



Universidad de Valladolid



**ESCUELA DE INGENIERÍAS
INDUSTRIALES**

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

Advanced automation concepts.

Autor:

Redondo Alonso, Miguel

Responsable de Intercambio en la UVa:

de la Fuente López, Eusebio

Universidad de destino:

University College Leuven-Limburg

Valladolid, Junio de 2019.

TFG REALIZADO EN PROGRAMA DE INTERCAMBIO

TÍTULO: Advanced automation concepts.

ALUMNO: Redondo Alonso, Miguel

FECHA: 20/06/2019

CENTRO: University College Leuven-Limburg (UCLL)

TUTOR: Wim Claes - UCLL

Robby Cloesen - Ciratec

Johan D'Hondt - Ciratec

Acknowledgments.

Thanks to all the UCLL for being my hosts during this amazing experience.

I would also thank to the people of Ciratec, especially to Robby Cloesen and Johan D'Hondt for their help and trust during these months.

To my home university for allowing me to live this experience, especially to Eusebio for his help and great effort to make this possible.

A toda esa maravillosa gente que he conocido durante esta experiencia y con la que he vivido tantos buenos momentos.

Y sobre todo, a mi familia, mis padres y mis hermanos que han estado ahí durante todos estos años y que me han apoyado en todo momento. También a Ana, por su incondicional apoyo y por aguantarme todo este tiempo.

Muchas gracias a todos.

Miguel Redondo Alonso.

Abstract

I did my internship at Ciratec BVBA in Diepenbeek. They do offline and online programming of different types of robots and, they are involved in making robot standards for various applications.

Ciratec wants to present their customers advanced automation concepts using collaborative robots (cobots), final assembly automation, line tracking, painting and gluing.

Cobots are starting to be used more frequently in industry due to their multiple advantages. Final assembly automation has a big potential as this part of the car assembly is still not automated as body-in-white, so the use of cobots could change this.

Further paint and glue processes are difficult to fine tune during production, so I investigated several ways to do this in an offline environment.

In my internship I used software as Siemens Process Simulate, ABB RobotStudio, Fanuc RoboGuide and UR Polyscope. I also worked with a real Universal Robot cobot, the UR10.

The concepts I elaborate are: gluing cell for glasses in final assembly and weld inspection in body-in-white using a Universal Robot; spare wheel insertion and battery insertion in final assembly using a Fanuc robot; body panel painting using an ABB robot, and paint coverage simulation with Process Simulate.

At the end of my internship I also developed a test environment for PLC equipment using a Beckhoff Twincat SoftPLC. For testing we used an analog sensor with Profinet which will be used during a bakery automation project

Table of contents.

| | |
|--|-----------|
| Acknowledgments. | 2 |
| Abstract | 3 |
| Table of contents. | 4 |
| List of figures. | 7 |
| List of tables | 10 |
| 1. Introduction. | 11 |
| 2. Universal Robots. UR10. | 13 |
| 2.1. CB-Series vs e-Series..... | 13 |
| 2.2. PolyScope. | 14 |
| 2.3. How use the robot..... | 16 |
| 2.4. Experience with a real robot..... | 18 |
| 3. Backup gluing cell. | 21 |
| 3.1. Development of the idea. | 23 |
| 3.1.1. Previous configuration..... | 24 |
| 3.1.2. Robot controller configuration. | 25 |
| 3.1.3. Creation of the process..... | 26 |
| 3.1.4. Programming of the operation. | 27 |
| 3.1.5. Upload a Process Simulate program to URSim..... | 28 |
| 3.1.6. Upload a program from URSim to PProcessSimulate. | 29 |
| 4. Weld check cell | 30 |
| 4.1. Idea development. | 31 |
| 4.1.1. Tool creation. | 31 |
| 4.1.2. Process programming. | 32 |
| 4.2. Conclusions..... | 32 |
| 5. Spare wheel and battery insertion. | 33 |
| 5.1. Developing of the idea without line tracking..... | 33 |

| | | |
|-----------|--|-----------|
| 5.1.1. | Create the gripper..... | 34 |
| 5.1.2. | Creation of the different parts..... | 34 |
| 5.1.3. | Programming the movements..... | 34 |
| 5.1.4. | Robot position..... | 35 |
| 5.2. | Developing of the idea with line tracking. | 36 |
| 5.2.1. | Line tracking. Definition..... | 36 |
| 5.2.2. | Create the conveyor | 37 |
| 5.2.3. | Tracking with FANUC. | 38 |
| 5.2.3.1. | RCS problem. | 39 |
| 5.2.3.2. | Programming | 40 |
| 5.2.3.3. | Conclusion..... | 41 |
| 5.2.4. | Line tracking with ABB. | 41 |
| 5.2.4.1. | MOC file | 41 |
| 5.2.4.2. | Conclusions | 42 |
| 6. | Paint and Coverage Simulation | 43 |
| 6.1. | Tool definition. | 43 |
| 6.2. | Brush definition..... | 44 |
| 6.3. | Creating the process. | 44 |
| 6.3.1. | Generate a continuous operation | 45 |
| 6.3.2. | Creating the mesh..... | 46 |
| 6.3.2.1. | XT-Brep exact geometry. | 47 |
| 6.3.3. | Programming and setting the simulation parameters. | 47 |
| 6.4. | Result and conclusions | 48 |
| 7. | Communication between a Turck module and TwinCat3. | 50 |
| 7.1. | Turck module..... | 50 |
| 7.2. | Connection to the Turck module | 52 |
| 7.2.1. | IP address setup..... | 52 |
| 7.2.1.1. | Web server..... | 53 |

| | |
|--|-----------|
| 7.2.2. Problem with the firmware. | 53 |
| 7.2.3. Read parameters from PACTware. | 54 |
| 7.3. Connect to TwinCat 3 | 55 |
| 7.3.1. Configuration of the devices on a TwinCat 3 project. | 56 |
| 7.4. Results and conclusions. | 59 |
| Appendix..... | 60 |
| A. Datasheets..... | 60 |
| 1. UR10..... | 60 |
| 2. CR-35iA..... | 61 |
| 3. Turck Module | 62 |
| Bibliography | 64 |

List of figures.

| | |
|--|----|
| Figure 1 Screenshot of Process Simulate environment | 12 |
| Figure 2. Left: UR10; Righth: UR10e | 13 |
| Figure 3. Screenshot of PolyScope | 15 |
| Figure 4. Pantalla de inicio de URSim | 16 |
| Figure 5. Installation Tab | 17 |
| Figure 6. Setup Robot window | 19 |
| Figure 7. Safety I/O tab..... | 20 |
| Figure 8: Robot on the limit position he could reach..... | 22 |
| Figure 9: Robot colliding with itself..... | 22 |
| Figure 10. PUHead | 23 |
| Figure 11. Tool Definition Window..... | 24 |
| Figure 12. Screenshot of the Placement Manipulator Tool | 25 |
| Figure 13. Screenshots of the windows for define the tool in the controller..... | 25 |
| Figure 14. Weld Points..... | 26 |
| Figure 15. Weld points projected | 27 |
| Figure 16. Path editor Screenshot | 28 |
| Figure 17. Structure tab in URSim | 29 |
| Figure 18. Screenshot of a . script file | 29 |
| Figure 19. Picture of the process..... | 30 |
| Figure 20. Simulation of the vision system..... | 31 |
| Figure 21. Check points on the bodywork..... | 32 |
| Figure 22. Capture of the process. | 33 |
| Figure 23. Gripper..... | 34 |
| Figure 24. Pick and place operation dialog window..... | 35 |

| | |
|---|----|
| Figure 25. Screenshot of the path editor with the properties of the movement..... | 35 |
| Figure 26. Basic simulation for line tracking with a FANUC robot | 36 |
| Figure 27. Representation of line tracking | 37 |
| Figure 28. Screenshot of the FANUC software on the option selection step. | 39 |
| Figure 29. Line Tracking Settings window | 40 |
| Figure 30. Screenshot of the ABB RCS server | 41 |
| Figure 31. Screenshot of the ABB robot setup | 42 |
| Figure 32. Representation of a paint gun and its frames | 43 |
| Figure 33. Picture of the brush on the paint gun used in the model | 44 |
| Figure 34. Picture of a bad definition of the paint gun and the brush..... | 44 |
| Figure 35. Continuous process generator window | 45 |
| Figure 36. Create mesh window | 46 |
| Figure 37. Error message when the par has not exact geometry | 47 |
| Figure 38. Screenshot of the OLP commands | 47 |
| Figure 39. Paint and coverage settings window..... | 48 |
| Figure 40. Result of the simulation..... | 48 |
| Figure 41. Face while creating the continuous operation..... | 49 |
| Figure 42. Painting simulation with different speeds: a) v50 b) v100 c) 60..... | 49 |
| Figure 43. Picture of the BLCEN-4M12MT-4AI-VI | 50 |
| Figure 44. Representation of the rotary switches..... | 51 |
| Figure 45. Screenshot of the web server..... | 53 |
| Figure 46. Screenshot of PACTware after searching the devices..... | 54 |
| Figure 47. Device added to the Project | 54 |
| Figure 48. Measured values. | 55 |
| Figure 49. Screenshot of TwinCat environment..... | 55 |

| | |
|---|----|
| Figure 50. Realtime compatible devices Window | 56 |
| Figure 51. Insert Device window | 57 |
| Figure 52. Setting tab | 57 |
| Figure 53. Insert Box Window | 58 |
| Figure 54. Topology of the device in TwinCat | 58 |

List of tables

| | |
|---|----|
| Table 1. Comparison between CB-Series and e-Series..... | 14 |
| Table 2. UR10's specifications | 21 |
| Table 3. Comparison between UR10 and CR-7iA | 30 |
| Table 4. Main characteristics of CR-35iA..... | 33 |
| Table 5. LED status of the Turck module | 51 |

1. Introduction.

I did my internship at *Ciratec BVBA* in Diepenbeek, where I developed advanced automation concepts using collaborative robots (cobots), final assembly automation, line tracking, painting and gluing. All these models were made with the offline environment software of Siemens, Process Simulate.

Ciratec is a company that do offline and online programming of different types of robots, which involved in making robot standards for various applications. The company was created in 1998 by Jean Cloesen. Since 2016, Bart Thys and Robby Cloesen became the directors. Also, in October 2016, they became part of the group FFT, keeping their independent business unit within the FFT group.

About the collaborative robots (cobots): they are a new generation of robots whose target is to work with humans in the same way. Those kind of robots are beginning to be used more frequently in industry because of their multiple advantages such as: reduced mode that can be activated when it detects a person in its work area to work with them, no need of security fences and detection of collisions with humans, which means that if the robot collides with anyone, it will stop immediately. Despite the great advantages that they have, they are still limited in some aspects like payload and reachability. During my internship I worked with two cobots: CR-35iA from FANUC and UR10 from Universal Robots, which I had the opportunity of trying.

With concepts, I am referring to models that only have the robot and the part where the process will be made. Those concepts are used for simulation test and to present to any customer.

Finally, the model design was made with Process Simulate. Process Simulate is a 3D design and offline robot programming environment. It makes the programming and process design easier because you can use any robot without using their own software, which gives to you more flexibility with customers and robot brands.

Also, Process Simulate allows you to import the programs that you make there to the robot format in order to upload it into the robot; and import robot programs, programmed with Teach Pendant or their own environment, to process simulate to be

modified. To make that, the robot controllers have to be correctly set up with the one that they need. Otherwise, the robot won't work as it is supposed to work.

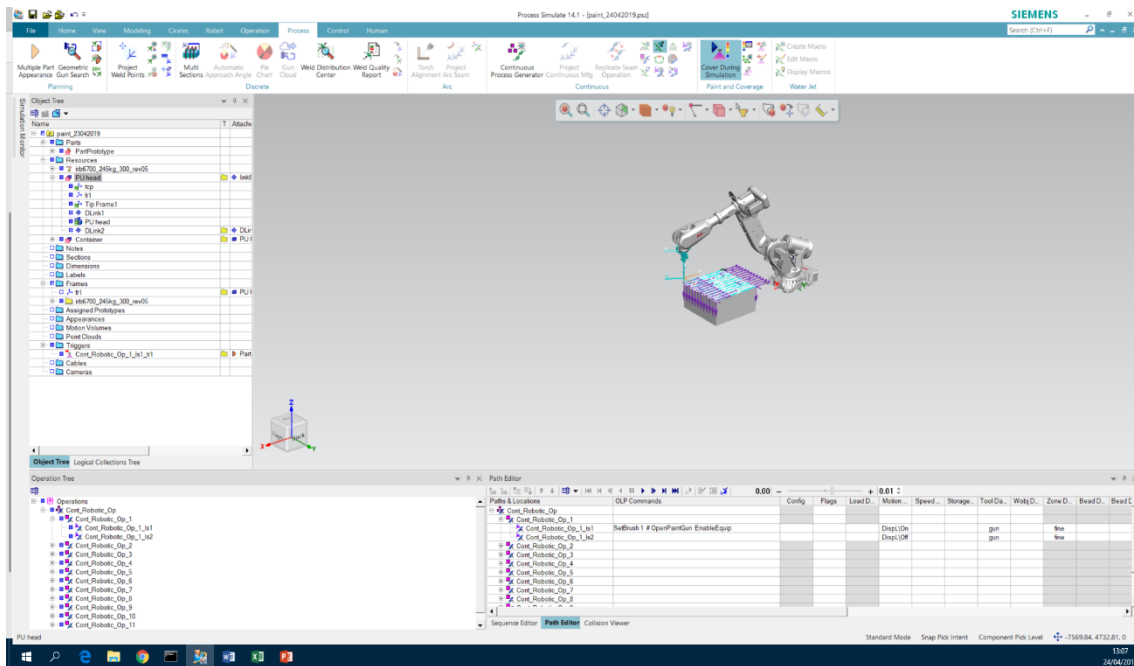


Figure 1 Screenshot of Process Simulate environment

On the following pages the different models I made will be developed and how I made them; and, a section about the UR10.

2. Universal Robots. UR10.

During the development of my practices, as I said, I have worked in simulations with Cobots. The Cobot that I used the most and the one I had to learn to handle it completely is the UR10 of Universal Robots.

The design of this robot is very versatile, because it is a light robot with a wide range of joint movements, it has an amplitude of ± 360 degrees in each joint axis. All of this allows movements in wide areas and even the possibility of installing the robot wherever it was necessary. However, the cobots have some limitation such as a limited reachability and payload. The reason is because working with humans the robot cannot exert too much force because by colliding with an operator it could cause serious damage.

2.1. CB-Series vs e-Series

Presently, Universal Robots have two different series of robots, CB-Series and e-Series. Despite the fact of being different series, they are similar robots, but with different controller.

As shown on at figure 2 both robots have similar appearances. The most notable difference between those robots is the base, but it is not the most important. The main differences are in the characteristics.

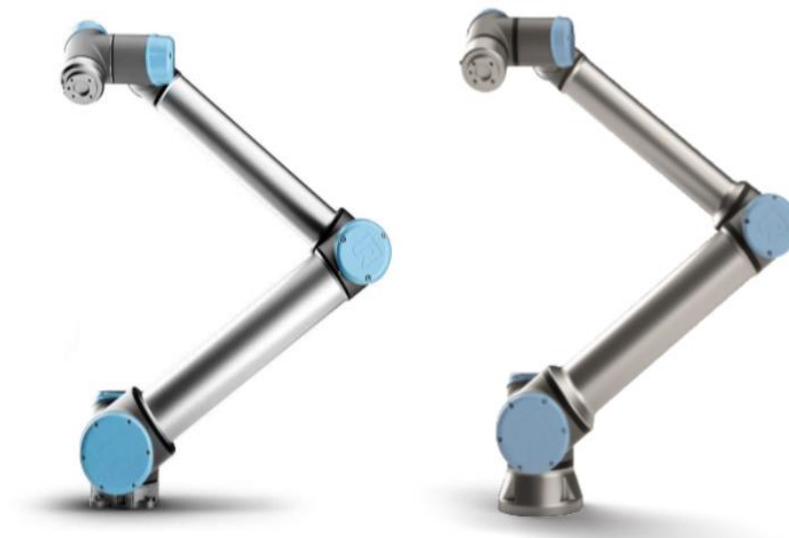


Figure 2. Left: UR10; Righth: UR10e

As can be observed on table 1, the differences are minimal. Although the comparison is made with the UR10 model, the main concepts could be applied on the rest of models. The main differences are to be founded in precision, weight, energy consumption and the number of security modes, basically.

| | | UR10 | UR10e |
|--------------------------|----------|---------|----------|
| Repeatability | | ±0.1 mm | ±0.05 mm |
| Mechanical weight | | 28.9 Kg | 33.5 Kg |
| Payload | | 10 kg | 10 kg |
| Reach | | 1300 mm | 1300 mm |
| Degrees of freedom | | 6 | 6 |
| Motion range | Base | ±360 | ±360 |
| | Shoulder | ±360 | ±360 |
| | Elbow | ±360 | ±360 |
| | Wirst1 | ±360 | ±360 |
| | Wirst2 | ±360 | ±360 |
| | Wirst3 | ±360 | ±360 |
| Maximum speed (mm/s) | Base | 1000 | 1000 |
| | Shoulder | 1000 | 1000 |
| | Elbow | 1000 | 1000 |
| | Wirst1 | 1000 | 1000 |
| | Wirst2 | 1000 | 1000 |
| | Wirst3 | 1000 | 1000 |
| Power consumption (typ.) | | 250W | 350W |
| Safety functions | | 15 | 17 |

Table 1. Comparison between CB-Series and e-Series

2.2. PolyScope.

As most of the robot brands, Universal Robots have also their own offline programming and simulation environment, PolyScope. Polyscope is an operative system based on Ubuntu specifically design for Universal Robots. With this OS you can launch an offline version of the software that the robots have in their Teach Pendant, URSim. In fact, there are three applications, one for each robot model. That is because the main way to program these robots are with the Teach Pendant, which is where you set all the robot configuration from.

Welcome to PolyScope

The icons for starting the simulator for different robot types are located on the desktop.

Links to the program folders are also available and it is possible to drag and drop files between the folder and Windows desktop.

