. Manual RobotStudio - screenshot 31

Then all the targets created will have the same properties. If the user wants to modify the instruction after creating a target, it can be done in two ways: on the RAPID code (deleting the property and writing it again) or pressing the name of the target on the path and edit it (see Figures 32 and 33).



1: The limits of joint 3 were updated to con
on): 10010 - Motors OFF state
on): 10040 - Program loaded

Figure 32. Manual RobotStudio - screenshot 32



Motion Type: Linear

\Conc	Disabled
ToPoint	Target_10
\ID	Disabled
Speed	v1000
\V	Disabled
\T	Disabled
Zone	z100
\Z	Disabled
\Inpos	Disabled
Tool	MyTool
\WObj	wobj0
\Corr	Disabled
\TLoad	Disabled

```
[Unsaved Station]
  Station Elements
  Controller15
    T_ROB1
      Tooldata
        MyTool
        tool0
      Workobjects & Targets
        wobj0
          wobj0_of
            Target_10
      Paths & Procedures
        Path_10
          MoveL Target_10
```

Figure 33. Manual RobotStudio - screenshot 33

9.7. Work Objects

All the targets created will have its reference on one Work Object which is one coordinate system. When the user creates a station and inserts a robot, the “wobj0” will be created. This coordinate origin will be on the central point of the station. For a basic station, there is not any problem to work with this coordinate origin but if it is necessary to work with more than one object/robot, then it is recommendable to create a new coordinate origin.

On the menu "home" there is the option for creating a new work object. A new settings window will appear once the user has pressed the first indication. Different properties can be changed as the name or the position of the new coordinate system.

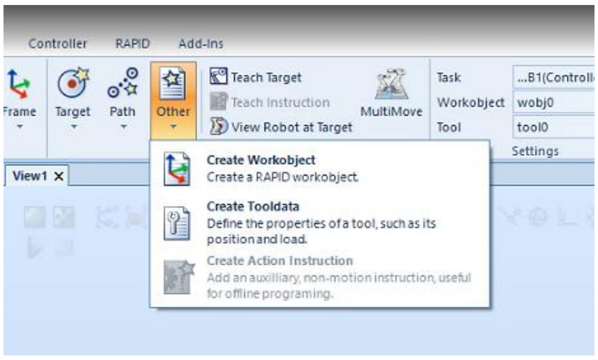
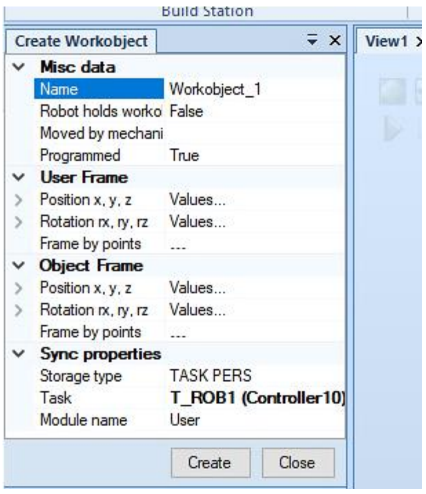

Controller
RAPID
Add-Ins
Frame
Target
Path
Other
Teach Target
Teach Instruction
View Robot at Target
MultiMove
Task
Workobject
Tool
...B1(Controll
wobj0
tool0
Settings
View1
Create Workobject
Create a RAPID workobject.
Create Tooldata
Define the properties of a tool, such as its position and load.
Create Action Instruction
Add an auxilliary, non-motion instruction, useful for offline programing.

Figure 34. Manual RobotStudio - screenshot 34



Misc data	
Name	Workobject_1
Robot holds worko	False
Moved by mechani	
Programmed	True
User Frame	
Position x, y, z	Values...
Rotation rx, ry, rz	Values...
Frame by points	...
Object Frame	
Position x, y, z	Values...
Rotation rx, ry, rz	Values...
Frame by points	...
Sync properties	
Storage type	TASK PERS
Task	T_ROB1 (Controller10)
Module name	User

Figure 35. Manual RobotStudio - screenshot 35

If the user creates new targets with the new work object previously created, it is possible to move all the targets moving the work object. It is the example of different targets created over an object and later the solid movement is needed. Once the work object is created, it will appear on the left list of the interface. The targets that the user wants to create in reference to the new work object can be added in two different ways. The first one is changing the station reference on the "Home" menu and then create the target and the other one is moving a target already created from one work object to another. Under the work object, another list with all its targets will be showed (see Figure 37).

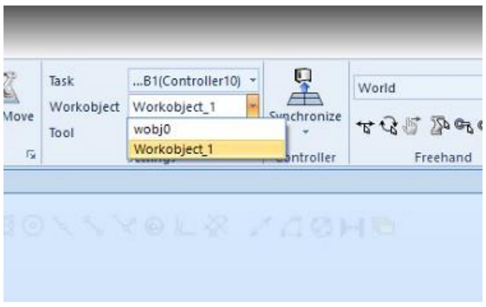


Figure 36. Manual RobotStudio - screenshot 36

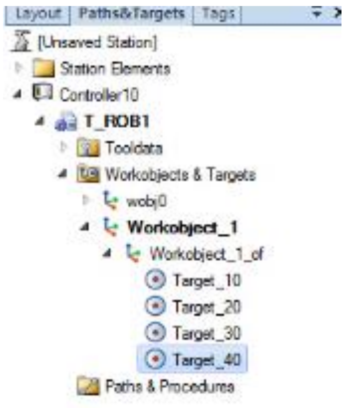


Figure 37. Manual RobotStudio - screenshot 37

9.8. Coordinate System

RobotStudio has different coordinate systems. All the coordinate systems will be represented in the same form. The different axes have a colour. Red for axis X, green for axis Y and blue for axis Z.

- World coordinate system. This system is the most important one and it is located on the central point of the station. When the user creates a station and starts to add robots, objects and tools on it, everything will be showed on the world coordinate system. Besides, when the user creates a station, four of the different coordinate system (Base frame, Task frame, world and work object) will be located on the same point: the centre of the station.
- TCP (Tool Center Point) coordinate system. This system is located on the sharp point of the robot or the tool (when this is attached to the robot).
- Work object. It is the coordinate system of the objects and the targets.
- Base frame. It is located on the base of the robot.
- Task frame. This coordinate system represents the origin of the robot controller world coordinate system in RobotStudio.

The base Frame and the Task frame are usually located on the same point and working together. However, if the user is working with more than one robot with multimove independent between them, the task frame allows working each robot in a different coordinate system.

When the user wants to set the position of a robot, object… it will be always possible to set the coordinate reference system of the new position.

9.9. Creation of a mechanism

The RobotStudio software has the possibility to create a new mechanism by the user, it would be a graphical representation of a robot, a tool, an external axis or a device. The new mechanism will move along or around axes. Once the user has created the different solids which will include the mechanism, a new window will appear pressing the option “Create mechanism” on the menu “Home”. The user has to configure the links, joints, frames/tools and calibration, and then compilable it. A more detailed explanation of how to create a mechanism can be found on the second simulation.

9.10. Configuration of Inputs and outputs.

The user can create different inputs and outputs on the controller. Going to the “Controller” menu, inside the section “configuration”, there is the option to configure the inputs and outputs of the system (see Figure 38).

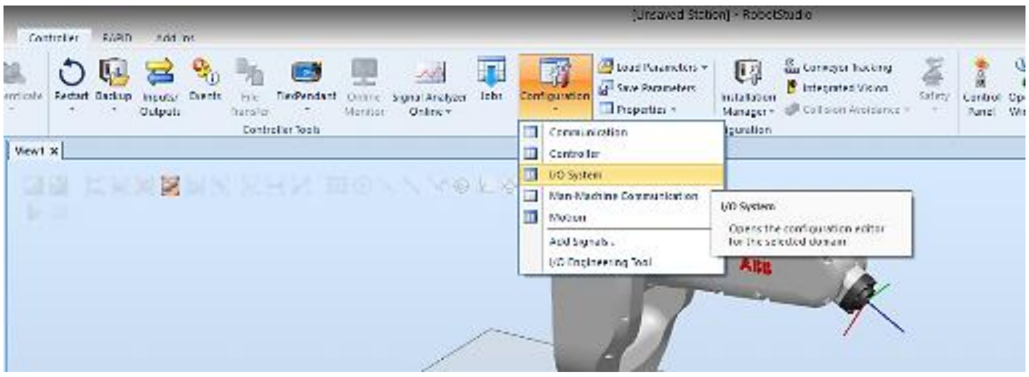
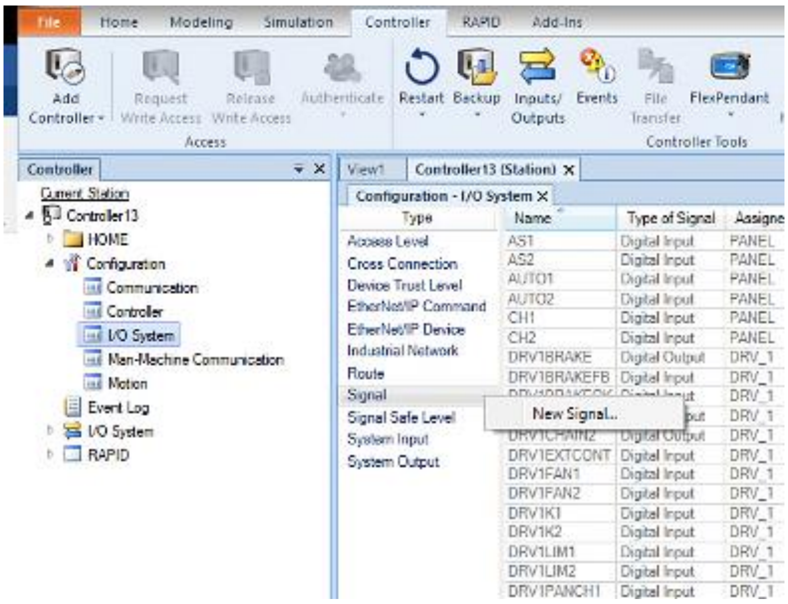


Figure 38. Manual RobotStudio - screenshot 38

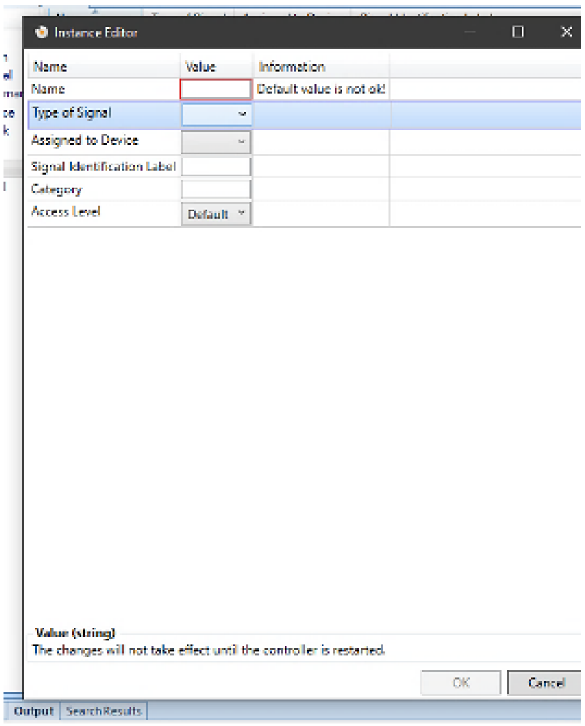
Once the user has click for configuring the inputs and outputs, a new window will appear. For create a new signal, it is enough with pressing over the name with the right bottom of the mouse and click “new signal”.



Type	Name	Type of Signal	Assigne
Access Level	AS1	Digital Input	PANEL
Cross Connection	AS2	Digital Input	PANEL
Device Trust Level	AUTO1	Digital Input	PANEL
EtherNet/IP Command	AUTO2	Digital Input	PANEL
EtherNet/IP Device	CH1	Digital Input	PANEL
Industrial Network	CH2	Digital Input	PANEL
Route	DRV1BRAKE	Digital Output	DRV_1
Signal	DRV1BRAKEFB	Digital Input	DRV_1
Signal Safe Level	[illegible]	Digital Input	DRV_1
System Input	[illegible]	[illegible]	DRV_1
System Output	DRV1CHAIN2	Digital Output	DRV_1
	DRV1EXTCONT	Digital Input	DRV_1
	DRV1FAN1	Digital Input	DRV_1
	DRV1FAN2	Digital Input	DRV_1
	DRV1K1	Digital Input	DRV_1
	DRV1K2	Digital Input	DRV_1
	DRV1LIM1	Digital Input	DRV_1
	DRV1LIM2	Digital Input	DRV_1
	DRV1PANCH1	Digital Input	DRV_1

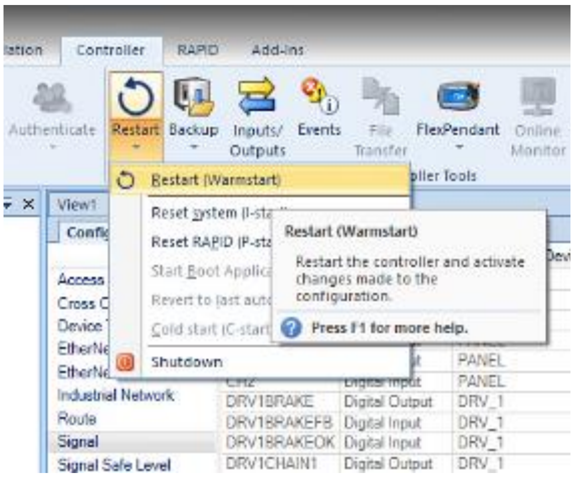
Figure 39. Manual RobotStudio - screenshot 39

A new pop-up window will be shown (see Figure 40) by the program and the user can complete the different parameters of the new signal. Once all the signals needed are created, to apply the changes it is needed to restart the virtual controller.



Name	Value	Information
Name		Default value is not ok!
Type of Signal		
Assigned to Device		
Signal Identification Label		
Category		
Access Level	Default	

Value (string)
The changes will not take effect until the controller is restarted.



Restart (Warmstart)

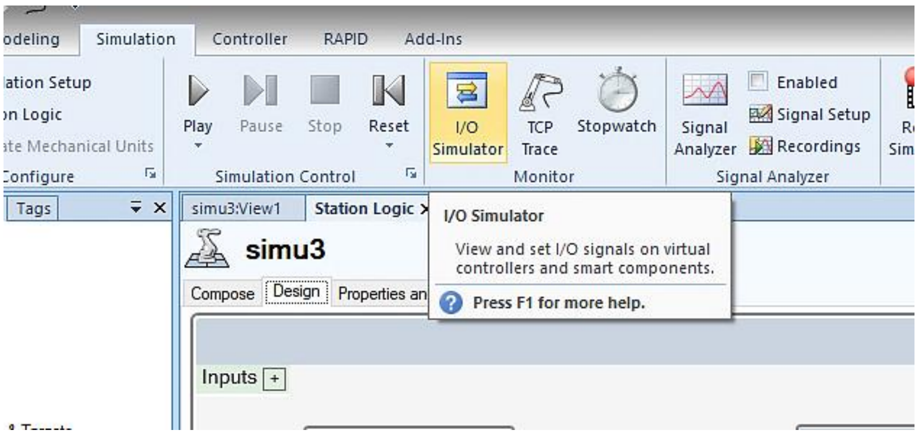
Restart the controller and activate changes made to the configuration.

Press F1 for more help.

Figure 40. Manual RobotStudio - screenshot 40

Figure 41. Manual RobotStudio - screenshot 41

The signals are already created and the user can program them to interact with the smart components and the whole simulation. It is possible to use a simulator of the signals during the simulation. The user would be able to active and inactive the different signals from the simulator window. For better use, it is possible to create a user list with just the signals created by the user.

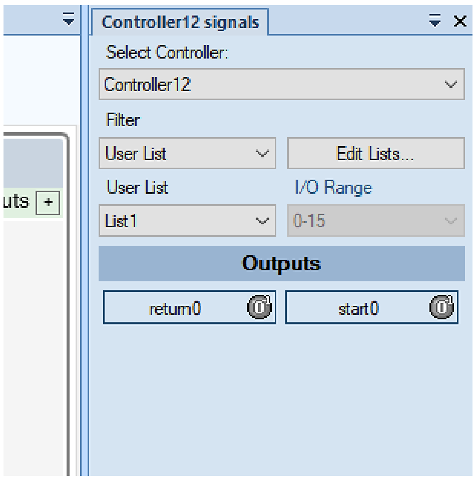


I/O Simulator

View and set I/O signals on virtual controllers and smart components.

Press F1 for more help.

Figure 42. Manual RobotStudio - screenshot 42



Controller12 signals

Select Controller: Controller12

Filter: User List | Edit Lists...

User List: List1 | I/O Range: 0-15

Outputs

return0 | start0

Figure 43. Manual RobotStudio - screenshot 43

For a better understanding of how to connect the signals with the simulation, the simulation number three uses these connections.

10. Simulation examples

10.1. Pick and place

The first simulation will be a simple case of Pick and Place. The robot will move the object from its initial position to a new one. To obtain this simulation the following steps have been followed:

1. Create an empty station on Robot Studio.
2. Insert the robot chosen by the user.
3. Insert the tool chosen by the user and attach it to the robot.
4. Configure the virtual controller.
5. Insert solids needed.

In this case, it is needed three solids for the actual simulation. The user can define the shape of each solid. Two of the solids will be equal, with the same form, but in different positions. The third object would be used as a base. On the current example, three boxes have been created. The user can also define different solid's properties, like the colour, pressing the right bottom of the mouse over the solid's name.

Table 2. Positions of the solids - Pick and Place

Solid's name	**Coordinates corner point (x,y,z) [mm]**	**Size (length, width, height) [mm]**
Solid1	0, 400, 0	200, 200, 150
Solid2	0, -600, 50	200, 200, 150
Base	-100, -700, 0	400, 400, 50

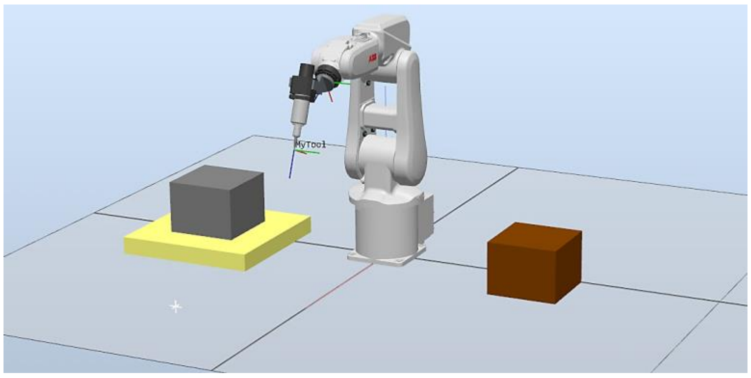


Figure 44. Pick and place - screenshot 1

6. Creation of targets.

The robot needs to follow a path with targets, so it is needed to create them. In this case, it has been created the following targets:

Table 3. Positions of the targets - Pick and place

Target's name	**Position (x,y,z) [mm]**	**Comments**
Home	-	Initial robot's sharp point with the tool.
Targsolid1	100, 500, 150	Target created over solid 1 on the centre.
Targsolid11	100, 500, 170	Target over the previous target on the solid 1. This target will be used for better visualization of the simulation. The robot will go to pick up the object in a vertical path.
Targsolid2	100, -500, 200	Target created over solid 2 on the centre.
Targsolid22	100, -500, 220	Target over the previous target on the solid 2.

Different targets for better visualization of the path can also be added. The program will define the path between two targets as a linear, but the user can add more targets creating a circular movement between two targets. In this case, four additional targets have been created. to create a circular movement between the two solids. In Figure 44 the representation of the different targets can be seen.

It is important to use one software tool during the creation of targets. The user can press on the name of the target with the right bottom of the mouse and press the option of "View Tool at Target" on the menu. That way, it can be seen the tool positioned on each target and change the orientation pressing in the same menu "Modify Target" and "Set Position". It is important to orientate the tool having as reference the robot and a good orientation, if not the robot will not be able to reach the position.

7. Create an empty path.

On the menu "Home" there is the option for creating a new path. Once the user has created the path, they will be able to insert all the targets in the correct order, repeating targets if it is necessary. The path will be shown on the station by a yellow route. The targets are also represented on the station by a small coordinate system with its names.

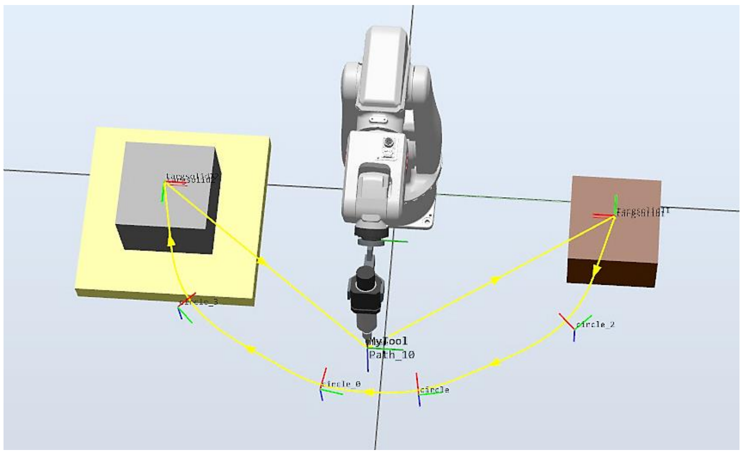


Figure 45. Pick and place - screenshot 2

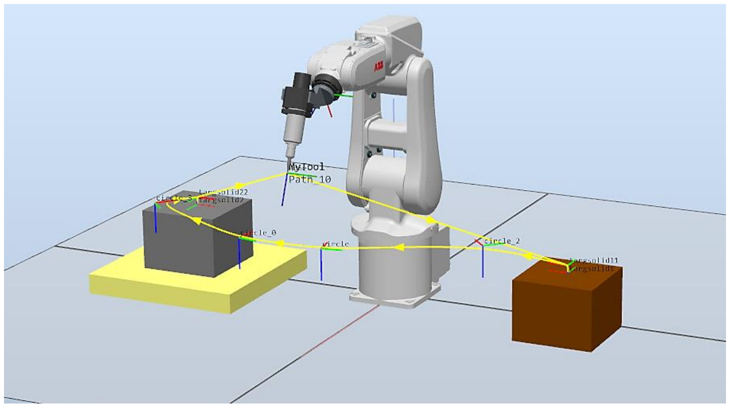


Figure 46. Pick and place - screenshot 3

The user can also define the different configuration of the path, as define the route as a “starting point” for start the simulation and change the properties of the different targets as the zone or the speed.

On the RAPID code, it can be seen the order of the targets on the path with its properties (the type of movement, name of the target, speed, zone, tool and work object used as reference).

```
PROC Path_10()
    MoveL Home,v1000,z100,MyTool\WObj:=wobj0;
    MoveL targsolid11,v1000,z100,MyTool\WObj:=wobj0;
    MoveL targsolid1,v1000,fine,MyTool\WObj:=wobj0;
    MoveL targsolid11,v1000,z100,MyTool\WObj:=wobj0;
    MoveL circle_2,v1000,z200,MyTool\WObj:=wobj0;
    MoveL circle,v1000,z100,MyTool\WObj:=wobj0;
    MoveL circle_0,v1000,z100,MyTool\WObj:=wobj0;
    MoveL circle_3,v1000,z200,MyTool\WObj:=wobj0;
    MoveL targsolid22,v1000,z100,MyTool\WObj:=wobj0;
    MoveL targsolid2,v1000,fine,MyTool\WObj:=wobj0;
    MoveL targsolid22,v1000,z100,MyTool\WObj:=wobj0;
    MoveL Home,v1000,z100,MyTool\WObj:=wobj0;
ENDPROC
```

8. Synchronize to RAPID.
9. Simulation. Station logic and smart components.

Before starting to programme the simulation, the user must prove that the path works correctly simulating it. If the robot does not have any problem to traverse the whole path, then the user can start with the configuration of the "station logic". This option can be found on the "Simulation" menu.

On the Station Logic, the user will be able to add smart components. In this case, it will be added four smart components with the following properties.

