U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Conception and simulation of a robotic cell based on the digital twin concept for industrial manufacturing

Tomás Alexandrino Oliveira Flores da Cunha

Faculdade de Engenharia da Universidade do Porto

Master's in Mechanical Engineering

Advisor:
Prof. Germano Veiga

Co-Advisor:
Eng. João Paulo Sousa

September 2021

The work presented in this dissertation was performed at the
Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial
Department of Mechanical Engineering
Faculty of Engineering
University of Porto
Porto, Portugal.

Tomás Alexandrino Oliveira Flores da Cunha
E-mail: up201605782@.up.pt

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Mecânica
Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial
Rua Dr. Roberto Frias 400
4200-465 Porto
Portugal

Resumo

Atualmente, existe uma elevada procura por eficiência na produção e por flexibilidade na linha de produção [1], consequentemente, o recurso a tecnologias de simulação tem vindo a aumentar rapidamente, uma vez que são uma solução rápida e de baixo custo para diversos problemas que a indústria de processos tem vindo a enfrentar. O uso de simulações robóticas permite o desenvolvimento de procedimentos robóticos sem a necessidade de utilizar o robô físico, prevendo o seu comportamento real num ambiente virtual. Esta tecnologia é extremamente útil em projetos industriais, uma vez que uma boa correlação entre os *digital twins* (ambiente de simulação e sistema real) permitirá a investigadores analisarem processos de um modo mais eficiente, reduzindo os tempos de desenvolvimento, e tornará possível o desenvolvimento completo de novas soluções sem a necessidade de parar o sistema real.

Esta dissertação pretende reportar o desenvolvimento de uma célula robótica para processamento de material metálico ao usar feixes laser. Para tal, serão analisados conceitos interdisciplinares relacionados com o desenvolvimento da célula robótica e o sistema de controlo e comando associado.

A primeira parte deste trabalho consiste no desenvolvimento de uma célula de produção virtual, idêntica ao sistema real, usando o software *RobotStudio*. A representação virtual é usada para programar as trajetórias robóticas necessárias e para verificar se o posicionamento de todos os elementos da estação é ideal. A segunda parte deste projeto consiste no desenvolvimento de uma *human machine interface (HMI)*, que foi projetada para providenciar ao utilizador um maior nível de controlo sobre o sistema, simplificando o processo de programação e integração. É importante também referir que vários testes foram realizados com este sistema, estando os resultados disponíveis nesta dissertação.

Para concluir, tendo em conta as soluções exploradas, esta tese reporta um avanço no desenvolvimento de células robóticas genéricas capazes de processamento material ao usar feixes laser. O derradeiro objetivo é implementar um sistema capaz de rapida e intuitivamente testar diferentes parâmetros do processo e criar peças personalizadas.

Keywords: Digital Twin, RobotStudio, HMI

Abstract

Nowadays, there is a growing demand for production efficiency and flexibility of the production line [1]. Therefor, the use of simulation technologies has been rapidly increasing, as it is a fast and low cost solution to several problems the manufacturing industry is facing. The use of a robotic simulations allows the development of robotic procedures without needing the physical robot, by predicting the real behaviour in a virtual environment. This technology is extremely useful in industrial setups, as a good correlation between the digital twins (simulation environment and real system) will enable researchers to explore processes in a more efficient manner, by greatly reducing development cycles, and allow the complete development of new solutions without the need to stop the real setup.

This dissertation aims to report the development of a robotic cell for laser material processing of metallic parts. To this end, the dissertation analyses cross-disciplinary concepts related to the development of the robotic cell and the associated command and control system.

The first part of this work consist in the development of a virtual production cell, identical to the real system, using *RobotStudio* software. The virtual representation is used to program the necessary robotic paths and to verify if the positioning of all the station elements is ideal. The second part of this project consist in the development of a human machine interface (HMI), which was designed to provide the user with a higher level of control over the entire process, simplifying the programming and integration process. It is also important to enlighten the fact that several testing procedures were concluded and the results are available in this dissertation.

To conclude, with the solutions explored, this thesis reports a step forward into the development of a fully functional generic laser material processing cell. The ultimate objective is to implement a processing system that is capable of quickly and intuitively test different process parameters and create custom parts.

Keywords: Digital Twin, HMI, RobotStudio

Acknowledgements

I would like to thank my supervisor professor Germano Veiga and my co-supervisor Engineer João Paulo Sousa for the chance to develop this dissertation, for all the holder demonstrated throughout this last months, and for the help solving the problems that arose.

I would also like to thank all INEGI personnel that helped me with the equipment necessary for this project, including Eng. João Martins, Eng. José Vieira, Eng. Filipe Alturas and Tiago Santos.

A special mention for the support of Smart2B project no. 38454 and CUT3D no. 69933, co-funded by the Programa Operacional Competitividade e Internacionalização (Compete2020) and Regional do Norte (NORTE2020), through the European Regional Development Fund (FEDER).

Abbreviations

CAD – *Computer Aided Design*;

CAM – *Computer Aided Manufacturing*;

DT – *Digital Twin*;

GUI - *Graphical User Interface*;

HMI - *Human Machine Interface*;

IFR - *International Federation of Robotics*;

TCP - *Tool Center Point*;

CSV - *Comma Separated Values*;

IIoT - *Industrial Internet of Things*;

PLC - *Programmable Logic Controller*;

CNC - *Computerized Numerical Control*;

OLP - *Off-line Programming (Robotics)*;

SDK - *Software Development Kit*;

FSoE - *Fail Safe over EtherCAT*;

WCS - *World Coordinate System*;

WPF - *Windows Presentation Foundation*;

WinForms - *Windows Forms*;

UI - *User Interface*;

XAML - *Extensible Application Markup Language*;

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The current state of Portugal's industry sector is being overwhelmed by several nations, that have continuously invested in the development of new technologies; therefore, a modernization of the production methods in Portugal is essential for it to compete for a place in the global market. To modernize, companies implement robotic stations to reduce the cost of labor in their factories as well as speed up production, allowing them to better compete and increase their market capitalization.

Analysing the *"World Robotic Report 2020"*, from *International Federation of Robotics (IFR)*, it is possible to visualize the number of robotic installations taking place each year and see their applications [2]. After analysing these results, it is possible to infer that the second most common application is welding and that several asian countries are more prolific with their installations, meaning that the western states are falling behind in their ability to increase production.

An obstacle in the current industries panorama is the ever-shifting society needs, which increases the necessity for production flexibility. This flexibility allows for more customization and individualization of products, helping companies to become more dynamic in modern markets. A flexible production system can quickly change its main structure to respond to any fluctuations in the market requirements, and it is critical for any high-end factory to be as flexible as possible, capable of a quick response to adversities and changes in the landscape of production [3].

Addionally, the fact that this system is using a state-of-the-art high-power laser and a tailored beam, improves the environmental sustainability of the industrial production, as it is using up to 30% lower consumption of resources compared to machines with similar functions. Laser-based technologies are considered agile and green manufacturing since the laser system can be deployed for different processes: cutting, welding, 3D printing, etc.) [4].

1.2 Objectives

The main objective of this dissertation was to develop a material laser processing system, including a virtual representation of the robotic cell with simulation capabilities and a control application, whose objectives are to simplify the interaction of the operator with the components of the robotic cell.