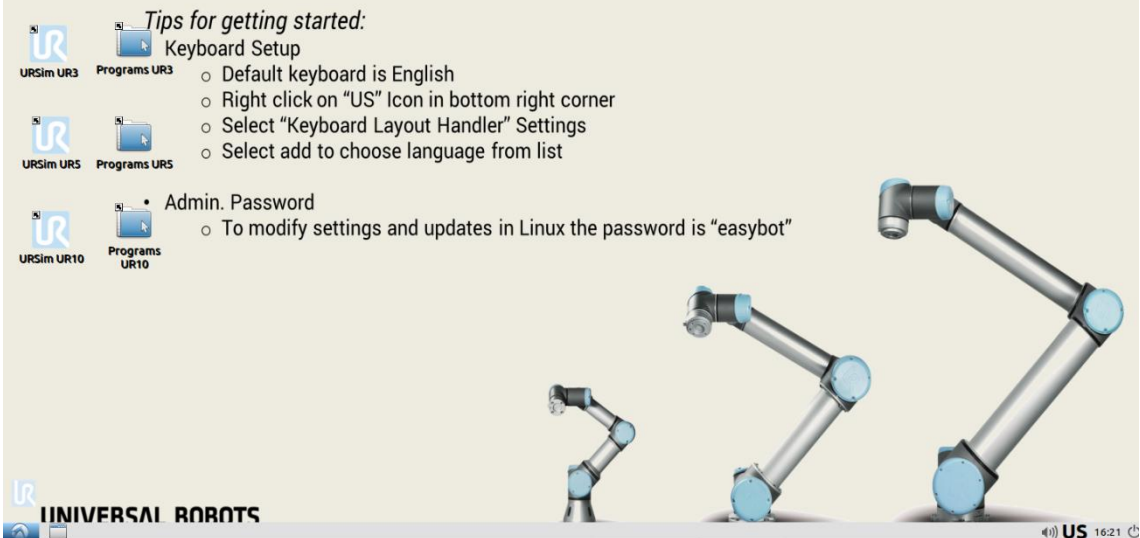


Figure 3. Screenshot of PolyScope

The software is free and available to download from the official Universal Robots web page. Also, it must be installed in a virtual machine. It is important to know that the software for the CB-Series and e-Series are different and the applications do not work in the same way due to the different controller between the series. The way of functioning of the robots are not the same. For example, the e-Series have more security functions than the CB-Series and trying to use one of those functions with a CB-Series robot that do not have could produce important errors.

As is said before, PolyScope is the OS; for programming the robot it must be used URSim. And in particular for the UR10, the URSim UR10.

The three different versions of URSim are exactly the same, but each one is for different robots, because the specific robot parameters are loaded in each application.

In figure 4, it is shown the aspect of the main screen of URSim. As can be seen, the appearance is the same as the one that is installed in the Teach Pendant of the real robot.

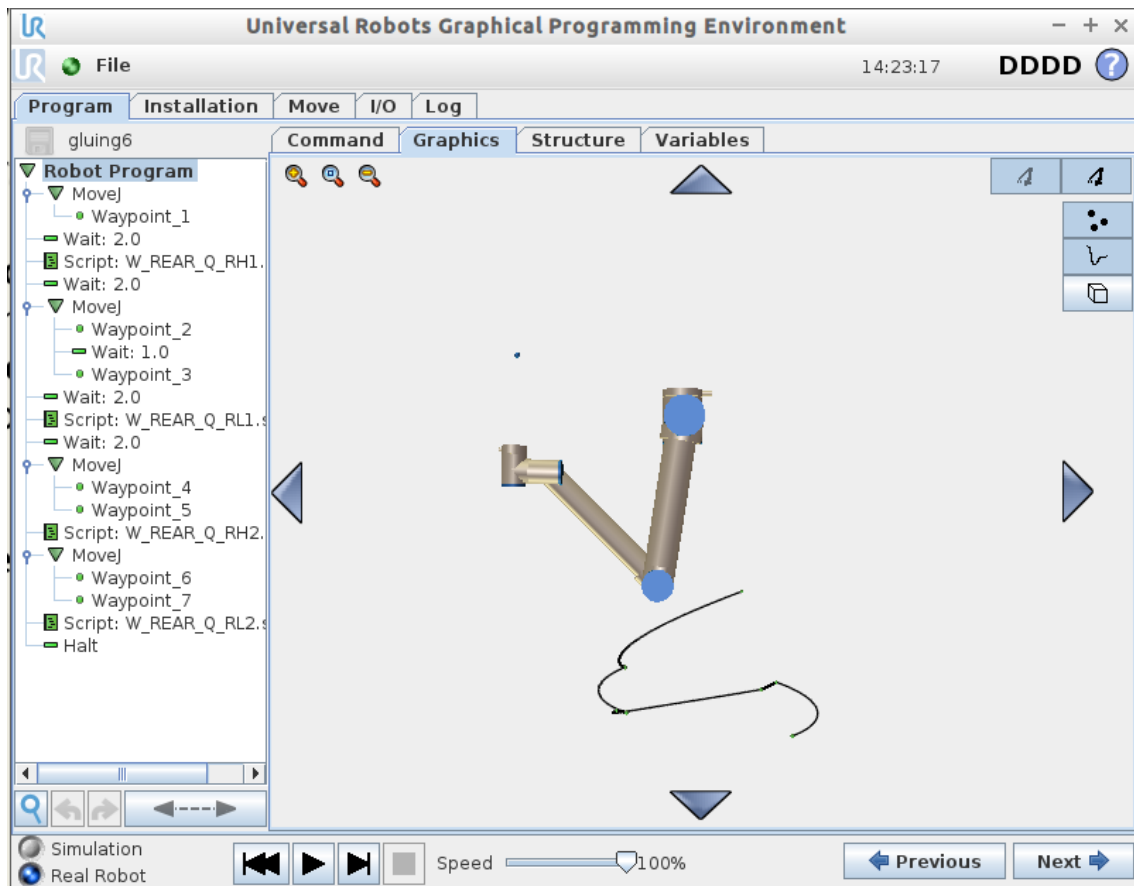


Figure 4. Pantalla de inicio de URSim

From URSim, it is possible to edit and simulate full robot programs.

It is interesting and useful to know which different file types exist in these robots. It is possible to differentiate 3 types of files:

1. *.urp*: this is the extension where the robot programs are saved. This is NOT a text file, this file was made for open and edit with URSim.
2. *.script*: text file where the robot moves are written. This is not possible to be edited whit the graphic interface of URSim, but can be added to bigger programs.
3. *.installation*: is a file where is saved the configuration of a robot: the position, safety limits, I/O, etc.
4. *.urcap*: this files are a kind of apps for the robot. Will be explained forward.

2.3.How use the robot.

The next steps that are going to be explained about how to use the robot, can be used either for the real robot or the use URSim in Polyscope

First thing to do is to set up the robot properly. At the *installation* tab there are several configuration options, as can be seen in figure 5.

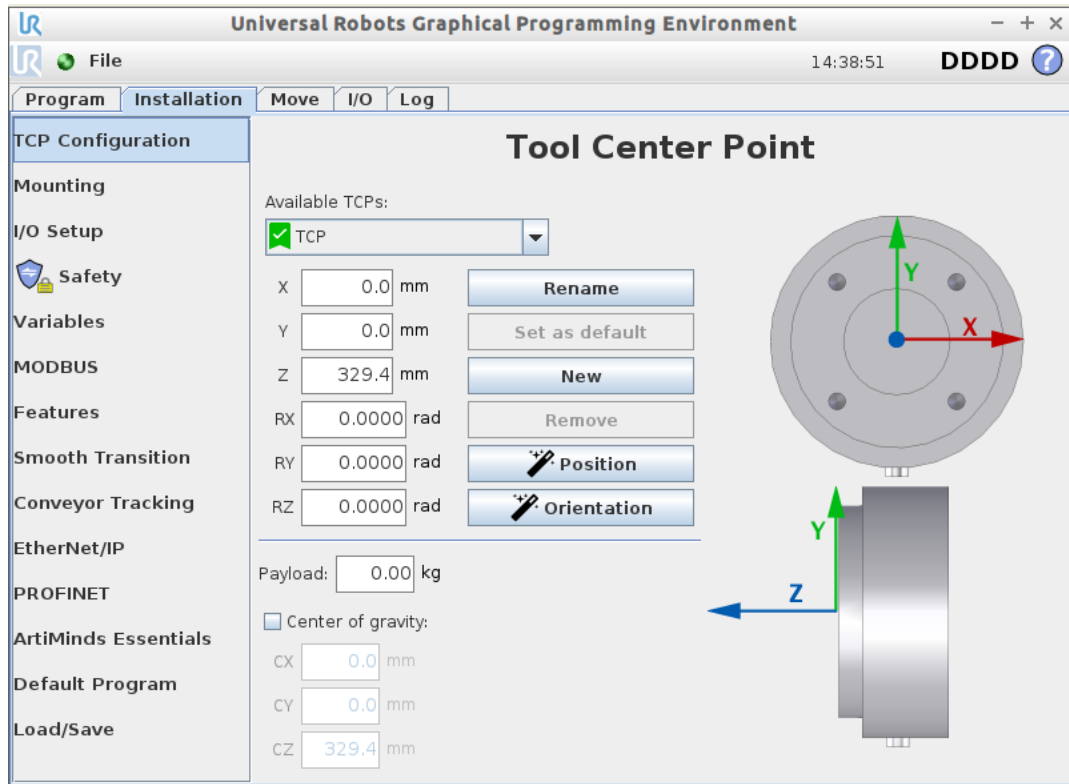


Figure 5. Installation Tab

Of all the options shown at this tab, the most important are:

- TCP configuration, where the data about position, orientation, weight and center of gravity of the TCP are introduced.
- I/O setup. Where the all inputs and outputs of the robot are set except for the security I/O.
- Mounting. Where the position of the robot is set.
- Safety. One of the most important section, on it you can set all the aspects for the robot related with the security.
 - General and joint limits: they are sections where is possible to set the speed and joint limits in either reduce and normal mode.
 - Safety boundaries: where is possible to define the safety planes for the robot motion. Can be set in 4 different ways:
 - Normal: the robot stops when is close to the plane only if the robot is in Normal mode.

- Reduce: the same as the previous one but with the robot in reduce mode.
 - Normal and reduce.
 - Trigger to reduce: when the safety system is either in normal or reduce, if this limit is activated the robot switch to the reduce mode as long as the robot TCP is located beyond it.
- Safety I/O: where is possible to link the inputs and outputs with the different safety actions.
- Load/Save: in that section is possible to load and save a robot configuration (.installation files).

2.4. Experience with a real robot.

During one week I had the opportunity to go to the FFT factory in Fulda, Germany, to use a real UR10 and make some tests with the programs I had made in the office. The difference with my model was that the robot wasn't handle above, but was enough for making the test. At that cell, the robot was mounted on a moving platform. Also, there were a table with a back window of a car and I could made some test with the robot and the widow that will be explain after.

The first day was the first contact with the robot. The first thing I did was to check the basic configuration as speed and joint limits, safety: moved the robot with the Teach Pendant and test the free drive mode. For the robot's free drive mode it must be pressed the teach Pendant button, but it is important to check before if the free drive function is associate with that button. It is important to know that the closer is the robot to the joint limits the harder is to move that join; and, if you move it to the limit the robot will turn on the emergency stop.

The following thing I did was try to connect the robot with my laptop with the Ethernet wire. It was not possible because Universal Robots do not support the TCP/IP connection and, for making the communication it should be used sockets or other communication elements with C++.

Other aspect I wanted to try was the record of movements. The basic software does not support that function, so an external application should be used; and those external

applications are the URCaps, which can be downloaded from the Universal Robots official web page. They are too easy to install: download the file into a USB drive and connect to the Teach Pendant; then, from the main window at **Setup Robot** → **URCaps** click the “+” button and select the file with the *.urcap* extension. The application selected was *Artiminds Essential* which can be used to record the movement and set the recording parameters as the points recorded per second, etc.

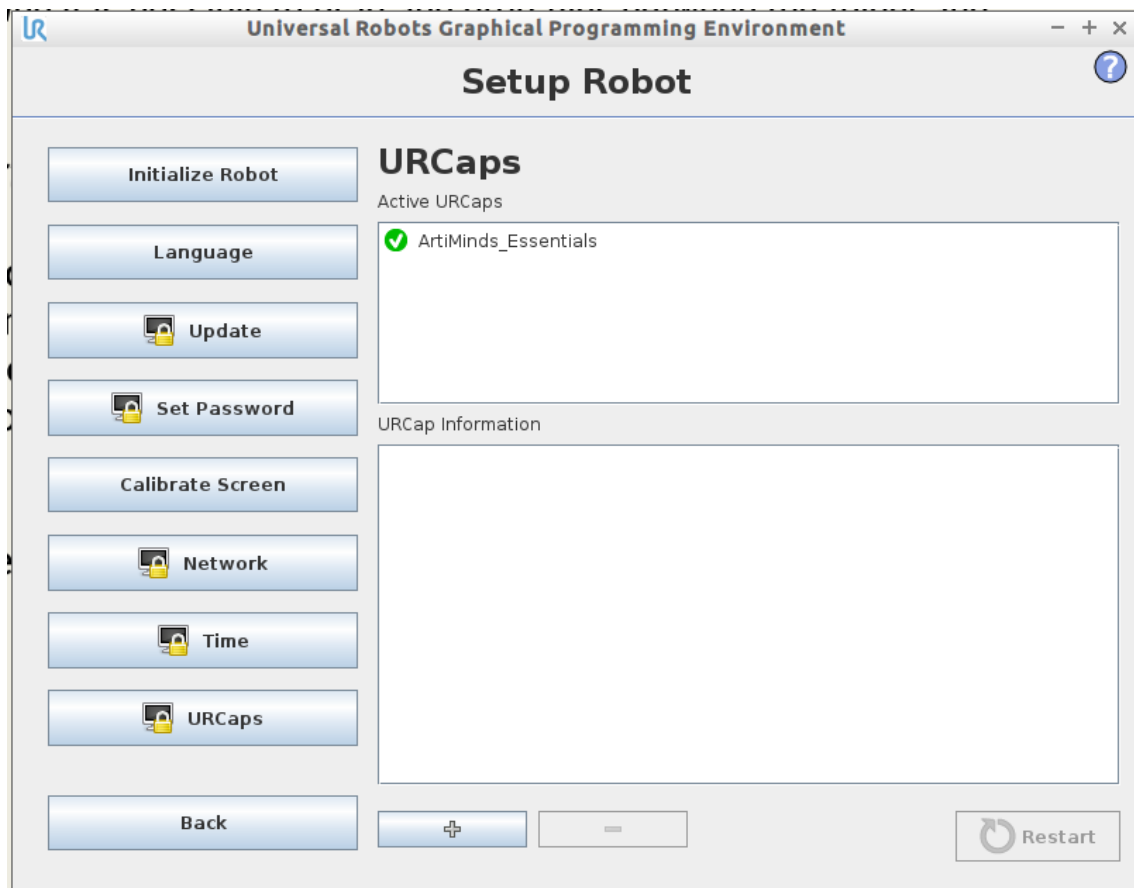


Figure 6. Setup Robot window

The reduced mode is something that define the cobots, which allow them to work with humans. To make this function available a security I/O must be defined or activate a security plane, because if not the reduce mode limits and options won't be available to modify. Those I/O are reserved to be linked with certain security related functions as the emergency stop, for example. The reduce mode will be active as long as the signal is on low level.

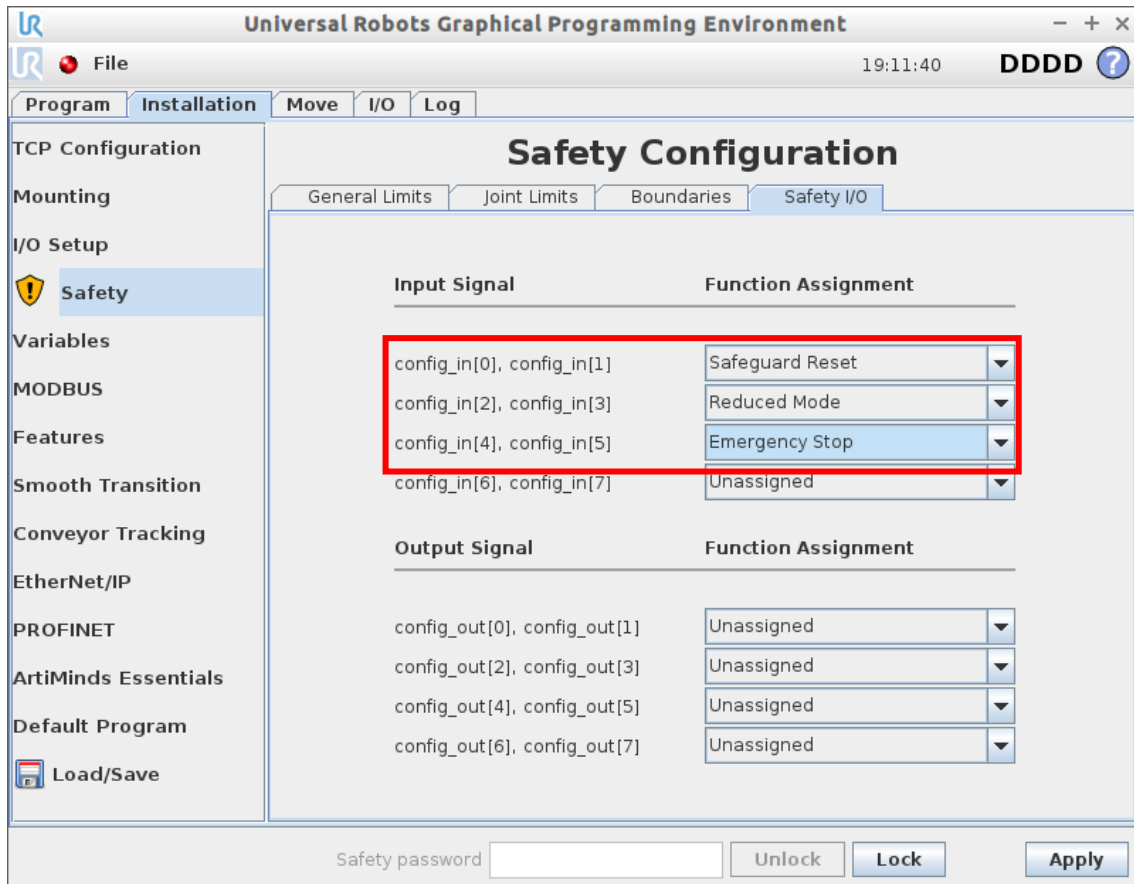


Figure 7. Safety I/O tab

After had all the safety related options set, I decided to upload some programs I made with Process Simulate and try them with the full configuration of the robot done. All the programs showed security errors. That was because all the programs were made using the **MoveP** command, which does not support I/O during the execution. To solve this all the **MoveP** commands have to be replaced by the command **MoveL**. In motion aspects, both do the same, move the robot between two points in a linear way; but **MoveL** allows I/O during execution.

3. Backup gluing cell.

The first model consist in a backup cell for the gluing of the different parts of a car that need it. A backup cell consist in a secondary process which can be activated when the main process is stopped because of any reason.

That cell is intended to be simple and with the lowest cost possible. That is why we decided to use a cobot, because the lack of security fences and the possibility of working together with a human makes the design easier.

The robot that is going to be used for the model is the UR10, from Universal Robots. These robots give more freedom in movement in comparison with other kind of robots; they have a ± 360 degrees of motion range in each joint.