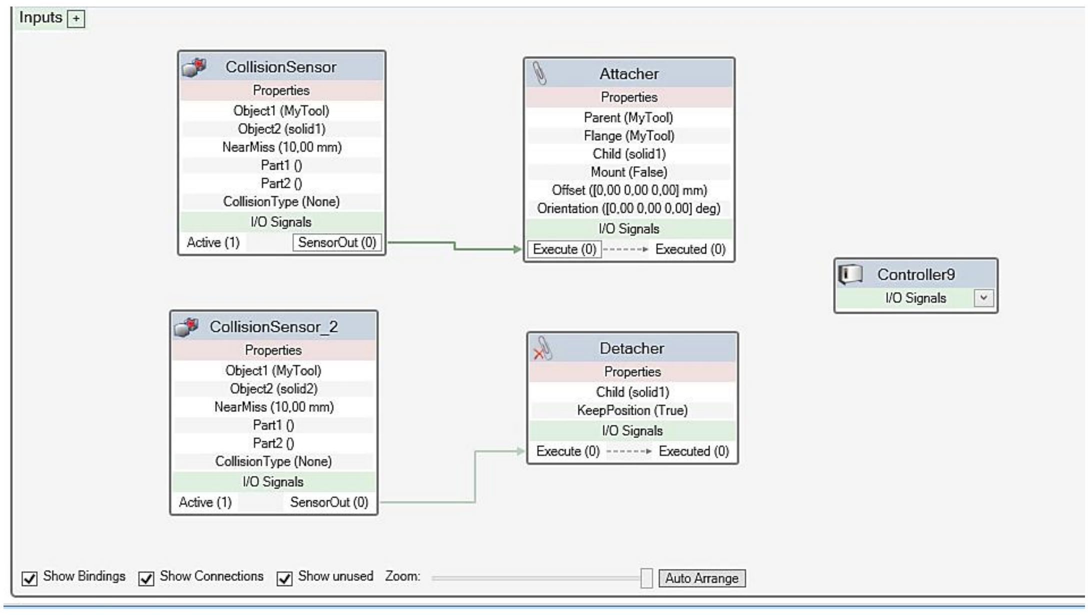
Table 4. Smart components properties - Pick and place

Smart Component name	**Object1**	**Object2**	**NearMiss [mm]**
CollisionSensor	MyTool	Solid1	10
CollisionSensor_2	MyTool	Solid2	10

Smart Component name	**Parent**	**Child**
Attacher	MyTool	Solid1
Detacher	-	Solid1

The collision sensor detects collisions on a specific distant (NearMiss) between the first object and the second object. The attach component is used for adhering the object "child" to another object which is the "parent". The detacher is used for separating the object "child" from its parent.

During the simulation of the robot, the collision sensors will be active in the two moments that the tool is near the objects. On the first activation, the output of the collision sensor will be used for active the attacher and move the box with the robot. In the moment that the collision sensor 2 will detect the tool near the object 2 (invisible during the simulation) will detach the first box on the place of the second one. The effect will then be that the robot takes an object and leaves it somewhere else in the station. On the window "design" inside the station logic, the user can configure the different outputs and inputs easily with connections between the different boxes observed in Figure 47.


Inputs

CollisionSensor
Properties
Object1 (MyTool)
Object2 (solid1)
NearMiss (10,00 mm)
Part1 ()
Part2 ()
CollisionType (None)
I/O Signals
Active (1) SensorOut (0)

Attacher
Properties
Parent (MyTool)
Flange (MyTool)
Child (solid1)
Mount (False)
Offset ([0,00 0,00 0,00] mm)
Orientation ([0,00 0,00 0,00] deg)
I/O Signals
Execute (0) Executed (0)

Controller9
I/O Signals

CollisionSensor_2
Properties
Object1 (MyTool)
Object2 (solid2)
NearMiss (10,00 mm)
Part1 ()
Part2 ()
CollisionType (None)
I/O Signals
Active (1) SensorOut (0)

Detacher
Properties
Child (solid1)
KeepPosition (True)
I/O Signals
Execute (0) Executed (0)

☑ Show Bindings ☑ Show Connections ☑ Show unused Zoom: Auto Arrange

Figure 47. Pick and place - screenshot 4

Once all the connections are done, pressing the play on the simulation, the robot will already pick the object and let it in the other location. However, it is important to have a reset bottom of the simulation. On the simulation control menu, there is a reset bottom but the user must configure an initial state on the option "Save current state", before starting the simulation. If not, the program will not remember where was the initial position of the first object that it had just moved. As mentioned before, the user will also configure the properties of the second object putting it invisible. Optionally, the path drawn can be also invisible; all these options can be found pressing the right bottom of the mouse over each name on the left list.

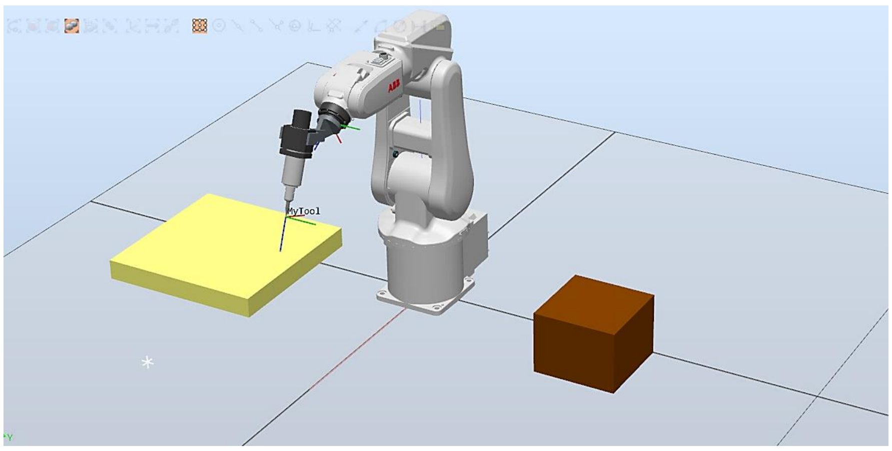


Figure 48. Pick and place - screenshot 5

Pressing the play on the simulation control, the program will work creating the effect of "Pick and Place". Pressing the reset bottom, the simulation will go to its initial place waiting for the next play.

Some images of the simulation:

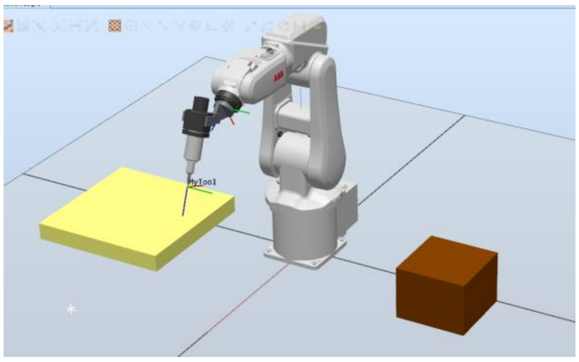


Figure 49. Pick and Place 1

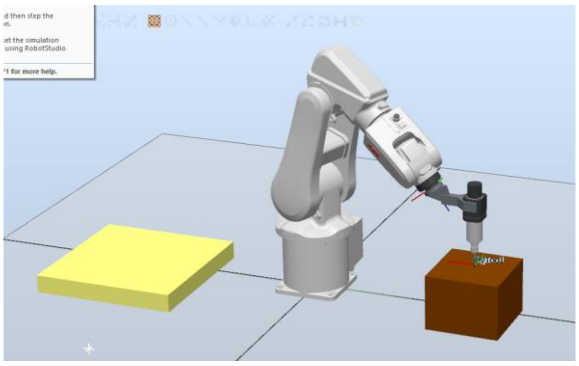


Figure 50. Pick and Place 2

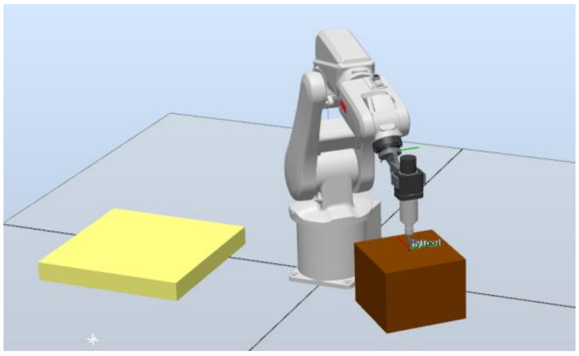


Figure 51. Pick and Place 3

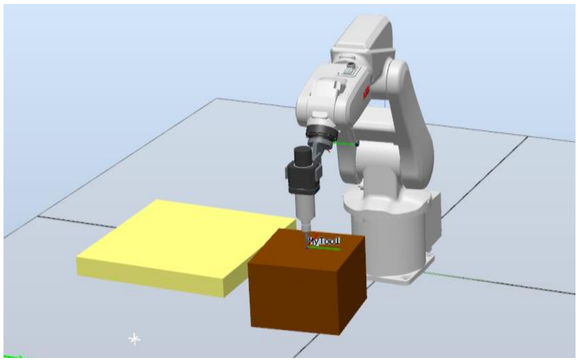


Figure 52. Pick and Place 4

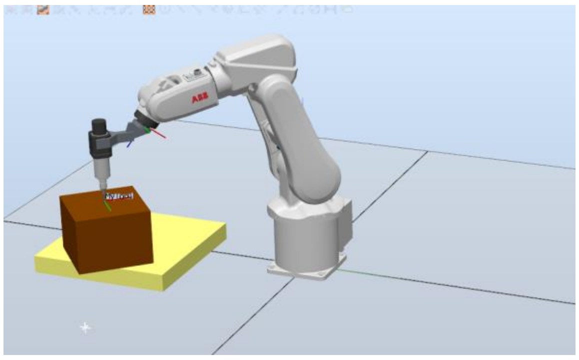


Figure 53. Pick and Place 5

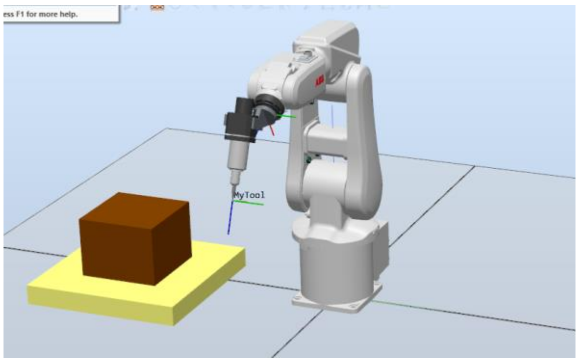


Figure 54. Pick and Place 6

Video of the simulation can be seen here:

https://drive.google.com/drive/folders/1VTs0rX9TSME7J_K1DgqRFnVHcndNq93c?usp=sharing

10.2. Mechanism creation

The second simulation will have as aim the creation of a mechanism and connect it with a robot on the station. When the robot will arrive in one target of its path, the mechanism will start moving. This simulation maybe does not have any real application outside the program. The following steps have been followed:

1. Create solids

The first step is to create the different solids that will compose the mechanism. In this case, it will be just three boxes (two fixes on their places and the other one will have the movement up and down).

Table 5. Positions of the solids - Mechanism creation

Solid's name	**Coordinates corner point (x,y,z) [mm]**	**Size (length, width, height) [mm]**
Part_1	100, 0, 0	50, 50, 700
Part_2	-150, 0, 0	50, 50, 700
Body	-100, -50, -50	200, 100, 30

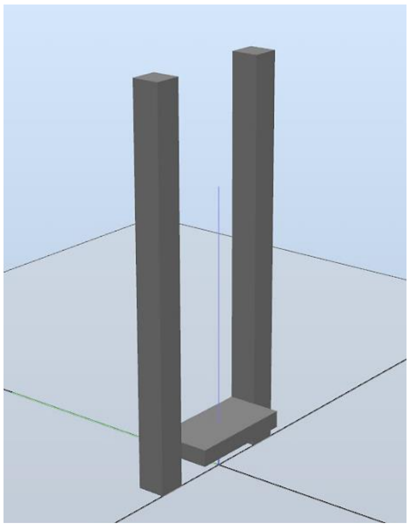
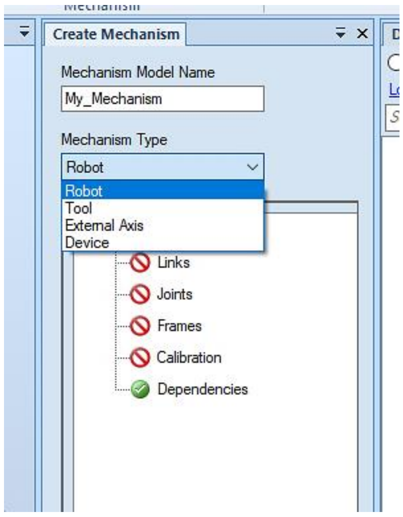


Figure 55. Mechanism creation - screenshot 1

2. Create a mechanism

Once all the solids have been created, on the menu "Modelling" the user can press the option "Create mechanism" located next to "create tool". The window showed in Figure 56 will appear and the user has to introduce the name of the mechanism, the type and start to configure the different Links, Joints and Frames. The type of mechanism, in this case, will be an External Axis. For an external axis, one calibration is required for each joint.



Create Mechanism

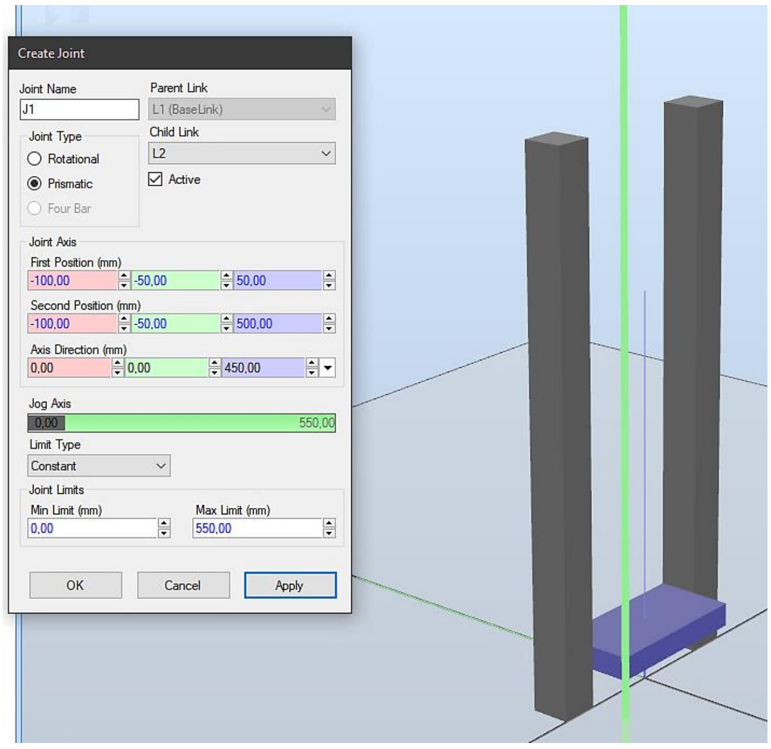
Mechanism Model Name: My_Mechanism

Mechanism Type: Robot

Figure 56. Mechanism creation - screenshot 2

To configure the different nodes, the user must click for example over Links with the right bottom of the mouse and press the option of "Add a link". It is the same with all the nodes. The links are defined as the different parts of the mechanism separated with the joints. In this case, two links can be identified: the Base Link (solids Part_1 and Part_2) which does not have any movement and the Body which is solid with the relative movement on the mechanism.

In this mechanism, just one joint needs to be added: the prismatic movement between the two links. The window on Figure 56 will be modified by the user choosing the movement axis, type, limits… On the station, a green line will appear meanwhile the configuration of the joint representing the direction of the movement.



Create Joint

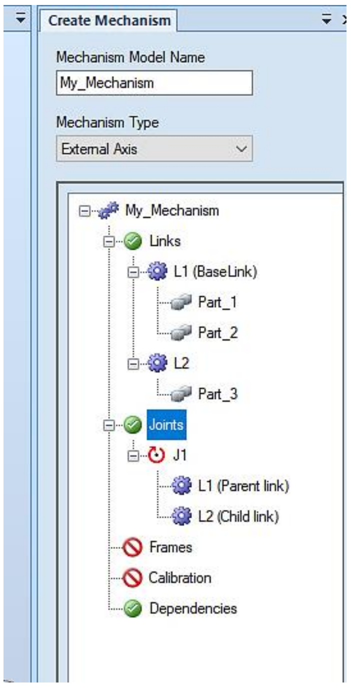
Joint Name: J1
Parent Link: L1 (BaseLink)
Child Link: L2
Active: ✓
Joint Type: Rotational [✓ Prismatic] Four Bar

Joint Axis
First Position (mm): -100,00 | -50,00 | 50,00
Second Position (mm): -100,00 | -50,00 | 500,00
Axis Direction (mm): 0,00 | 0,00 | 450,00

Jog Axis: 0,00 … 550,00
Limit Type: Constant

Joint Limits
Min Limit (mm): 0,00
Max Limit (mm): 550,00

Figure 57. Mechanism creation - screenshot 2



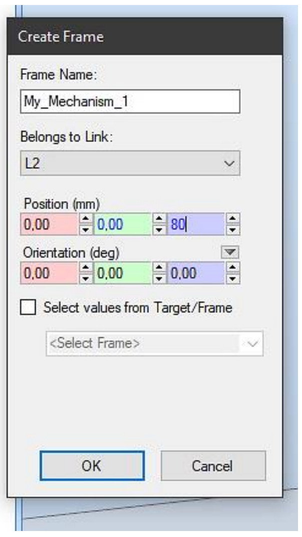
Create Mechanism

Mechanism Model Name: My_Mechanism
Mechanism Type: External Axis

```
My_Mechanism
  Links
    L1 (BaseLink)
      Part_1
      Part_2
    L2
      Part_3
  Joints
    J1
      L1 (Parent link)
      L2 (Child link)
  Frames
  Calibration
  Dependencies
```

Figure 58. Mechanism creation - screenshot 3

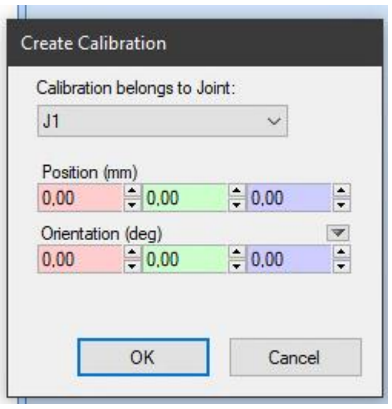
The last step before compiling the mechanism is choosing a frame, which is the reference point of the mechanism, and calibrate the joint (it is just necessary to press "add calibration" and then accept, the user does not need to modify anything in this case). If the mechanism has been created correctly, the compilation will not be a problem and the mechanism will be already created.



Create Frame

Frame Name: My_Mechanism_1
Belongs to Link: L2
Position (mm): 0,00 | 0,00 | 80
Orientation (deg): 0,00 | 0,00 | 0,00
Select values from Target/Frame: ☐
<Select Frame>

Figure 59. Mechanism creation - screenshot 4



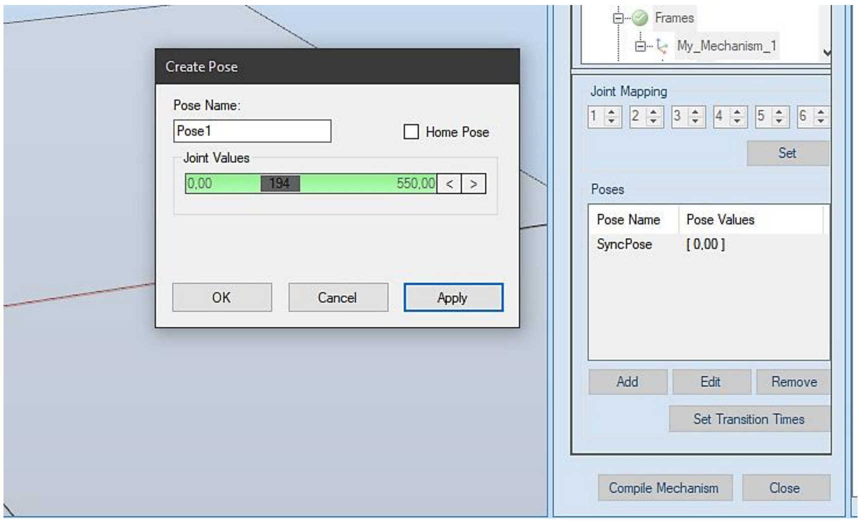
Create Calibration

Calibration belongs to Joint: J1
Position (mm): 0,00 | 0,00 | 0,00
Orientation (deg): 0,00 | 0,00 | 0,00

Figure 60. Mechanism creation - screenshot 5

3. Create poses on the mechanism

Once the mechanism is created, the user can programme different positions between the joint limits of the mechanism and save it. In this case, three positions will be saved (SyncPose, Pose1 and Pose2).


Create Pose

Pose Name: Pose1

☐ Home Pose

Joint Values: 0,00 — 194 — 550,00

Pose Name	Pose Values
SyncPose	[0,00]

Figure 61. Mechanism creation - screenshot 6

4. Insert robot on the station with a tool
5. Configure the virtual controller
6. Create a simple path for the robot
7. Create an invisible solid

As mentioned before, when the robot will be in one of the targets of its path, the mechanism will start the movement. A small box has been created and positioned on one target. The solid which has Part_5 as the name will be invisible during the simulation.

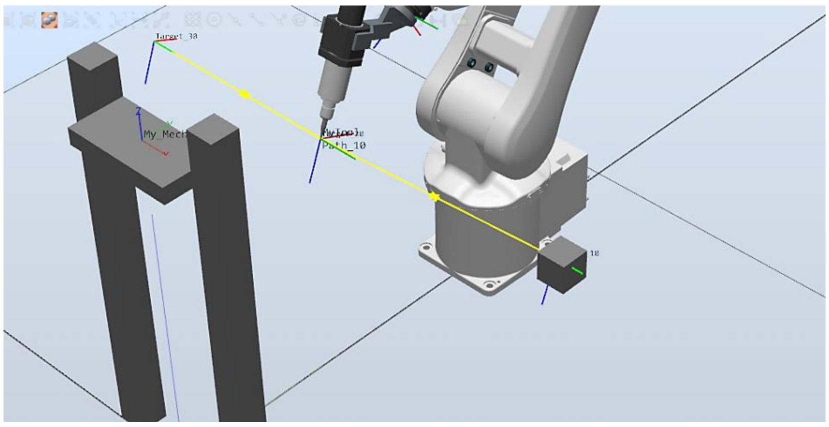


Figure 62. Mechanism creation - screenshot 7

8. Create a smart component and attach the mechanism into it

On the layout list located on the left side of the station, the name of the mechanism should appear under the smart component name.

9. Insert the different smart components

On this simulation, four smart components will be used (three Pose Mover and one Collision Sensor). The Pose Mover has as properties the mechanism (the one that will move), the pose (the position of the mechanism that the smart component will go) and the duration (the time that the mechanism has for arriving in the pose). The collision sensor will be used to detect the moment that the tool is near the small box, invisible, and the program will use the digital output from the sensor to execute the mechanical movement.

Table 6. Smart Components properties - Mechanism creation

Smart Component name	**Object1**	**Object2**	**NearMiss [mm]**
CollisionSensor	MyTool	Part_5	10

Smart Component name	**Mechanism**	**Pose**	**Duration [s]**
PoseMover	My_Mechanism	SyncPose	2
PoseMover_2	My_Mechanism	Pose1	2
PoseMover_3	My_Mechanism	Pose2	2

10. Program the smart components on the logic station

On the window “design” inside the station logic, the user can configure the different outputs-inputs easily with connections between the different boxes which are representing the different smart components (see Figure 63). When the sensor detects the collision, the first Pose Mover will change the mechanism to the programmed position. The three Pose Mover will run one after the other.