Although at the moment only one tool has been acquired, a tool changing mechanism has been added to the station to help improve the flexibility of this system, which required the creation of the necessary robot procedures to add and remove the laser tool.

The creation of a virtual representation of the robotic cell allowed testing of diverse robot movements and facilitated the ideal positioning of some components, including the tool changing station and the laser cutting table.

The interaction of the control application with the laser processing equipment can be divided based on the purpose of the desired procedure. On the one hand, if the procedure's purpose is to test different process parameters, then the system is capable of creating parallel welding and cutting lines with different sets of process parameters. On the other hand, if the procedure's purpose is to create a custom part, then the system is capable of importing a robotic path and the adequate processing parameters for the available workpiece.

Another goal of this dissertation was the creation of an operator manual that would provide an example on how the testing procedures and the custom procedures can be calibrated and started.

1.3 Project Methods

In the first part of the dissertation, it was analyzed the different aspects of the station where the laser cutting, laser welding and tool changing procedures were going to be implemented.

It was researched:

- The constructive aspects of the industrial robot used in this application, including performance parameters, dimensions, maximum payload and number of joints.
- The tool changer that was going to be coupled to the different tools, exploring the possible configurations.
- The laser source and the parameters required to achieve a reliable laser beam.
- The laser tool attached to the robot's arm, responsible for the focusing of the laser beam.
- The software required for this integration, which was TwinCAT 3, RobotStudio (RAPID) and VisualStudio (C#)

It was then analyzed the communication between all the components of robotic station. Several tests were conducted to fully understand the implementation of the required communication protocols.

Afterwards, the virtual representation of the robotic cell was built using *RobotStudio*, which suffered several iterations while determining the best position for the tool changer holder and the laser cutting table. The required functionalities of the robot and the HMI were also explored during this phase.

Subsequently, the tool changing procedures to pick and place the laser tool were added to the robot controller, and the respective buttons to start these procedures were also added to the the HMI.

Although it is possible to jog the robot's arm using the *Flexpendant*, several robotic procedures and buttons, in the control application, were created to jog the tool center point (TCP). The advantage of jogging the robot using the C# application instead of the *Flexpendant* is to further simplify the use of this system by not requiring the use of several devices.

Eventually, the methodology to test different process parameters was developed, consisting in the generation of several cutting or welding parallel lines, with each line having a different set of process parameters. The data being used for each line is stored in a CSV file, which implies that the file must firstly be deciphered in the C# application and the information is then transmitted to the robot controller.

The methodology to import custom cutting and welding procedures consists of using the virtual representation of the robotic cell to import the CAD files of the desired parts, program the necessary trajectories for the tool and add the necessary functions to control all the necessary components of the robotic station. The creation of the robotic trajectories is not ideal because it is only using *RobotStudio* basic functionalities, which necessarily requires a trained user to operate. Several software options were researched to generate the robotic paths but due to the high price and bad compatibility with this project they were not used.

After all the methodology was created for the required processes, the possibility of moving the robot to extremely useful positions by pressing a button in the C# application was added to the system. The advantage of this functionality is to avoid constantly jogging the robot to common positions, decreasing the time to start a procedure.

The final part consists of adding the final details into the C# application, including a list with all the messages coming from the robot controller and some events from the C# application. A major revamp of the HMI design was also done, changing letter fonts, colours and the position of several components, resulting in a simpler and more elegant interface.

1.4 Dissertation Structure

The dissertation begins in *Chapter 2* by exploring the capabilities of robotic simulation software and the impact of its implementation. It is also studied the potential of laser material processing systems, pointing out limitations of these technologies and some crucial concepts to maximize the productivity.

The *Chapter 3* begins by listing every component of the robotic cell and defining their properties and functions in this system. Every component was chosen for their particular set of characteristics, which will also be explored in this section. Given the fact that every element is exchanging information, it is crucial to explore the communication architecture of this system; hence, in the second part of this chapter the data exchange between every component will be closely analysed, defining the communication protocols and creating tables for all the shared variables. In this chapter, the objectives of the tool changing system will also be defined and the most adequate configuration for the available robotic station will be selected.

Chapter 4 is used to explore the developed C# application and all the required elements to establish a viable and intuitive laser processing system. First of all, the application objectives are listed and a brief description of the used software is given. The framework of the C# application is described and the structure of the application is explained, with the functions of each tab being reported as well. Additionally this chapter explores the structure of the *RAPID* program and gives several explanations of the methodology used to create the laser procedures.

Chapter 5 displays the results from the initial tests done with the laser processing system. Some problems were encountered during this simple procedures; however, solutions were found for each one. The cause of the problems and the respective solutions are also detailed in this chapter.

Chapter 2

State of the Art

In this chapter the concepts and processes used to develop this system will be analyzed, which will include the ”digital twin” concept and the laser cutting/welding processes. Some applications of this technologies will also be explored throughout this chapter.

2.1 Digital Twin

Industrial Internet of Things (IIoT) systems allow the connectivity of various devices and other assets into one system to achieve more intelligent actions from the data input. The application of the IIoT in industrial production systems is known as Industry 4.0 and a key enabler for this new industrial revolution is the Digital Twin, which is going to be discussed going forward.

The first definition of the term ”Digital Twin” (DT) was given by the National Aeronautics and Space Administration (NASA) in 2010. It was described as an *“integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor update, fleet history to mirror the life of its flying twin”* [5]. Taking this definition into consideration, the digital twin was an incredibly detailed simulation model of a spacecraft/airplane which tries to recreate their physical behavior as close as possible in the virtual world.

Evolution of the term digital twin

The digital twin was originally considered to be a set of high-fidelity mathematical models that would mimic the real behavior of the assets as closely as possible. This idea then evolved into incorporating the simulated dynamic 3D models of the real world assets [6].

In more recent years, through innovations of modern simulations tools in the automation field, it has become possible to build incredibly accurate simulation models of several physical devices and even entire production plants. Given the history of the DT, there are several interpretations in the sector of manufacturers of machines and systems; however, as the original understanding suggests, it is a complete and operational virtual representation of an asset, system, or sub-system, combining digital aspects of the building process (design models, product life cycle management data, manufacturing data) with real-time properties of how the equipment is being operated and maintained [7]. In this context, the DT is not only a huge collection of information, but it utilizes connections between the information and meta information to explicitly add semantics [8].

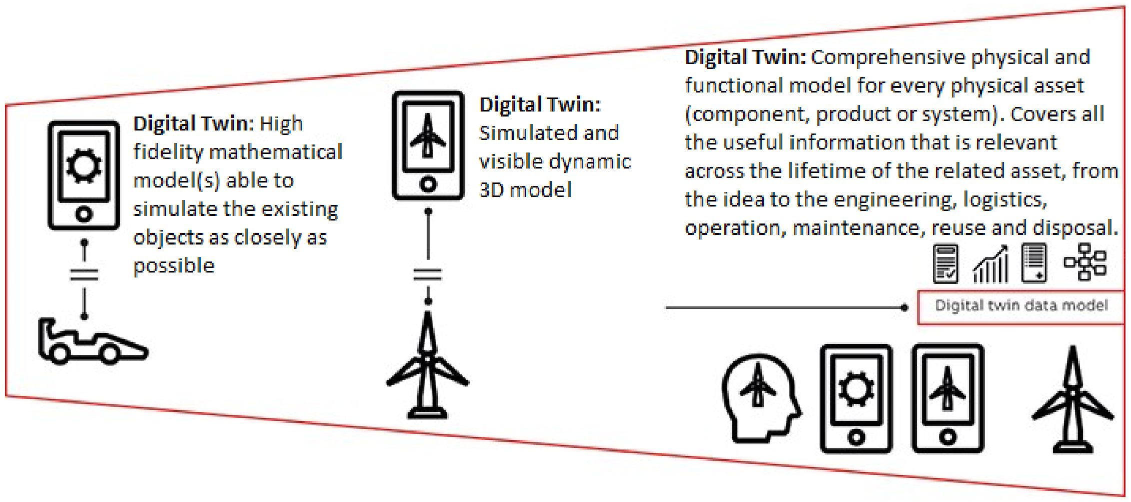

Digital Twin: High fidelity mathematical model(s) able to simulate the existing objects as closely as possible
Digital Twin: Simulated and visible dynamic 3D model
Digital Twin: Comprehensive physical and functional model for every physical asset (component, product or system). Covers all the useful information that is relevant across the lifetime of the related asset, from the idea to the engineering, logistics, operation, maintenance, reuse and disposal.
Digital twin data model