Table 2. UR10's specifications

| UR10 | Mechanical weight | Payload | Reach (mm) | Motion range | | | | | |
|------|-------------------|---------|------------|--------------|-----------|-----------|-----------|-----------|-----------|
| | | | | Base | Shoulder | Elbow | Wirst1 | Wirst2 | Wirst3 |
| | 28.9 Kg | 10 kg | 1300 | ± 360 | ± 360 | ± 360 | ± 360 | ± 360 | ± 360 |

For the design process of the model, I was given the biggest parts of a car that need gluing in order to understand and make some tests about the reachability of the robot and ensure that the idea is viable.

I had 3 ideas for this project: put the robot in a common position, put it in a wall with 45 degrees of inclination, and with the robot above the work table.

- **Robot in a common position.**

The most common position for a robot in a factory is horizontal.

As you can see in *Table 1* the reachability of the robot is 1300 mm, which could be not long enough for reaching the biggest pieces in a factory. And that was what happened on simulations.

As can be seen in *Figure 8* this is the furthest position that the robot could reach. Even with the design it has, the following points need a configuration that is impossible to reach with the robot, so it can produce the block of the robot. Due to those reasons, and because it happened with most of the parts, the idea was discarded.

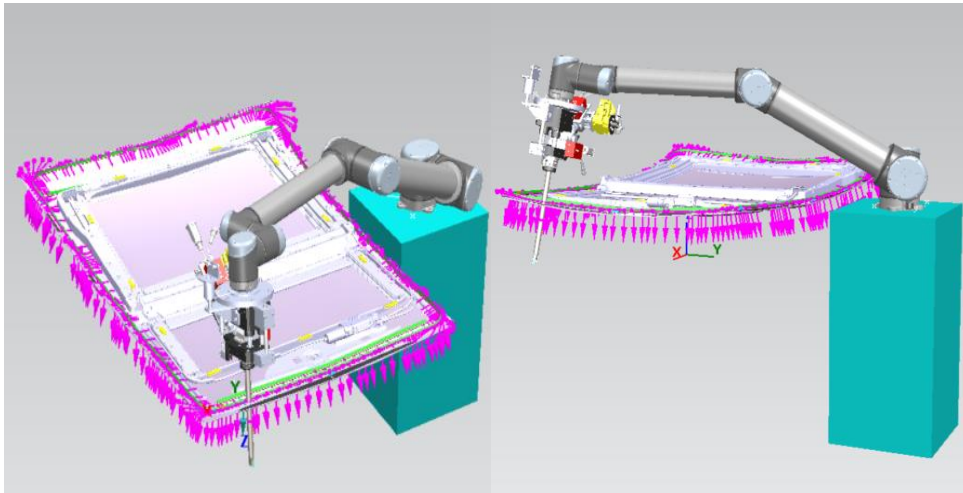


Figure 8: Robot on the limit position he could reach

- **Robot on the wall with 45 degrees of inclination.**

It was a feasible idea. The problem of the reachability and joint configuration of the robot on the pieces was solved; in some cases with little difficulty, but acceptable.

Nevertheless, there was a huge problem. After making some test with the location of the robot and the work table most of the pieces were reached, but the robot collides with itself in some cases. In *Figure 9* it is possible to see the collision.

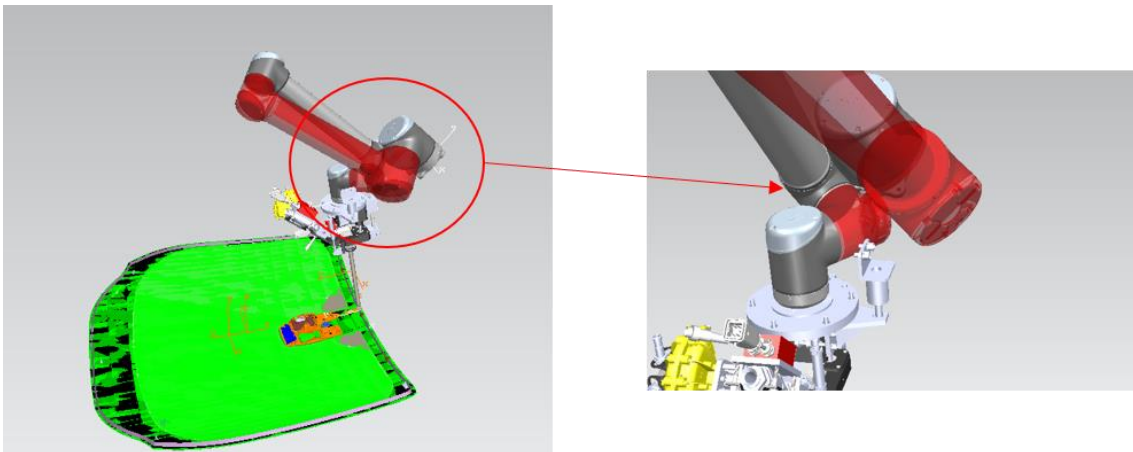


Figure 9: Robot colliding with itself

The problem was not the collision at all. The main problem was that if the piece was moved, the robot didn't reach the whole piece. That is why the idea was discarded.

- **Robot above the work table.**

Another idea was to put the robot on above the pieces, face down and with the base frame in the middle of the work table handled by a structure. In that location, the robot can reach all the pieces.

That idea wouldn't be possible with other robot but a Universal Robots'; and that is because of its motion range which gives the possibility of doing a hole 360 degrees turn of the base.

Those reasons are the ones that makes the idea available and what I developed. How I developed the idea (model, simulation, etc.) will be explained in the following lines.

3.1. Development of the idea.

The Process Simulate tool was used to carry out this concept.

As a concept, all that is needed at the beginning is the robot and the parts on which the process is going to be carried out

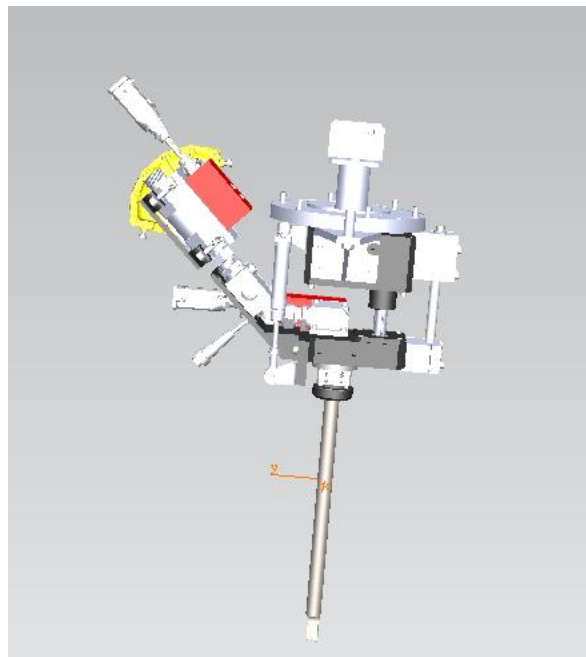


Figure 10. PUHead

3.1.1. Previous configuration.

The first step is to open one of the parts that we are going to use, in my case the file 'PANOROOF_LIGHT.cojt', and place it at a normal height for a work table, around 1m more or less. Then place the robot at a suitable height so that the process can be carried out. For this type of process it is necessary to use a gluing gun. As the gun we use the file 'PUHead.cojt' which represents the glue gun of the *Figure 10*.

After adding the tool, it must be defined. To do this, activate the *Set Modeling Scope* mode of the part; and, selecting the tool, go to **Modeling** → **Kinematic Device** → **Tool Definition** and pop up window of the *Figure 11* will appear. In the 'Tool Type' field select Gun, because it is the most similar option to the use it will have. Then, in the fields 'TCP Frame' and 'Base' we select, respectively, the frames TCP and fr1, which is the frame that is in the base of the gun.

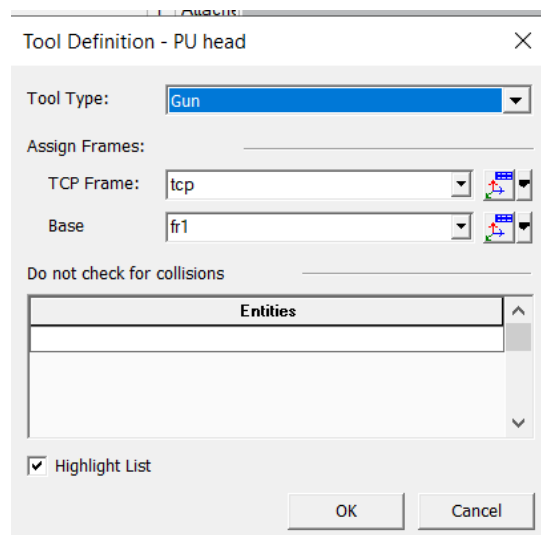



Figure 11. Tool Definition Window

Although it is a concept, the glue gun is too long for what it is supposed to be a real one, so the TCP was approached for not being so far away. Keeping the tool in edit mode we select the TCP base and, using the **Placement Manipulator Tool**  we place the TCP at an appropriate distance.

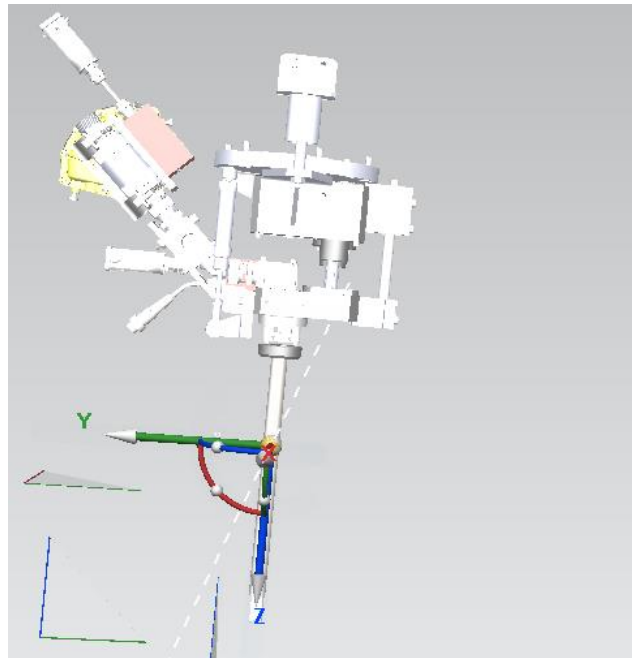


Figure 12. Screenshot of the Placement Manipulator Tool

3.1.2. Robot controller configuration.

Finally, before programming and carrying out the necessary processes, it is necessary to define the tool, but this time in the robot controller. This is because when a program is generated, the parameters of the system frames, tools, etc. are in the robot controller because those are the necessary ones for the proper functioning of the robot.

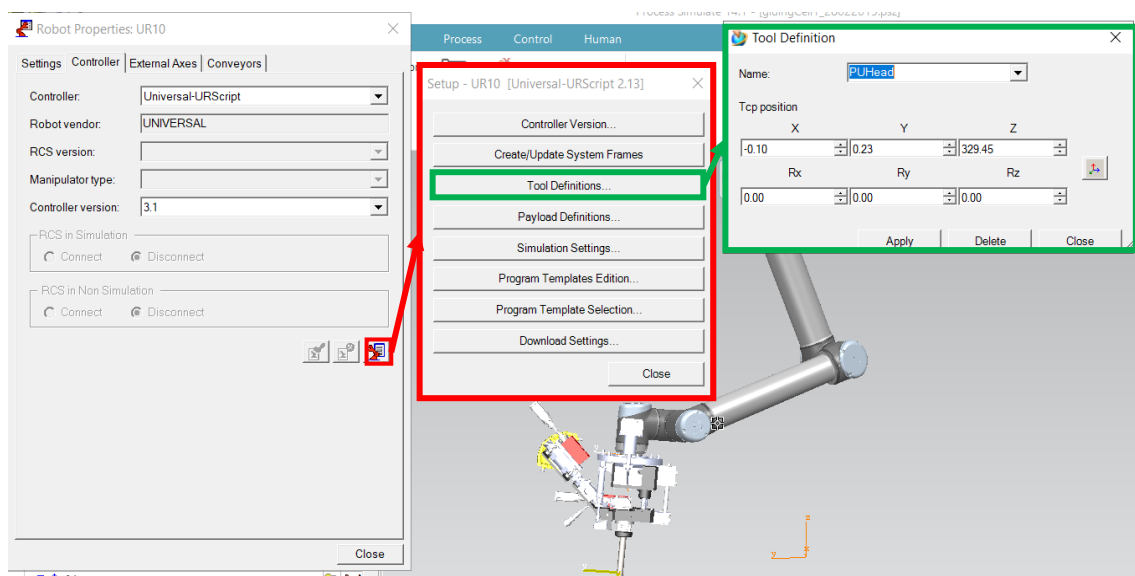


Figure 13. Screenshots of the windows for define the tool in the controller

In **Robot**→**Setup**→**Robot Properties**, the window on the left in Figure 13, pops up and you can choose the desired driver from the ones you have installed. We chose *Universal*

URScript and version 3.1. Pressing the **Robot Setup** button takes us to the controller window where we can configure multiple aspects. Within them we select the **Tool Definitions** option. In the **Tool Definition** window type the name in the **Name** frame and then select the TCP axes of the tool.

3.1.3. Creation of the process.

Now that we have the robot and the part in place and the controller and the tool configured, we can proceed to the creation of the process itself.

In Process Simulate there is no option to create a gluing operation, the closest there is to be a welding operation. The first thing to do is add the points where you want the robot to move using the **Process→Discrete→Create Weld points by pick** tool. Select the desired points and some garnet cubes will appear on the selected locations. The dots must be projected onto a part so that they are attached to it and can be adapted to its shape

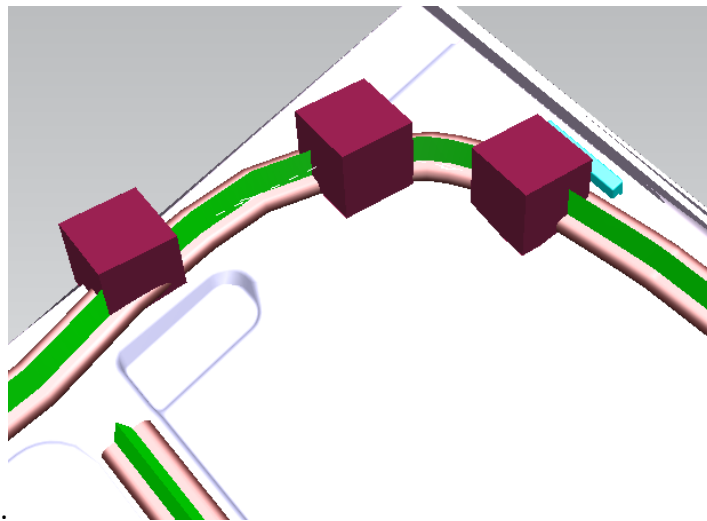


Figure 14. Weld Points

Select the points and use the **Process→Discrete→Project Weld Points** option, select the part on which you want to project and, thus, there are some visible points with orientation and manipulable (Figure 15).

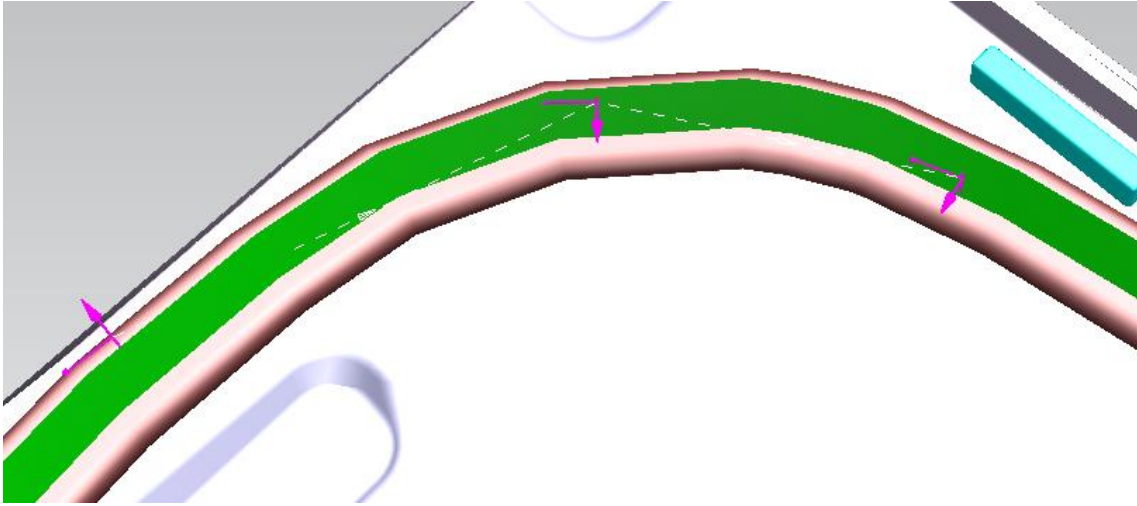


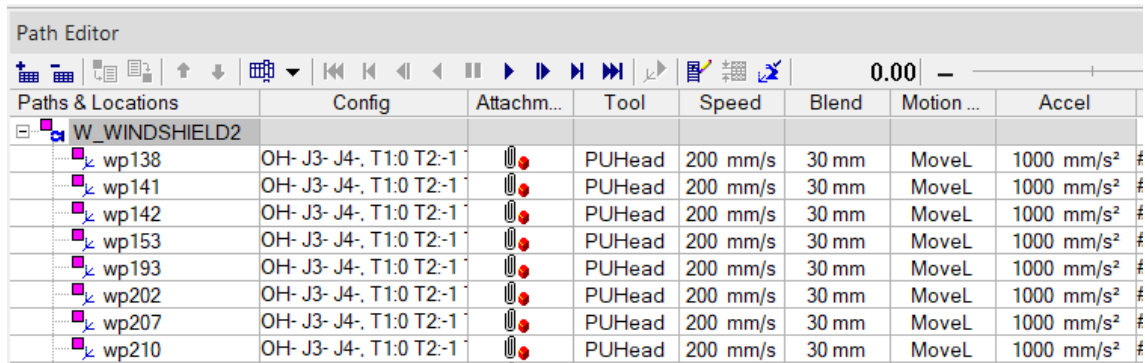
Figure 15. Weld points projected

With the points already created and projected on a part you must create a welding operation in **operation** → **Create operation** → **new operation** → **New Weld Operation**. In this way, an operation is already created with a series of points that the robot must follow, otherwise it could not be executed. Since this is not a real welding operation, the points can simply be converted into 'via' points with the function **Convert to Via Location**.

As the last step before going on to program the operation, we must place and orient the points correctly, because after the projection they may be in an undesirable position or with an incorrect orientation, as can be seen in *Figure 15*. Several Process Simulate tools can be used to modify positions, but the most useful are: **Location Manipulator** and **Flip Location**. When repositioning the points, orientation is very important, since the orientation of the robot's TCP is what it will have when it tries to reach the points; they must also always be oriented in such a way that the x-axis is in the direction of movement.