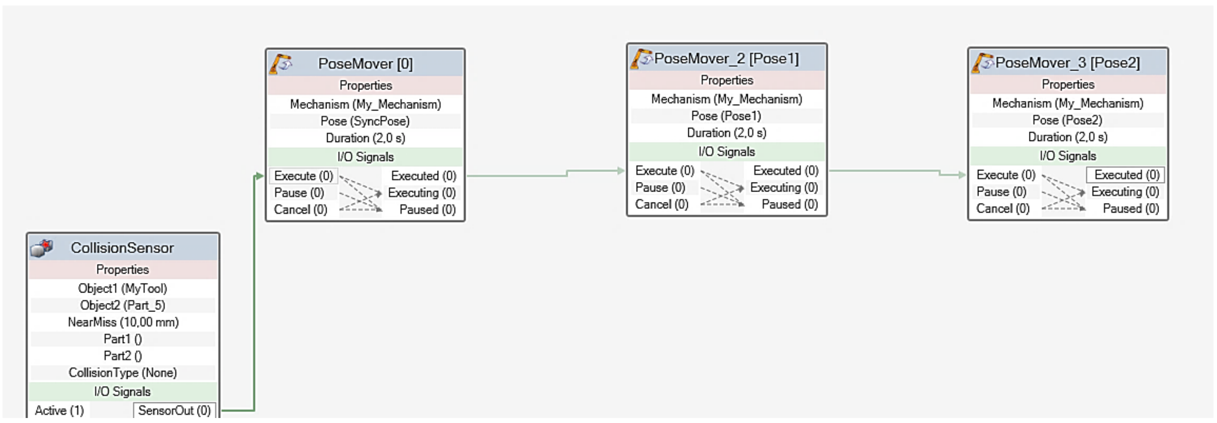


Figure 63. Mechanism creation - screenshot 7

11. Synchronize to RAPID
12. RAPID language

Once the simulation is created, the software will look at the process “main” of the code. The user can choose between set the path as a starting point or write the name of the path inside the “main” on the code. In this case, the code has been modified writing the path’s name and using a new function: “WaitTime”. This function is used to wait time between two instructions, the number written after the function is the seconds.

```
PROC main()
        WaitTime 3;
        Path_10;
ENDPROC
PROC Path_10()
        MoveL Target_20,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_10,v1000,fine,MyTool\WObj:=wobj0;
        MoveL Target_20,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_30,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_20,v1000,z100,MyTool\WObj:=wobj0;
ENDPROC
```

13. Synchronize to Station.
14. Simulate

Images from the simulation:

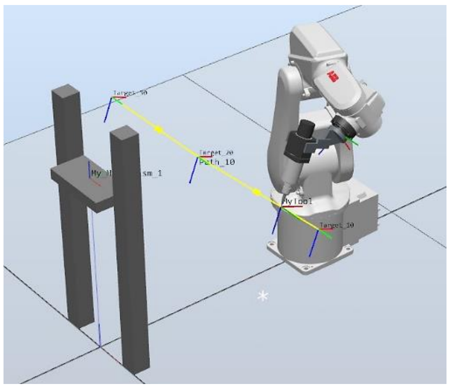


Figure 64. Mechanism creation 1

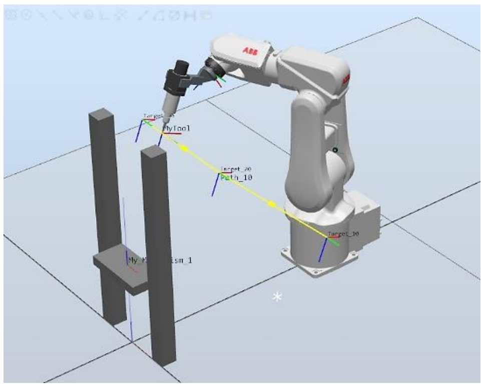


Figure 65. Mechanism creation 2

Access to the video of the simulation on the following link:

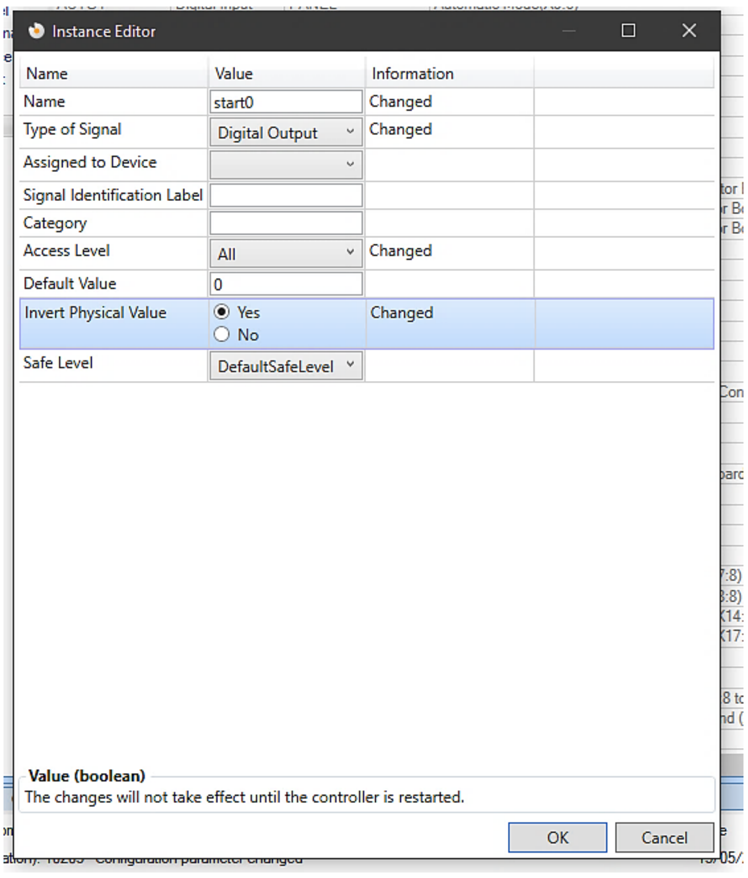
https://drive.google.com/drive/folders/1VTs0rX9TSME7J_K1DgqRFnVHcndNq93c?usp=sharing

10.3. Mechanism interaction

The third simulation has some similarities with the previous one, but now the mechanism will be moved first and the robot will react after. The mechanism will transport an object and the robot will take the solid from the mechanism and change its place. To create the simulation the following steps have been followed:

1. Create an empty station and insert the robot.
2. Configure the virtual controller.
3. Add new signals: start0 and return0

The two signals created on this simulation will be of the type of digital output. On the controller window, it is possible to configure the I/O of the system and create the new signals. The following configuration will be inserted for both signals.



Name	Value	Information
Name	start0	Changed
Type of Signal	Digital Output	Changed
Assigned to Device		
Signal Identification Label		
Category		
Access Level	All	Changed
Default Value	0	
Invert Physical Value	[✓ Yes] No	Changed
Safe Level	DefaultSafeLevel	

Value (boolean)
The changes will not take effect until the controller is restarted.

Figure 66. Mechanism interaction - screenshot 1

4. Restart controller (Warmstart).

Once the signals are created, it is important to restart the controller to apply the changes.

5. Create solids. Create a mechanism.

The user will create different solids which will compose the mechanism. They consist of one base and one solid which will have movement.

Table 7. Positions of the solids - Mechanism interaction

Solid's name	**Size (length, width, height) [mm]**	**Corner point (x, y, z) [mm]**	**Orientation (x,y,z) [degrees]**
Base	1500 x 100 x 300	1000, -1000, 0	0, 0, 90
Solid1	500 x 300 x 100	500, -1000, 100	0, 0, 0

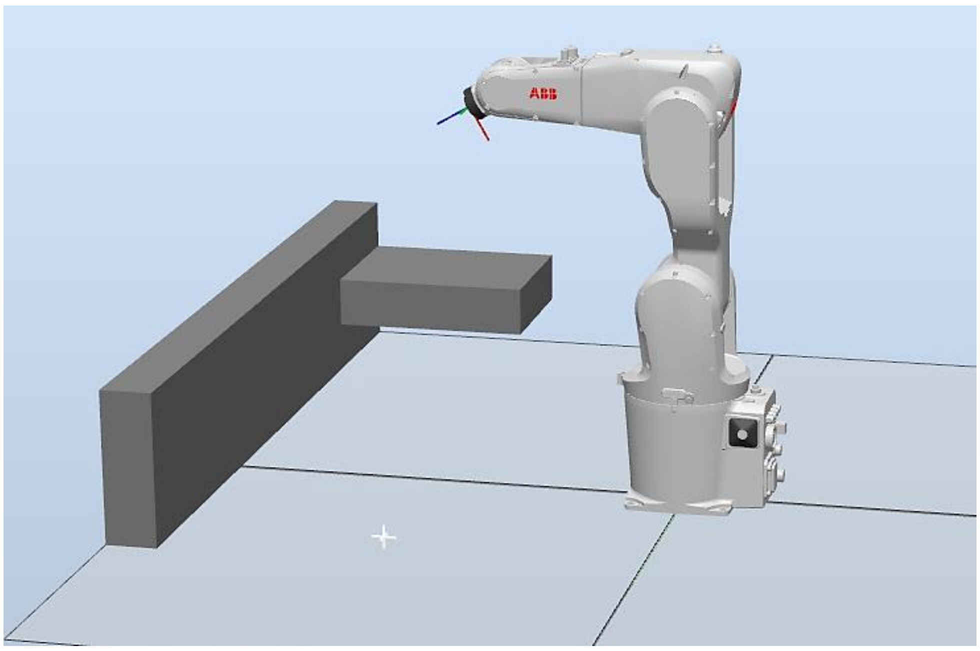
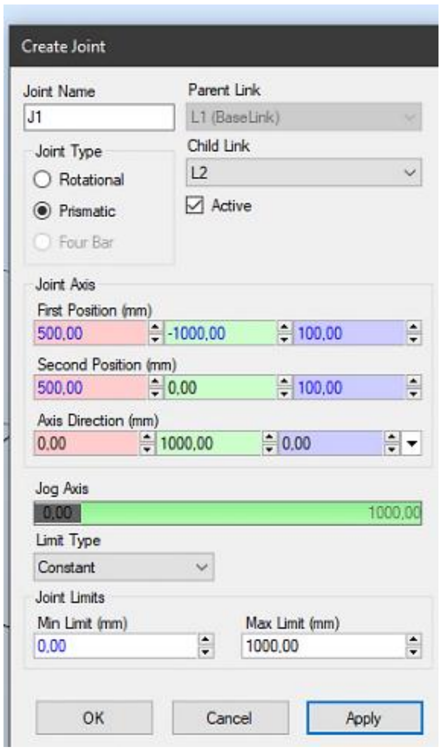


Figure 67. Mechanism interaction - screenshot 2

Once the solids are created, the user can create the mechanism adding the links, joints and frames. There will be just two links, one for each solid, and the "Base" solid will be the base-link. The joint will be the prismatic movement between the two links and the frame will be located over the Solid1.



Create Joint

Joint Name: J1

Parent Link: L1 (BaseLink)

Child Link: L2

Joint Type: Rotational [✓ Prismatic] Four Bar

[✓ Active]

Joint Axis

First Position (mm): 500,00 | -1000,00 | 100,00

Second Position (mm): 500,00 | 0,00 | 100,00

Axis Direction (mm): 0,00 | 1000,00 | 0,00

Jog Axis: 0,00 – 1000,00

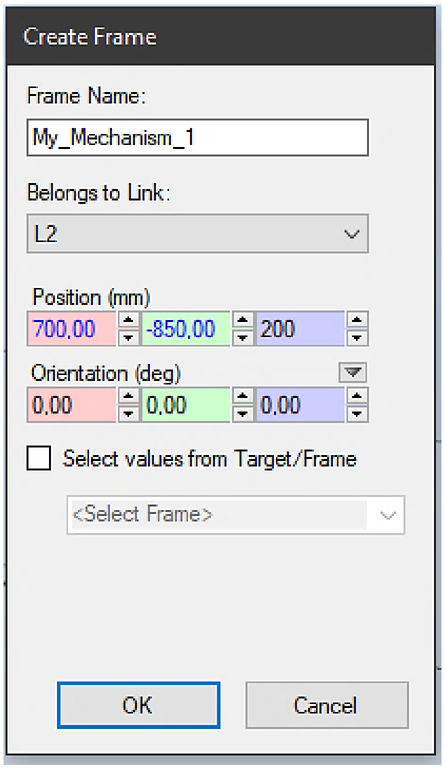
Limit Type: Constant

Joint Limits

Min Limit (mm): 0,00

Max Limit (mm): 1000,00

Figure 68. Mechanism interaction - screenshot 3



Create Frame

Frame Name: My_Mechanism_1

Belongs to Link: L2

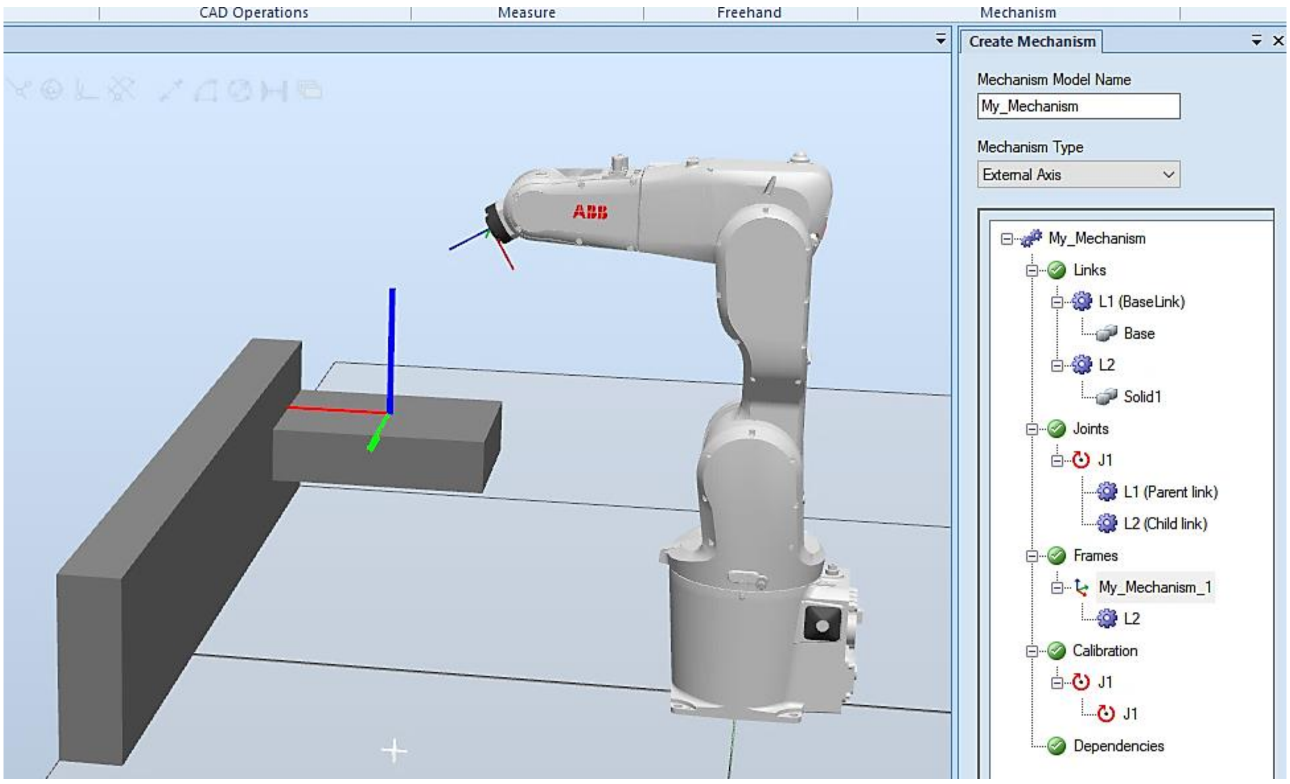
Position (mm): 700,00 | -850,00 | 200

Orientation (deg): 0,00 | 0,00 | 0,00

[ ] Select values from Target/Frame

<Select Frame>

Figure 69. Mechanism interaction - screenshot 4



Create Mechanism

Mechanism Model Name: My_Mechanism

Mechanism Type: External Axis

- My_Mechanism
  - Links
    - L1 (BaseLink)
      - Base
    - L2
      - Solid1
  - Joints
    - J1
      - L1 (Parent link)
      - L2 (Child link)
  - Frames
    - My_Mechanism_1
      - L2
  - Calibration
    - J1
      - J1
  - Dependencies

Figure 70. Mechanism interaction - screenshot 5

6. Create Pose1 in front of the robot.

As mentioned in the previous simulation, it is possible to save one position of the mechanism. The position that the user should save for this simulation is the one with the solid in front of the robot (Pose1). The other pose saved is SyncPose and it is the initial position of the mechanism.

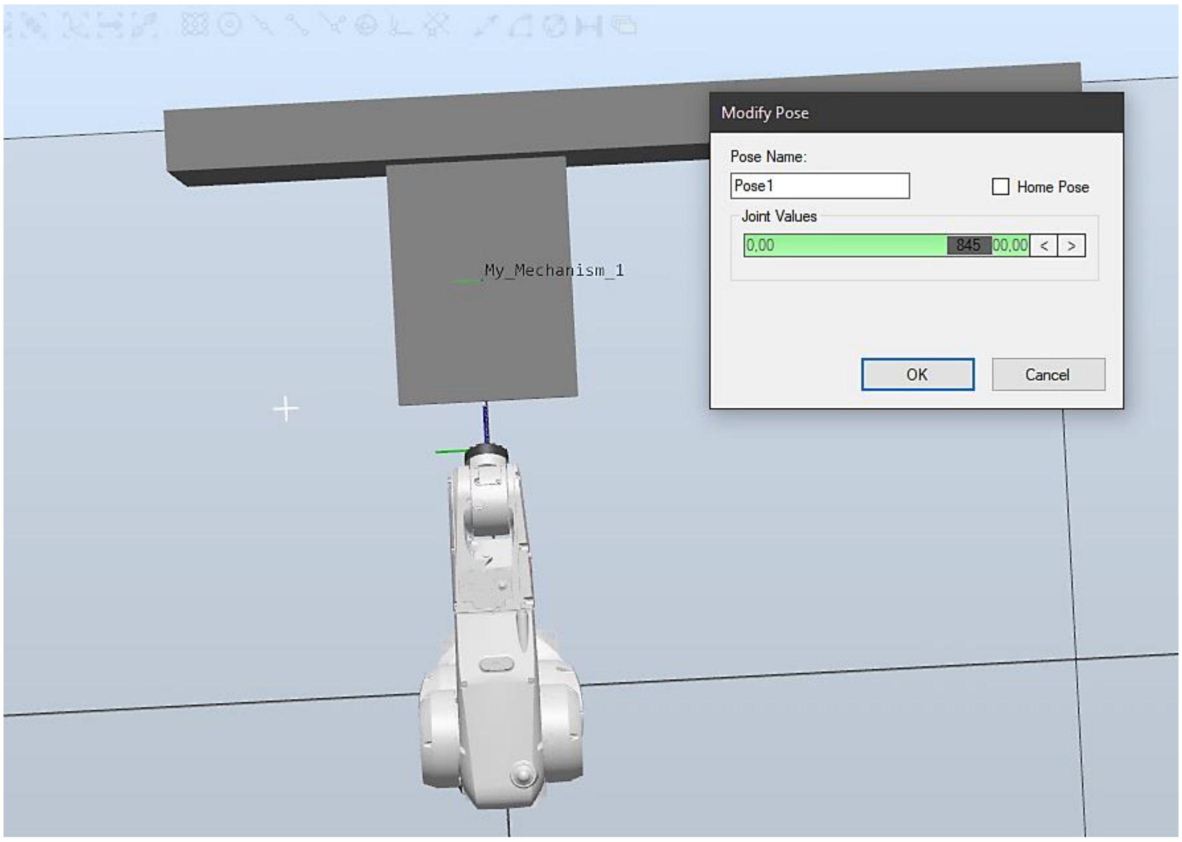


Figure 71. Mechanism interaction - screenshot 6

It is possible to move the mechanism into its limits, pressing the right bottom of the mouse over its name and press the option “Mechanism Joint Jog”. Once the mechanism is on the desirable location, the pose can be saved.

7. Create new solid “solidcyl” and attach it on the mechanism

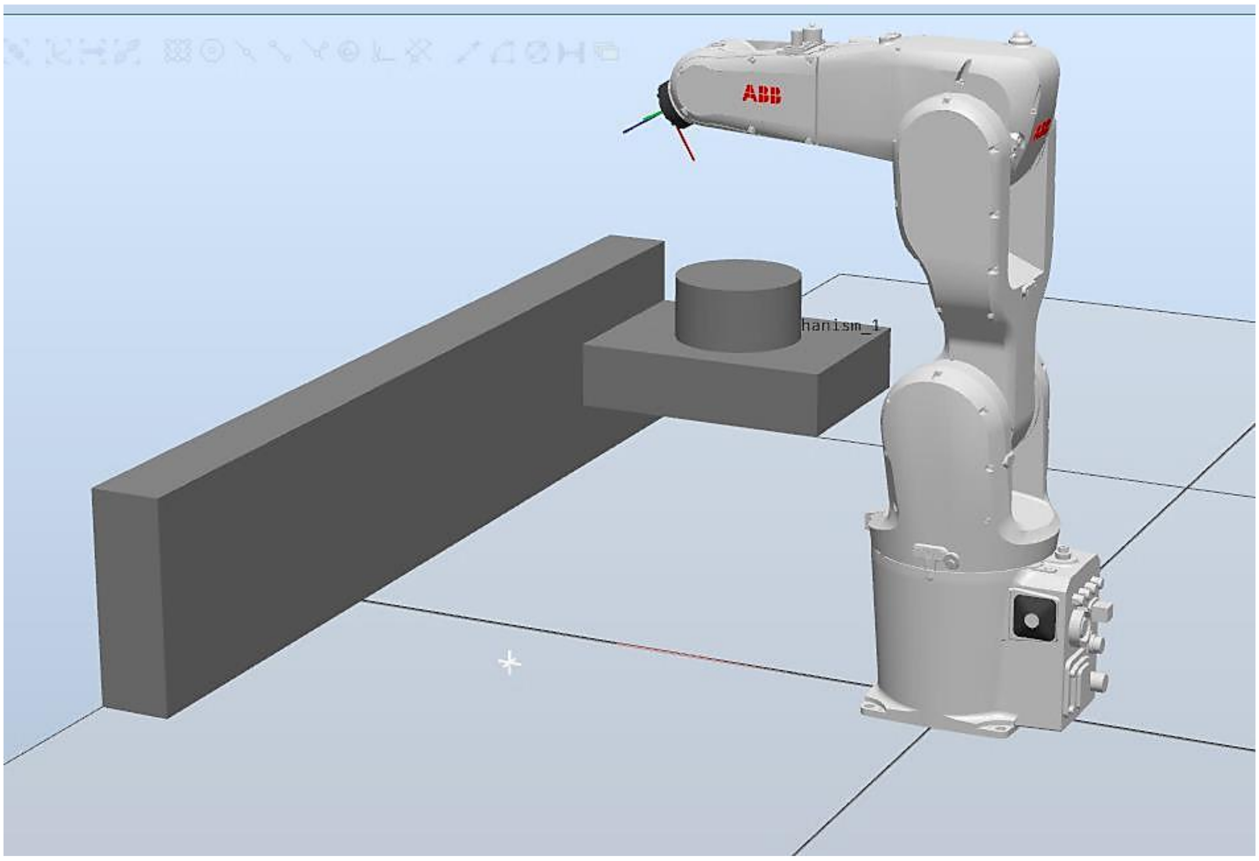


Figure 72. Mechanism interaction - screenshot 7

The new solid created will be a cylinder with a radius of 100 mm and a height of 100 mm. The solid called “solidcyl” will be attached on the mechanism dragging with the mouse the name of the solid over the name of the mechanism. The solid will be located on the frame of the mechanism automatically.

8. Creation of the second solid.

Create a copy of the previous solid “solidcyl” and set the new position of it (it does not matter the position; it is just needed that the robot can reach it). In this case, it has been used the position (100,-500,0).