Figure 2.1: Evolution of the digital twin definition [6]

The ultimate value of a DT comes from the interactions with other digital twins and software tools. The DT on the manufacturer site contains various models for the design and manufacturing of a product type; whereas, the digital twin on the customer site contains several models to buy, install, maintain, operate, and dispose of different products. This data exchange between digital twins allows for both parties to better comprehend the systems being used.

Digital Twin Applications:

- **Design -** During the design phase, tools for simulation and visualization can be used to verify and inspect the 3D design and guarantee that all the parts assemble correctly. These simulations might include mechanical, thermal, and electrical properties as well as the correlation between these properties.

- **System integration -** The 3D visualization on a system level can help verify several constrains, such as spatial footprint and physical connections. By connecting digital twins from different components it is possible to simulate their interactions, including data transfer, control functionalities, mechanical and electrical behavior. It is also possible to test scenarios with unforeseen consequences without jeopardizing the real-world systems.

- **Prediction -** All operational and sensor data, past and present, is stored and combined with predictive algorithms to provide insights into the condition of the equipment and the probability of it failing.

- **Diagnostic -** Utilizing the many functionalities of the digital twins it is possible to troubleshoot several problems. For example, trough the usage of 3D visualization, technicians can have an overlay over the real equipment to visualize parameters, such as temperature of non-accessible parts or material stress, which contributes to the solution of complex problems.

- **Advanced Services -** In the cases where all the advanced service parameters (IoT connectivity, analytic algorithms, etc.) are configured in the digital twin, the parameters can be subscribed by the client and subsequently activated when the equipment is installed, reducing significantly further engineering work [7].

Advantages:

The main advantages that the digital twin solutions offer are the potential to commission a project right the first time and speed up the entire process. Most of the technical issues can be solved in advance due to the possibility of extensive testing prior to installing the physical devices. It is also possible to integrate all automation assets (PLC, Robot, etc.) by adjusting the parameters in different simulations to optimize for the real world and additionally eliminate the need for reprogramming, since the testing software and the actual software are identical. According to ABB the main advantages of using digital twins are available in table 2.1 .

Table 2.1: ABB's estimate benefits from introducing a digital twin to a project [7]

Reduced engineering time	20%
Reduced capital outlay	25%
Reduced training costs	50%

In this project, the digital twin concept was explored by using *RobotStudio* software during the design phase of the robotic cell to check the positioning of the different components, to test laser cutting/welding procedures as well as to verify the tool changing capabilities of the system.

Examples of digital twins in the manufacturing sector

Several projects exist where the usage of simulation technology is prevalent; however, this example will focus on the development of a robotic station capable of bending metal sheets [9]. A simulation environment was created using the software tools made available by the manufacturer, which was ABB. This environment, which can be seen in figure 2.2, used the *Bending PowerPac* and other functionalities made available in *RobotStudio* to simulate the robot movements and visualize the bending results.

The virtual station can be used as a blueprint for the real world system, helping with the placement of all the necessary components in their ideal position and using the programmed trajectories to speed up the integration process. The real system used to test the bending procedures and the result from a procedure are available in figure 2.3 .

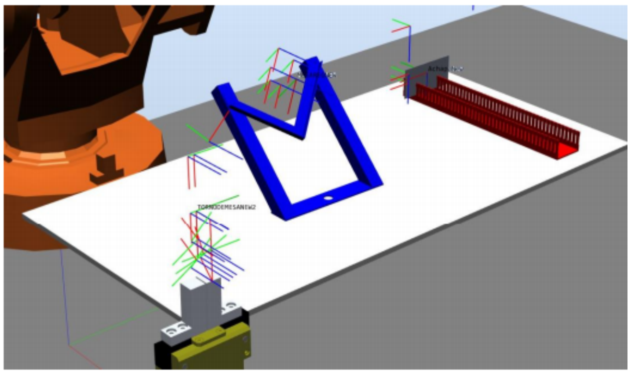


Figure 2.2: Virtual robotic cell for bending using *Bending PowerPac* [9]

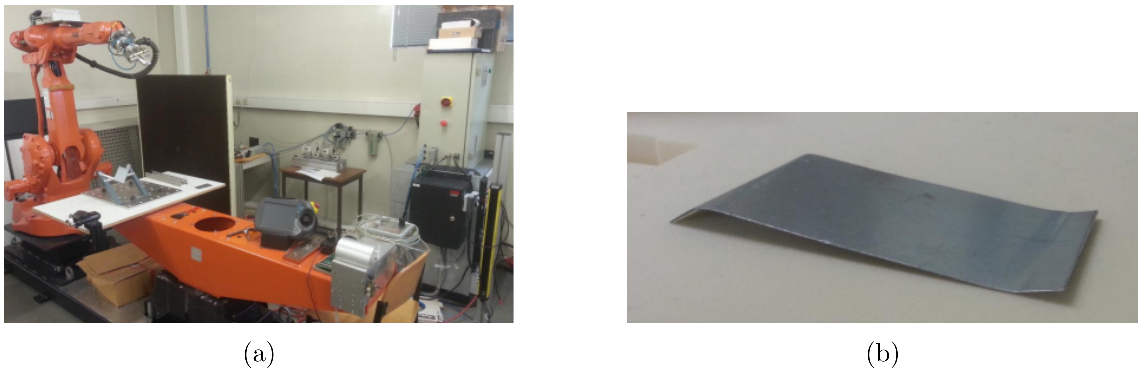


Figure 2.3: (a) Real robotic bending station; (b) Results from a testing procedure [9]

Following this practical example of a manufacturing robotic station that uses digital twins to aid in the creation of procedures, it is also important to refer more complex applications of digital twins developed by industry giants. Therefor, it will be explored some digital twin applications by Siemens and KUKA.

Siemens

Siemens is one of the biggest industrial conglomerates in the world and as expected they have been exploring the concept of digital twins for many years, having designed several products. The many applications of Siemens's digital twins include gas turbines, factories, new buildings and many others. Including the Siemens Building Technologies headquarters in Zug, Switzerland [10].

One of the major users of digital twin technology is Siemens's new operating company *Digital Industries (DI)*. For instance, technologies from DI have helped to create the new *"Solo"* electric car from Canadian startup *Electra Meccanica*, which is available in fig.2.4 . The car was designed, simulated and manufactured with the help of Siemens digital twin software, enabling *Electra Meccanica* to test and optimize all the vehicle's elements - mechanical, electronic, software and system performance - in advance of starting production [10].