3.1.4. Programming of the operation.

With all the above, the operation is already defined. All that remains to be done is to configure the parameters of the robot movement which, although it is more appropriate to program with the robot itself, can be done with Process Simulate.



| Paths & Locations | Config | Attachm... | Tool | Speed | Blend | Motion ... | Accel |
|-------------------|-------------------------|------------|--------|----------|-------|------------|--------------|
| W_WINDSHIELD2 | | | | | | | |
| wp138 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |
| wp141 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |
| wp142 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |
| wp153 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |
| wp193 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |
| wp202 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |
| wp207 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |
| wp210 | OH- J3- J4-, T1:0 T2:-1 | | PUHead | 200 mm/s | 30 mm | MoveL | 1000 mm/s² # |

Figure 16. Path editor Screenshot

With the **path editor** tool, it is much easier to modify the parameters. Here you can modify the important aspects for the execution of the program such as speed, acceleration, type of movement, etc. Since in general all parameters are the same, it is enough to select all the points and using the **Set Location Properties** tool all the necessary properties are configured.

3.1.5. Upload a Process Simulate program to URSim

Once the operation is programmed and with the parameters well configured, the program can be downloaded and Process Simulate will automatically generate the file in the appropriate format and program language for the robot. This file is generated with the **download to robot** function. The file it generates is a .script, not an .urp. As explained in section 2, .script files are those that are text files with the robot movements, so we cannot open them directly with URSim.

To open them with URSim and include them in a functional program, the following steps should be followed:

- First, open a new program.
- Second: add the script file to the program in **structure** → **advance** → **Script code** and select the option 'file'.

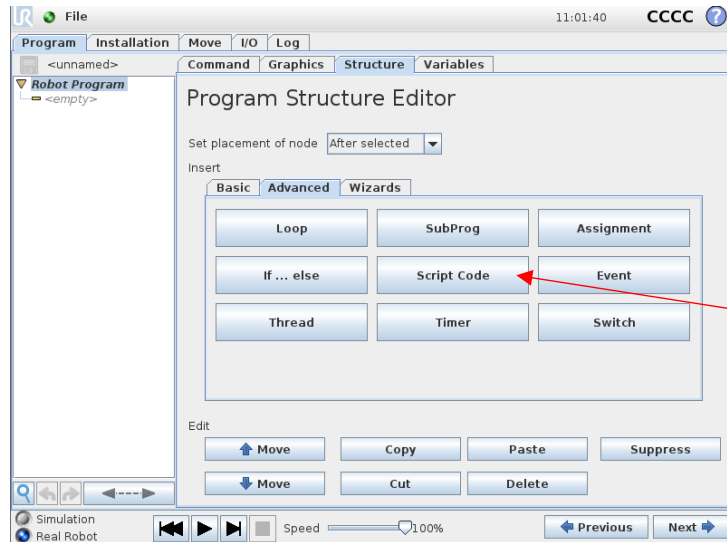


Figure 17. Structure tab in URSim

- Third: before including it, delete the first and last lines of the code, because if you add it as it is, it will cause errors when running the program.

```

10 def W_BACKSHIELD():
11     set_tcp(p[-0.0001004599,0.0002301469,0.3294465624,0.0000000000,0.0000000000,0.0000000000])
12     # $ "wp695"
13     movep(p[0.0213114485,0.1989209301,1.2537783046,-0.0187575204,0.1235261397,3.1008550200], a=0.05, v=0.2, r=0)
14     set_tcp(p[-0.0001004599,0.0002301469,0.3294465624,0.0000000000,0.0000000000,0.0000000000])
15     # $ "wp696"
16     movep(p[-0.1014307212,0.2027250535,1.2501663918,-0.0668975905,0.1223421639,3.1014640393], a=0.05, v=0.2, r=0)
17     set_tcp(p[-0.0001004599,0.0002301469,0.3294465624,0.0000000000,0.0000000000,0.0000000000])
18     # $ "wp697"
19     movep(p[-0.2234049599,0.2100653041,1.2431285811,-0.1064726561,0.1126735900,3.0560351791], a=0.05, v=0.2, r=0)
20 end

```

Figure 18. Screenshot of a script file

3.1.6. Upload a program from URSim to Process Simulate.

The opposite process is also simple. After creating a program with URSim, multiple files are generated, including .urp and .script. As mentioned before, .urp files will only be used to open and modify them with URSim, so the file we are interested in is the .script.

We copy it to the directory of our Process Simulate project and with the **robot** → **tool program** → **Upload Program** we select the copied file.

Once uploaded, in a similar way to the previous one, it is necessary to eliminate 2 OLP Commands that say *while(true)* and *end*. They are usually the first and last points on the list.

4. Weld check cell

The second concept is about checking bodywork welds. It is another process where the human and robot works can be combined. In this case, the robot checks the welds appearance, position, length and pores in places that it is impossible or difficult for operators to get to.

The main idea was developed with the Fanuc's CR-7iA, which is a smaller size cobot and with very approximate characteristics with the UR10's.

Table 3. Comparison between UR10 and CR-7iA

| UR10 | Repeatability | Mechanical weight | Payload | Reach | DOF | Motion range | | | | | |
|--------|---------------|-------------------|---------|---------|-----|--------------|----------|-------|--------|--------|--------|
| | | | | | | Base | Shoulder | Elbow | Wirst1 | Wirst2 | Wirst3 |
| | ±0.1 mm | 28.9 Kg | 10 kg | 1300 mm | 6 | ±360 | ±360 | ±360 | ±360 | ±360 | ±360 |
| CR-7iA | | | | | | J1 | J2 | J3 | J4 | J5 | J6 |
| | ±0.01 mm | 55 Kg | 7 Kg | 911 mm | 6 | 340 | 166 | 383 | 380 | 240 | 720 |

As I did not have the Process Simulate model for the CR-7iA, the model was developed with the UR10 model.

As it is a small robot, it can be handled above like in the previous model (section 2), because the inspection zone will be the bodywork of the car and from that position the robot can reach easier all the points. To check the weld, on the robot will be mounted a vision system.

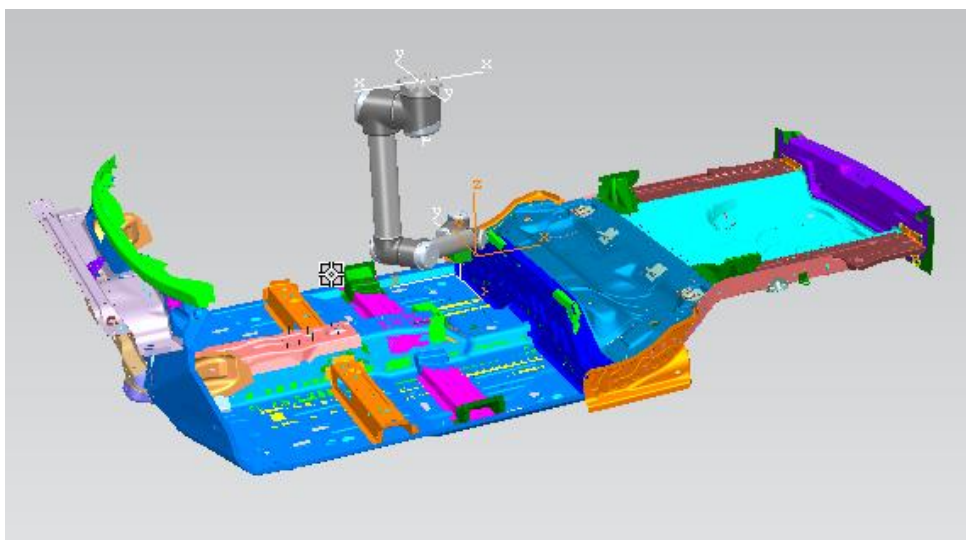


Figure 19. Picture of the process.

4.1. Idea development.

It was an easy model to make. First, a complete car model, COJT file, was placed only using the bodywork. Then, the robot was placed above the bodywork, in a middle point to reach the important parts of the piece.

After positioning all the parts needed there were only two things to do left: creating a tool to simulate de vision system and the programming of the movements.

4.1.1. Tool creation.

As I did not have particular Vision System for the process I created a resource to simulate the camera of the inspection system¹ [FANUC, 2018].

For doing that I created a new resource at **Modeling** → **Components** → **Create new resource** and within this resource I created two elements to simulate the camera: a box, which pretends to be the body of the camera; and a cylinder to simulate the part which joints with the robot. Finally, I added a frame as the TCP; and it is separated of the object because, normally, the vision systems need to be on a certain distance to analyze the devices they want to check.

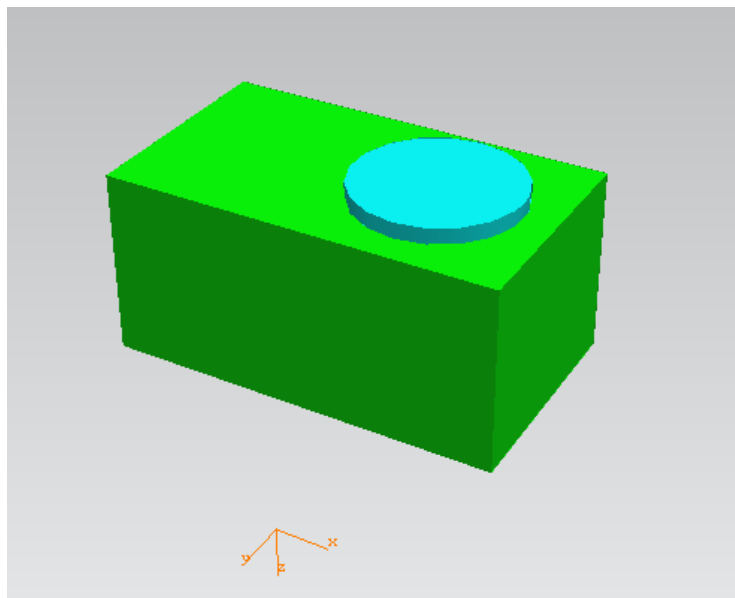


Figure 20. Simulation of the vision system.

¹ <https://www.fanuc.eu/bg/en/customer-cases/audi>

4.1.2. Process programming.

The way of programming the process was the same as in the gluing cell using weld points and projecting them to the part.

On a bodywork the weld parts are not as big as the car length, there is only a few parts. In this model I select five imaginary weld parts; especially on the most difficult shapes of the bodywork because I wanted to show how the robot can perform the action there.

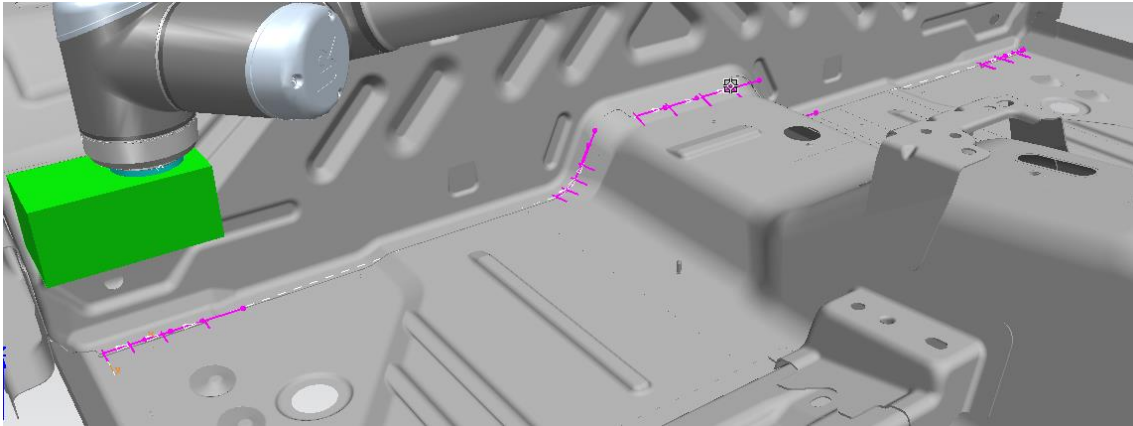


Figure 21. Check points on the bodywork.

Going in depth with the motion parameters of the process, the robot needs to go slower

4.2. Conclusions

The main target of this simulation was to analyze if this process can be made by the company and to go in depth with different cobot processes and performances; and the result was successful.

5. Spare wheel and battery insertion.

The following concept is about the spare and battery insertion. Nowadays, it is not very common to see these processes automatized, but with cobots that could change.

Normally, cobots cannot handle with big payloads. FANUC has one which has a maximum payload of 35 kg, the CR-35iA.

Table 4. Main characteristics of CR-35iA

| CR-35iA | Repeatability | Payload | Reach (mm) | DOF | Motion range(degrees) | | | | | |
|---------|---------------|---------|------------|-----|-----------------------|-----|-----|-----|-----|-----|
| | | | | | J1 | J2 | J3 | J4 | J5 | J6 |
| | ±0.03 mm | 35 Kg | 1813 | 6 | 370 | 165 | 258 | 400 | 220 | 900 |

The main idea is to have two of those cobots that insert at the same time the spare wheel and the battery in the car. Also, if it is possible, do that on line tracking. Line tracking is an automation practice in which the process is made on a certain part while it is moving, without the need to stop the line.

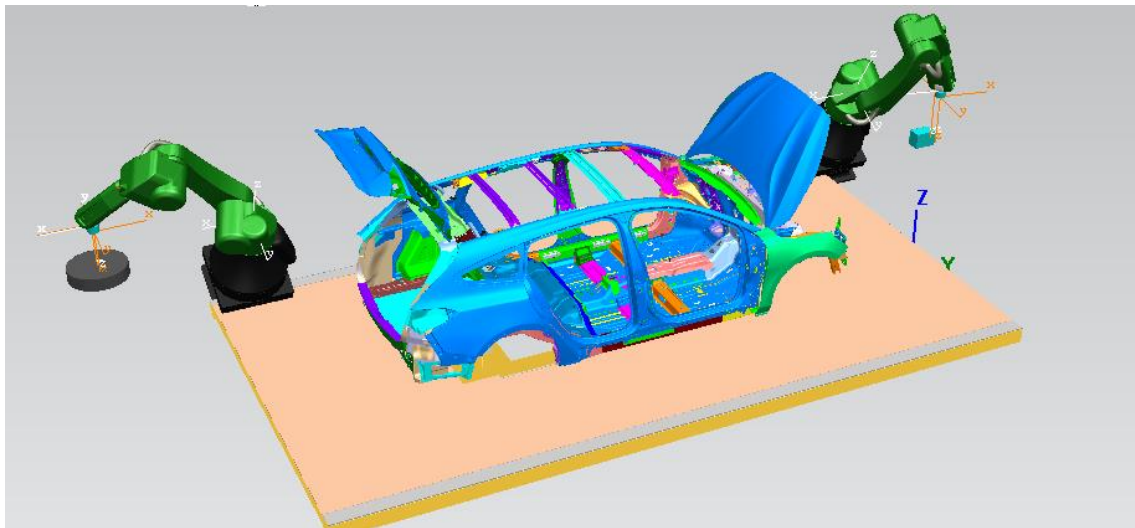


Figure 22. Capture of the process.

5.1. Developing of the idea without line tracking.

The first thing to do was the simple idea, to program the insertion of the devices. But before that, it is necessary to prepare the gripper and mounted it on the robots.

5.1.1. Create the gripper.

The idea for this gripper was to make one similar to the one that is used on the robot demonstration [Fanuc, 2019]² by FANUC. For doing this I create a new resource and within it I added a cylinder. Instead of creating more parts, I added a TCP separate of the cylinder, as can be seen on *Figure 23*. Then, the tool is mounted on the robot and set as a gripper. This gripper is used by both robots.

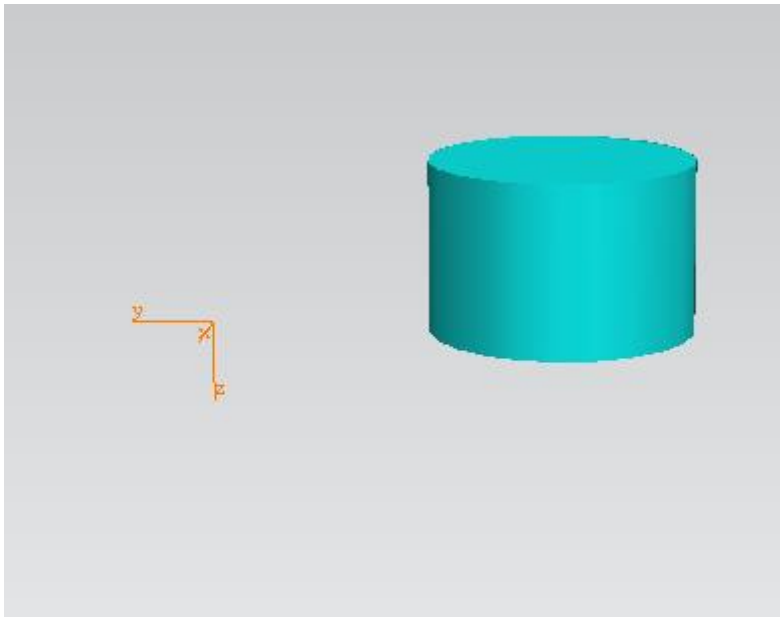


Figure 23. Gripper

5.1.2. Creation of the different parts.

The way to create the parts are almost the same as to create a resource. On **Modeling** → **Components** → **create new part** and adding new geometry; for the wheel a cylinder, and a box for the battery. To make them I used standard dimensions for this products.

5.1.3. Programming the movements.

This is a simple pick and place operation where the robots take the battery and the spare wheel and place it inside the car. To make it in Process Simulate, clicking on **operation** → **Create operation** → **new operation** → **New pick and place operation**, a dialog window

² <https://www.fanuc.eu/bg/en/customer-cases/audi>

appears, in which you have to select the robot and the pick and place point, and the operation is created automatically.

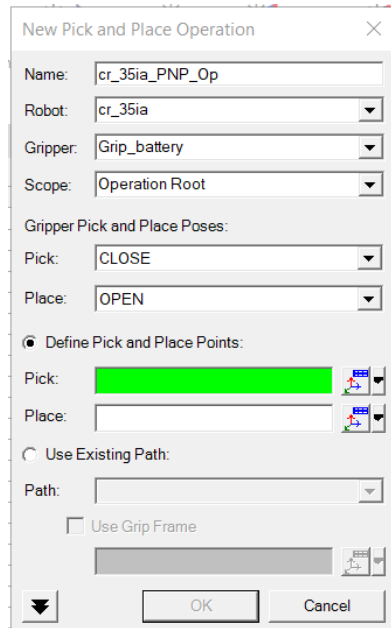


Figure 24. Pick and place operation dialog window

The only problem was that it creates an operation only with these two point, and for this process more points are needed, mainly for the insertion part, in which is necessary accuracy; otherwise, the robot can hit the car instead of insert the battery and the wheel. That is because I added 3 points between the pick and place ones. One for rise the object, one in the middle of the trajectory and one last to place the parts above the place point. In this processes I used the linear movement for the rise and down of the object; and, circular for the middle trajectory.

| Paths & Locations | Config | Attachm... | Utool | Uframe | Speed | Acceler... | Motion ... | Term ty... |
|-------------------|--------|------------|------------|--------------|------------|------------|------------|------------|
| cr_35ia_PNP_wheel | | | | | | | | |
| pick1 | | | 2-gripper2 | 0-WorldFrame | 100 % | 90 % | Joint | FINE |
| via | | | 2-gripper2 | 0-WorldFrame | 200 mm/... | 90 % | Linear | FINE |
| via1 | | | 2-gripper2 | 0-WorldFrame | 200 mm/... | 90 % | Circular | CNT0 |
| via2 | | | 2-gripper2 | 0-WorldFrame | 200 mm/... | 90 % | Circular | FINE |
| place1 | | | 2-gripper2 | 0-WorldFrame | 200 mm/... | 90 % | Linear | FINE |

Figure 25. Screenshot of the path editor with the properties of the movement

5.1.4. Robot position

Having done all this, you could see that the robot did not have a wide range of motion, so it was necessary to modify the position in which it was so that it could perform all movements with more freedom and without danger of colliding with the car. After

different tries the final option was chosen, putting the robot at 30 degrees of inclination in such a way where it has more opportunities of movement and avoiding collision.