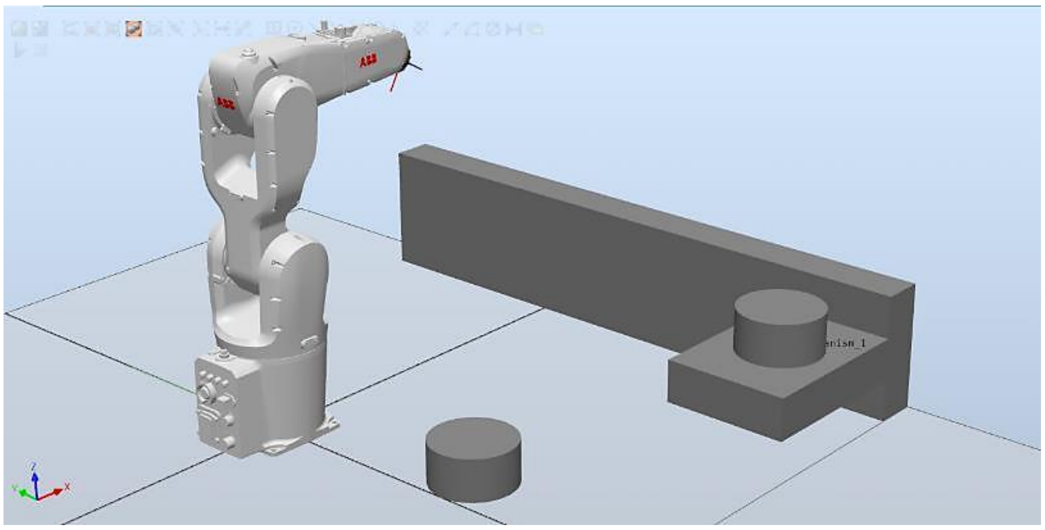


Figure 73. Mechanism interaction - screenshot 8

9. Insert tool and attach it on the robot. In this case, it will be used the tool called AW_Gun_PSF_25.

10. Create targets needed.

The different targets have been created on the station. It has been needed just five targets: the Home position of the robot and two targets for each cylinder solid. To select the centre point of the cylinder surface it has been used the tool Snap object found on the top of the station interface shown on Figure 74.

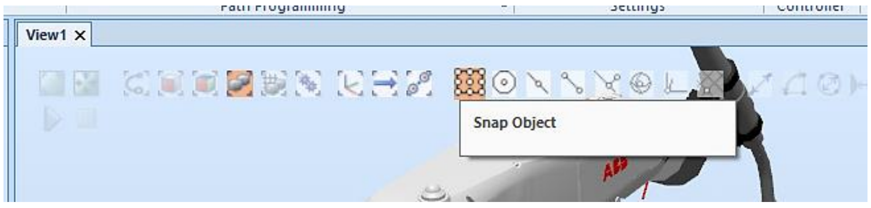


Figure 74. Mechanism interaction - screenshot 9

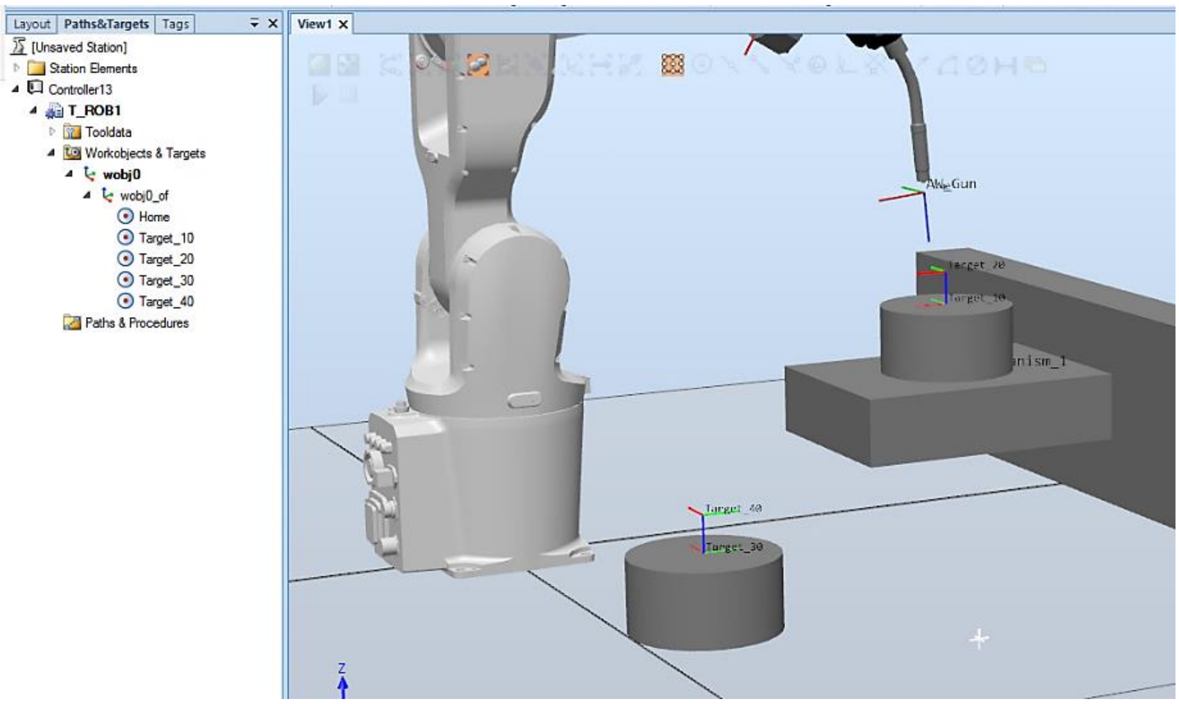


Figure 75. Mechanism interaction - screenshot 10

11. Create a path.

Creation of the path with all the targets included. In this case, it has been needed to repeat some of the positions. It can be seen the order of the targets on the left side of the station with its repetitions.

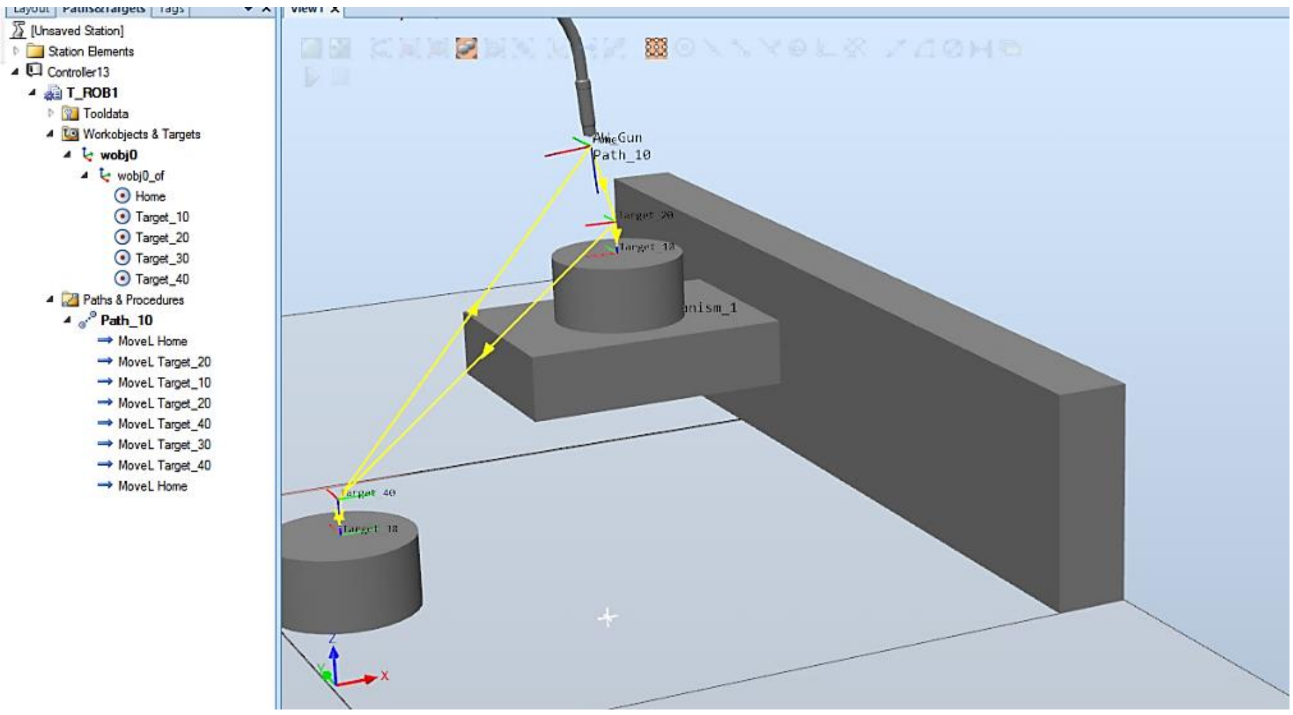


Figure 76. Mechanism interaction - screenshot 11

12. Add smart components and edit their properties.

On the menu “Simulation” it can be found the station logic to add the smart components. In this case, it will be adding two PoseMover, two CollisionSensor, one Attacher and one Detacher. The user has to edit their properties.

Table 8. Smart Components properties - Mechanism interaction

Smart Component name	**Mechanism**	**Pose**	**Duration (s)**
PoseMover	My_Mechanism	Pose1	2
PoseMover_2	My_Mechanism	SyncPose	2

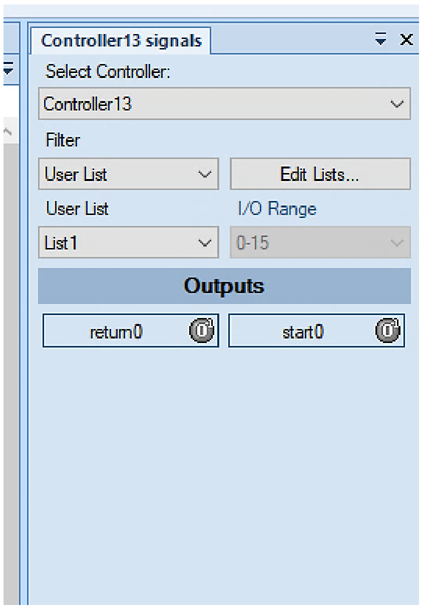
Smart component name	**Object1**	**Object2**	**Near miss (mm)**
CollisionSensor	AW_Gun_PSF_25	Solidcyl	50
CollisionSensor_2	AW_Gun_PSF_25	Solidcyl_2	50

Smart component name	**Parent**	**Child**
Attacher	AW_Gun_PSF_25	Solidcyl
Detacher	-	Solidcyl

13. Simulation Setup

On the simulation Setup which can be found on the simulation menu, the user will select the Single-cycle. That means that the simulation will stop after the program has run one cycle the process main on the RAPID code.

On the same simulation menu, the user can also use the I/O Simulator to see the outputs and inputs when are they active or not. The user can change if the signals are active or not manually through this window simulation. It is possible also to insert a user list for just visualize the signals created by the own user.


Controller13 signals

Select Controller: Controller13

Filter: User List

User List: List1

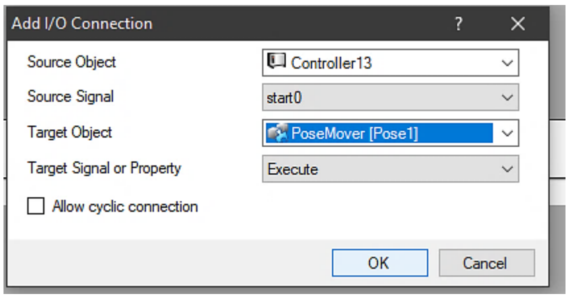
I/O Range: 0-15

Outputs

Figure 77. Mechanism interaction - screenshot 12

14. Station logic. 1. Add I/O connection on “Signals and Connections”.

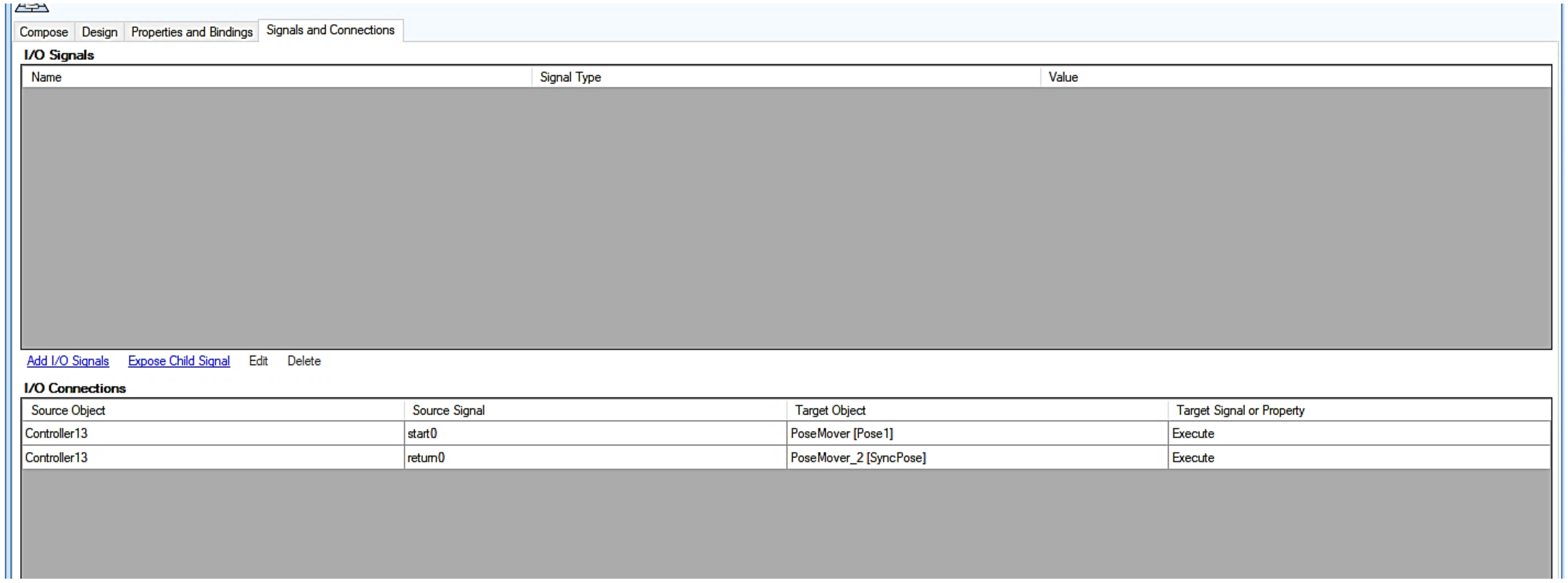
On the simulation menu, inside the station logic, there is the window of “signals and connections”. In this space, the user will add the different connections between the objects and the signals. It is just needed to click on add connection and complete the new pop-up window (see Figure 78). The window is just asking where is the signal from (the source), which signal, in which object is going to actuate and which property of that object is going to be active.


Add I/O Connection

Source Object	Controller13
Source Signal	start0
Target Object	PoseMover [Pose1]
Target Signal or Property	Execute

☐ Allow cyclic connection

Figure 78. Mechanism interaction - screenshot 13



I/O Signals

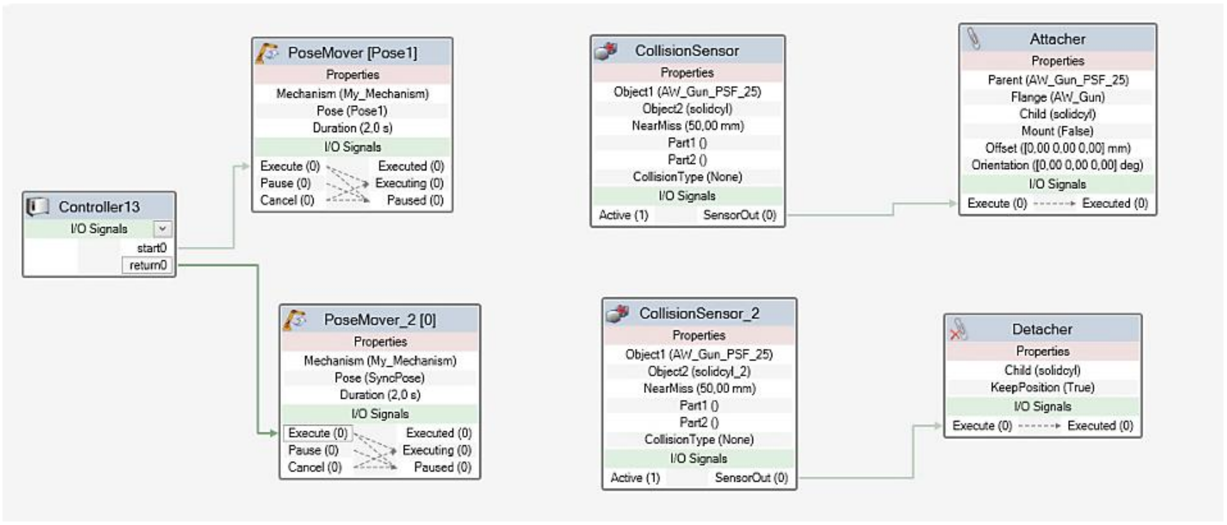
Name	Signal Type	Value

I/O Connections

Source Object	Source Signal	Target Object	Target Signal or Property
Controller13	start0	PoseMover [Pose1]	Execute
Controller13	return0	PoseMover_2 [SyncPose]	Execute

Figure 79. Mechanism interaction - screenshot 14

The I/O connections will appear automatically on the window Design after configuring them on “signals and connections” window. The next step is to edit the latest connections between the smart components on the Design window, as done on the previous simulation.


Controller13
I/O Signals
start0
return0

PoseMover [Pose1]
Properties
Mechanism (My_Mechanism)
Pose (Pose1)
Duration (2,0 s)
I/O Signals
Execute (0) Executed (0)
Pause (0) Executing (0)
Cancel (0) Paused (0)

PoseMover_2 [0]
Properties
Mechanism (My_Mechanism)
Pose (SyncPose)
Duration (2,0 s)
I/O Signals
Execute (0) Executed (0)
Pause (0) Executing (0)
Cancel (0) Paused (0)

CollisionSensor
Properties
Object1 (AW_Gun_PSF_25)
Object2 (solidcyl)
NearMiss (50,00 mm)
Part1 ()
Part2 ()
CollisionType (None)
I/O Signals
Active (1) SensorOut (0)

CollisionSensor_2
Properties
Object1 (AW_Gun_PSF_25)
Object2 (solidcyl_2)
NearMiss (50,00 mm)
Part1 ()
Part2 ()
CollisionType (None)
I/O Signals
Active (1) SensorOut (0)

Attacher
Properties
Parent (AW_Gun_PSF_25)
Flange (AW_Gun)
Child (solidcyl)
Mount (False)
Offset ([0,00 0,00 0,00] mm)
Orientation ([0,00 0,00 0,00] deg)
I/O Signals
Execute (0) Executed (0)

Detacher
Properties
Child (solidcyl)
KeepPosition (True)
I/O Signals
Execute (0) Executed (0)

Figure 80. Mechanism interaction - screenshot 15

The two digital outputs created will be the signals which will connect the two PoseMover to move the mechanism from one Pose to another. When the CollisionSensor will detect that the tool is near to the solid over the mechanism (solidcyl), the attacher will active. When the CollisionSensor_2 will detect that the tool is near the second solid located on the floor of the station (solidcyl_2), the detacher will active. The mechanism will change its position with the solid over it, the robot will follow its path to attach the solid and change its location. Meanwhile, the mechanism will go back to its first position. After the user has configured the connection on the window design, they must synchronize to RAPID.

15. RAPID code

On the RAPID window, it can be found the MainModule code (shown in Figure 81). The user can double-click over it to edit it.

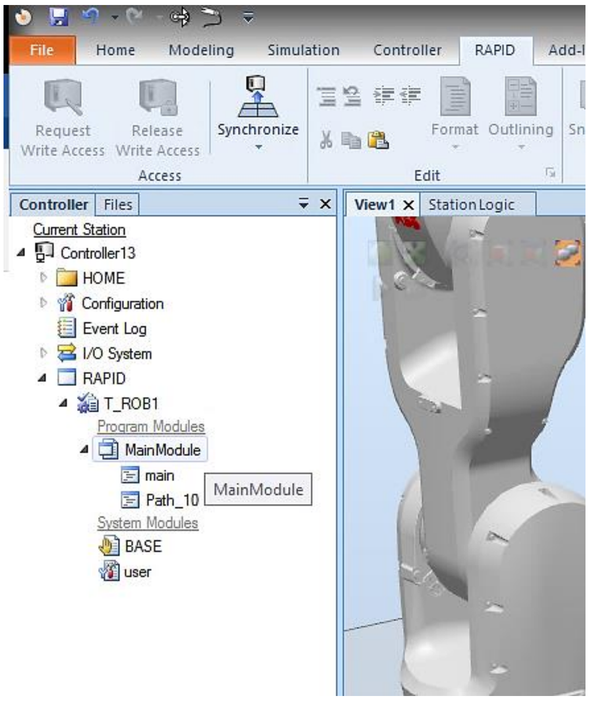


Figure 81. Mechanism interaction - screenshot 16

After everything explained is configure, the RAPID code will just show the path of the robot and the process main will be empty. It is important to remember that the simulator follows the instructions within the "main" process. If there is no path or function within the process, the simulation will not work.

Just two functions will be used on this simulation: SetDO and WaitTime. The last function mentioned has been already explained in the other simulation. SetDO just changes the value of the signal: on the first line, the digital output “start0” will change its state to active. Depends on the type of signal, the function will change its name: SetDO (digital output), SetDI (digital input)…

When the simulation will start, the first signal will execute the first PoseMove and the mechanism will change its position. After three seconds, the robot will start its move following the Path programmed. It will take the object and change its position. Then the other signal will execute the second PoseMover and the mechanism will come back to its first position. The two signals will be inactive when the simulation finishes. The code can be seen in the following square:

```
PROC main()
    SetDO start0,1;
    WaitTime 3;
```

```
        Path_10;
        SetDO return0,1;
        SetDO start0,0;
        WaitTime 1;
        SetDO return0,0;
    ENDPROC
PROC Path_10()
    MoveL Home,v1000,z100,AW_Gun\WObj:=wobj0;
    MoveL Target_20,v1000,fine,AW_Gun\WObj:=wobj0;
    MoveL Target_10,v1000,fine,AW_Gun\WObj:=wobj0;
    MoveL Target_20,v1000,fine,AW_Gun\WObj:=wobj0;
    MoveL Target_40,v1000,fine,AW_Gun\WObj:=wobj0;
    MoveL Target_30,v1000,fine,AW_Gun\WObj:=wobj0;
    MoveL Target_40,v1000,fine,AW_Gun\WObj:=wobj0;
    MoveL Home,v1000,z100,AW_Gun\WObj:=wobj0;
ENDPROC
```

Once all the changes have been added, the user can apply changes, synchronize to the station, and save the code before the simulation.

16. Reset simulation. The user should save the current state of the simulation, also mentioned in the previous simulation. It is important to return the solid moved to its original place (over the mechanism).
17. Edit the properties of the object “solidcyl_2” and make it invisible. The path can be also invisible. The user can also change the colour of the solids, for example.