Figure 2.4: *Solo* from Electra Meccanica : Developed with digital twins [11]

The final example of a Siemens's digital twin application is *SINUMERIK ONE*, which is the future-proof CNC for highly productive machine tools. This system grants faster innovations through the seamless interaction of virtual sphere and real world. The interaction is a result of using digital twins that help to simulate and test work processes entirely in a virtual environment, optimizing engineering processes based on consistent end-to-end workflows, saving time and money. The digital twin created of a machining process optimized the machine tool production capacity, thus minimized unproductive times and continually moving them to the work preparation phase, creating room for new approaches. An example of an application of the *SINUMERIK ONE* system is available in figure 2.5 .

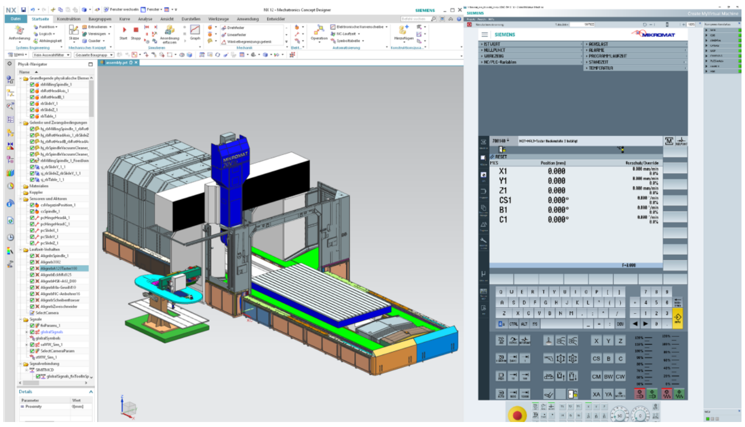


Figure 2.5: Example of a *SINUMERIK ONE* System application [12]

KUKA

The manufacturer of industrial robots KUKA has an identical approach to the digital twin concept as ABB, as they both believe that by using virtual commissioning it is possible to develop projects with increased complexity and achieve a more diverse product range in shorter periods of integration. The planned designs can be analysed precisely and tested in advance on the digital image, enabling the detection of errors in the simulation and their correction prior to the real installation. This gives the user increased flexibility in planning and avoids downtimes, resulting in significant cost savings [13]. An example of the virtual commissioning that can be done with *KUKA.Sim* software in available in figure 2.6 .

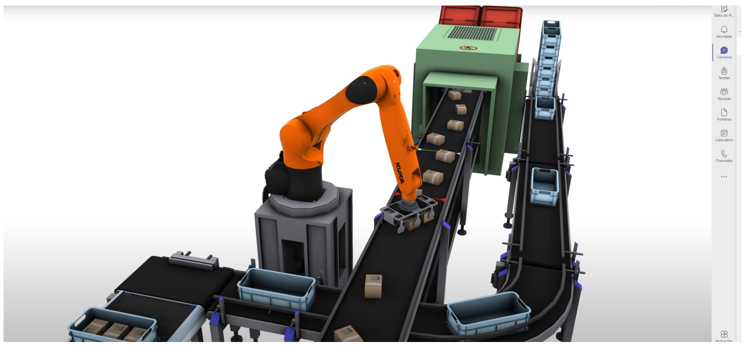


Figure 2.6: Pick and place simulation with *KUKA.Sim* Software [14]

2.2 Robot Programming

The method used to program the necessary robotic trajectories was off-line programming (OLP); therefor, the robot program is created independent from the actual robot cell and is then uploaded to the real industrial robot for execution. In off-line programming the robot cell is represented with a graphical 3D model in a simulation that helps robot integrators create the optimal program paths for the robot to perform a specific task [15]. Using OLP in this project is justified since the production of this cell will be low quantity with high diversity, with the creation of new procedures being simplified.

There are several off-line programming languages; however, *RobotStudio* was selected to be used in this project due to the following reasons:

- Software created by the manufacturer of the industrial robot used in the system.
- Free premium license was made available by the Universidade do Porto.
- Programs built on the ABB virtual controller, the same software used in the real ABB robot programs and configurations to be easily and seamless transferable from the virtual controller to the real one [16].
- Ability to create more than one program module/task and running them simultaneously, effectively creating a multi-thread application and asynchronous program [16].

A few examples of the possible alternatives would be KUKA.Sim, which is very similar to *RobotStudio*, and *RobotDK*, which has a wide range of robot manipulators brands and tool that can be used to create generic tasks.

An important factor to consider is the suite of SDKs and Web services to communicate with ABB Robots, which can vastly improve the quality of the robotic station. In this project it was used PC SDK, which enabled the creation of a windows application that is used as an interface for the robot, being capable of starting/stopping robotic procedures and much more.

2.3 Laser Processing

The robotic cell being developed is capable of laser cutting and laser welding, as such several components are required to create a laser system capable of metal processing. This section will explore the required components of the laser system and some concepts to take into consideration for an optimized production.

The main component of any laser system is the laser source and several different types exists; however, the one being used in this project is a fiber laser, which is ideal for metal processing. The development of high-brightness disk and fiber lasers in the past years has improved three-dimensional material laser processing, with the advantages being the fiber-coupled beam guidance, the good pulsability, the small size and the decreased investment costs. The brightness is relevant for combined cutting and welding due to the enlarged operating window allowed for changing head distance. A “slim” focal zone of a bright enough laser allows the cutting of narrow *kerfs* with a small nozzle standoff and grants excellent welding conditions with a larger standoff using the same nozzle, without changing the focal distance relative to the nozzle exit [17].

In today's current market situation, the possibility of having laser sources with the power and beam quality to cut and join metal in an expeditious and repetitious manner is greatly valued and as since become a basic requirement in modern laser devices, granting, in most applications, a more flexible process. In addition to these reasons, the market demands for a multi-functional processing device have led to the development of a combination head capable of laser cutting and welding 3D metal work pieces [18].

Laser Power Modulation

The speed of the tool center point (TCP) can vary dramatically in 3D applications, with a factor of 10 or 20 between speeds in straight paths and around small radiuses not being unusual. Reduced quality occurs in low-speed sections in the form of burrs when cutting, and in the form of irregular joints during welding.

An effective answer for some of the problems resulting from speed variation can be solved with a simple laser power control; however, to achieve a burr-free cut quality over the entire speed range, an adaptive laser power modulation is necessary. The creation of a programmable laser modulation control can adapt modulation frequency, duty-cycle and amplitude of the laser power to the effective speed. By controlling the pulse frequency and duty cycle, it is possible to adjust both the average power and spatial overlapping of pulses individually for each velocity, resulting in a processes with far better results [17].

Another crucial component in a laser system is the head that guides the focused laser beam to the desired target on the workpiece. The head used in this application is capable of laser cutting and laser welding and it is named *Combi-head*.

Standard cutting head

Standard cutting heads have an intense cutting gas flow, travelling coaxially to the focused laser beam, which provides for an effective melt ejection out of the *kerf*. A high pressure nozzle guides the laser beam in such a way that the beam focus and the gas jet meet in the workpiece below the nozzle exit. A lens in the nozzle's laser aperture grants a gas tight sealing of the nozzle chamber and focuses the laser beam. The nozzle tip serves as a capacitive clearance sensor, which is used in a closed-loop control of the machine axis ensuring a constant distance to the working surface [18].