5.2. Developing of the idea with line tracking.

For develop this idea it is going to start from the previous part, because the structure of the cell is the same; but first, it is necessary to understand how the line tracking works and how must be done on Process Simulate. Although the aim was to add line tracking to the spare wheel and battery insertion concept, all the simulations were done creating basic elements to experiment and investigate how it works. Due to the different problems, the process was tested with 2 different robots.

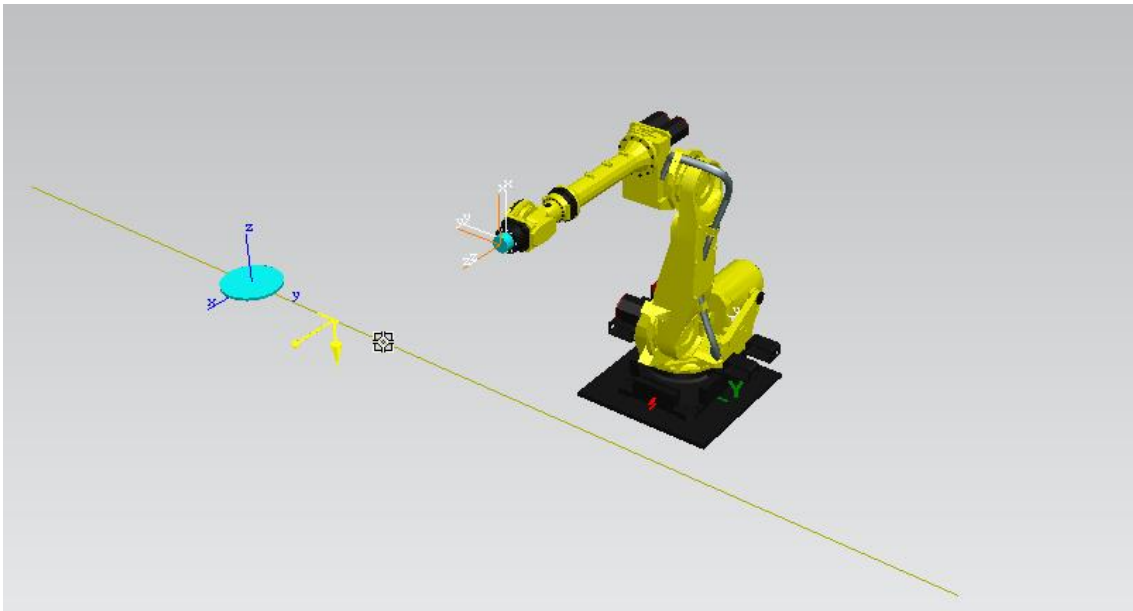


Figure 26. Basic simulation for line tracking with a FANUC robot

5.2.1. Line tracking. Definition.

There are some actions that need to be done in a certain moment and stop the line for that can make fail the process. A line tracking process is one in which the different actions needed are done while the production line is moving; giving the advantage of not stopping the line.

The line tracking is based on the setting of a moving reference frame which modifies its position in the world with the line movement. And referring the action, which the robot must do, to that frame. Normally the movement is measured by an encoder and its signal is used to modify the global position of where the robot has to go.

The robot is not constantly following the same piece, it has a tracking space or range in which it follows the pieces. It normally works as follow: the robot waits until a piece enter in the tracking space (trigger); then, the robot starts to do the action following the piece and when it is finished the robot stop following the piece and waits again until the next piece. It can be explained as the robot catch the moving frame whit the trigger signal and drops it when the action is done. If the action is not done when the piece is on the track limit, the line stops until it finish.

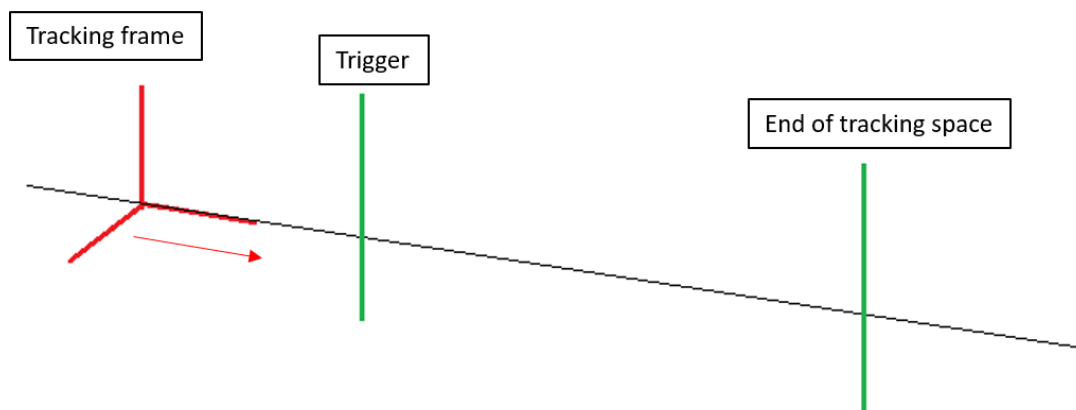



Figure 27. Representation of line tracking

5.2.2. Create the conveyor

To do a line tracking on Process Simulate the first thing to do is to create a conveyor. The conveyor is the moving part of the process, which you can simulate the line with. To create it I followed a tutorial from the Siemens help platform [Siemens, 2019]³. When the conveyor is created, the next step is to add an object that moves along the conveyor, a skid. To create it, first a resource has to be created; and, after that define it as a conceptual skid at **control** → **conveyor** → **Define as a conceptual skid**.

With all this created, it is time to activate the controller's line tracking option. At the robot properties window, on the **conveyors** tab, there will appear one of the conveyor signals that was created before, following the tutorial, and the opportunity to add that signal to a conveyor clicking on the **associate conveyor with signal** button .

³https://docs.plm.automation.siemens.com/tdoc/tecnomatix/14.1/tecnomatix_eMS/#uid:RoboticsMenu_RobotConveyorTracking

You can also add control points on the conveyor. There are three different types of points: control point, 'generate skid appearance' point and 'destroy skid appearance' point.

- Control point: Creates a new control point on a selected point on the conveyor. With them you can modify the speed and the orientation of the skid when it pass through the point.
- 'Generate skid appearance' point: enables you to create a skid appearance on that point, it is important to select a skid resource as the prototype for the appearance.
- 'Destroy skid appearance' point: generates a skid deletion on that point.

After creating and setting all the basic things of the conveyor, the control of the process featuring with the conveyor depends only on the robot controller type.

5.2.3. Tracking with FANUC.

On Process Simulate there is the opportunity to connect the robot controller to the RCS module. The RCS, or Real-time Control System, module is a reference model architecture suitable for real-time control problem domains. It defines the types of functions that are required in a real-time intelligent control system and how these functions are related to each other. Each robot brand has its own RCS module. This RCS module is helpful to make more accurate simulations, because it shows the real limits of the robots.

In order to make the most accurate simulation, the first thing to do is connect the robot to this module. For this, I had to create a virtual robot using a FANUC software. This software is installed in Window XP virtual machine. The first thing to do after opening the virtual machine is to check the communication between the virtual machine and the host doing a ping⁴ from the virtual machine to the host and vice versa. Then, with the FANUC software you can create a robot: first, you select the robot model, and, then, select the options you want you robot to have.

⁴ Ping (Packet Internet Groper) is used to check if a certain network interface, from our computer or another, is active. The PING sends packets to the IP or host that is indicated, and tells us how long it took the packet to go and return, among other little information.

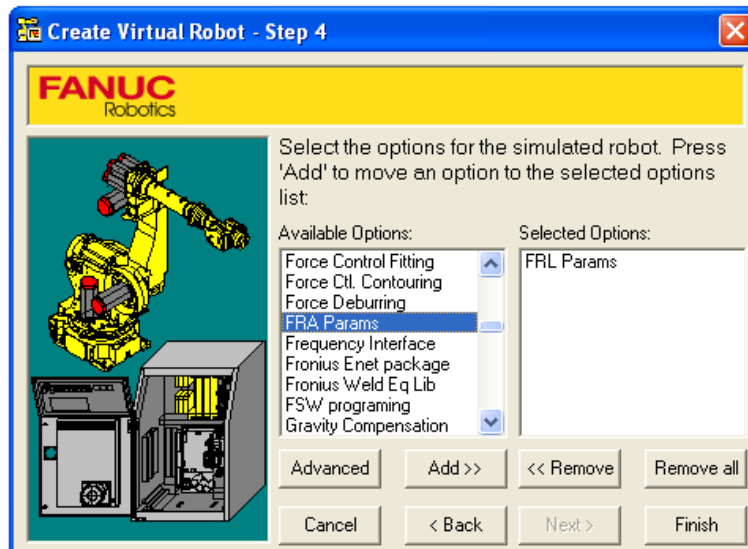


Figure 28. Screenshot of the FANUC software on the option selection step.

After this, it is necessary to check if the robot was successfully created opening the web browser and writing the following direction: <http://localhost::9080>. In the direction it will appear the option to visit two sites: Named Robots and Active Virtual Robots.

- Named Robots: Virtual robots successfully created.
- Active Virtual Robots: The ones that are currently in use. In other words, those to whom we have connected through Process Simulate.

Before connect the robot controller from Process Simulate to the virtual robot it is necessary to modify the file called “*specific.cfg*” located in the following path: `C:\apps\robcad\rss_bin\rscfr13\robcad.bin\VERSION\8.30`. That file is a list of robots that are created, so the only thing left is to add the name of the virtual robot that has just been created.

Now, for connect the controller to the RCS module, on **Robot Properties** → **Controller** switch to *Connect* the option on the boxes: ‘RCS in simulation’ and ‘RCS in Non Simulation’. Now when you want to go to the **Robot Setup** the controller is going to connect with the RCS module.

5.2.3.1. RCS problem.

The first idea was to do the simulation with the CR-35iA, but when it tries to connect with its RCS module there appear an error. After trying different things, the cobot was replaced by another robot model with similar characteristics, the r2000ib/165F. That

was made because the target of the simulation was to check how the line tracking can be implemented with Process Simulate, and, in the end, both are FANUC robots that have the same controller.

5.2.3.2. Programming

As explained in point 5.2.1, the line tracking programming consists in making the robot wait until the skid object arrives to the trigger point and, thereafter, follow it until finish the task.

In Process Simulate, first the line tracking option has to be configured on the robot controller. In the **robot setup** window there appears the line tracking option. In that settings you can select the track frame, normally one that exists yet; select the scale and the teach distance, and, the different boundaries for the tracking task.

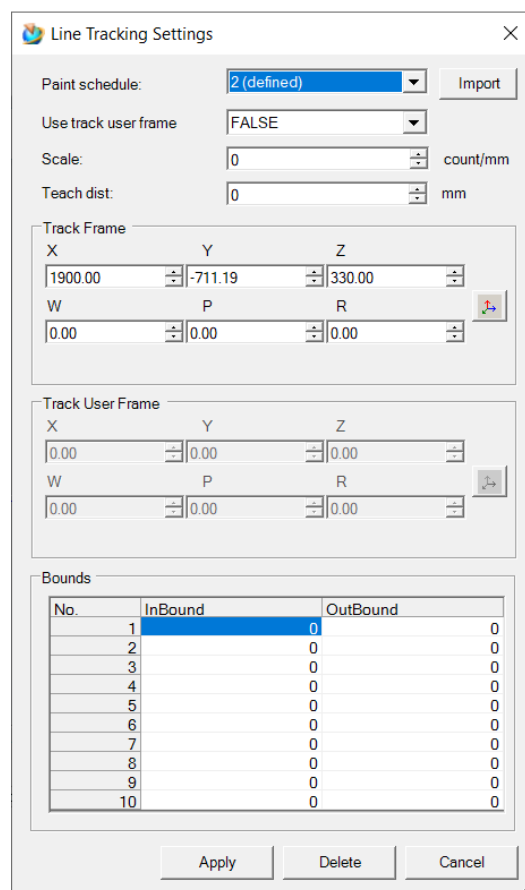


Figure 29. Line Tracking Settings window

For the programming of the action I create a simple movement of the robot in order to follow the skid resource. For create the movements, the steps to follow are the same as for another process, but attaching via point to the moving resource. The final thing to

do is to add to the point the OLP commands to make the robot wait to the trigger and to drop the tracking frame.

5.2.3.3. Conclusion.

After different changes on the programming and configuration I was not able to make the line tracking works on Process Simulate. The most possible problem is that there is something missing on the robot configuration or in the programming, but I did not find it.

With the multiple problems I had, I decided to replace the robot by an ABB; because I found evidences that with these robots is possible to do it.

5.2.4. Line tracking with ABB.

With ABB there is also the opportunity to connect the robot with the RCS module. In this case it is a software that has to be active while you want to use it.

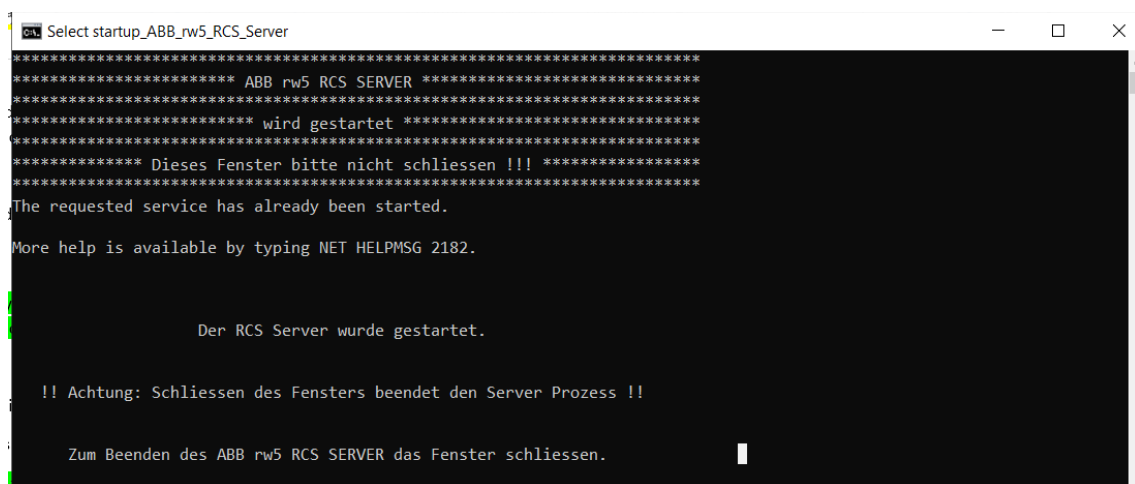


Figure 30. Screenshot of the ABB RCS server

Opposite with FANUC, this RCS server can be used for any ABB robot, is not necessary to create virtual robots to use it.

5.2.4.1. MOC file

For ABB, apart from the main things that are explained before, it is necessary to add some extra information, because it is not enough to associate the conveyor on the robot properties. There is needed which is called a MOC file. A MOC file is a file generated by RobotStudio in which there are defined extra axes for the robot, conveyors, etc. For

generate this file it is necessary to create one conveyor on RobotStudio first and then load the file on Process Simulate. The MOC file should be uploaded from the **robot setup** window in the *Load Machine Data* option.

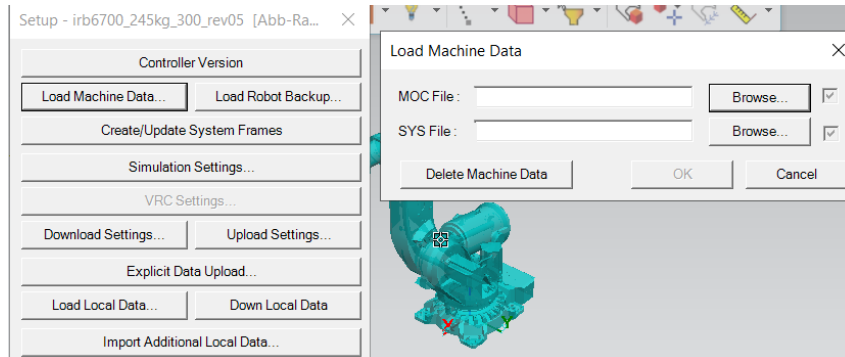


Figure 31. Screenshot of the ABB robot setup

After that, I programmed a similar task as the one I made with the FANUC robot. Also in this case it has to add OLP commands. The ones that have to be used are *WaitWObj* and *DropWObj*, which are used for waiting for the part and for drop the reference when the task is done.

5.2.4.2. Conclusions

Here happened the same as with the FANUC model, it did not work. I also think that the problem has to be in some missing data or because a bad configuration in Process Simulate.

In this case I previously made a simulation in Process Simulate and it worked perfectly, so I am sure that it is possible to make it.

6. Paint and Coverage Simulation

Nowadays, the final assembly processes are starting to be automatized. One of those processes is paint and coverage. Besides the process itself, it is important to optimize it; and, here is where the simulation can help.

With Process Simulate is possible to simulate the paint and coverage due to the tools it has designed to do so. The goal of this concept is to know how to use those tools and help to optimized the processes and, also, make more visual the simulations in order to present them.

6.1.Tool definition.

First thing to do is define the tool. This is one of the most important parts of this concept, because with a bad definition of the tool the coverage simulation will not work.

On Process Simulate painting tools are defined by three frames: base, tip and TCP.

- Base: where is attached to the robot.
- Tip: where the paint brush starts.
- TCP: with it is possible to manage the distance between the gun and the painting face.