Some images of the simulation:

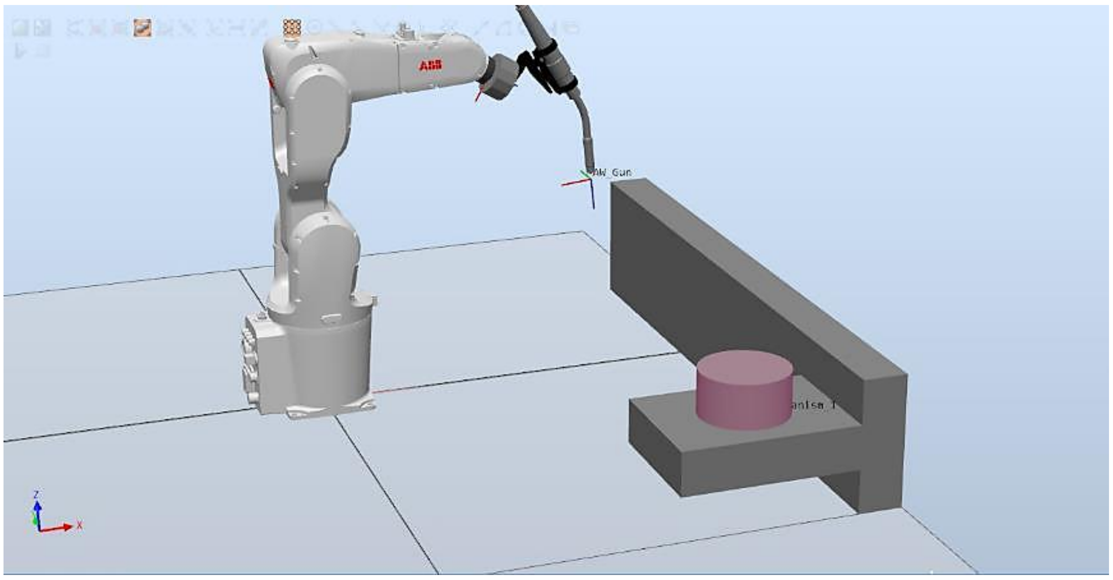


Figure 82. Mechanism interaction 1

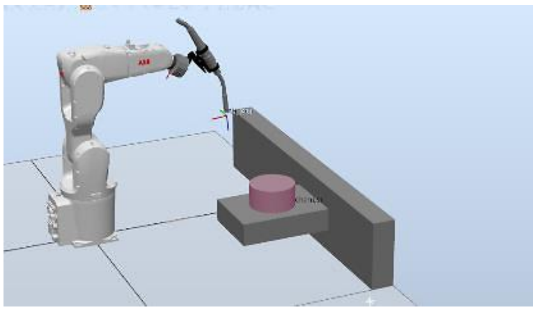


Figure 83. Mechanism interaction 2

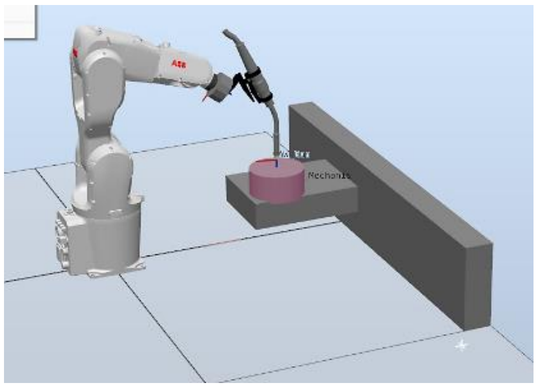


Figure 84. Mechanism interaction 3

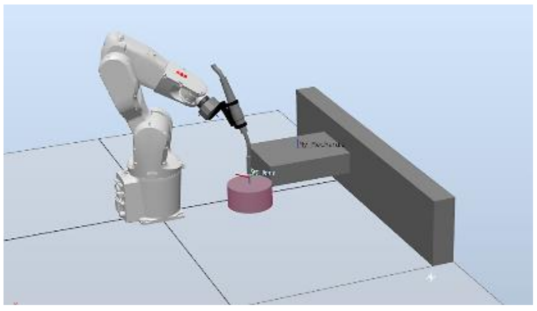


Figure 85. Mechanism interaction 4

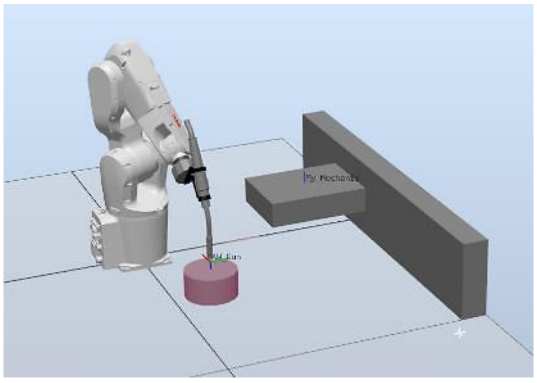


Figure 86. Mechanism interaction 5

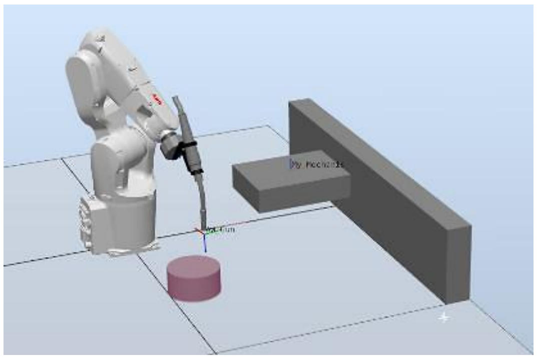


Figure 87. Mechanism interaction 6

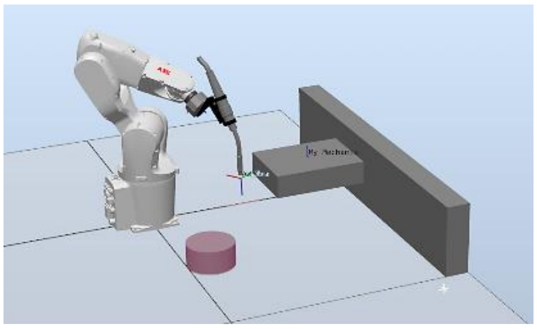


Figure 88. Mechanism interaction 7

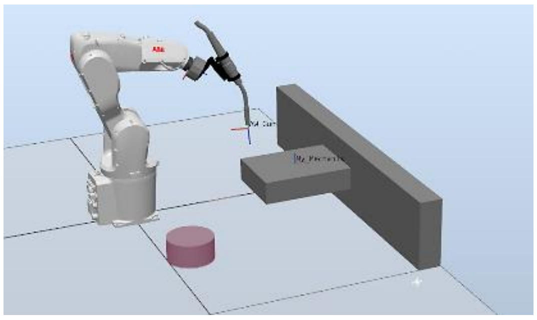


Figure 89. Mechanism interaction 8

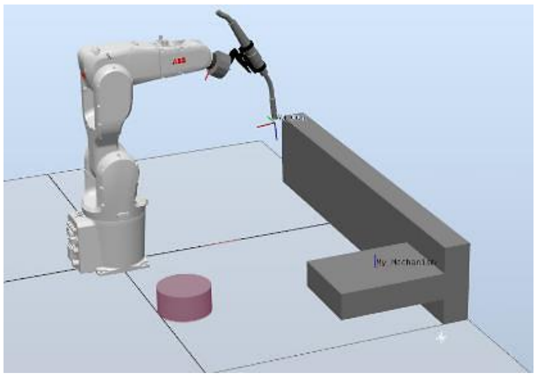


Figure 90. Mechanism interaction 9

Access to the video of the simulation on the following link:

https://drive.google.com/drive/folders/1VTs0rX9TSME7J_K1DgqRFnVHcndNq93c?usp=sharing

10.4. Transportation guides

This simulation consists of a solid moving through two transportation guides. Once the simulation is started, the object will cross the first guide. The different sensors will stop the box at the end of the first guide and the robot will attach it and move it to the other guide. After the solid has crossed the second guide, the object will be deleted. To create this simulation the following steps have been followed:

1. Insert the robot and the tool.

As showed and explained in other simulations, the tool should be attached to the robot. In this case, it will be used “MyTool” tool.

2. Insert conveyor guide

The transportation guides could be found on the equipment from the library. For this simulation, two guides will be inserted. The location of each guide should be one on each side of the robot but with enough distance for the robot to reach it.

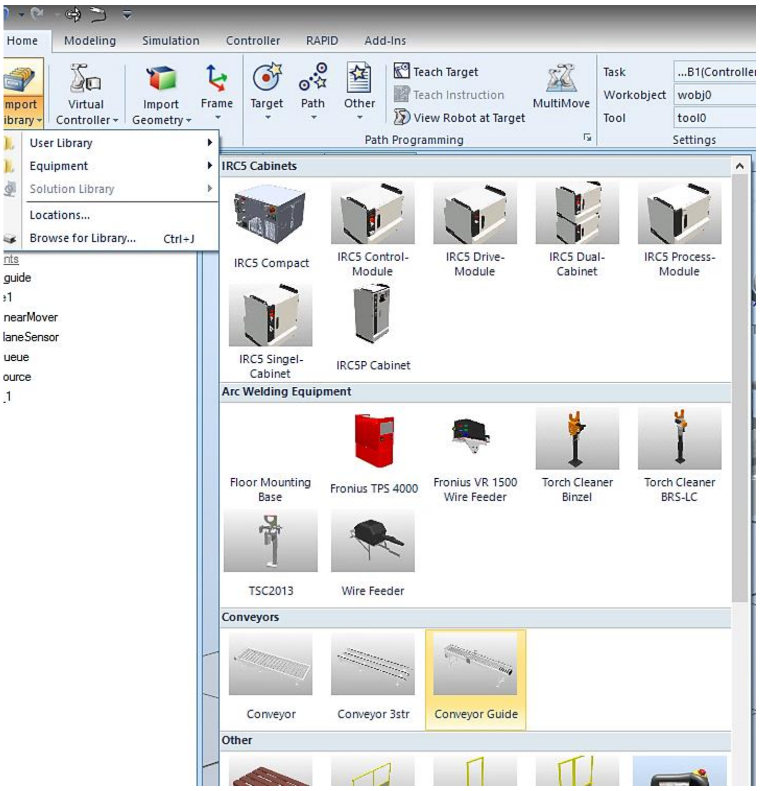


Figure 91. Transportation guides - screenshot 1

3. Create a solid.

The user needs to create a solid that will be moved through the guides. In this case, it will be created a box (size: 300x300x200mm). The location of it does not matter, could be wherever. The object that will be over the guides, it will be a copy from the solid created.

The station should look like the following images will all the items inserted.

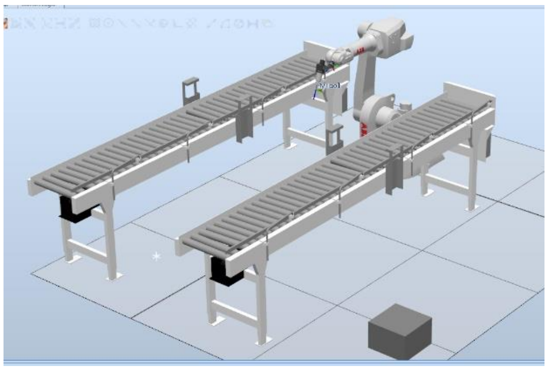


Figure 92. Transportation guides - screenshot 2

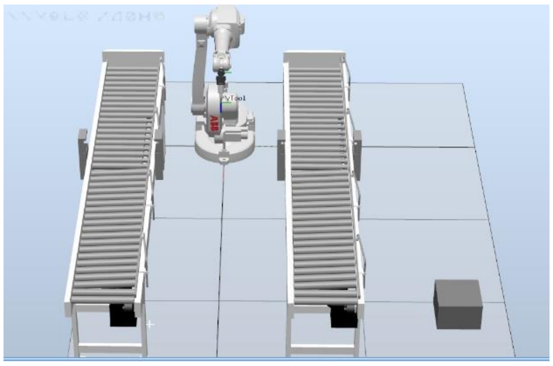
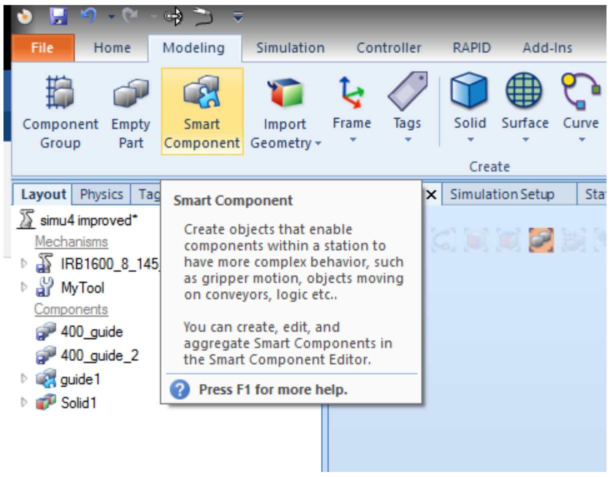


Figure 93. Transportation guides - screenshot 3

4. Initialize a virtual controller from the layout.
5. Create a smart component

It will be created a smart component for the guides and it will be added the components and logic gates needed. After the smart component will have all its connections between its components, the user should connect it with the controller through the different inputs and outputs. All the components listed in Figure 95 will be added. In addition, a LogicSRLatch gate will be added.



Smart Component

Create objects that enable components within a station to have more complex behavior, such as gripper motion, objects moving on conveyors, logic etc..

You can create, edit, and aggregate Smart Components in the Smart Component Editor.

Press F1 for more help.

Figure 94. Transportation guides - screenshot 4

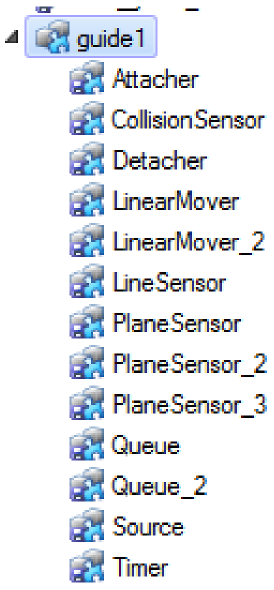


Figure 95. Transportation guides - screenshot 5

6. Configure the components and connections

The user should configure the properties of each component. In this case, the following properties have been inserted. The position of each component could change depending on the location of all the items previously created (solid, guides…)

Table 9. Smart Components properties - Transportation guides

SOURCE					
Source	**Copy**	**Parent**	**Position [mm]**	**Transient**	**Physics Behaviour**
Solid1	-	Guide1	500x500x750	YES	Kinematic

QUEUE		
Back	**Front**	**NumberOfObjects**
(copy source)	-	1
QUEUE_2		
(object LinearMover_2)	-	1

LINE SENSOR				
Start [mm]	**End [mm]**	**Radius [mm]**	**Sensed Part**	**Sensed Point [mm]**
3000x600x1000	800x600x1000	100	(front queue)	0x0x0

LINEAR MOVER			
Object	**Direction [mm]**	**Speed [mm/s]**	**Reference**
(front queue)	-1000x0x0	200	global
LINEAR MOVER_2			
(copy source)	1000x0x0	200	global

COLLISION SENSOR		
Object1	**Object2**	**NearMiss [mm]**

MyTool	(object LinearMover)	20

ATTACHER	
Parent	**Child**
MyTool	(object2 Collision Sensor)

PLANE SENSOR		
Origin [mm]	**Axis 1 [mm]**	**Axis 2 [mm]**
500x550x800	0x300x0	0x0x50
PLANE SENSOR_3		
500x-850x775	0x300x0	0x0x50
PLANE SENSOR_2		
2695,08x-850x800	0x300x0	0x0x50

TIMER		
Start Time [s]	**Interval [s]**	**Repeat**
0	1	YES

DETACHER	
Child	**Keep position**
(Sensed Part PlaneSensor_3)	YES

There will be created also some inputs and outputs of the smart controller: one input and two outputs.

The input will be used to give a signal to the component "Source" when the simulation is started. The component will create a copy of the solid already created and the user should configure its properties as the location desirable: over the guide on the beginning. Once the "Source" is executed, the copy will be added on the end of the queue. For that, it is needed a connection between the copy of the source and the back of the queue besides connecting the executed of one component to the next execute.

As it is just one object each time, the queue will be just the copy. So, the back and the front of it will be the same solid. In some other simulation with more objects at the same time, this component would make more sense. Regardless, it will be created a connection from the front of the queue to two connections: the sensed part of the “LineSensor” and the object of the “LineMover”. The Sensor out of the “LineSensor” will execute the “LineMover”. In other words, meanwhile the sensor is detecting the object, the mover component will move it on the direction and with the speed configured.

The object moved for the mover component should be connected on the “CollisionSensor” as the second object. The first object of the sensor is the tool attached to the robot. The sensor is always active, and when it detects that the objects are the near-miss previously configured, it will execute the “Attacher” component.

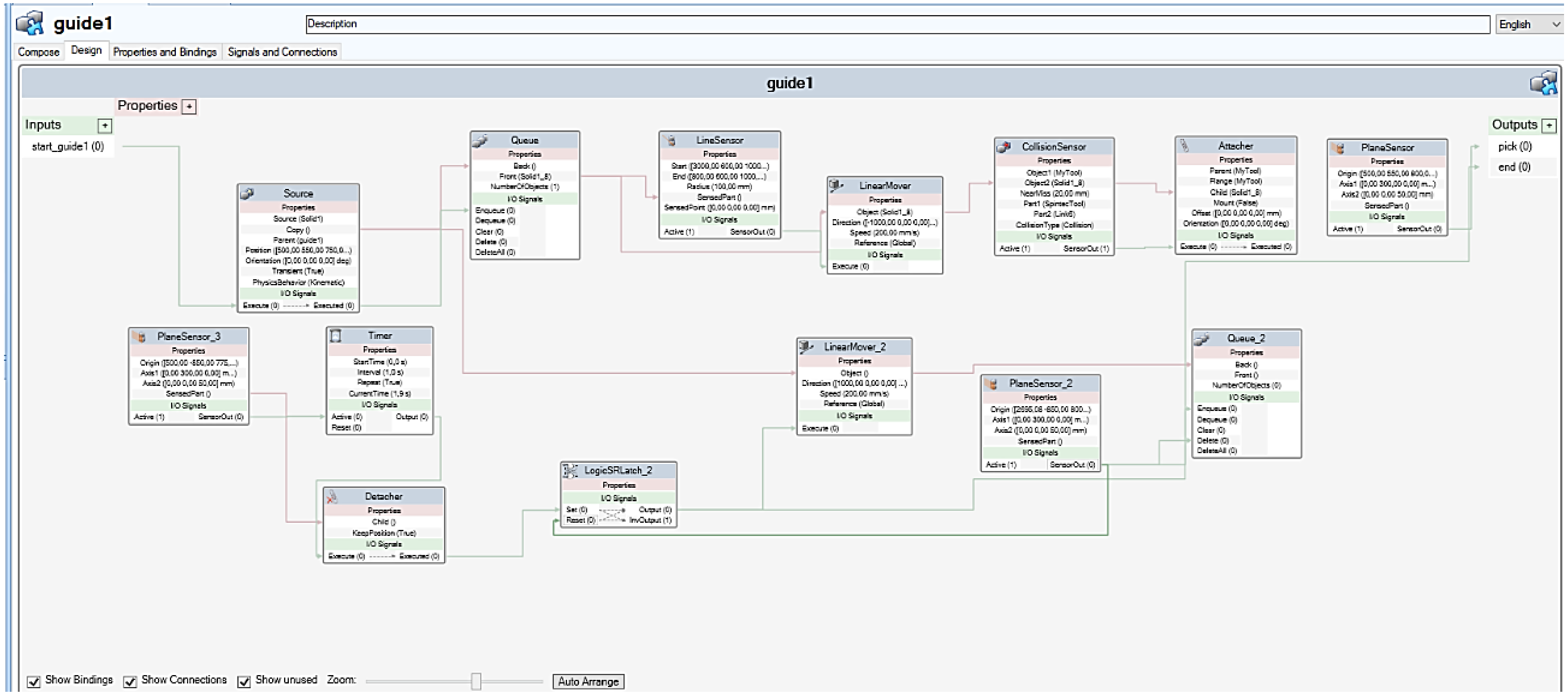
There is the “PlaneSensor” which detects if there is some object on a 2D space. The sensor is located on the end of the first guide, and when it detects the object, it will active the output “pick” used for active the robot.

The next sensor is the “PlaneSensor_3” and it is located on the beginning of the second guide (where the robot should leave the box). When the sensor detects the object a “Timer” of one second will start and then active the “Detacher”. The sensed part of the sensor must be the child that de detacher should use. If it is not used the timer, the sensor detects the box before it arrives on the desirable place over the guide. Once the detacher is executed, a logic gate is used: the LogicSRLatch. The utility of the gate is to prolong the signal from the detacher.

The signal prolonged will execute the other “LineMover” but in the opposite direction as the other one. The object that it will be moved is the copy from the source. The signal prolonged will also activate the enqueue from the second “Queue” component. The object that the “Queue” will put on the back, is the one moved by the “LineMover”.

The last component left for mention is the sensor “PlaneSensor_2” which is located on the end of the second guide. When the sensor detects the object, it will active the delete option of the “Queue” component and the reset of the gate. Besides, the sensor will active also the output “end” switching off the simulation.

Design of the smart component:



Source
Properties
Source (Solid1)
Copy ()
Parent (guide1)
Position ([500,00 550,00 750,0...)
Orientation ([0,00 0,00 0,00] deg)
Transient (True)
PhysicsBehavior (Kinematic)
I/O Signals
Execute (0) → Executed (0)

Queue
Properties
Back ()
Front (Solid1_8)
NumberOfObjects (1)
I/O Signals
Enqueue (0)
Dequeue (0)
Clear (0)
Delete (0)
DeleteAll (0)

LineSensor
Properties
Start ([3000,00 600,00 1000...)
End ([800,00 600,00 1000,...)
Radius (100,00 mm)
SensedPart ()
SensedPoint ([0,00 0,00 0,00] mm)
I/O Signals
Active (1) SensorOut (0)

LinearMover
Properties
Object (Solid1_8)
Direction ([-1000,00 0,00 0,00]...)
Speed (200,00 mm/s)
Reference (Global)
I/O Signals
Execute (0)

CollisionSensor
Properties
Object1 (MyTool)
Object2 (Solid1_8)
NearMiss (20,00 mm)
Part1 (SpintecTool)
Part2 (Link6)
CollisionType (Collision)
I/O Signals
Active (1) SensorOut (1)

Attacher
Properties
Parent (MyTool)
Flange (MyTool)
Child (Solid1_8)
Mount (False)
Offset ([0,00 0,00 0,00] mm)
Orientation ([0,00 0,00 0,00] deg)
I/O Signals
Execute (0) → Executed (0)

PlaneSensor
Properties
Origin ([500,00 550,00 800,0...)
Axis1 ([0,00 300,00 0,00] m...)
Axis2 ([0,00 0,00 50,00] mm)
SensedPart ()
I/O Signals
Active (1) SensorOut (0)

PlaneSensor_3
Properties
Origin ([500,00 -650,00 775,...)
Axis1 ([0,00 300,00 0,00] m...)
Axis2 ([0,00 0,00 50,00] mm)
SensedPart ()
I/O Signals
Active (1) SensorOut (0)

Timer
Properties
StartTime (0,0 s)
Interval (1,0 s)
Repeat (True)
CurrentTime (1,9 s)
I/O Signals
Active (0) Output (0)
Reset (0)

Detacher
Properties
Child ()
KeepPosition (True)
I/O Signals
Execute (0) → Executed (0)

LogicSRLatch_2
Properties
I/O Signals
Set (0) → Output (0)
Reset (0) → InvOutput (1)

LinearMover_2
Properties
Object ()
Direction ([1000,00 0,00 0,00] ...)
Speed (200,00 mm/s)
Reference (Global)
I/O Signals
Execute (0)

PlaneSensor_2
Properties
Origin ([2696,08 -650,00 800...)
Axis1 ([0,00 300,00 0,00] m...)
Axis2 ([0,00 0,00 50,00] mm)
SensedPart ()
I/O Signals
Active (1) SensorOut (0)

Queue_2
Properties
Back ()
Front ()
NumberOfObjects (0)
I/O Signals
Enqueue (0)
Dequeue (0)
Clear (0)
Delete (0)
DeleteAll (0)

Figure 96. Transportation guides - screenshot 6

7. Create the outputs and the inputs of the controller.

It is needed to create two digitals inputs (*pickcontroller* and *endcontroller*) and one digital output of the controller (*startcontroller*).

8. Configure the connections between the controller and the smart component.

The inputs of the smart controller will be connected to the outputs of the controller and vice-versa.