An example of a standard cutting head is the *SolidCutter*, which enables machining of spacial geometries with high accuracy, being mainly used in 3D laser cutting systems for the automobile industry. The method of robot guided laser cutting is a flexible and highly productive method for the processing of high-strength body components and profiles. The compact and weight-optimized design of this laser cutting head results in high dynamics and great freedom of movement, as it can be seen in figure 2.7 . The integrated fine adjustment of the working distance ensures that any inaccurate guidance is reliably compensated, ensuring high cutting quality even at high feed rates [19].

Figure 2.7: Cutting head *SolidCutter* [19]

Standard welding heads

Standard welding heads with solid-state and diode lasers, the lens optics serves for focusing; however, with CO_2 and fiber lasers mirror optics are used instead. The creation of a smooth shielding and process gas flow necessary for the welding process requires the application of off-axis nozzles a few millimeters from the laser impingement point. Additionally, a compressed air crossjet is arranged between the optics and the process gas nozzle blowing transversely to the laser axis to protect the optics from the smoke and spatter emitted from the weld zone. In contrast to the gas tight optics in standard cutting heads, a welding head requires an open flow section dynamically decoupling the gas from the processing zone [18].

The welding head *YW52*, available in the figure 2.8, has a modular design, meaning that it is possible to integrate very simple or very complex welding optic's designs. Another important feature of this head is the reliability in an everyday industrial environment, being useful in "low-power" and "high-power" laser applications, and always achieving a high degree of economic efficiency. This laser welding head can be combined with a variety of process monitoring systems; hence, enabling a high degree of automation of the laser welding process. The monitoring systems can acquire and document a wealth of information relevant to quality and productivity while the tool is being used [20].

Figure 2.8: Welding heads *YW52* [20]

Combination heads

Taking into consideration a traditional approach to laser technology, it is possible to conclude that a standard cutting head cannot be used for welding, and a laser welding head cannot match the laser cutting speeds and quality demanded in most industrial applications.

Although it is possible to design a robotic laser setup that changes the heads for each process, either cutting or welding, the increase in integration complexity is cause for the use of a head that combines both processes. The increased complexity is a result of the two heads having to be mounted together and the need for a beam switch to be incorporated between the beam delivery system and the two heads [18].

In principle, there are no differences between the capabilities and parameters of standard cutting/welding heads and a combination head; however, some details are worth mentioning to avoid needless confusion. It is sometimes believed that adaptive optics or motorized nozzles to change nozzle distance and focal position independently are obligatory when switching between cutting and welding procedures; nevertheless, it has been determined that using the same focal distance from the nozzle tip for both cutting and welding processes is appropriate in many situations. The focal position and nozzle distance relative to the workpiece surface are changed simultaneously by simply lifting the entire head when switching from cutting to welding [17].

Identical Path Concept

Whenever cutting and welding operations use the same combination head, both processes are sharing an identical tool center point. If both processes can use an identical path with the same clamping, the precision when repeating the cut or weld contour is extremely good; hence, it is possible to achieve great and reliable results. For this reason, machines with moderate accuracy, such as industrial robots, can operate at higher speeds than expected [17].

2.4 Example of a robotic laser processing station

In this sub-chapter a robotic fiber laser 3D processing machine, manufactured by the company Golden Laser [21], will be analyzed. Every component of this fiber laser processing machine is available in figure 2.9 .

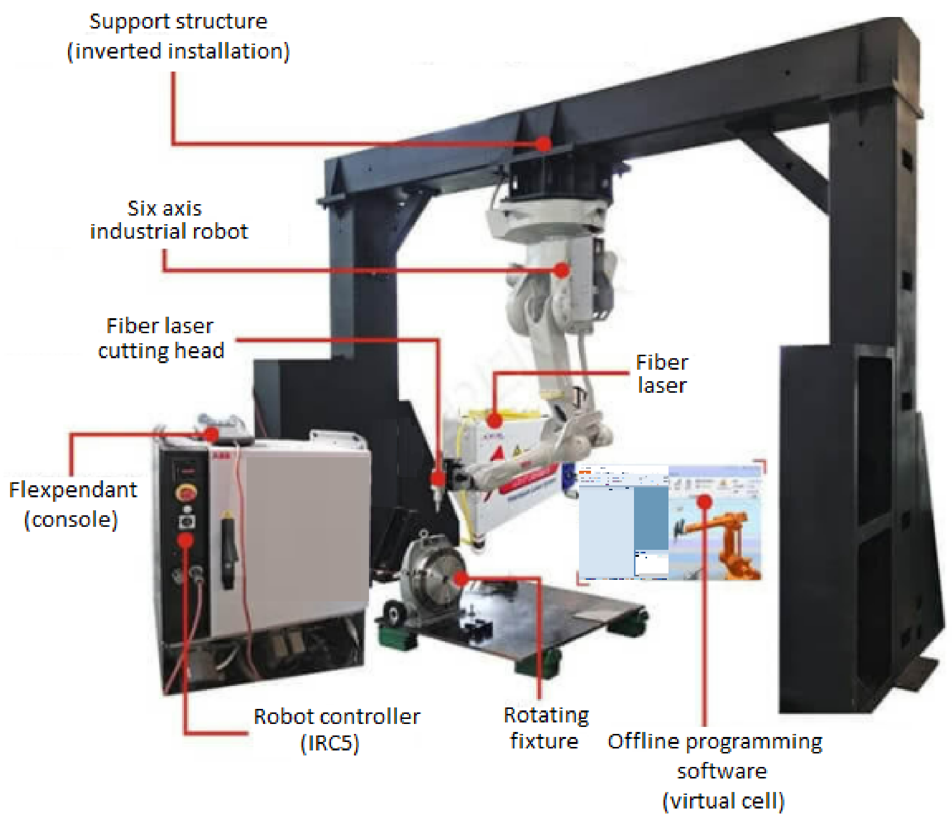


Figure 2.9: Robotic Fiber Laser 3D Cutting Machine manufactured by Golden Laser [21]

The robotic station is formed by an ABB robot, a fiber laser cutting head, a support structure for the robot, which is inverted, and fixture system for the necessary workpieces. This system is very similar to the robotic station developed in this dissertation; however, the robot is inverted, which grants the robot an increased movement flexibility. Although the increased flexibility grants the system the ability to produce a wider variety of products, it also makes the system installation more complex, due to the robot's support structure requiring a mechanical project and because of the inverted robotic installation. According to the manufacturer, the main features of this machine are the following [21]:

- Wide range of work with load-bearing capacity.
- Compact and wrist slim tool area that makes it possible to achieve high performance operations.
- Process speed and positioning that can be adjusted to achieve the best manufacturing accuracy, high yield, and high speed.
- Low noise without the need for short routine maintenance intervals, indicating a long service life.

- Usage of a simulation environment, developed using *RobotStudio*, to program the necessary robotic trajectories without jeopardizing the real system's safety.

As previously mentioned, the great tool mobility of this machine guarantees enumerable applications, which include all kinds of uneven metal pipe and sheet metal, making it suitable for the automotive industry, mold manufacturing, kitchen utensils, etc. It is ideal for automotive sheet metal coverings, chassis parts and other smaller production batches, such as the maintenance market, trucks, buses, construction machinery and others [21].