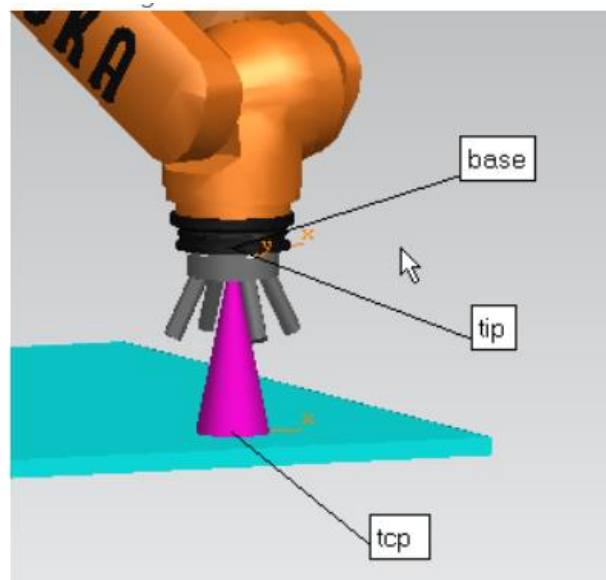


Figure 32. Representation of a paint gun and its frames

6.2. Brush definition.

To simulate the brush, the paint gun model must have a cone with the vertex on the tip frame and the base on the TCP frame. To define the brush, at **Process** → **Paint and Coverage** → **Paint Brush Editor** select the cone and the origin frame, which is the tip frame.

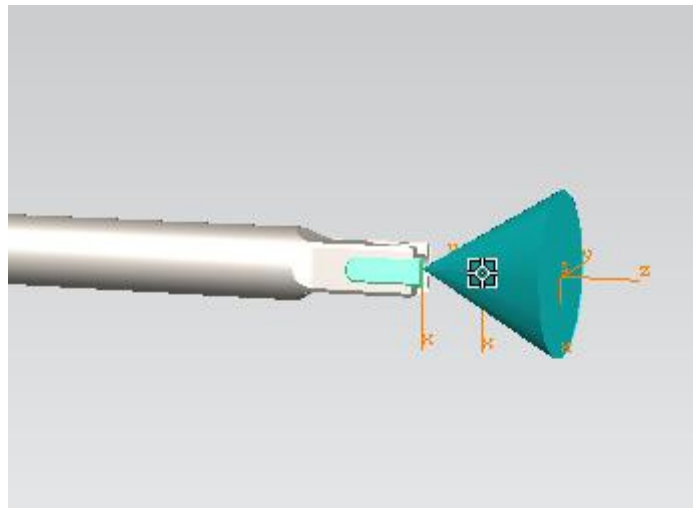


Figure 33. Picture of the brush on the paint gun used in the model

It is important to define the brush and the paint gun with the correct frames, otherwise the simulation will not work properly and the cone will become purple that indicates that there is a problem with the brush definition.

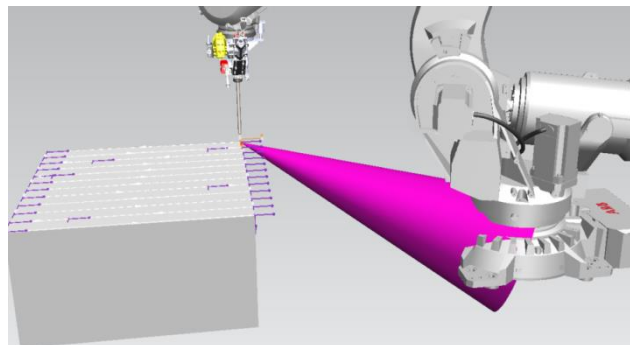


Figure 34. Picture of a bad definition of the paint gun and the brush

6.3. Creating the process.

For this part there are three tasks: generate the robot movements along the painting face, create a point mesh, which will help for simulate the paint; and set the OLP commands on the path editor for chose, open and close the paint gun.

6.3.1. Generate a continuous operation

A continuous operation is one that is developed in one or more faces of a part. After selecting the faces, you select the length of the operation and the times that you want to repeat it along the face. This is very useful for painting, because is an easy way to program the robot to move through a face to cover it.

At **Process**→**Continuous**→**Continuous Process Generator** its dialog window appears. There you can select between *Coverage pattern* and *Arc*, depending on whether it is for painting or an arc operation.

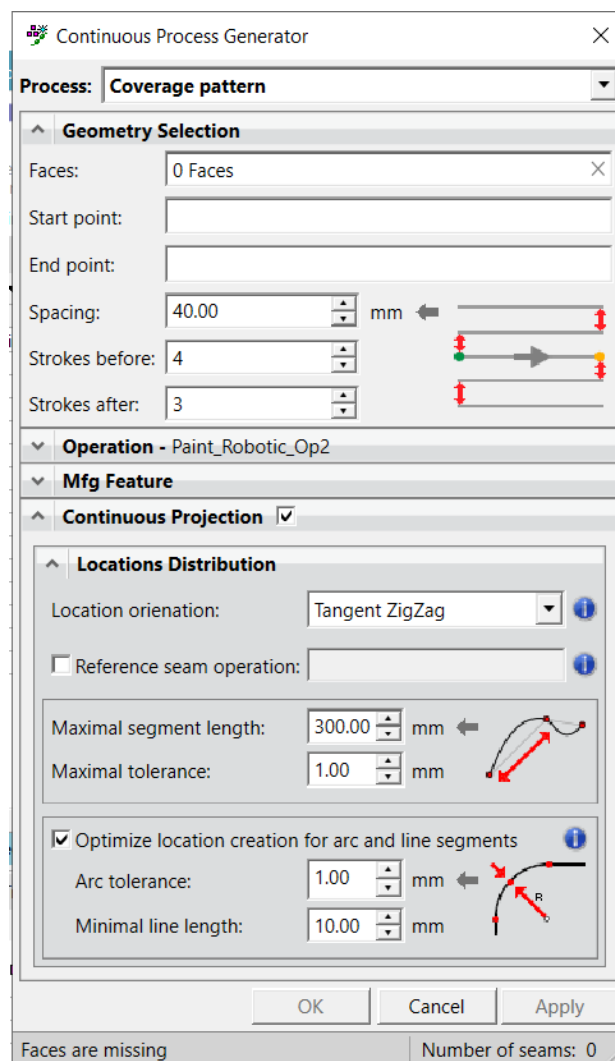


Figure 35. Continuous process generator window

On the coverage pattern first you can select the faces you want to cover. Then, the start and end points of the movements. Also, you can select the spacing between the strokes and the number of it that you want to be before and after. On the *Operation* tab you

should select the robot and the tool. Also, on the *Continuous projection* tab is possible to define how to distribute the locations setting their orientation, segment length and tolerance; and, in addition to that, it is possible to select the optimization parameters for arcs and line segment creations.

6.3.2. Creating the mesh.

With the operation created the next step is to create a 3D mesh of points. It is for the measuring of the coverage. It is essential for the coverage simulation.

To create the mesh, go to **process → Paint and Coverage → Create Mesh**. Then, select the parts you want to create the mesh on. In the *Create mesh* window it is possible to see if the part has an exact geometry and if it has already a mesh. Also it is possible to modify the *tessellation tolerances* with you can make the mesh more accurate. The creation of the mesh takes time, so it is better to set the appropriate values of the parameters. Those parameters are:

- Distance: The maximum distance allowed between adjacent vertices of the mesh.
- Deviation: This sets the maximum allowed divergence of the approximate geometry from the exact geometry
- Angle: This sets the maximum angle allowed between adjacent approximation items

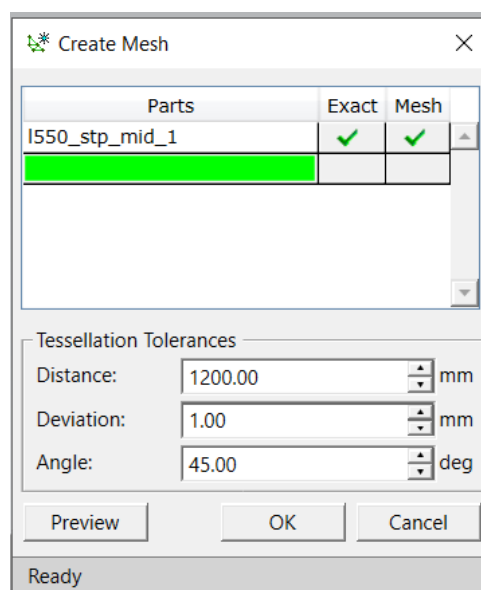


Figure 36. Create mesh window

6.3.2.1. XT-Brep exact geometry.

At first I had a different part to paint on, but it did not have an exact geometry, what caused an error and the no process of the mesh.

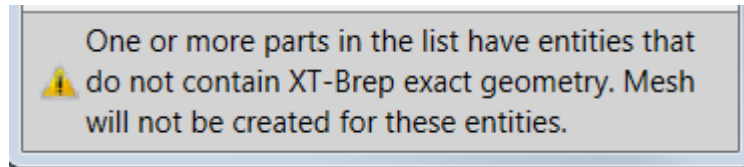


Figure 37. Error message when the par has not exact geometry

The XT-Brep exact geometry is a data format that Siemens use and is one of the best in providing solid representation for JT files. The meshes are created thanks to the information that this kind of formats give.

This exact geometry is something similar to the writing permission on a document, because it gives lot of information about the design of the parts of the products. Some factories do not want to put at risk a product that will go on sale soon, products such as cars, and because of that they usually do not give models with this information, to avoid the copies.

6.3.3. Programming and setting the simulation parameters.

After creating the mesh and the continuous operation, the only things left are the OLP commands and the option for the covering during the simulation.

On the continuous operation, in the first point it must be added two commands: *ChangeBrush* and *OpenPaintGun*. The first one is for selecting the brush wanted, here is not possible to select the brush, it must be written with the exact name; and, the second one is for activate the painting.

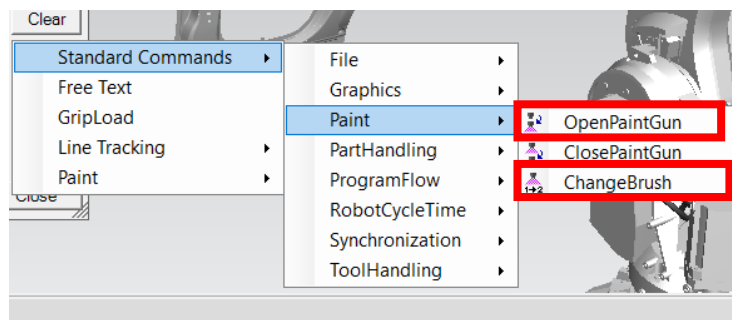


Figure 38. Screenshot of the OLP commands

Finally, at **process**→**Paint and Coverage**→ **Paint and coverage settings** we can configure the color of the different layers and what we want to see as the mesh (only while the window is open), paint or the brush while the simulation is not running. And for activate the Paint and Coverage it is important to activate the option **Paint and Coverage during simulation**.

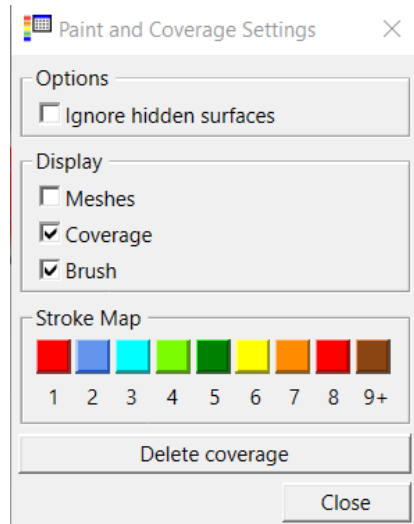


Figure 39. Paint and coverage settings window

6.4. Result and conclusions

The results of the simulations were not as I expected. The coverage does not fit to the face shapes, as can be seen on *Figure 40*. If the geometry is one created with Process Simulate there is no problem, but with an imported one yes.

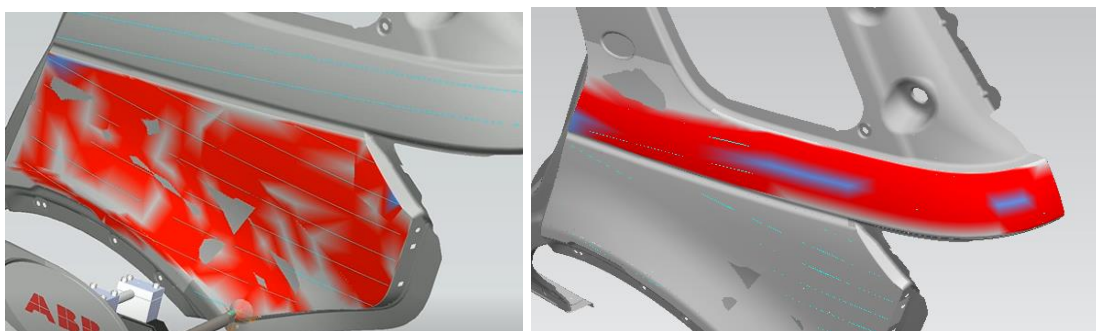


Figure 40. Result of the simulation

In *Figure 40* it is possible to see that the coverage is not good, because the paint is not equal along the faces. For example, on the picture of the right can be seen that there are some zones where a second layer (color blue) appears and that is because of a bad paint distribution.

At first I thought that the problem was in the mesh creation, but I realized that Process Simulate does not recognize the shape while the creation of the continuous operation (Figure 41).

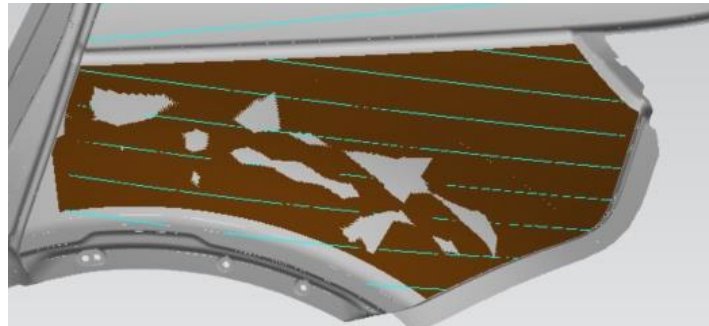


Figure 41. Face while creating the continuous operation

You can also see that depending on the speed that you program the movements, the result of the painting is different. With less speed the painting will have a better distribution.

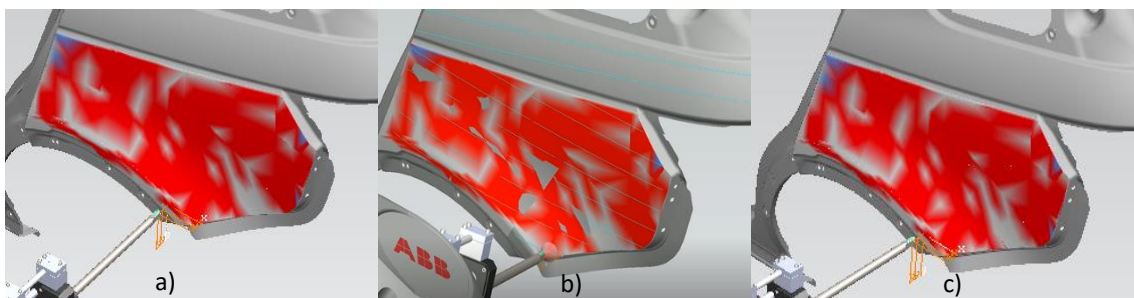


Figure 42. Painting simulation with different speeds: a) v50 b) v100 c) 60

To sum up, the tool is not as useful as I thought. The intended use of this application was to optimize the painting processes and for making a more realistic simulation. But now this is not good enough for that.

7. Communication between a Turck module and TwinCat3.

The aim of this part is to read the values of a distance sensor from Beckhoff's TwinCat software via a Turck input module.

The sensor is a *Keyence IL-600*, a laser sensor that is connected to one of the analog input channels of the Turck *BLCEN-4M12MT-4AI-VI*. The Turck module is connected with the laptop via a PROFINET adapter to Ethernet.

To reach the final result of reading the sensor data in TwinCat it is necessary to go step by step. First, check the configuration of the Turck module and whether it is possible to communicate with it via the Ethernet connection. Then, configure a project in TwinCat and read the data received from the module.

7.1. Turck module.

The *BLCEN-4M12MT-4AI-VI* is an input module from Turck. It has four analog inputs, two fieldbus ports, which will be connected to the Ethernet port and auxiliary power, one input and one output connectors. About the last ones, the input one is connected to a power source.

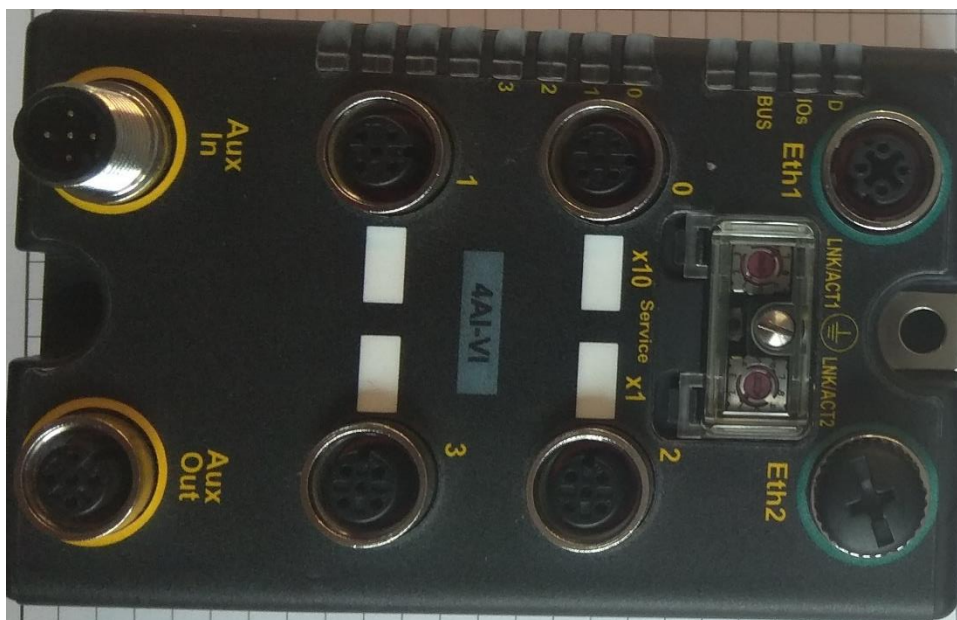


Figure 43. Picture of the *BLCEN-4M12MT-4AI-VI*

Apart from the different ports, there are status LEDs that give information about the status of the different parts of the module such as inputs, bus connection, etc. For

example, if the measure is out of range, it is possible to know because the status LED of the channel starts to blink.

Table 5. LED status of the Turck module

Station LED status

| LED | Color | Status | Description |
|---------|--------|-------------------|---------------------------------|
| IOs | | OFF | No power |
| | RED | ON | Low power or station error |
| | RED | FLASHING (1 Hz) | I/O module configuration error |
| | RED | FLASHING (4 Hz) | No I/O module bus communication |
| | GREEN | ON | Station ok |
| | GREEN | FLASHING | Force mode active |
| BUS | | OFF | Power Off |
| | GREEN | ON | Connected to Master |
| | GREEN | FLASHING | Ready |
| | GREEN | FLASHING 3x (1Hz) | ARGEE Running |
| | RED | ON | Error |
| | RED | FLASHING | WINK |
| | YELLOW | ON | DHCP/BOOTP Search |
| LNK/ACT | | OFF | No Link |
| | GREEN | ON | Link |
| | GREEN | FLASHING | Traffic |
| | YELLOW | ON | 100 Mbit Linked |

I/O LED status

| LED | Color | Status | Description |
|---------------------|-------|-------------------|---|
| D * | | OFF | No diagnostics active |
| | RED | ON | Station error/ module bus communication failure |
| | RED | FLASHING (0.5Hz) | Diagnostics active |
| AI channel 0...3 | | OFF | Not active |
| | GREEN | ON | Active |
| | GREEN | FLASHING (0.5 Hz) | Underflow in measuring range |
| | GREEN | FLASHING (4 Hz) | Overflow in measuring range |

* D LED also indicates gateway diagnostics

Apart from all this there are two rotary switches, covered by a small glass, that are used either for setting the IP address or the device mode operation. With them it is possible to set the configuration parameters necessary for the connection.