Design of the controller:

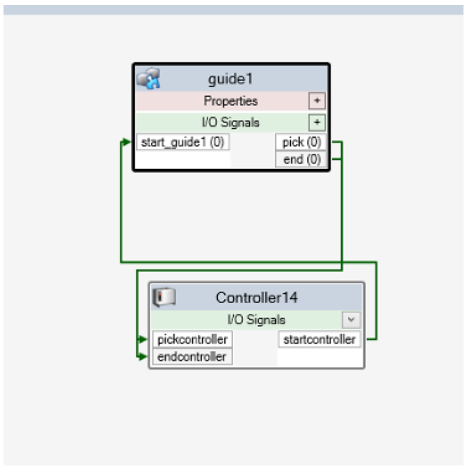
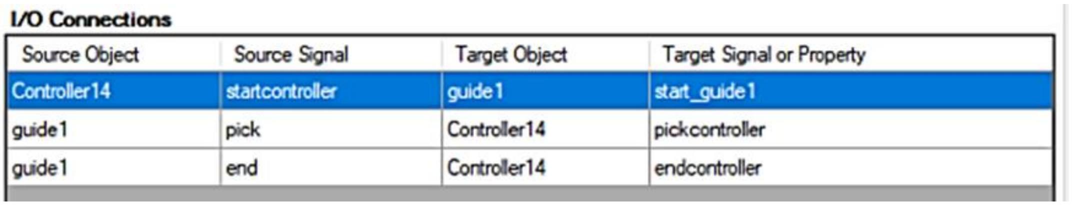


Figure 97. Transportation guides - screenshot 7

I/O connections of the controller:



I/O Connections

Source Object	Source Signal	Target Object	Target Signal or Property
Controller14	startcontroller	guide1	start_guide1
guide1	pick	Controller14	pickcontroller
guide1	end	Controller14	endcontroller

Figure 98. Transportation guides - screenshot 8

9. Create targets and the path.

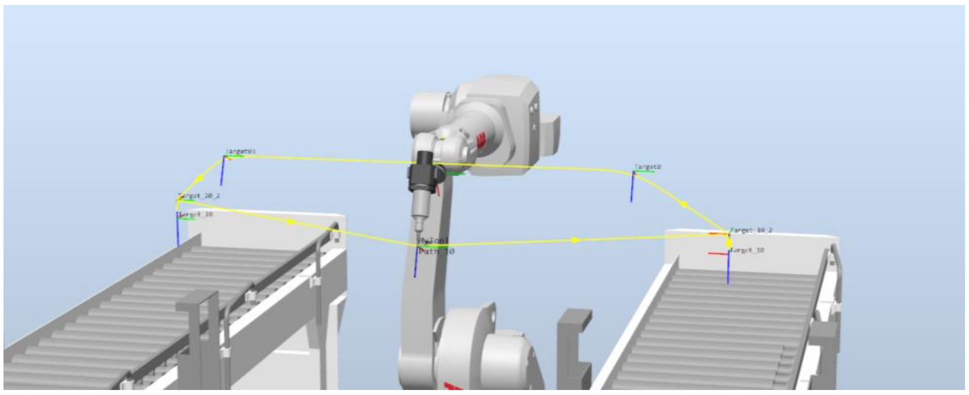


Figure 99. Transportation guides - screenshot 9

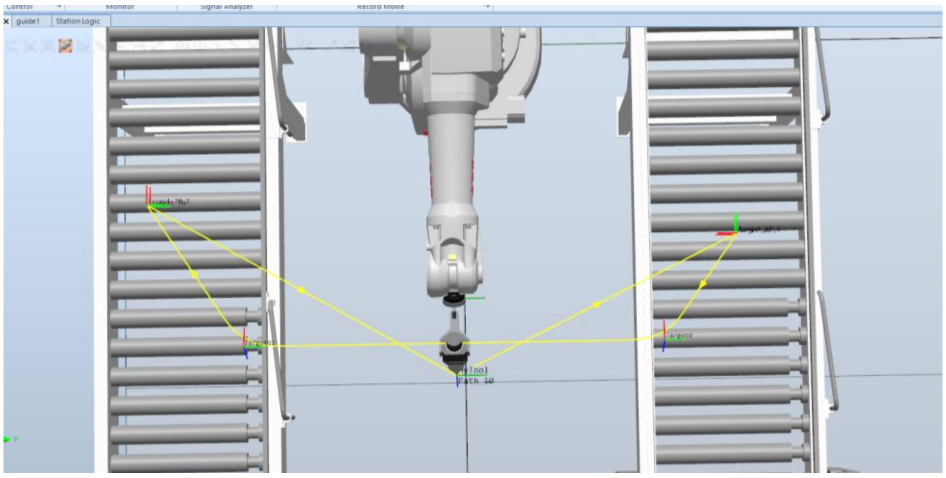


Figure 100. Transportation guides - screenshot 10

Seven targets have been created for this simulation. On the locations that the robot has to attach and detach the solid, there are two targets on each (one some mm higher in axis z than the other). One target is the home position and the other two targets are created for avoiding collisions when the robot is moving the box. The positions of the different targets could be seen on the code.

10. Synchronize to RAPID
11. Edit RAPID code

On the first part of the code, there is always the data of all the targets saved. In this case, it has been also created a variable called "counter" initializing on 0. The digital output "startcontroller" should be active when the user wants to start the simulation and that output gives the signal to the "source" component. It is just needed that the output switches on once during the simulation, therefore it is used the variable “counter”, for being sure that the output will switch on just once. When the user resets the simulation, the counter will start again.

The robot will wait until the digital input “pickcontroller” is active to move through the path and the code will wait until the signal “endcontroller” for finishing the simulation.

```
MODULE MainModule
   CONST robtarget
Home:=[[989.268392129,0,1011.647370713],[0.190809008,0,0.981627181,0],[0,
0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_10_2:=[[648.4,700,1000],[0,-
0.707106781,0.707106781,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_10:=[[648.4,700,950],[0,-
0.707106781,0.707106781,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_20_2:=[[554.623,-761.226,1007.96],[0,0,1,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_20:=[[554.623,-761.226,957.96],[0,0,1,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget
Target0:=[[914.347068677,500,1197.0841397],[0.190809008,0,0.981627181,0],[0
,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target01:=[[914.347068677,-
500,1197.0841397],[0.190809008,0,0.981627181,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   VAR num counter:=0;

   PROC main()
      IF counter=0 THEN
         SetDO startcontroller,1;
         counter:=counter+1;
         SETDO startcontroller,0;
      ENDIF
```

```
        waitDI pickcontroller,1;
        Path_10;
        waitDI endcontroller,1;
    ENDPROC

    PROC Path_10()
        MoveL Home,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_10_2,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_10,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_10_2,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target0,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target01,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_20_2,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Target_20,v1000,z100,MyTool\WObj:=wobj0;
        WaitTime 2;
        MoveL Target_20_2,v1000,z100,MyTool\WObj:=wobj0;
        MoveL Home,v1000,z100,MyTool\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

Apply changes in the code and synchronize to the station. The simulation is ready for running.

10.5. Multimove tool

This simulation aims to test one of the software's tools which is the MultiMove. This function enables the interaction between two robots; one robot is holding the workpiece and another robot is operating on it. This simulation is a simple example to work with this function. The following steps have been followed:

1. Insert two robots.

Insert two robots and position it one in front of the other. In this case, the distance between them is 1000 mm.

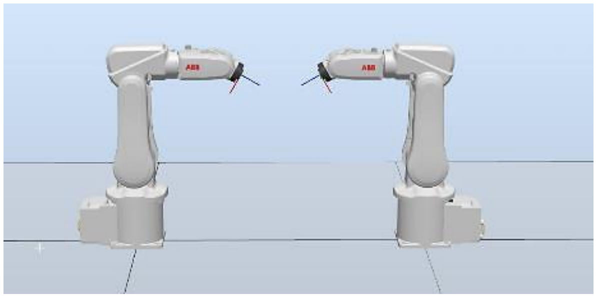


Figure 101. Multimove tool - screenshot 1

2. Insert tool

Insert the tool MyTool and attach it to the robot 1 (the one on the left on the previous figure).

3. Insert solid

Create a box with a size of 200x200x200 mm and modify its local origin to locate it on the central point of the solid. Attach the object to the robot 2.

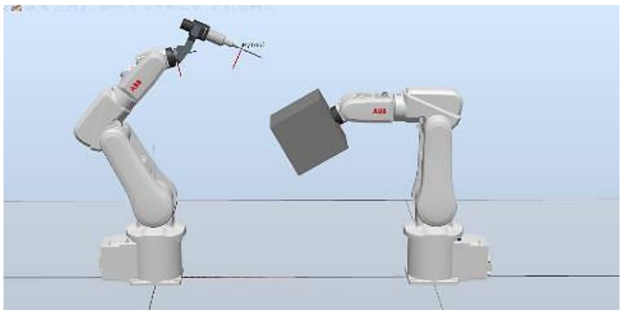
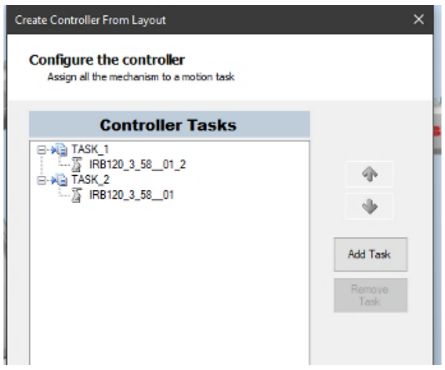


Figure 102. Multimove tool - screenshot 2

4. Insert a virtual controller

Create a virtual controller from layout and configure the tasks of it. On task 1 should be the robot that is holding the object and on task 2 should be the one which has the tool.


Create Controller From Layout

Configure the controller

Assign all the mechanism to a motion task

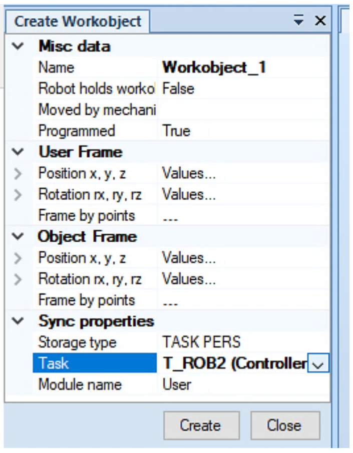
Controller Tasks

- TASK_1
  - IRB120_3_58__01_2
- TASK_2
  - IRB120_3_58__01

Figure 103. Multimove tool - screenshot 3

5. Create a new work object.

Create a new work object synchronized to robot 2 which is holding the object.

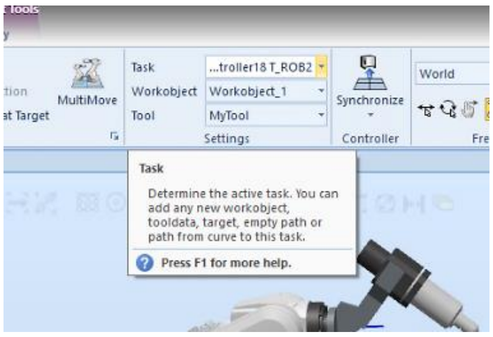

Create Workobject

Misc data	
Name	**Workobject_1**
Robot holds worko	False
Moved by mechani	
Programmed	True
User Frame	
Position x, y, z	Values...
Rotation rx, ry, rz	Values...
Frame by points	...
Object Frame	
Position x, y, z	Values...
Rotation rx, ry, rz	Values...
Frame by points	...
Sync properties	
Storage type	TASK PERS
Task	**T_ROB2 (Controller**
Module name	User

Figure 104. Multimove tool - screenshot 4

6. Change the settings

Modify the settings on the “Home” menu. Select the task of the second robot which is holding the box, the tool of MyTool (which is grabbed by the robot 1) and the work object recently created.



Task	...troller18 T_ROB2
Workobject	Workobject_1
Tool	MyTool

Settings

Task

Determine the active task. You can add any new workobject, tooldata, target, empty path or path from curve to this task.

Press F1 for more help.

Figure 105. Multimove tool - screenshot 5

7. Create targets

Add four targets on each corner of the box and change its orientation for the robot 1 being able to reach all of them. It can be used the “Snap object” tool for better precision on the location of the targets. It is useful also to create a fifth target as the home position.

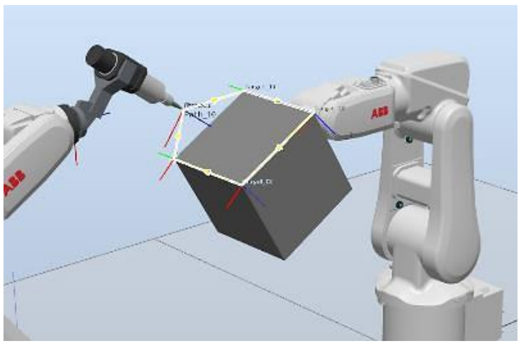


Figure 106. Multimove tool - screenshot 6

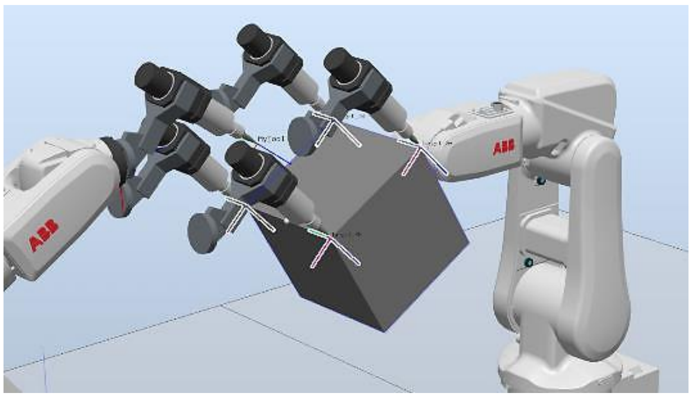
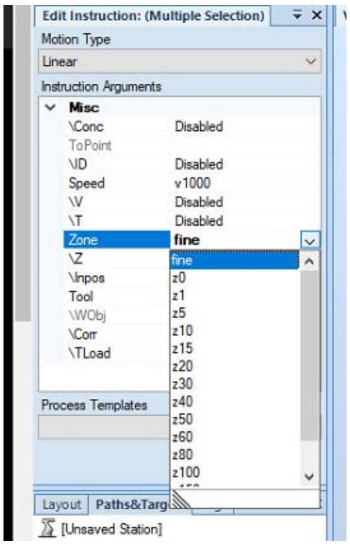


Figure 107. Multimove tool - screenshot 7

8. Create the path

Insert all the targets in the desirable order into an empty path. Change the instruction of each movement for a precision “fine”.



Edit Instruction: (Multiple Selection)

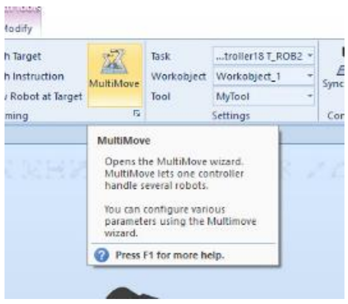
Motion Type: Linear

Instruction Arguments

Misc	
\Conc	Disabled
ToPoint	
\ID	Disabled
Speed	v1000
\V	Disabled
\T	Disabled
Zone	fine
\Z	
\Inpos	
Tool	
\WObj	
\Corr	
\TLoad	

Figure 108. Multimove tool - screenshot 8

9. Use the MultiMove function



Task: ...troller18 T_ROB2
Workobject: Workobject_1
Tool: MyTool

MultiMove

Opens the MultiMove wizard. MultiMove lets one controller handle several robots.

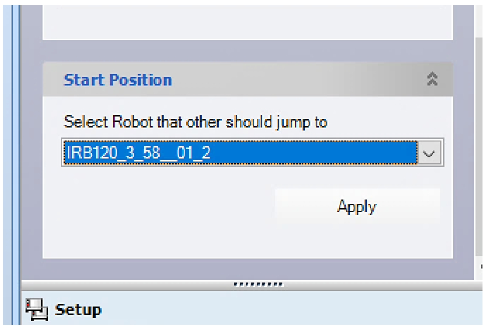
You can configure various parameters using the Multimove wizard.

Press F1 for more help.

Figure 109. Multimove tool - screenshot 9

A new pop-up window will appear pressing this function to configure the program. The user should insert that the robot2 has the object and the robot1 has the tool. Another information that the program will ask for is the work object and the user should insert the one created (WorkObject_1). Once all the information has been added, the user can close the pop-up window.

A MultiMove setup is appearing on the right side of the screen. On the section path configuration in the setup, the user should press "Update". On the section start position, the program is asking which robot the other should jump to, the user should select the robot 2.

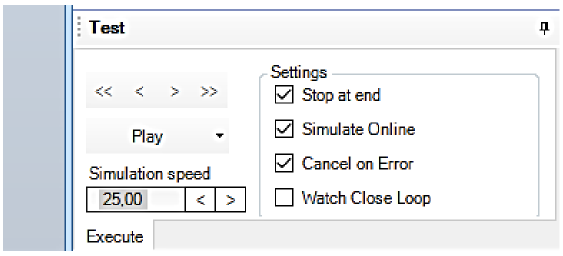

Start Position

Select Robot that other should jump to

IRB120_3_58__01_2

Figure 110. Multimove tool - screenshot 10

On the end of this menu, there is the section for test the function multimove and simulate it. It can also be changed the speed and other settings.


Test

Simulation speed 25,00

Settings
- [✓] Stop at end
- [✓] Simulate Online
- [✓] Cancel on Error
- [ ] Watch Close Loop

Figure 111. Multimove tool - screenshot 11

10.6. Pick and place with two robots

This simulation is similar to the first one, an action of pick and place, however, this time the program is working with two robots on the same controller. The first robot will pick the object from the first platform and leave it to the second. The second robot will modify the position of the object from the second to the third base. The following steps have been followed:

1. Insert two robots.

Insert two robots and locate them in a distance of 1000 mm with the same orientation.

2. Insert two tools.

Import from the library two tools "MyTool" and attach each with one robot.

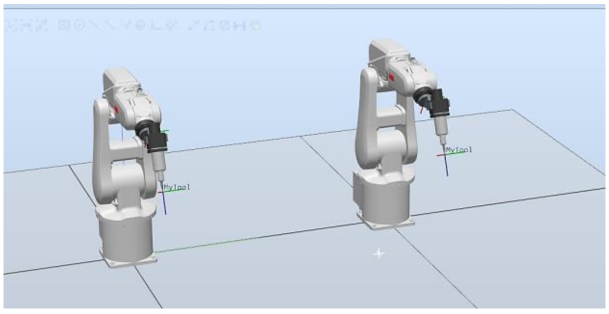


Figure 112. Pick and place with two robots - screenshot 1

3. Create solids.

Add to the station the three platforms that the robot will pick up or leave the object over it. In this case, it has been created solids with a size of 300 x 300 x 100 mm. The three solids are located in a distance of 500 between the others, with a robot in the middle.

4. Create a cylinder.

The cylinder is the object that the robot will move from one platform to another. The solid has a radius of 100mm and a height of 200mm. It will be copied and located one on each platform previous created. The user should set the local origin of these objects in the centre.

5. Create new work object.

It should be created two work objects located on the robot's base. When the user is creating them, it is important to select the task choosing which robot is going to use each work object.

6. Create targets and path

It is important to modify the settings on the menu “Home” selecting the task, work object and tool before creating the targets. It will be created two targets over each cylinder, with a distance of 30 mm for instance. The middle cylinder will have over itself four targets, two for each robot (and for each work object). Change the orientation of the targets. Insert an empty path for each robot and put the targets in the desired order.

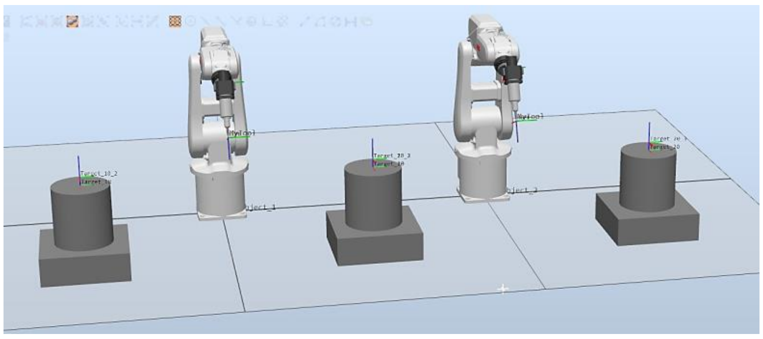


Figure 113. Pick and place with two robots - screenshot 2

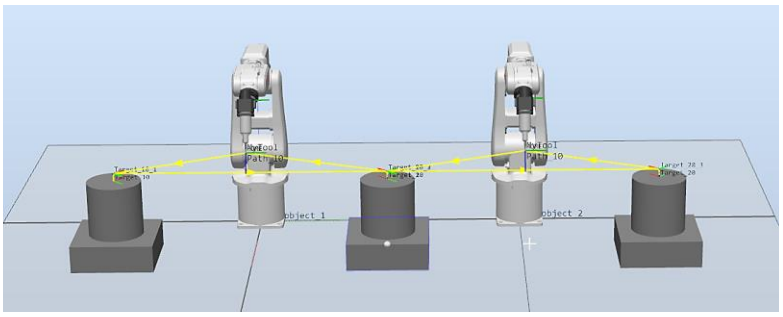


Figure 114. Pick and place with two robots - screenshot 3

7. Create signals in the controller.

It will be created two digital inputs (*rob1* and *rob2*) and one digital output (*startcontroller*) in the controller configuration. After modifying the configuration, it is needed always to restart the controller to apply the changes.

11. Use of the FlexPendant

The robot system can be modified by the RobotStudio or the FlexPendant. This last device is a handheld operator unit for manipulating the different tasks of the robot system. The physical tool can be used to manage a physical robot. However, RobotStudio has the option to manipulate the simulations with a virtual FlexPendant. A few tasks using this new instrument will be shown.

First, the user should insert the robot and create a virtual controller from the layout. The virtual FlexPendant can be found on the menu "Controller" (see Figure 115).

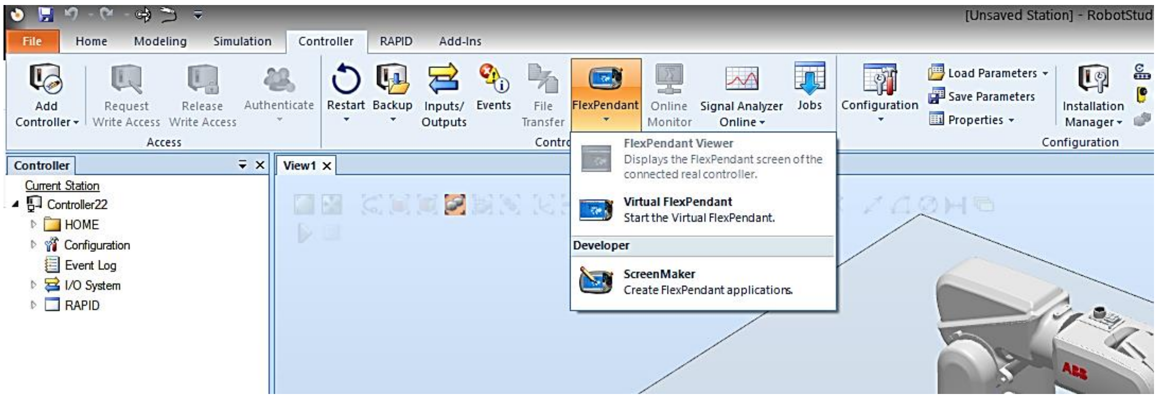
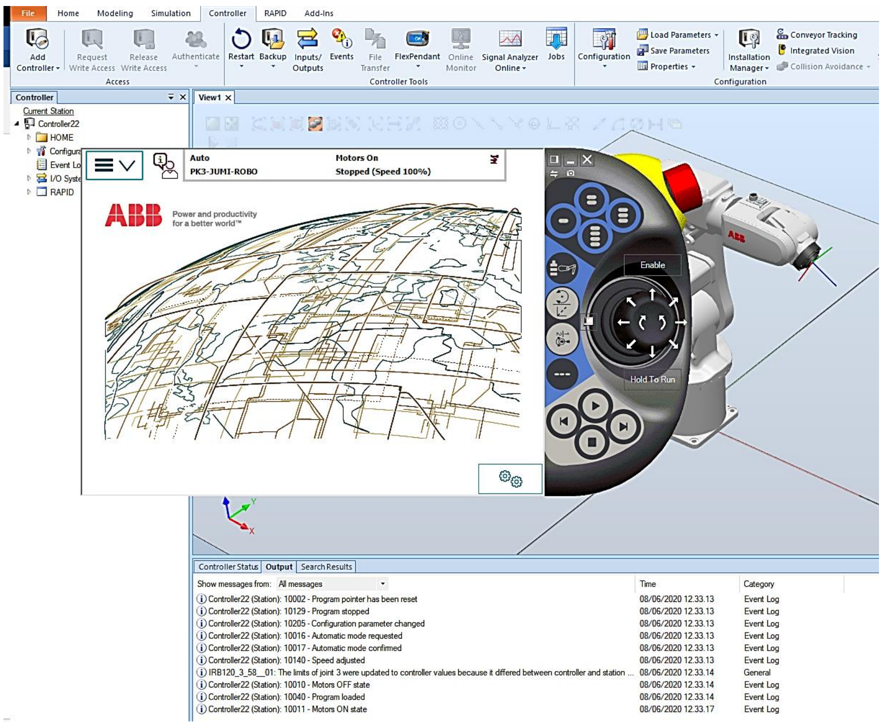


Figure 115. Use of the FlexPendant - screenshot 1