The use of low cost, high efficiency, alternative imports of five-axis laser cutting machines, significantly reduces the cost of the installation and makes it more competitive in global markets. The flexible production system is suitable for personalized parts, by shortening the production development cycle; hence, being capable of a rapid response to market [21].

The disadvantages of this system when compared to the system reported during this dissertation are the absence of a robotic positioner and a tool changing system. The lack of an industrial positioner and the limitation of having a single tool diminishes the range of products that this system can produce; nevertheless, the cost of the system does not get inflated by including this products.

The next chapter will discuss the equipment required for the robotic cell developed during this dissertation and will explain the architecture of the industrial communication system used. Additionally, a study of the available configurations for the tool changing station will be done to determine the best design for this cell.

Chapter 3

Robotic Station

In the third chapter every crucial component of the robotic cell and the communication system architecture used will be described. Additionally, a study of the tool changing station solutions will be done to determine the best configuration for this robotic cell.

3.1 Components

After analyzing the state of the art and the components available on the market, it is possible to start developing a robotic station that meets the system requirements. To achieve 3D combination processing with high quality results at economically attractive processing times, the following elements are essential [17]:

- 3D machine or robot with high dynamic response, such as a gantry robot with linear direct drives and integrated beam guidance
- Laser source with a fiber coupled beam delivery and good beam quality such as a fiber or disk laser
- Combination head, featuring optimized 3D capacities with a slender design, short length and fast distance control with a dynamic z-axis
- Laser power modulation control based on the speed of the tool center point

Subsequently, all the essential components of the robotic cell will be displayed and described. The reason for selecting each component will also be included in their description.

3.1.1 Industrial Robot

The industrial robot chosen for this project was the IRB 4600, its weight is 435 kg, its reach is 2,55 m and the nominal payload is 40 Kg [22]. This type of robots move by rotating their joints, which they have a minimum of two and a maximum of ten [23]; however, the IRB 4600 has six joints, as it can be seen in figure 3.1.

Figure 3.1: ABB IRB 4600 40kg 2,55m [22]

An important parameter to consider is the pose accuracy, which details the robot's ability to reach the desired pose. This parameters expresses the deviation between the command pose and the mean of attained poses when approaching the command pose from the same direction. This parameter's value is divided into two parts [24]:

- the distance between the command pose and the barycentre of the cluster of attained points (positioning accuracy)
- the difference between the command angular orientation and the average of attained angular orientations

$$\Delta L = \sqrt{(\bar{x} - x_c)^2 + (\bar{y} - y_c)^2 + (\bar{z} - z_c)^2}$$

Where:

$$\bar{x} = \frac{1}{n} * \sum_{j=1}^{n} x_j$$

$$\bar{y} = \frac{1}{n} * \sum_{j=1}^{n} y_j$$

$$\bar{z} = \frac{1}{n} * \sum_{j=1}^{n} z_j$$

Where:

- x,y,z are the coordinates of the barycentre of the cluster of points obtained after repeating the same pose **n** times
- x_c, y_c, z_c are the coordinates of the command pose
- x_j, y_j, z_j are the coordinates of the obtained j- pose

A high value of this parameter implies that the robot is less likely to reach the desire pose, resulting in bad pose accuracy.

Another crucial parameter to analyze is the pose repeatability, which measures the ability of the robot to consecutively reach the same position, and expresses the closeness between the positions and orientations of the attained poses after **n** repeated visits to the same command pose in the same direction. For a particular pose, the repeatability is expressed by [24]:

- the value of r, which is the radius of the sphere whose center is the barycenter of the cluster of attained points
- the spread of the angles $\pm 3S_a, \pm 3S_b, \pm 3S_c$, about the mean values a, b, c where S_a, S_b, S_c are the standard deviations:

$$r = \bar{x} + 3S_d$$

$$\bar{D} = \frac{1}{n} * \sum_{j=1}^{n} D_j$$

$$S_d = \sqrt{\frac{\sum_{j=1}^{n}(D_j - \bar{D})^2}{n-1}}$$

$$r_a = \pm 3S_a = 3 * \sqrt{\frac{\sum_{j=1}^{n}(a_j - \bar{a})^2}{n-1}}$$

$$r_b = \pm 3S_b = 3 * \sqrt{\frac{\sum_{j=1}^{n}(b_j - \bar{b})^2}{n-1}}$$

$$r_c = \pm 3S_c = 3 * \sqrt{\frac{\sum_{j=1}^{n}(c_j - \bar{c})^2}{n-1}}$$

A high value of this parameter implies that the robot is less likely to reach repeatedly the same desired pose, resulting in bad pose repeatability.

A good pose repeatability and a bad pose accuracy implies that the robot will always reach the same pose; however, the pose is not the programmed one. Contrarily, a bad pose repeatability and a good pose accuracy implies that the robot will always be near the programmed pose but always in different poses.

The importance of these parameters when creating the tool changing procedures is extreme. If the robot tries to pick the tool with a slightly different orientation, the connecting pins necessary to attach the tool to the robot will be slightly off, which can result in damages to the laser tool or the robot.

Other parameters can also be considered:

- Linear path repeatability - Evaluates robot's ability to go through the same linear path.
- Linear path accuracy - Evaluates robot's ability to maneuver though the desired linear path.
- Pose stabilization time - Measures the time between the robot reaching the desired pose and the moment the robot stops in that pose.

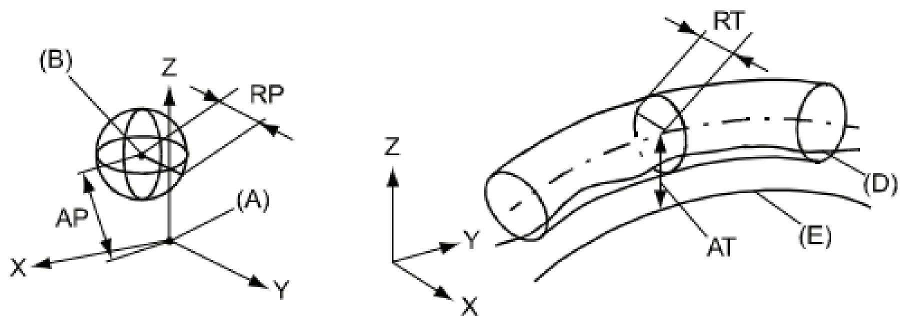
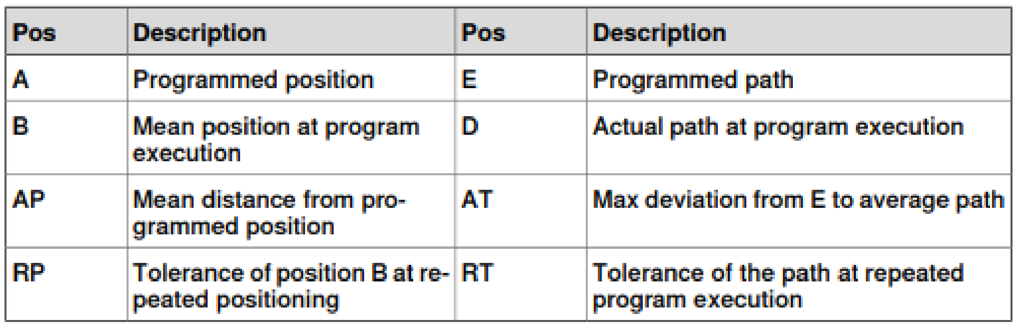
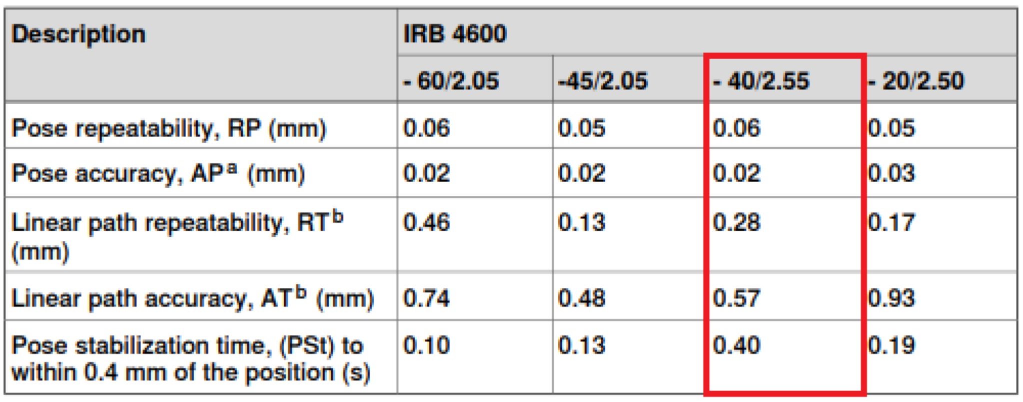