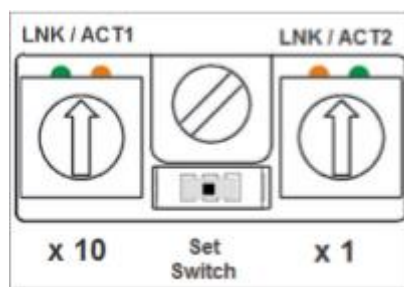


Figure 44. Representation of the rotary switches

The different operation modes of the device that can be determined by the switches are the following:

- 00: default configuration

- 01-92: rotary mode 1...92 that correspond to the last octet of the module's IP address. The 100,...,254 are available with the PGM mode
- 93: BOOTP, that obtains the IP address from the BOOTP server.
- 94: DHCP, that obtains the IP address from the DHCP server.
- 95: PGM, in which you can change the IP address from the web server of the device. That is what I chose, and will be explained before more in depth.
- 96: PGM-DHCP, which is the out-of-the-box mode and provides the customer with powerful and convenient IP address setup.
- 97-99: Vendor specific address.

7.2. Connection to the Turck module

As I said before, the first step is to connect the Turck module and check if it is possible to read from it. For it, the first step is to configure the IP address, then configure the general settings and, finally, check if it is possible to read from the device.

7.2.1. IP address setup

As the connection is via Ethernet, it is important to set up correctly the IP address, and for that there are the rotary switches (*Figure 44*). Normally, the rotary switches are in the position 00, which is the default configuration that has the following parameters:

- IP address: 192.168.1.254
- Subnet mask: 255.255.255.0
- Default gateway: 192.168.1.1

The device mode operation I chose was the PGM (95) because it is possible to change the IP address from the web server of the device, giving me the possibility of select the whole range of addresses in an easy way.

To select this, I firstly had to rotate the switches to the 95 position and cycle the power to the device. With this mode the device keeps the last address used, in my case the default one. Once the device is on, the IP address can be changed from the web server. The IP address I chose for the device was 192.168.1.190.

7.2.1.1. Web server.

To have access to the web server it is necessary to enter the current IP address into the web browser. From the web server it is possible to see the device information about the gateway, network, Ethernet and the parameters and values of the inputs. For the last ones is necessary to log in as administrator with a password, which by default is “password”. Also from here is where is possible to change the IP address, at **gateway** → **Network configuration**.

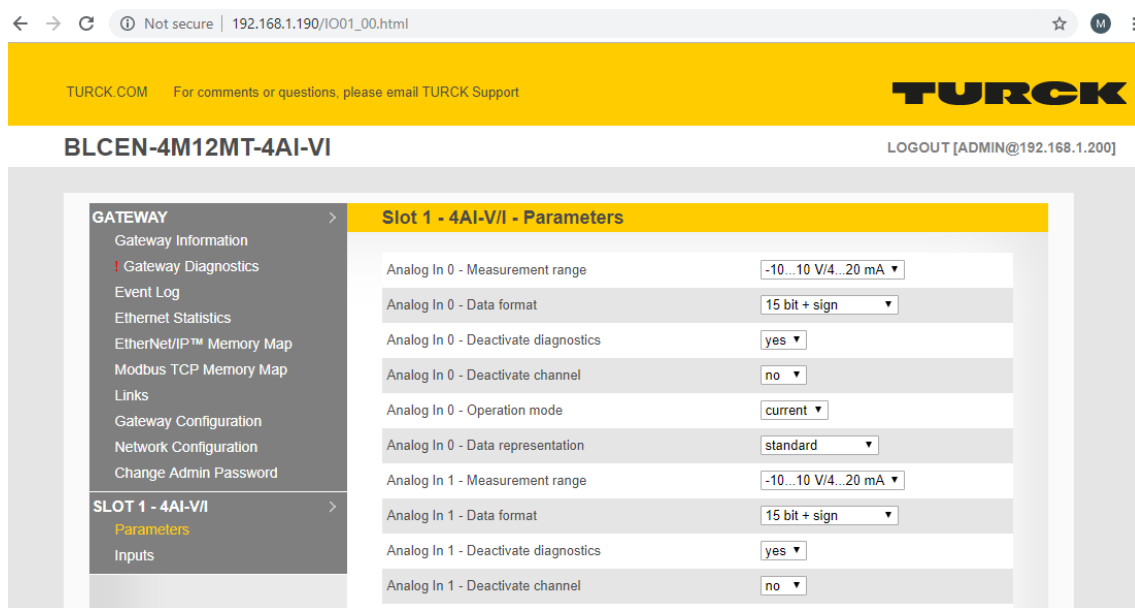



Figure 45. Screenshot of the web server

7.2.2. Problem with the firmware.

One of the biggest problems I had with the device was that I could not connect to the ARGEE⁵ environment and I could not see and change the parameters of the inputs. Without that I could not see if the device was receiving data. All of that was because of the device’s firmware was not update.

For the firmware update I had to use PACTware, a software supported by Turck for device management. For doing that, first it is necessary to search the devices. On **Project** window, with right click on “HOST PC” add device and select “BL Service Ethernet”. Then,

⁵ ARGEE: Programming environment of Turck.

on *Online available devices* tab it has to be selected the network in with it is going to search for devices and, then, click on the *Search* button  and the device appears.

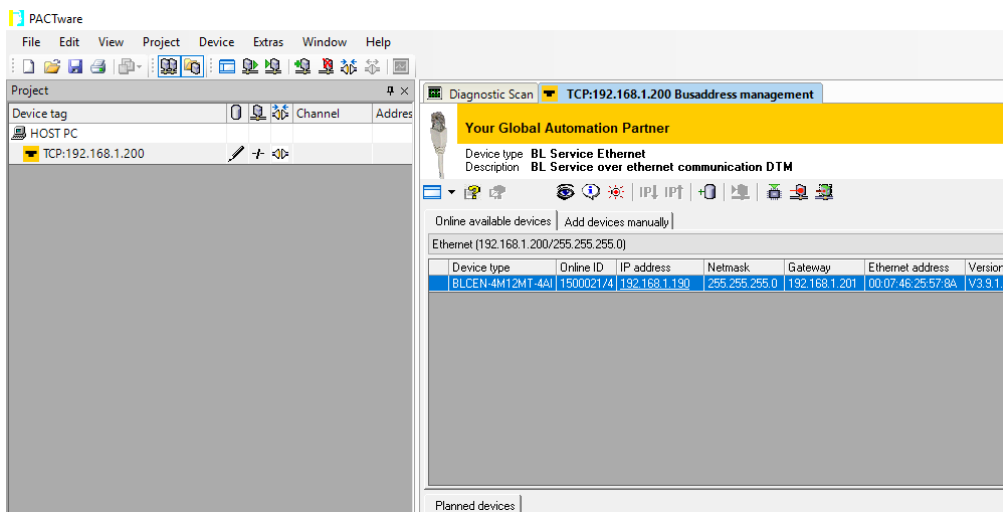




Figure 46. Screenshot of PACTware after searching the devices.

When the device is found, to update the firmware click on the *Firmware download* button . Then, the latest firmware file should be selected and, when the download is finished, search again and check the firmware version it has.

7.2.3. Read parameters from PACTware.

After solving the problem with the firmware I was able to have access to ARGEE and to all the features of the web server.

Also, I was able to read values from PACTware. Starting from what has been done for the firmware update, it is possible to connect to the device with the *Add device/DTM to project* button . Now, the device is added to the project and, automatically, PACTware scan the device and find its full topology.

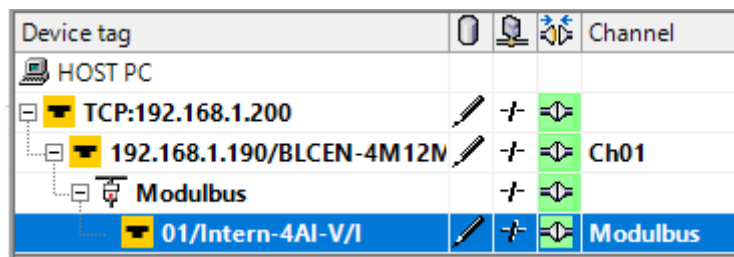


Figure 47. Device added to the Project

Now that the device is added and connected, it is possible to read the values of the different channels. For that, with a right click on the input module part and select the *Measured value* option the value read will be displayed.

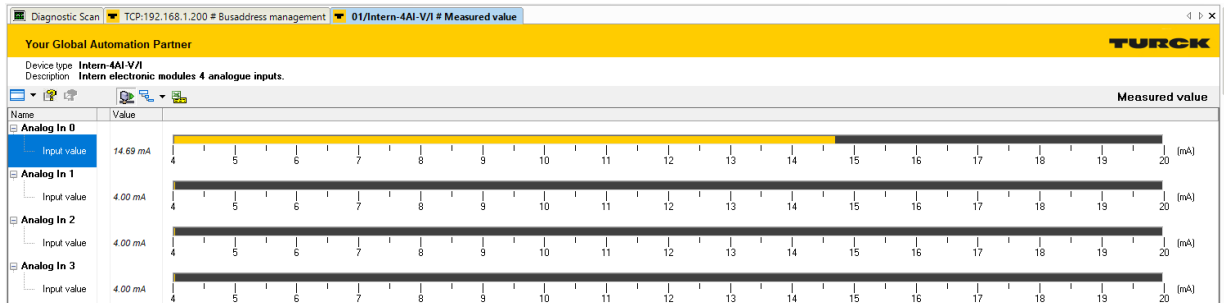


Figure 48. Measured values.

7.3. Connect to TwinCat 3

After ensure that the module received the signal from the sensor and I could read it with PACTware, the next step was to connect the signal with TwinCat 3.

TwinCat 3 is a SoftPLC developed by Beckhoff. TwinCat turns almost any PC-based system into a real-time control.

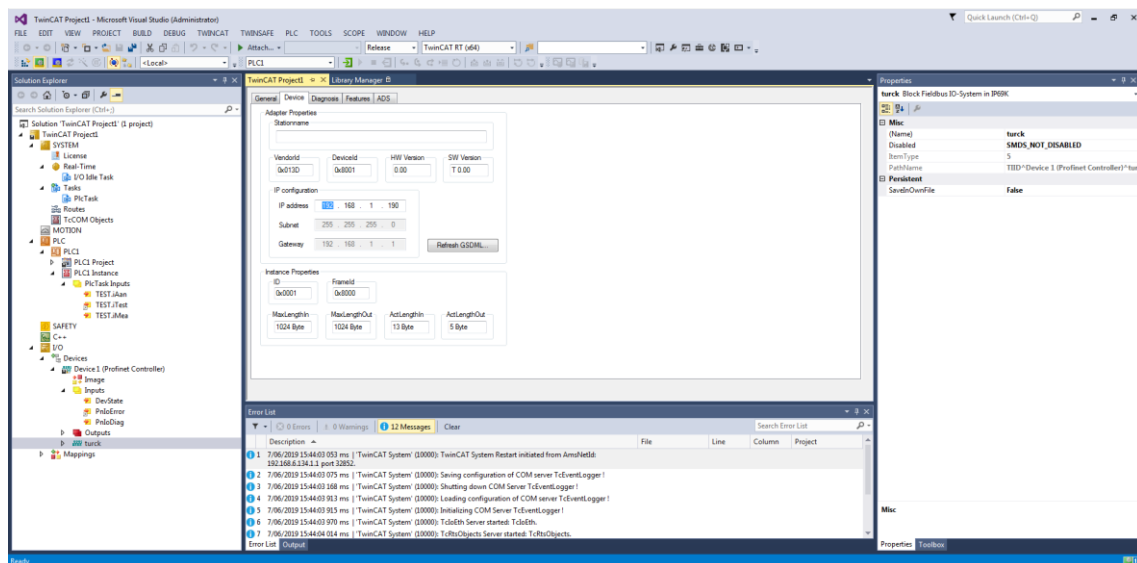


Figure 49. Screenshot of TwinCat environment.

Most of the information about TwinCat is for previous versions of Windows so that I used TwinCat in a virtual machine with Windows 7. For the communication between the virtual machine and the computer, the network has to be configured as a bridge.

The laptop has not a PROFINET adapter itself, but with the PROFINET wire to Ethernet it is possible to connect the module to the laptop. Because of that, the PC needs a specific driver to make it able to receive data from a PROFINET device. Just in case I install the drivers on the virtual machine and the laptop itself. It is possible to do from the network connection setting of Windows and, also, from TwinCat. There, at **TWINCAT** tab in the **Show Realtime Ethernet Compatible Devices...** option appears a window like *Figure 50* in which is possible to see the compatible devices. There, on the *incompatible devices* group appears the full name of the Ethernet device; selecting it and clicking on the *Install* button the driver will be installed and the name of the Ethernet appears on the compatible devices group.

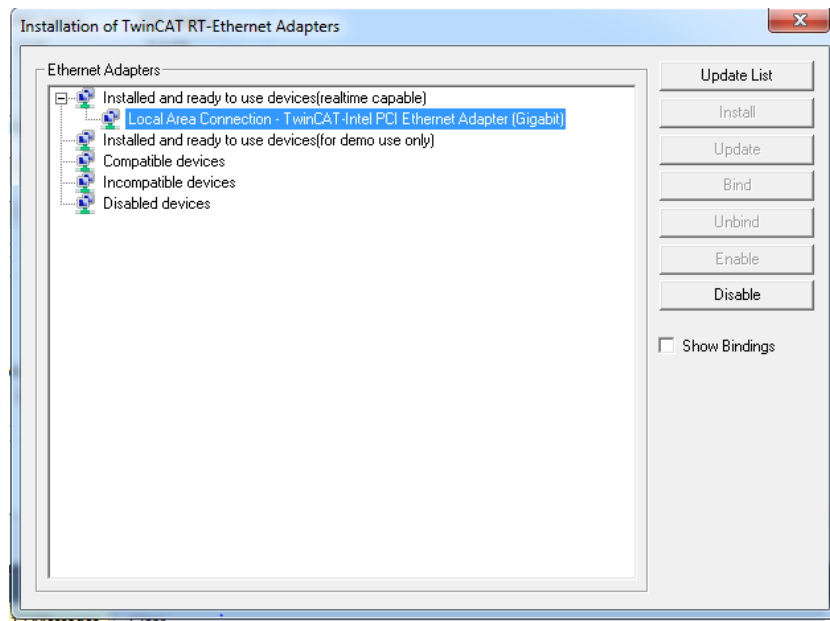


Figure 50. Realtime compatible devices Window

7.3.1. Configuration of the devices on a TwinCat 3 project.

In TwinCat it is possible to add the Turck module. For doing that, the controller must be declared first and within the device itself. From the Solution Explorer at **I/O → Devices** doing right click selecting *Add New Item* option there appear a Window. On that window select “Profinet I/O Controller (RT)”.

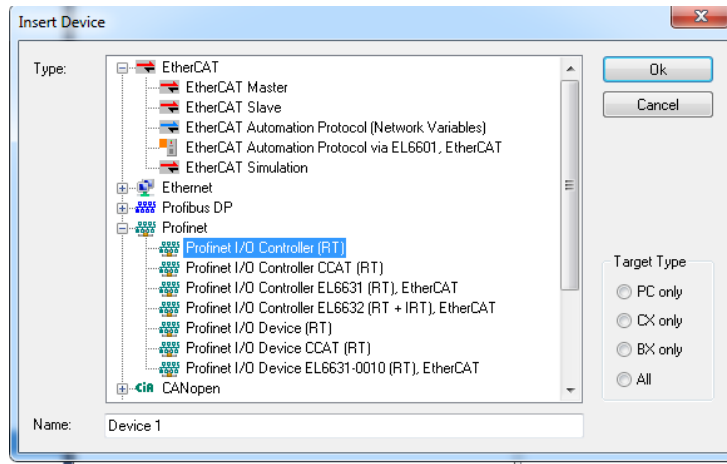


Figure 51. Insert Device window

Now, on the Solution Explorer appears the controller. In its options, on the *Adapter* tab, click on the *Search...* button to select in which port the device is located and select the Ethernet adapter. Then, on the *Settings* tab it is possible to configure the IP of the controller. There it must appear the IP of the controller (not the module's IP) and the subnet mask and Gateway directions. Those last ones are the ones configured before on the module's web server. As I did not make any change on that parameters, the subnet mask and the gateway are the default ones, as can be seen on *Figure 52*.

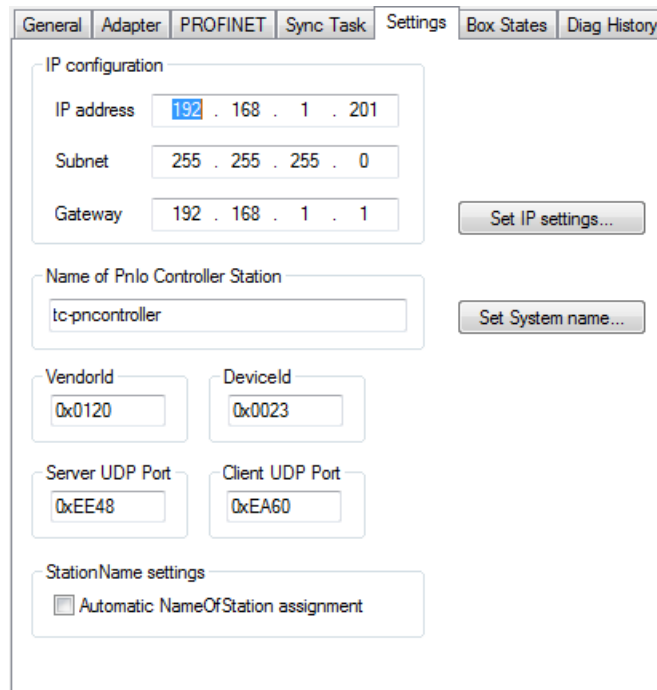


Figure 52. Setting tab

The controller is already configured. Now it is turn of the module. First, add a new item within the controller and select "PROFINET IO Device" within the *miscellaneous* group.