```
Show messages from:  All messages                                                        Time                  Category
Controller22 (Station): 10002 - Program pointer has been reset                           08/06/2020 12.33.13   Event Log
Controller22 (Station): 10129 - Program stopped                                          08/06/2020 12.33.13   Event Log
Controller22 (Station): 10205 - Configuration parameter changed                          08/06/2020 12.33.13   Event Log
Controller22 (Station): 10016 - Automatic mode requested                                 08/06/2020 12.33.13   Event Log
Controller22 (Station): 10017 - Automatic mode confirmed                                 08/06/2020 12.33.13   Event Log
Controller22 (Station): 10140 - Speed adjusted                                           08/06/2020 12.33.13   Event Log
IRB120_3_58__01: The limits of joint 3 were updated to controller values because it differed between controller and station ...   08/06/2020 12.33.14   General
Controller22 (Station): 10010 - Motors OFF state                                         08/06/2020 12.33.14   Event Log
Controller22 (Station): 10040 - Program loaded                                           08/06/2020 12.33.14   Event Log
Controller22 (Station): 10011 - Motors ON state                                          08/06/2020 12.33.17   Event Log
```

Figure 116. Use of the FlexPendant - screenshot 2

Editing the program by the FlexPendant is editing the RAPID code of the system. For being able to start working through this new device, two properties have to been changed. The first one is that the controller must be in manual mode. There is a bottom next to the Joystick which a pop-up window will appear for modifying it (see Figure 117). The other one is just to put the motors on with the ENABLE bottom. The changes will be shown on the FlexPendant top (see Figure 118).

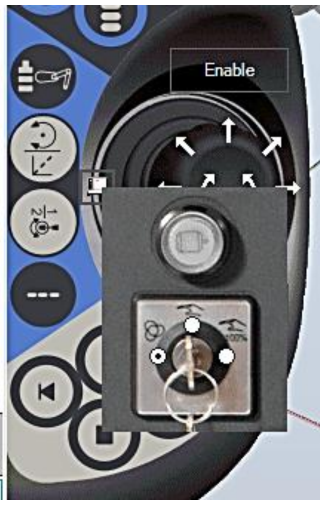
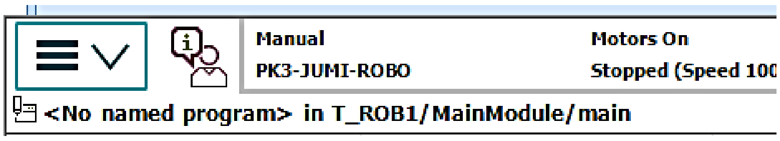


Figure 117. Use of the FlexPendant - screenshot 3

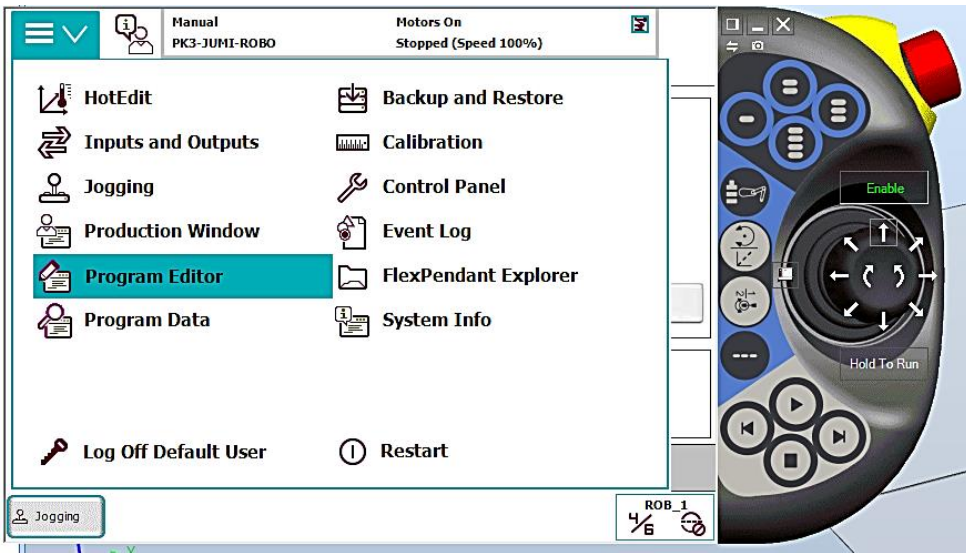


Manual
PK3-JUMI-ROBO
Motors On
Stopped (Speed 100

<No named program> in T_ROB1/MainModule/main

Figure 118. Use of the FlexPendant - screenshot 4

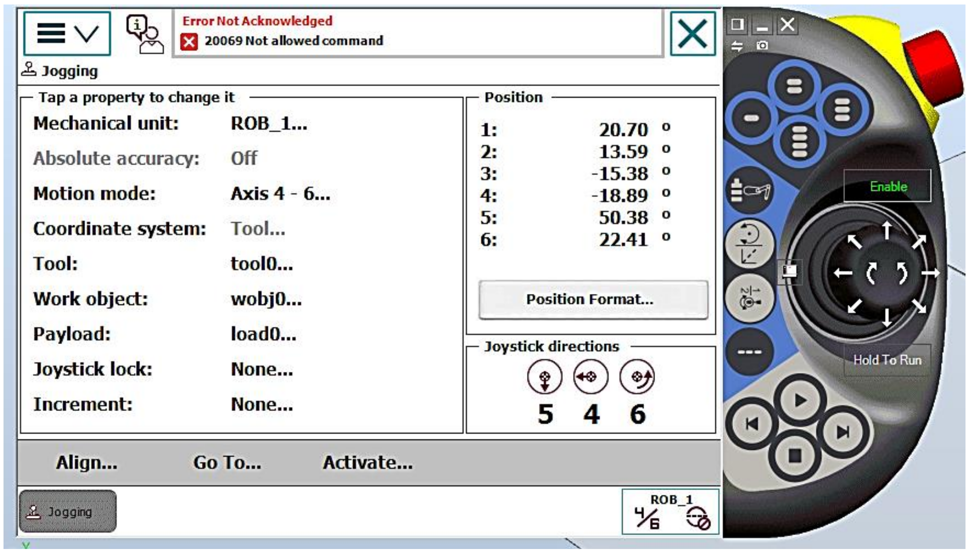
The user can unfold the following menu to find the different tasks that the FlexPendant is able to do.



Manual
PK3-JUMI-ROBO
Motors On
Stopped (Speed 100%)

Figure 119. Use of the FlexPendant - screenshot 5

On the Jogging menu (see Figure 120), different properties can be changed. These properties are included in each instruction of the program, as the tool or the WorkObject.


Error Not Acknowledged
20069 Not allowed command
Jogging
Tap a property to change it

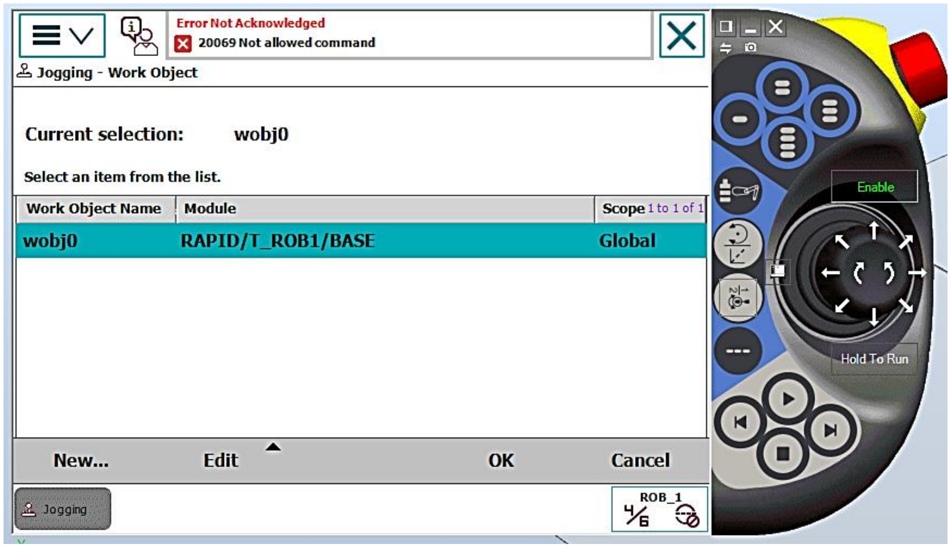
Mechanical unit:	ROB_1...
Absolute accuracy:	Off
Motion mode:	Axis 4 - 6...
Coordinate system:	Tool...
Tool:	tool0...
Work object:	wobj0...
Payload:	load0...
Joystick lock:	None...
Increment:	None...

Position

1:	20.70 °
2:	13.59 °
3:	-15.38 °
4:	-18.89 °
5:	50.38 °
6:	22.41 °

Figure 120. Use of the FlexPendant - screenshot 6

For example, it is needed to create a new WorkObject before inserting new targets on the program. Pressing over the property name on the Jogging menu, there is the option (see Figure 121) to edit (its position for instance) or add a new one.


Error Not Acknowledged
20069 Not allowed command
Jogging - Work Object

Current selection: wobj0

Select an item from the list.

Work Object Name	Module	Scope 1 to 1 of 1
wobj0	RAPID/T_ROB1/BASE	Global

Figure 121. Use of the FlexPendant - screenshot 7

It is common to program the targets moving the robot manually. On the menu Jogging, the different joystick directions can be changed using the two grey bottoms next to the joystick, for move in the different axes the robot with the joystick. The bottom which is over is for change to the axis and the other is for change to the robot joints motors.

12. Results

On the final section of the report could be found the different simulations of this project, which are the final results. There are a total of six simulations which it has been tried to increase the level on each.

The first simulation is the typical case Pick and Place which can be found a lot of examples on the Internet. This is about a robot taking an object from one location and move it to a different one.

The second and third simulations are about one RobotStudio tool which is the Mechanism. A mechanism is an object with their movements on the different axis on their joints. The second simulation is more focused on the creation of this material and how to work with it. The third one is a mix between the first and second simulation. The robot will do the action of pick and place over the mechanism.

Increasing the level on the next simulation, the “Transportation guides” was the longer one to implement. It is about the action of pick and place over two guides. This simulation has been designed to work with the Smart components which are a huge tool on the software.

On the last simulations, the number of robots is changed for work with a pair of them. The next simulation is the introduction to the Multimove tool. It is about working with two robots on the same controller, that means that the movement of them will be synchronized. The last simulation has two controllers, one for each robot. That means that each robot has to be programmed separately with their signals (input and output) and their smart components.

Each simulation has been implemented and explained for a future copy of them to learn about robots and programming. A simple manual about simple actions with the software could be also found on this report, before the simulations section. It is important to have a basic idea of how the program work before start simulating.

13. Discussion

To meet the aim of the project three objectives were established. The first objective planned was to acquire knowledge about the program, get used to it and learn about RAPID, the programming language used on this software. It has been needed to watch as many videos and tutorials as possible from different Youtube accounts. Most of them were explaining the different tools of the program, as there has been explained on the simple manual of this report. It has also been used some complete manuals from ABB about the software.

The next two objectives are to develop a range of examples of robot simulation and establish a simple manual step by step of how to reach each simulation. They are connected and they have worked in parallel. The process followed to prepare each simulation was the following: first of all, it is important to have an idea of the simulation which it is wanted to implement and for it a range of videos have been watched and a lot of information have been read. Once the idea was chosen, the design of the simulation is created with the interface (how all the objects will be located on the station of the software) and the idea of the code (what is needed for work the simulation). The next step was to go to Technobotnia and make the first attempt of creating the simulation on RobotStudio. The issues and doubts emerged and the following plan was to figure them out. Depending on the simulation, this process took one day or a few days. Once the simulation is done and works perfectly, the document is written with all the steps followed.

The time was the biggest limitation on this project because it was working against the clock. The first setback on this project was the software availability. It was not able to work from home on the program because it was not free, so every time it was needed to go to Technobotnia and work from there. The pandemic did not either help with the laboratory access. The tutor had to provide a key to the student to keep working form the university on the software.

While working on the practical part of the laboratory, some problems showed up. The minimum problem that occurred was that sometimes the computer on which the project was being worked on was not available. Then, in some simulations, there were problems with the commissioning. The connections with the smart components did not work or did not go as they should, for example. After some time dedicated to each one of these problems, sometimes it was possible to solve them and sometimes it was necessary to look for another way to reach the objective. Other times, for instance, the difficulty of the mechanism which was designed first, it was unreachable with the knowledge acquired until the moment.

The simulation ideas were taken from the internet by watching some videos of industrial robots and some of the other users who work with RobotStudio. The first idea to get ideas was to go to a company with the customer, Roger Mäntylä, and see how the robots work on the industry. However, as a result of the coronavirus, it was not possible to make any visits.

Fortunately, the three objectives have been accomplished and as a consequent, the aim has been completed. It is important to consider that this project is something introductory to the software RobotStudio and the material worked on it is basic. The possibilities which the program gives to the user are huge and require much more time to work with it.

14. Conclusion

The use of simulations is crucial for most of the experiments and studies. It has a pile of advantages related to the time and the cost. On the industrial robot and its programme, the simulation is an utmost interest in the companies. ABB has created its software to create simulations of the programming of its robots installed on different companies.

In this project, a new way of programming has been worked. It has learned to get smart to find all the information about what was needed about the program and the code. The thesis was initially quite open and could cover many possibilities of simulation, as the program allows. It is the student who has been working to focus the project and to be able to reach a reasonable result: six simulations in which the knowledge is increasing in each one of them and in which concepts and tools are repeated. This last part is important because this work is focused on future use in an initiation course in the university about programming in RobotStudio.

The future simulations will be to work with more examples of smart components in different cases, as there are a lot of them. If there had been more time in Finland, the following simulation would have been adding a robot to the simulation four “Transportation guides”. This way it would have worked with the smart components in two different controllers. The simulation was tried and it was worked in its moment, but it did not arrive at the objective with the time that there was.

This project has helped to increase the programming knowledge of the student, and not only as a new language but also as a new way of thinking. The programming in this project has ceased to be a code to be a combination of different concepts.

15. References

ABB, 2020. *ABB.* [Online] Available at: https://fortune.com/global500/abb/
[Accessed 1 August 2020].

ABB, 2020. *ABB Ability Smart Sensor - ABB Advanced Services (ABB Service for Motors and Generators).* [Online] Available at: https://new.abb.com/motors-generators/service/advanced-services/smart-sensor?_ga=2.3369461.452141795.1596301595-622681349.1596301595
[Accessed 1 August 2020].

ABB, 2020. *History of ABB.* [Online] Available at:
https://global.abb/group/en/about/history
[Accessed 1 August 2020].

ABB, 2020. *Innovation, T. and innovations, A.*. [Online]
Available at: https://global.abb/group/en/technology/innovations
[Accessed 1 August 2020].

ABB, 2020. *Our businesses - ABB group.* [Online]
Available at: https://new.abb.com/about/our-businesses
[Accessed 1 August 2020].

ABB, 2020. *RobotStudio - ABB Robotics.* [Online]
Available at: https://new.abb.com/products/robotics/robotstudio
[Accessed 10 September 2020].

ABB, 2020. *Smart charging infrastructure for Singapore port's automated guided vehicles.*. [Online]
Available at: https://new.abb.com/news/detail/63868/smart-charging-infrastructure-for-singapore-ports-automated-guided-vehicles
[Accessed 1 August 2020].

ABB, 2020. *Technology: Digital Products. About ABB Ability IoT Platform.* [Online]
Available at: https://ability.abb.com/about/technology/
[Accessed 1 August 2020].

Anon., 2019. *If robots take our jobs, what will it mean for climate change?.* [Online]
Available at: https://theconversation.com/if-robots-take-our-jobs-what-will-it-mean-for-climate-change-123507
[Accessed 17 April 2020].

Anon., 2020. *ABB.* [Online]
Available at: https://fortune.com/global500/abb/
[Accessed 20 April 2020].

Anon., 2020. *Simulation Team Table of Contents.* [Online]
Available at: https://uh.edu/~lcr3600/simulation/contents.html
[Accessed 19 July 2020].

Ayers, Whitlow & Dressler, 2020. *NHTSA: Nearly all car crashes are due to human error.* [Online]
Available at: https://www.ayersandwhitlow.com/blog/2018/01/nhtsa-nearly-all-car-crashes-are-due-to-human-error.shtml
[Accessed 17 April 2020].

BostonDynamics, 2020. *Spot, Boston Dynamics.* [Online]
Available at: https://www.bostondynamics.com/spot
[Accessed 16 April 2020].

Builtin, 2020. *What is Artificial Intelligence? How does AI work?.* [Online]
Available at: https://builtin.com/artificial-intelligence
[Accessed 19 July 2020].

Cox, B., 2020. *The Evolution of Industrial robotics.* [Online]
Available at: https://www.sutori.com/story/the-evolution-of-industrial-robotics--ESrMeSubei76BonChmqmtgkr
[Accessed 9 September 2020].

Deahl, D., 2018. *Daisy is Apple's New iPhone-recycling robot.* [Online]
Available at: https://www.theverge.com/2018/4/19/17258180/apple-daisy-iphone-recycling-robot
[Accessed 19 April 2020].

European Engineering Industries Association, 2020. *Robotics Market Introduction.* [Online]
Available at: https://www.eu-nited.net/eunited+aisbl/sectors/robotics/robotics-market-introduction/index.html
[Accessed 16 April 2020].

Executive Summary World Robotics 2019 Industrial Robots, 2019. *Executive Summary World Robotics 2019 Industrial R0bots.* [Online]
Available at:
https://ifr.org/downloads/press2018/Executive%20Summary%20WR%202019%20Industrial%20Robots.pdf
[Accessed 25 April 2020].

Gabbatiss, J., 2019. *Robot "Shark" that eats plastic waste launched to tackle pollution.* [Online]
Available at: https://www.independent.co.uk/environment/robot-shark-plastic-pollution-wasteshark-wwf-devon-ilfracombe-a8805681.html
[Accessed 17 April 2020].

IEEE Spectrum, 2020. *ROBOTS: your guide to the world of robotics.* [Online]
Available at: https://robots.ieee.org/learn/
[Accessed 20 April 2020].

infoPLC, 2020. *ABB ofrece el simulador RobotStudio de forma gratuita ante COVID-19.* [Online]
Available at: https://www.infoplc.net/noticias/item/107658-abb-simulador-robotstudio-gratis-ante-covid-19
[Accessed 10 September 2020].

Internation Federation of Robotics, 2020. *International Federation of Robotics.* [Online]
Available at: https://www.ifr.org/robot-history
[Accessed 9 September 2020].

IQBAL, J., KHAN, Z. & KHALID, A., 2017. Food science and Technology.. In: *Prospects of robotics in food industry..* s.l.:s.n., pp. 159-165.
Kedia, A., 2020. *The quest for artificial general intelligence.* [Online]
Available at: https://towardsdatascience.com/the-quest-for-artificial-general-intelligence-34ecf9e7e88e
[Accessed 20 July 2020].

Kudlak, L., 2019. *Protecting the environment with automation and robotics engineering.* [Online]
Available at: https://medium.com/tech4planet/protecting-the-environment-with-automation-and-robotics-engineering-44c0a176c18f
[Accessed 17 April 2020].

Linder, C., 2020. *Suction Cups Kinda Suck. This Wall-Climbing robot could make them better..* [Online]
Available at:
https://www.popularmechanics.com/technology/robots/a30609685/wall-climbing-robot/
[Accessed 17 April 2020].

Linder, C., 2020. *What This PigeonBot tells us about the future of the flight.* [Online]
Available at:
https://www.popularmechanics.com/technology/robots/a30564900/pigeonbot-future-of-flight/
[Accessed 17 April 2020].

Massachusetts Institutes of Technology, 2020. *MIT engineers develop a new way to remove carbon dioxide from air.* [Online]
Available at: http://news.mit.edu/2019/mit-engineers-develop-new-way-remove-carbon-dioxide-air-1025 [Accessed 16 Apr. 2020].

McSweeney, K., 2017. *How and why Apple's robot Liam disassembles iPhone.* [Online]
Available at: https://www.zdnet.com/article/how-and-why-apples-robot-liam-disassembles-iphones/
[Accessed 9 September 2020].

Michelini, R. C., 2014. *Robots in Medicine: a survey of in-body nursing aids introductory overview and concept design hints.*. [Online]
Available at:
https://www.researchgate.net/profile/Rinaldo_Michelini/publication/237682894_ROBOTS_IN_MEDICINE_A_SURVEY_OF_IN-BODY_NURSING_AIDS_INTRODUCTORY_OVERVIEW_AND_CONCEPT_DESIGN_HINTS/links/00b495278c27cdb7e2000000/ROBOTS-IN-MEDICINE-A-SURVEY-OF-IN-BODY-NURSING-A
[Accessed 9 September 2020].

Mohammed Bin Rashid Initiative for Global Prosperity, 2020. *Recycling Robots by Dorabot - About Solution.* [Online]
Available at: https://makingprosperity.com/solutions-details/recycling-robots-by-dorabot
[Accessed 18 April 2020].

NRDC, 2020. *Water Pollution: Everything You need to know.*. [Online]
Available at: https://www.nrdc.org/stories/water-pollution-everything-you-need-know
[Accessed 17 April 2020].

O'Carroll, B., 2020. *What are the 3 types of AI? A guide to narrow, general, and super artificial intelligence.* [Online]
Available at: https://codebots.com/artificial-intelligence/the-3-types-of-ai-is-the-third-even-possible
[Accessed 19 July 2020].

Ohio University, 2018. *Top 5 Industries Utilizing Robotics.* [Online]
Available at: https://onlinemasters.ohio.edu/blog/5-industries-utilizing-robotics/
[Accessed 20 April 2020].

Philamore, H., 2015. *A row-bot that loves dirty water.* [Online]
Available at: https://phys.org/news/2015-11-row-bot-dirty.html
[Accessed 17 April 2020].

Robot Worx SCOTT company, 2020. *Industrial Robot History.* [Online]
Available at: https://www.robots.com/articles/industrial-robot-history
[Accessed 9 August 2020].

Robotics, L. C. f. S., 2015. *Defining robots and robotics.* [Online]
Available at: http://www.leorobotics.nl/definition-robots-and-robotics
[Accessed 16 April 2020].

RobotWorx, 2020. *Benefits of Using Robotics.* [Online]
Available at: https://www.robots.com/articles/benefits-of-using-robotics
[Accessed 20 April 2020].

Rogers, C. D., 2018. *The Effects of Carbon Dioxide on Air Pollution.* [Online]
Available at: https://sciencing.com/list-5921485-effects-carbon-dioxide-air-pollution.html
[Accessed 18 April 2020].

Schiller, B., 2015. *These experimental micro-robots are designed to take CO2 out of the ocean.* [Online]
Available at: https://www.fastcompany.com/3051560/these-experimental-micro-robots-are-designed-to-take-co2-out-of-the-ocean
[Accessed 17 April 2020].

School of Electrical and Electronic Engineering Nanyang Technological University, 2020. *A new generation of military robots.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/1333028
[Accessed 20 April 2020].

ScienceDaily, 2020. *Living robots built using frog cells: Tiny "xenobots" assembled from cells promise advances from drug delivery to toxic waste clean-up..* [Online]
Available at: https://www.sciencedaily.com/releases/2020/01/200113175653.htm
[Accessed 17 April 2020].

Simon, M., 2017. *What is a robot?.* [Online]
Available at: https://www.wired.com/story/what-is-a-robot/
[Accessed 16 April 2020].

Sorensen, C. G., 2015. *Mission planner for agricultural robotics.* [Online]
Available at:
https://www.researchgate.net/profile/Claus_Sorensen3/publication/266499487_MISSION_PLANNER_FOR_AGRICULTURAL_ROBOTICS/links/5560603d08ae86c06b643119.pdf
[Accessed 9 September 2020].

SPARC The Partnership for Robotics in Europe, 2020. *Why is Robotics important?.* [Online]
Available at: https://www.eu-robotics.net/sparc/about/robotics-in-europe/index.html
[Accessed 16 April 2020].

Team, R. O. M., 2017. *Robotics in Agriculture: Types and Applications.* [Online]
Available at: https://www.robotics.org/blog-article.cfm/Robotics-in-Agriculture-Types-and-Applications/74
[Accessed 17 April 2020].

Temming, M., 2020. *"PigeonBot" is the first robot that can bend its wings like a real bird..* [Online]
Available at: https://www.sciencenews.org/article/new-robot-pigeon-can-bend-wings-like-real-bird
[Accessed 17 April 2020].

TESLA, 2020. *Tesla.* [Online]
Available at: https://www.tesla.com/es_MX/autopilot?redirect=no
[Accessed 9 September 2020].

ThePermacultureResearchInstitute, 2020. *Pollution-Eating Robots Could Help Protect the Planet.* [Online]
Available at: https://www.permaculturenews.org/2017/04/17/pollution-eating-robots-help-protect-planet/
[Accessed 17 April 2020].

Thesen, A., Grant, H. & Kelton, W. D., 2020. *Discrete, continous and combined simulation.* [Online]
Available at:
https://repository.lib.ncsu.edu/bitstream/handle/1840.4/7464/1987_0004.pdf?sequence=1
[Accessed 19 July 2020].

TWI, 2020. *What is industry 4.O? How does it work? (A beginners guide).* [Online]
Available at: https://www.twi-global.com/what-we-do/research-and-technology/technologies/industry-4-0
[Accessed 1 August 2020].

Wallén, J., 2008. *The history of the industrial robot.* [Online]
Available at: http://liu.diva-portal.org/smash/get/diva2:316930/FULLTEXT01.pdf
[Accessed 9 September 2020].

WAYMO, 2020. *Technology - Waymo.* [Online]
Available at: https://waymo.com/tech/
[Accessed 17 April 2020].

Worldmeters, 2020. *World Population Clock: 7.8 Billion People.* [Online]
Available at: https://www.worldometers.info/world-population/
[Accessed 16 April 2020].

Yeung, J. C., 2020. *Scientist have built the world's first living, self-healing robots..* [Online]
Available at: https://edition.cnn.com/2020/01/13/us/living-robot-stem-cells-intl-hnk-scli-scn/index.html
[Accessed 17 April 2020].

ABB AB, 2011. *Operating manual RobotStudio 5.14.* E ed. Västerâs, Sweden: ABB AB Robotic Products. Available at:
https://library.e.abb.com/public/e5ad00148905fb58c1257b4b00523751/3HAC032104-001_revE_en.pdf. [Accessed 17 May 2020].

ABB AB Robotics Products, 2010. *Technical reference manual. RAPID instructions, functions and data types. RobotWare 5.13.* J ed. Västerâs, Sweden: ABB AB.

ABB AB, 2007. *Operating manual. Introduction to RAPID. RobotWare 5.0.* 5.0 ed. Västerâs, Sweden: ABB AB. Available at: https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf. [Accessed 17 May 2020].

ABB AB, 2011. *Operating manual RobotStudio 5.14.* E ed. Västerâs, Sweden: ABB AB Robotic Products. Available at: http://dl.icdst.org/pdfs/files3/db9fddeb58803077290aa2538c54333d.pdf. [Accessed 17 May 2020].

16. Appendices

16.1. Smart components

Following the section of basic smart components with its uses and its properties could be found. This section has been used a lot during the simulation examples on the thesis.

Source:

ABB AB, 2011. *Operating manual RobotStudio 5.14.* E ed. Västerâs, Sweden: ABB AB Robotic Products. Available at:
https://library.e.abb.com/public/e5ad00148905fb58c1257b4b00523751/3HAC032104-001_revE_en.pdf.

Pages 258 – 273

9.4.7. Basic Smart Components

Overview

The base components represent a complete set of basic building block components. They can be used to build user defined Smart Components with more complex behavior.

This lists the basic Smart Components available and are described in the following sections:

- *Signals and Properties on page 258*
- *Parametric Primitives on page 262*
- *Sensors on page 264*
- *Actions on page 266*
- *Manipulators on page 268*
- *Other on page 270*

Signals and Properties

LogicGate

The signal Output is set by the logical operation specified in Operator of the two signals InputAand InputB, with the delay specified in Delay.

Properties	**Description**
Operator	The logical operator to use. The following lists the various operators: • AND • OR • XOR • NOT • NOP