Figure 3.2: Illustration of the performance parameters [22]



Pos	Description	Pos	Description
A	Programmed position	E	Programmed path
B	Mean position at program execution	D	Actual path at program execution
AP	Mean distance from pro-grammed position	AT	Max deviation from E to average path
RP	Tolerance of position B at re-peated positioning	RT	Tolerance of the path at repeated program execution

Table 3.1: Performance parameters [22]



Description	IRB 4600			
	- 60/2.05	-45/2.05	- 40/2.55	- 20/2.50
Pose repeatability, RP (mm)	0.06	0.05	0.06	0.05
Pose accuracy, AP[a] (mm)	0.02	0.02	0.02	0.03
Linear path repeatability, RT[b] (mm)	0.46	0.13	0.28	0.17
Linear path accuracy, AT[b] (mm)	0.74	0.48	0.57	0.93
Pose stabilization time, (PSt) to within 0.4 mm of the position (s)	0.10	0.13	0.40	0.19

Table 3.2: Values of the performance parameters [22]

Analysing table 3.1, table 3.2 and figure 3.2 it is possible to determine that the pose accuracy for the chosen industrial robot is 0,02 mm and the pose repeatability is 0,06 mm; hence, when positioning the holder for the laser tool, the programmed paths must assure a minimum distance of 0.06 from the tool to all possible obstacles. If this properties are taken into consideration it is possible to guarantee that the robot will always succeed in reaching its destination without crashing.

The main reasons for choosing this robot were the ability to program it using Robot-Studio software, which is regularly used by the industry and education sector, and the fact that the manufacturer is a large corporation known for facilitating the creation of very flexible and personalized systems by providing specialized tools, such as PCSDK.

ABB makes available several optional add-ins to improve the robot's performance. The more relevant ones acquired for this project were the *Multitasking*, which grants the capacity of having more than a single task running simultaneously, and *Absolute Accuracy*, which improves the robot's accuracy dramatically. Several tools designed to assist in the generation of the robot's paths were also explored, including the ones available in *Robot-Studio*, named *PowerPacs*. The *PowerPacs* studied in this project were the *Cutting* and *Welding*, which facilitate the programming of cutting and welding procedures, respectively.

The manufacturer points out that the compact and optimized design of the IRB 4600 resulted in a low weight articulated robot that can cut cycle times of the industry benchmark by up to 25%. The ability to position the robot's base in several configurations such as floor mounted, inverted or even tilted is incredible powerful when combined with the simulation software, as it becomes possible to determine the most favorable position with regard to reach, cycle time and auxiliary equipment [25]. The main applications for this robot are available in the table 3.3 .

Table 3.3: Main applications of an IRB 4600 industrial robot [25]

Machine tending	Assembly
Material handling	Palletizing
Arc welding	Measuring
Cutting	Deburring
Dispensing	Polishing

3.1.2 Industrial Positioner

The workpiece positioner used in this application was the IRBP A, its weight is 850 kg, its maximum handling capacity is 500 kg and the maximum continuous torque is 650 Nm. It was designed for workpieces that have to be rotated around two axes to reach the optimal processing position. The axes movements are coordinated with the robot when programming and during operation [26]. The model used in the robotic station can be seen in the figure 3.3.

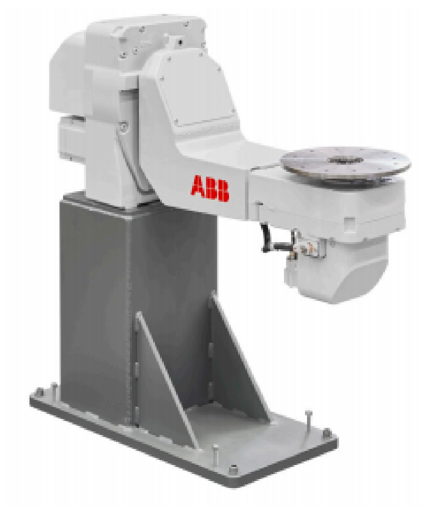


Figure 3.3: ABB IRBP A [27]