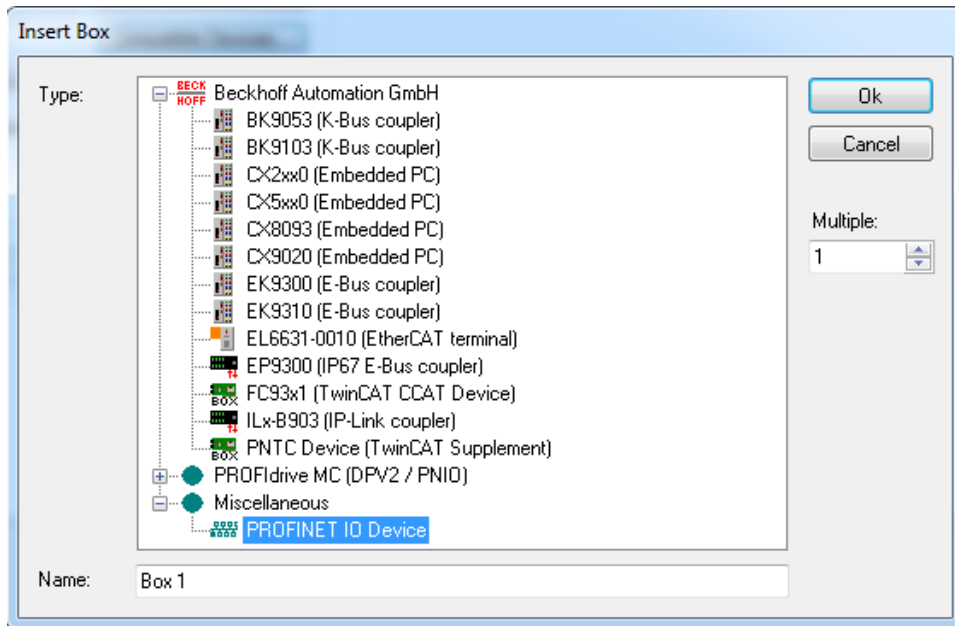


Figure 53. Insert Box Window

After that, the GSDML ⁶file of the module has to be selected to charge all the options and configuration of the file. In my case I selected the file: **GSDML-V2.3-Turck-BLCEN-20180921-010400.xml**. Now on the Solution explorer appears the full topology of the Turck module.

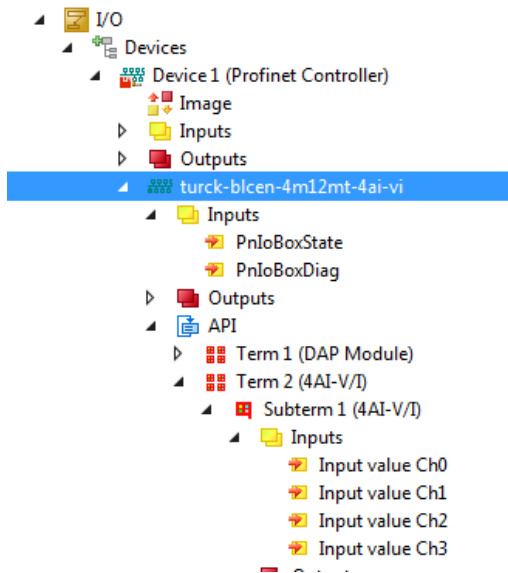


Figure 54. Topology of the device in TwinCat

⁶ General Station Description Markup Language file is the device description technology for PROFINET devices, regarding its identification, its structure, its communication features, its process data, its parameters and its diagnosis. [Profibus, 2017]

The last thing to do is the IP address configuration of the device. On the *Device* tab within the module options it is possible to do that. There, the subnet mask and the gateway are the same as in the controller; only it is possible to change the IP address. So, there I wrote the one I set before with the web server: 192.168.1.190.

7.4. Results and conclusions.

After doing all the configuration correctly I could not read the measuring data from the module. The only values I could read were the state values, which inform that there were an error and that the device was not found.

One of the problems could be in the driver, because the driver were developed for Intel Ethernet cards and mine was a Realtek. Nevertheless I was able to install the drivers in my laptop. Also, I tried to run TwinCat from my laptop instead from the virtual machine, but when I tried to switch TwinCat to run mode the PC break down and a blue screen appears. Apparently, the drivers cause the problem so I had to uninstall all.

I tried to use another laptop with an Intel card and it did not work either. Because, first, I had problems installing the drivers, and when the drivers were installed I did not read anything. The only software I could read from was PACTware.

All this could be because some configuration were wrong or, it is possible that read a Turck module is still not supported by TwinCat or that it is only possible to be read from their own software.

Appendix

A. Datasheets

1. UR10

UR10

Performance

| | |
|---------------------------|---|
| Repeatability | ±0.1 mm / ±0.0039 in (4 mils) |
| Ambient temperature range | 0-60° |
| Power consumption | Min 90W, Typical 260W, Max 600W |
| Collaboration operation | 16 advanced adjustable safety functions. TUV NORD Approved Safety Function Tested in accordance with: EN ISO 13849-2:2008 PL d |

Specification

| | |
|--------------------|---|
| Payload | 10 kg / 22 lbs |
| Reach | 1800 mm / 61.2 in |
| Degrees of freedom | 6 rotating joints |
| Programming | Polyscope graphical user interface on 12 inch touchscreen with mounting |

Movement

| Axis movement robot arm | Working range | Maximum speed |
|-------------------------|---------------|-------------------------|
| Base | ± 360° | ± 120°/Sec. |
| Shoulder | ± 360° | ± 120°/Sec. |
| Elbow | ± 360° | ± 180°/Sec. |
| Wrist 1 | ± 360° | ± 180°/Sec. |
| Wrist 2 | ± 360° | ± 180°/Sec. |
| Wrist 3 | ± 360° | ± 180°/Sec. |
| Typical tool | | 1 m/Sec. / 39.4 in/Sec. |

Features

| | |
|--------------------------|--|
| IP classification | IP64 |
| ISO Class Cleanroom | 6 |
| Noise | 72dB |
| Robot mounting | Any |
| I/O ports | Digital in 2 Digital out 2 Analog in 2 Analog out 0 |
| I/O power supply in tool | 12 V/24 V 600 mA in tool |

Physical

| | |
|------------------------|------------------------|
| Footprint | Ø 190mm |
| Materials | Aluminium, PP plastics |
| Tool connector type | M8 |
| Cable length robot arm | 6 m / 236 in |
| Weight with cable | 28,9 kg / 63.7 lbs |

CONTROL BOX

Features

| | |
|---------------------------|--|
| IP classification | IP20 |
| ISO Class Cleanroom | 6 |
| Noise | <65dB(A) |
| I/O ports | Digital in 16 Digital out 16 Analog in 2 Analog out 2 |
| I/O power supply | 24V 2A |
| Communication | TCP/IP 100Mbit, Modbus TCR Profinet, EthernetIP |
| Power source | 100-240 VAC, 50-60 Hz |
| Ambient temperature range | 0-60° |

Physical

| | |
|--------------------------|--|
| Control box size (WxHxD) | 476mm x 428mm x 268mm / 18.7 x 16.7 x 10.6 in |
| Weight | 17 kg / 37.6 lbs |
| Materials | Steel |

TEACH PENDANT

Features

| | |
|-------------------|------------------|
| IP classification | IP20 |
| Materials | Aluminium, PP |
| Weight | 1,6 kg / 3.3 lbs |
| Cable length | 4,6 m / 177 in |



CR-35iA



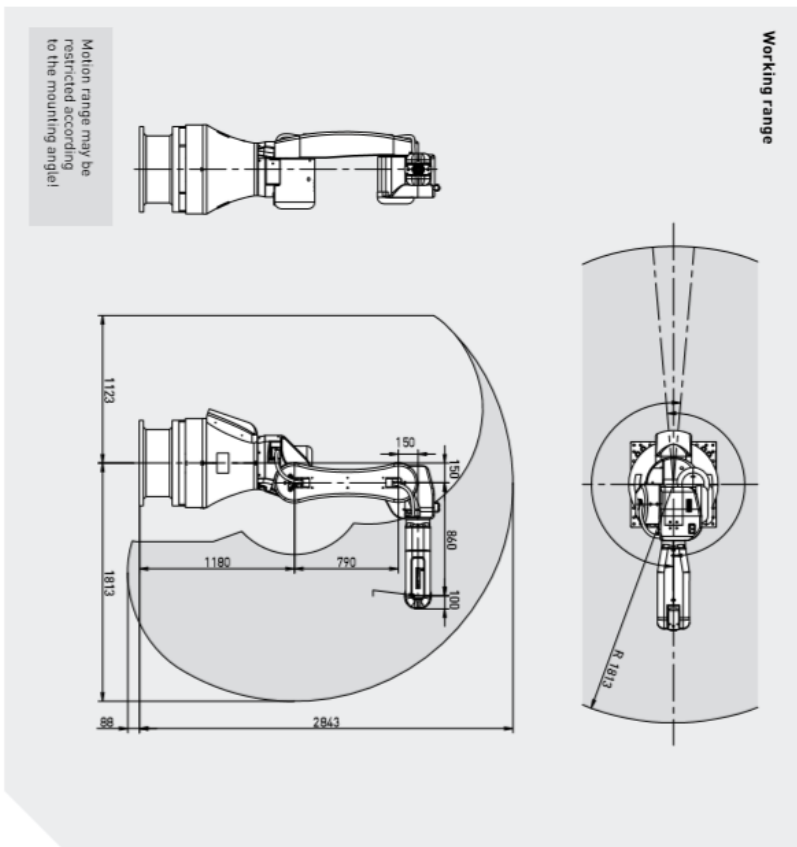
Max. load capacity
at wrist: **35 kg**



Max. reach:
1813 mm

| Controlled axes | Repeatability (mm) | Mechanical weight (kg) | Motion range (°) | | | | | | Maximum speed (mm/s) ¹⁾ | | | | | | | | | | |
|-----------------|--------------------|------------------------|------------------|-----|-----|-----|-----|-----|------------------------------------|----|----|----|----|----|---|---|---|-------|----------|
| | | | J1 | J2 | J3 | J4 | J5 | J6 | J1 | J2 | J3 | J4 | J5 | J6 | | | | | |
| 6 | ± 0.03** | 990 | 370 | 165 | 258 | 400 | 220 | 900 | 750*2 | | | | | | J4 Moment/ Inertia (Nm/Kgm ²) | J5 Moment/ Inertia (Nm/Kgm ²) | J6 Moment/ Inertia (Nm/Kgm ²) | | |
| | | | | | | | | | | | | | | | | | 110/4 | 110/4 | 60.0/1.5 |

Working range



MDS-00105-EN



Robot footprint [mm]

CR-35iA
650 x 650

Mounting position Floor

•

Mounting position Upside down

-

Mounting position Wall

-

Controller

R-30iB Plus

Open air cabinet

-

Mate cabinet

-

A-cabinet

•

B-cabinet

•

Pendant Touch

•

Electrical connections

Voltage 50/60Hz 3phase [V]

380-575

Voltage 50/60Hz 1phase [V]

-

Average power consumption [kW]

1

Integrated services

Integrated signals on upper arm In/Out

6/4

Integrated air supply

-

Environment

Acoustic noise level [dB]

< 70

Ambient temperature [° C]

0-45

Protection

Body standard/optional

IP54

Wrist & J3 arm standard/optional

IP67

¹⁾ In case of short distance motion, the speed may not reach the maximum value stated.
²⁾ It is necessary to set a motion speed according to risk assessment of system considering pinching with the surroundings.

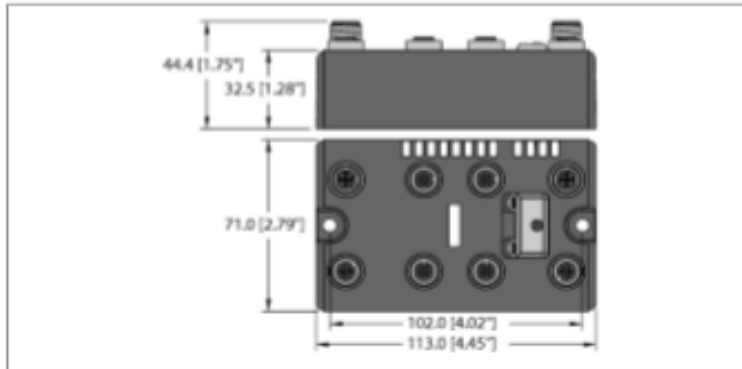
• standard ○ on request - not available | | with hardware and/or software option

**Based on IS09283

3. Turck Module



BL compact™ multiprotocol fieldbus station for Industrial Ethernet
4 Analog Inputs for Current or Voltage
BLCEN-4M12MT-4AI-VI



- On-machine Compact fieldbus I/O block
- EtherNet/IP™, Modbus® TCP, or PROFINET slave
- Integrated Ethernet Switch
- 10 Mbps / 100 Mbps supported
- Two 4-pole M12, D-coded, connectors for fieldbus connection
- 2 rotary switches for node address
- IP67, IP69K
- M12 I/O connectors
- LEDs indicating status and diagnostics
- Electronics galvanically separated from the field level via optocouplers
- 4 analog inputs for current or voltage
- 0/4...20 mA or -10/0...+10 VDC (selectable per channel)
- FLC/ARGEE programmable

| | |
|---------------------------------|---|
| Type designation | BLCEN-4M12MT-4AI-VI |
| Ident-No. | 6811468 |
| Nominal system voltage | 24 VDC |
| System power supply | Via auxiliary power |
| Voltage supply connection | 2 x M12, 5-pin |
| Admissible range VI | 11...30VDC |
| Nominal current VI | 137 mA |
| Max. current VI | 1 A |
| Fieldbus transmission rate | 10/100 Mbps |
| Adjustment transmission rate | Automatic detection |
| Fieldbus address range | 1...92 0 (192, 168, 1, 254) 93 (BootP) 94 (DHCP) 95 (PGM) 96 (PGM-DHCP) *Recommended for PROFINET 97...98 (manufacturer specific) |
| Fieldbus addressing | 2 decimally coded rotary switches |
| Fieldbus connection technology | 2 x M12 4-pole, D-coded |
| Protocol detection | automatic |
| Web server | Integrated |
| Service interface | Ethernet |
| Vendor ID | 48 |
| Product type | 12 |
| Product code | 11468 |
| Modbus TCP | |
| Addressing | Static IP, BOOTP, DHCP |
| Supported function codes | FC1, FC2, FC3, FC4, FC5, FC6, FC15, FC16, FC23 |
| Number of TCP connections | 6 |
| Input Data Size | max. 6 register |
| Input register start address | 0 (0x0000 hex) |
| EtherNet/IP™ | |
| Addressing | acc. to EtherNet/IP™ specification |
| Device Level Ring (DLR) | supported |
| Class 1 connections | 6 |
| Input Assembly Instance | 103 |
| Input Data Size | 7 INT |
| Output Assembly Instance | 104 |
| Output Data Size | 1 INT |
| Configuration Assembly Instance | 106 |
| Configuration Size | 0 |
| Comm Format | Data - INT |

Edition - Rev. C - 2019-02-08T10:11:29+01:00



BL compact™ multiprotocol fieldbus station for Industrial Ethernet 4 Analog Inputs for Current or Voltage BLCEN-4M12MT-4AI-VI



| | |
|-----------------------------------|--|
| PROFINET | |
| Addressing | DCP |
| Conformance class | B (RT) |
| MinCycleTime | 1 ms |
| Diagnostics | acc. to PROFINET alarm handling |
| Topology detection | supported |
| Automatic addressing | supported |
| Media Redundancy Protocol (MRP) | supported |
| Input Data Size | max. 8 BYTE |
| <hr/> | |
| Analog Inputs | |
| Operating modes | from 4AI-VI |
| Type of Input diagnostics | 0/4 ... 20 mA or -10/0 ... 10 VDC |
| Sensor supply | Channel diagnostics |
| Input resistance | 24 VDC, 1 amp max. |
| Maximum limiting frequency analog | Current: < 0.125 KΩ, Voltage: < 98.5 KΩ |
| Basic fault limit at 23 °C | < 20 Hz |
| Repeatability | < 0.3 % |
| Temperature coefficient | < 0.05 % |
| Resolution | < 300 ppm / °C of full scale |
| Measuring principle | 16 bit |
| Measurement display | Sigma Delta |
| | 16 bit signed integer |
| | 12 bit full range left-justified |
| <hr/> | |
| Dimensions | |
| Mounting | 113 x 71 x 32.5 mm |
| Weight | 2 x 5.4 mm diameter holes, 1.7 Nm torque |
| Housing material | 390 ± 20 g |
| Housing color | Glass-filled nylon, nickel plated brass connectors |
| Window material | Black |
| Material screw | Lexan |
| Material label | Nickel-plated brass |
| Ground label material | Polyester with polycarbonate overlay |
| Protection class | Nickel plated brass |
| | IP67 |
| | IP65K |
| Operating temperature | -40...+70 °C |
| Storage temperature | -40...+85 °C |
| Relative humidity | 15 to 95% (non-condensing) |
| Vibration test | according to IEC 61131-2 |
| Extended vibration resistance | |
| - up to 20 g (at 10 up to 150 Hz) | For mounting on base plate or machinery |
| Shock test | according to IEC 61131-2 |
| Electromagnetic compatibility | according to IEC 61131-2 |
| MTTF | 122 years |
| MTTF note | acc. to EN 25500 (Ed. 99) 20 °C |
| Approvals and certificates | CE, cULus, Class I Div.2 |

Bibliography

Universal Robots. (2009-2018) *Manual of UR10 version 3.8*.

Fanuc. (2019). *Audi uses FANUC collaborative robots to check welds*. Available at:

<https://www.fanuc.eu/bg/en/customer-cases/audi>

Siemens Product Lifecycle Management Software Inc. (2018). *Tracking objects on conveyors*. Available at:

https://docs.plm.automation.siemens.com/tdoc/tecnomatix/14.1.2/tecnomatix_eMS/#uid:RoboticsMenu_RobotConveyorTracking

Siemens Product Lifecycle Management Software Inc. (2018). *Coverage during Simulation*. Available at:

https://docs.plm.automation.siemens.com/tdoc/tecnomatix/14.1.2/tecnomatix_eMS/#uid:xid1179708

Siemens Product Lifecycle Management Software Inc. (2018). *Generating a continuous coverage pattern process*. Available at:

https://docs.plm.automation.siemens.com/tdoc/tecnomatix/14.1.2/tecnomatix_eMS#uid:index_xid1015765:xid1095247:xid1579323:xid1579170

Siemens Product Lifecycle Management Software Inc. (2018). *Create a paint trigger*.

Available at: https://docs.plm.automation.siemens.com/tdoc/tecnomatix/14.1.2/tecnomatix_eMS#uid:index_xid1015765:xid1095247:xid1183005:Operations_Menu_CreatePaintTrigger

Juckes, J. (2010). *XT B-Rep: Making it real*. Available at:

https://www.plm.automation.siemens.com/ja_jp/Images/XT-in-JT_tcm821-115289.pdf

Turck Inc. (2016). *BLCEN Ethernet/IP. Configuration guide*.

Profibus. (2017). *GSDML Specification for PROFINET*. Available at:

<https://www.profibus.com/download/gsdml-specification-for-profinet/>