Delay	Time to delay the output signal.

Signals	**Description**
InputA	The first input.
InputB	The second input.
Output	The result of the logic operation.

LogicMux

Output is set according to: Output = (Input A * NOT Selector) + (Input B * Selector)

Signals	**Description**
Selector	When low, the first input is selected. When high, the second input is selected.
InputA	Specifies the first input.
InputB	Specifies the second input.
Output	Specifies the result of the operation.

Continues on next page

LogicSplit

The LogicSplit takes Input and sets OutputHigh to the same as Input, and OutputLow as the inverse of Input.

PulseHigh sends a pulse when Input is set to high, and PulseLow sends a pulse when Input is set to low.

Signals	**Description**
Input	Specifies the input signal.
OutputHigh	Goes high (1) when input is 1.
OutputLow	Goes high (1) when input is 0.
PulseHigh	Sends pulse when input is set to high.
PulseLow	Sends pulse when input is set to low.

LogicSRLatch

TheLogicSRLatch has one stable state.

- When Set=1, Output=1 and InvOutput=0
- When Reset=1, Output=0 and InvOutput=1

Signals	**Description**
Set	Sets the output signal.
Reset	Resets the output signal.
Output	Specifies output signal.
InvOutput	Specifies Inverse output signal.

Converter

Converts between property values and signal values.

Properties	**Description**
AnalogProperty	Converts to AnalogOutput.
DigitalProperty	Converts to DigitalOutput.
GroupProperty	Converts to GroupOutput.
BooleanProperty	Converts from DigitalInput and to DigitalOutput.

Signals	**Description**
DigitalInput	Converts to DigitalProperty.
DigitalOutput	Converted from DigitalProperty.
AnalogInput	Converts to AnalogProperty.
AnalogOutput	Converted from AnalogProperty.
GroupInput	Converts to GroupProperty.
GroupOutput	Converted from GroupProperty.

VectorConverter

Converts between Vector3 and X, Y,and Z values.

Properties	**Description**
X	Specifies the X-value of Vector.

Continues on next page

Properties	Description
Y	Specifies the Y-value of Vector.
Z	Specifies the Z-value of Vector
Vector	Specifies the vector value..

Expression

The Expression consists of numeric literals (including PI), parentheses, mathematical operators +,-,*,/,^ (power) and mathematical functions sin, cos, sqrt, atan, abs. Any other strings are interpreted as variables, which are added as additional properties. The result is displayed in Result.

Signals	Description
Expression	Specifies the expression to evaluate.
Result	Specifies the result of evaluation.
NNN	Specifies automatically generated variables.

Comparer

The Comparer compares First value with Second value, using Operator. Output is set to 1 if the condition is met.

Properties	Description
ValueA	Specifies the first value.
ValueB	Specifies the second value.
Operator	Specifies the comparison operator. The following lists the various operators: • == • != • > • >= • < • <=

Signals	Description
Output	True if the comparison evaluates to true, otherwise false.

Counter

Count is increased when the input signal Increase is set, and decreased when the input signal Decrease is set. Count is reset when the input signal Reset is set.

Properties	Description
Count	Specifies the current count.

Signals	Description
Increase	Adds one to the Count when set to True.
Decrease	Subtracts one from the Count when set to True.
Reset	Resets the Count to zero when set to high.

Continues on next page

Continued

Repeater

Pulses Output signal Count number of times.

Properties	Description
Count	Number of times to pulse Output.

Signals	Description
Execute	Set to high (1) to pulse Output Count times.
Output	Output pulse.

Timer

The Timer pulses the Output signal based on the given interval.

If Repeat is unchecked, one pulse will be triggered after the time specified in Interval. Otherwise, the pulse will be repeated at the interval given by Interval.

Properties	Description
StartTime	Specifies the time to pass before the first pulse.
Interval	Specifies the simulation time between the pulses.
Repeat	Specifies if the signal should be pulsed repeatedly or only once.
Current time	Specifies the current simulation time.

Signals	Description
Active	Set to True to activate the timer, and False to deactivate it.
Output	Sends pulses at the specified time intervals.

StopWatch

The StopWatch measures time during simulation (TotalTime). A new lap can be started by triggering the Lap input signal. LapTime shows the current lap time. The time is only measured when Active is set to 1. The times are reset when the input signal Reset is set.

Properties	Description
TotalTime	Specifies the accumulated time.
LapTime	Specifies the current lap time.
AutoReset	If true, TotalTime and LapTime will be set to 0 when the simulation starts.

Signals	Description
Active	Set to True to activate the stop watch, and False to deactivate it.
Reset	Resets Total time and Lap time when set to high.
Lap	Starts a new lap.

Continues on next page

Parametric Primitives

ParametricBox

The ParametricBox generates a box with dimensions specified by length, width, and height.

Properties	Description
SizeX	Specifies the length of the box in the X-axis direction.
SizeY	Specifies the length of the box in the Y-axis direction.
SizeZ	Specifies the length of the box in the Z-axis direction
GeneratedPart	Specifies the generated part.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

Signals	Description
Update	Set to high (1) to update the generated part.

ParametricCircle

The ParametricCircle generates a circle with a given radius.

Properties	Description
Radius	Specifies the radius of the circle.
GeneratedPart	Specifies the generated part.
GeneratedWire	Specifies the generated wire object.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster

Signals	Description
Update	Set to high (1) to update the generated part.

ParametricCylinder

The ParametricCylinder generates a cylinder with the dimensions given by Radius and Height.

Properties	Description
Radius	Specifies the radius of the cylinder.
Height	Specifies the height of the cylinder.
GeneratedPart	Specifies the generated part.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

Signals	Description
Update	Set to high (1) to update the generated part.

Continues on next page

Continued

ParametricLine

The ParametricLine generates a line with a given end point or a given length. If either of them is changed, the other one will be updated accordingly.

Properties	Description
EndPoint	Specifies the end point for the line.
Length	Specifies the length of the line.
GeneratedPart	Specifies the generated part.
GeneratedWire	Specifies the generated wire object.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

Signals	Description
Update	Set to high (1) to update the generated part.

LinearExtrusion

The LinearExtrusion extrudes SourceFace or SourceWire along the vector given by Projection.

Properties	Description
SourceFace	Specifies the face to extrude.
SourceWire	Specifies the wire to extrude.
Projection	Specifies the vector to extrude along.
GeneratedPart	Specifies the generated part.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

CircularRepeater

The CircularRepeater creates a number of given copies of Source around the center of the SmartComponent with a given DeltaAngle.

Properties	Description
Source	Specifies the object to copy.
Count	Specifies the number of copies to create.
Radius	Specifies the radius of the circle.
DeltaAngle	Specifies the angle between the copies.

LinearRepeater

The LinearRepeater creates a number of copies of Source, with the spacing and direction given by Offset.

Properties	Description
Source	Specifies the object to copy.
Offset	Specifies the distance between copies.
Count	Specifies the number of copies to create.

Continues on next page

MatrixRepeater

The MatrixRepeater creates a specified number of copies in three dimensions, with the specified spacing of the object in Source.

Properties	Description
Source	Specifies the object to copy.
CountX	Specifies the number of copies in the X-axis direction.
CountY	Specifies the number of copies in the Y-axis direction.
CountZ	Specifies the number of copies in the Z-axis direction.
OffsetX	Specifies the offset between the copies in the X-axis direction.
OffsetY	Specifies the offset between the copies in the Y-axis direction.
OffsetZ	Specifies the offset between the copies in the Z-axis direction.

Sensors

CollisionSensor

The CollisionSensor detects collisions and near miss events between the First object and the Second object. If one of the objects is not specified, the other will be checked against the entire station. When the Active signal is high and a collision or a near miss event occurs and the component is active, the SensorOut signal is set and the parts that participate in the collision or near miss event are reported in the first colliding part and second colliding partof the Property editor.

Properties	Description
Object1	The first object to check for collisions.
Object2	The second object to check for collisions.
NearMiss	Specifies the near miss distance.
Part1	The part of First object that has a collision.
Part2	The part of Second object that has a collision.
CollisionType	• None • Near miss • Collision

Signals	Description
Active	Specifies if the CollisionSensor is active or not.
SensorOut	True if there is a NearMiss or Collision.

Continues on next page

Continued

LineSensor

The LineSensor defines a line by the Start, End, and Radius. When an Active signal is high, the sensor detects objects that intersect the line. Intersecting objects are displayed in the ClosestPart property and the point on the intersecting part that is closest to the line sensors start point is displayed in the ClosestPoint property. When intersection occurs the SensorOut output signal is set.

Properties	**Description**
Start	Specifies the start point.
End	Specifies the end point.
Radius	Specifies the radius.
SensedPart	Specifies the part that intersects the line sensor. If several parts intersect, then the one closest to theStart point is listed.
SensedPoint	Specifies the point on the intersecting part, closest to the Start point.

Signals	**Description**
Active	Specifies if the LineSensor is active or not.
SensorOut	True if the sensor intersects with an object.

PlaneSensor

The PlaneSensor defines a plane by Origin, Axis1, and Axis2. When the Active input signal is set the sensor detects objects that intersect this plane. Intersecting objects are displayed in the SensedPart property and when intersection occurs the SensorOut output signal is set.

Properties	**Description**
Origin	Specifies the origin of the plane.
Axis1	Specifies the first axis of the plane.
Axis2	Specifies the second axis of the plane.
SensedPart	Specifies the part that intersects the PlaneSensor. If several parts intersect, then the one listed first in the Layout browser is selected.

Signals	**Description**
Active	Specifies if the PlaneSensor is active or not.
SensorOut	True if the sensor intersects with an object.

PositionSensor

The PositionSensor monitors the position and orientation of an object.

NOTE! The position and orientation of an object is updated only during the simulation.

Properties	**Description**
Object	Specifies the object to monitor.
Reference	Specifies the reference coordinate system (Parent or Global).
ReferenceObject	Specifies the reference object, if Reference is set to Object.

Continues on next page

Properties	Description
Position	Specifies the position of the object relative to Reference.
Orientation	Specifies the orientation (Euler ZYX) relative to Reference.

ClosestObject

The ClosestObject defines a Reference object or a Reference point. When the Execute signal is set, the component finds the ClosestObject, ClosestPart, and the Distance to the reference object, or to the reference point if the reference object is undefined. If RootObject is defined, the search is limited to that object and its descendants. When finished and the corresponding properties are updated the Executed signal is set.

Properties	Description
ReferenceObject	Specifies the object to get the closest object to.
ReferencePoint	Specifies the point to get the closest object to.
RootObject	Specifies the object whose children to search. Empty means entire station.
ClosestObject	Specifies the object closest to Reference object or Reference point.
ClosestPart	Specifies the part closest to Reference object or Reference point.
Distance	Specifies the distance between the Reference object and the Closest object.

Signals	Description
Execute	Set to True to find the Closest part.
Executed	Sends a pulse when completed.

Actions

Attacher

The Attacher will attach Child to Parent when the Execute signal is set. If the Parent is a mechanism, the Flange to attach to must also be specified. When the input Execute is set, the child object is attached to the parent object. If Mount is checked, the child will also be mounted on the parent, with the Offset and Orientation specified. The output Executed will be set when finished.

Properties	Description
Parent	Specifies the object to attach to.
Flange	Specifies the Index of mechanism flange to attach to.
Child	Specifies the object to attach.
Mount	If true, the object to attach mounts on the attachment parent.
Offset	Specifies the position relative to the attachment parent when using Mount.
Orientation	Specifies the orientation relative to the attachment parent when using Mount.

Continues on next page

Continued

Signals	Description
Execute	Set to True to attach.
Executed	Sends a pulse when completed.

Detacher

The Detacher will detach the Child from the object it is attached to when the Execute signal is set. If Keep position is checked, the position will be kept. Otherwise the child is positioned relative to its parent. When finished, the Executed signal will be set.

Properties	Description
Child	Specifies the object to detach.
KeepPosition	If false, the attached object is returned to its original position.

Signals	Description
Execute	Set to True to remove the attachment.
Executed	Sends a pulse when completed.

Source

The Source property of the source component indicates the object that should be cloned when the Execute input signal is received. The parent of the cloned objects is specified by the Parent property and a reference to the cloned object is specified by the Copy property. The output signal Executed signifies that the clone is complete.

Properties	Description
Source	Specifies the object to copy.
Copy	Specifies the copied object.
Parent	Specifies the parent to the copy. If not specified, the copy gets the same parent as the source.
Position	Specifies the position of the copy relative its parent.
Orientation	Specifies the orientation of the copy relative its parent.
Transient	Marks the copy as transient if created during simulation. Such copies are not added to the undo queue and are automatically deleted when the simulation is stopped. This is used to avoid increased memory consumption during simulation.

Signals	Description
Execute	Set to True to create a copy of the object.
Executed	Sends a pulse when completed.

Sink

The Sink deletes the object referenced by the Object property. Deletion happens when the Execute input signal is received. The Executed output signal is set when the deletion is finished.

Properties	Description
Object	Specifies the object to remove.

Continues on next page

Continued

Signals	Description
Execute	Set to True to remove the object.
Executed	Sends a pulse when completed.

Show

When the Execute signal is set, the object referenced in Object appears. When finished, Executed signal will be set.

Properties	Description
Object	Specifies the object to show.

Signals	Description
Execute	Set to True to show the object.
Executed	Sends a pulse when completed.

Hide

When the Execute signal is set, the object referenced in Object will be hidden. When finished, Executed signal will be set.

Properties	Description
Object	Specifies the object to hide.

Signals	Description
Execute	Set to True to hide the object.
Executed	Sends a pulse when completed.

Manipulators

LinearMover

The LinearMover moves the object referenced in the Object property with a speed given by the Speed property in the direction given by the Direction property. The motion starts when the Execute input signal is set and stops when Execute is reset.

Properties	Description
Object	Specifies the object to move.
Direction	Specifies the direction to move the object.
Speed	Specifies the speed of movement.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the reference object, if Reference is set to Object.

Signals	Description
Execute	Set to True to start move the object, and False to stop.

Continues on next page

Continued

Rotator

The Rotator rotates the object referenced in the Object property with an angular speed given by the Speed property. The axis of rotation is given by CenterPoint and Axis. The motion starts when the Execute input signal is set and stops when the Execute is reset.

Properties	Description
Object	Specifies the object to rotate.
CenterPoint	Specifies the point to rotate around.
Axis	Specifies the axis of the rotation.
Speed	Specifies the speed of the rotation.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the object which are relative to CenterPoint and Axis, if Reference is set to Object.

Signals	Description
Execute	Set to True to start rotating the object, and False to stop.

Positioner

The Positioner takes an Object, Position, and Orientation as properties. When the Execute signal is set the object is repositioned in the given position relative to the Reference. When finished the Executed output is set.

Properties	Description
Object	Specifies the object to position.
Position	Specifies the new position of the object.
Orientation	Specifies the new orientation of the object.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the object which are relative to Position and Orientation, if Reference is set to Object.

Signals	Description
Execute	Set to True to start move the object, and False to stop.
Executed	Set to 1 when operation is completed.

PoseMover

The PoseMover has a Mechanism, a Pose, and Duration as properties. When the Execute input signal is set the mechanism joint values are moved to the given pose. When the pose is reached the Executed output signal is set.

Properties	Description
Mechanism	Specifies the mechanism to move to a pose.
Pose	Specifies the Index of the pose to move to.
Duration	Specifies the time for the mechanism to move to the pose.

Continues on next page

Continued

Signals	Description
Execute	Set to True, to start or resume moving the mechanism.
Pause	Pauses the movement.
Cancel	Cancels the movement.
Executed	Pulses high when the mechanism has reached the pose.
Executing	Goes high during the movement.
Paused	Goes high when paused.

JointMover

The JointMover has a Mechanism, a set of Joint Values and a Duration as properties. When the Execute input signal is set the mechanism joint values are moved to the given pose. When the pose is reached the Executed output signal is set. The GetCurrent signal retrieves the current joint values of the mechanism.

Properties	Description
Mechanism	Specifies the mechanism to move to a pose.
Relative	Specifies if J1-Jx are relative to the start values, rather than absolute joint values.
Duration	Specifies the time for the mechanism to move to the pose.
J1 - Jx	Joint values.

Signals	Description
GetCurrent	Retrieve current joint values.
Execute	Set to True to start moving the mechanism.
Pause	Pauses the movement
Cancel	Cancels the movement
Executed	Pulses high when the mechanism has reached the pose.
Executing	Goes high during the movement.
Paused	Goes high when paused.

Other

GetParent

The GetParent return the parent object of the input object. The executed signal is triggered if a parent is found.

Properties	Description
Child	Specifies the object to whose parent is to be found.
Parent	Specifies the parent of the child

Signals	Description
Output	Goes high (1) if the parent exists.

Continues on next page

Continued

GraphicSwitch

Switches between two parts, either by clicking on the visible part in the graphics or by setting and resetting the input signal.

Properties	Description
PartHigh	Displayed when the signal is high.
PartLow	Displayed when the signal is low.

Signals	Description
Input	Input signal.
Output	Output signal.

Highlighter

The Highlighter temporarily sets the color of the Object to the RGB-values specified in Color. The color is blended with the original color of the objects as defined by Opacity When the signal Active is reset, Object gets its original colors.

Properties	Description
Object	Specifies the object to highlight.
Color	Specifies the RGB-values of the highlight color.
Opacity	Specifies the amount to blend with the object's original color (0-255).

Signals	Description
Active	True sets the hightlight. False restores the original color.

Logger

Prints a message in the output window.

Properties	Description
Format	Format string. Supports variables like {id:type}, where type can be d (double), i (int), s (string), o (object)
Message	Formatted message.
Severity	Message severity: 0 (Information), 1 (Warning), 2 (Error).

Signals	Description
Execute	Set to high (1) to print the message.

MoveToViewPoint

Moves to the selected viewpoint in the given time, when the input signal Execute is set. The output signal Executed is set when the operation is completed.

Properties	Description
Viewpoint	Specifies the viewpoint to move to.
Time	Specifies the time to complete the operation.

Continues on next page

Continued

Signals	Description
Execute	Set to high (1) to start the operation.
Executed	Goes high (1) when the operation is completed.

ObjectComparer

Determines if ObjectA is the same as ObjectB.

Properties	Description
ObjectA	Specifies the object to compare.
ObjectB	Specifies the object to compare.

Signals	Description
Output	Goes high if the objects are equal.

Queue

The Queue represents a FIFO (first in, first out) queue. The object in Back is added to the queue when the signal Enqueue is set. The front object of the queue is shown in Front. The object in Front is removed from the queue when the signal Dequeue is set. If there are more objects in the queue, the next object is shown in Front. All objects in the queue are removed from the queue when the signal Clear is set.

If a transformer component (such as LinearMover) has a queue component as its Object, it will transform the contents of the queue, rather than the queue itself.

Properties	Description
Back	Specifies the object to enqueue.
Front	Specifies the first object in queue.
Queue	Contains unique IDs of the queue's elements.
NumberOfObjects	Specifies the number of objects in the queue.

Signals	Description
Enqueue	Adds the object in Back to the end of the queue.
Dequeue	Removes the object in Front from the queue.
Clear	Removes all objects from the queue.
Delete	Removes the object in Front from the queue and from the station.
DeleteAll	Clears the queue and removes all objects from the station

SoundPlayer

Plays the sound specified by Sound Asset when the input signal Execute is set. The asset must be a .wav file

Properties	Description
SoundAsset	Specifies the sound file that should be played. Must be a .wav file.

Continues on next page

Continued

Signals	Description
Execute	Set to high to play the sound.

StopSimulation

Stop a running simulation when the input signal Execute is set.

Signals	Description
Execute	Set to high to stop the simulation.

Random

Random generates a random number between Min and Max in Value when Execute is triggered.

Properties	Description
Min	Specifies minimum value.
Max	Specifies maximum value.
Value	Specifies a random number between Min and Max.

Signals	Description
Execute	Set to high to generate a new random number.
Executed	Goes high when the operation is completed.

Continues on next page