A scheme of the variant used in this project can be seen in the figure 3.4

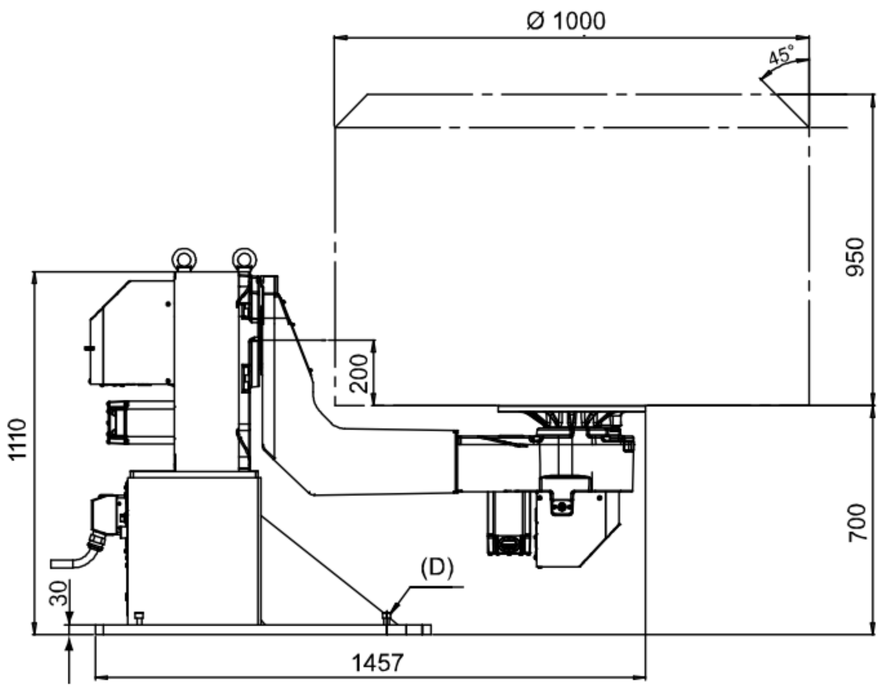
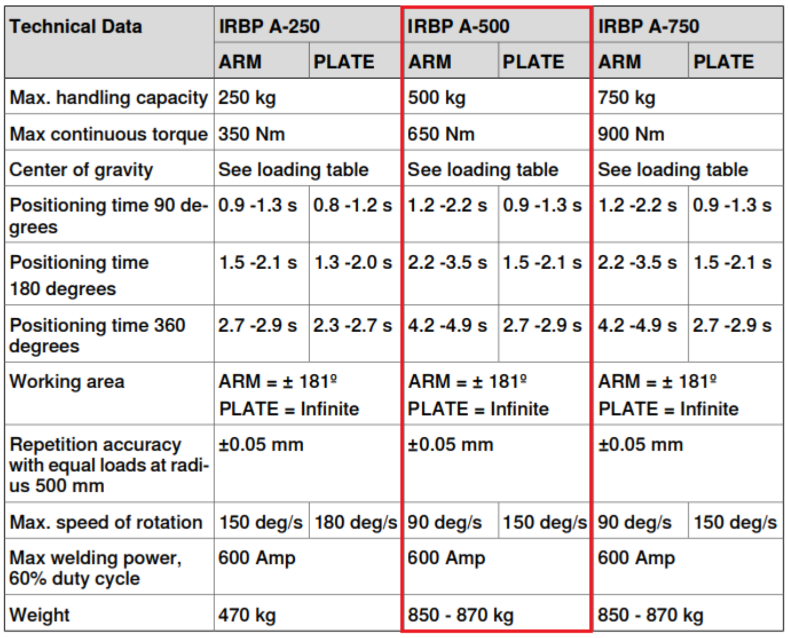


Figure 3.4: Scheme of the IRBP A A500 D1000 H700 [26]

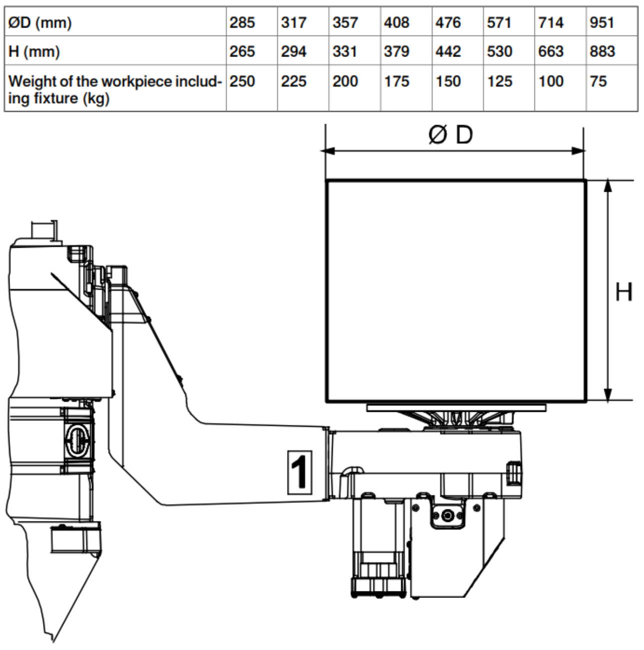
The combination of this equipment's modular design, consisting of few and heavy-duty moving parts, and the minimal maintenance demands turns this equipment in a service friendly device. In addition, the dynamically adaptive software plus high-speed drives results in fast changeover and high productivity. The technical data of the selected positioner can be seen in table 3.4



Technical Data	IRBP A-250		IRBP A-500		IRBP A-750	
	ARM	PLATE	ARM	PLATE	ARM	PLATE
Max. handling capacity	250 kg		500 kg		750 kg	
Max continuous torque	350 Nm		650 Nm		900 Nm	
Center of gravity	See loading table		See loading table		See loading table	
Positioning time 90 degrees	0.9 -1.3 s	0.8 -1.2 s	1.2 -2.2 s	0.9 -1.3 s	1.2 -2.2 s	0.9 -1.3 s
Positioning time 180 degrees	1.5 -2.1 s	1.3 -2.0 s	2.2 -3.5 s	1.5 -2.1 s	2.2 -3.5 s	1.5 -2.1 s
Positioning time 360 degrees	2.7 -2.9 s	2.3 -2.7 s	4.2 -4.9 s	2.7 -2.9 s	4.2 -4.9 s	2.7 -2.9 s
Working area	ARM = ± 181º PLATE = Infinite		ARM = ± 181º PLATE = Infinite		ARM = ± 181º PLATE = Infinite	
Repetition accuracy with equal loads at radius 500 mm	±0.05 mm		±0.05 mm		±0.05 mm	
Max. speed of rotation	150 deg/s	180 deg/s	90 deg/s	150 deg/s	90 deg/s	150 deg/s
Max welding power, 60% duty cycle	600 Amp		600 Amp		600 Amp	
Weight	470 kg		850 - 870 kg		850 - 870 kg	

Table 3.4: Values of the performance parameters of the IRBP A-500 [26]

An important factor to consider when using the IRBP A is the maximum permitted center of gravity displacement from the center of rotation and the rotary unit's face-plate at different loads. For the IRBP A-500, if the load is 450 kg the center of gravity must be within the area limited by the measurement ØD and the measurement H (294 mm and 748 mm, respectively), as it can be seen in figure 3.5.



ØD (mm)	285	317	357	408	476	571	714	951
H (mm)	265	294	331	379	442	530	663	883
Weight of the workpiece including fixture (kg)	250	225	200	175	150	125	100	75

Figure 3.5: Values for the maximum permitted center of gravity displacement [26]

Furthermore, ABB believes that the most captivating features of this positioner are the intuitive programming, the advanced dynamic modelling and the Load ID-function, which is used to calculate the center of gravity and the inertia of the workpiece and the fixture. The positioner's programming is easy to understand because of the simple programming instructions, the control equipment being located in the robot controller and the drive systems and software being identical to the ones used with the industrial robots. The dynamic modelling used is capable of rapid acceleration, fast movements and re-orientation so that cycle times are kept to a minimum. Additionally, the positioner automatically compensates for the effects of gravity, inertia and friction to provide fast and accurate movements [27].

3.1.3 IRC5 Controller

The robot controller used in this application is the IRC5, which has been used for more than four decades and is the robotic industry's benchmark in robot controller technology. According to ABB, what makes this controller one of the best in the market is its unique motion control and an array of crucial properties such as flexibility, safety, modularity and others [28].

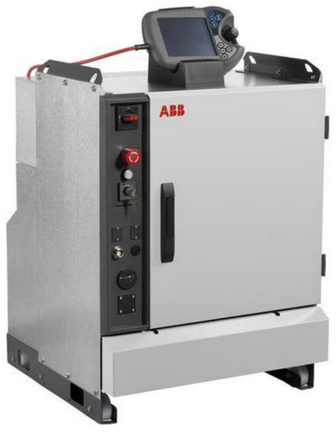


Figure 3.6: ABB IRC5 Controller [28]

- **Speed and Accuracy:** The IRC5 gives the robots the ability to perform tasks in a highly efficient manner by using advanced dynamic modelling. It automatically optimizes the performance of the robot by reducing cycle times and providing precise path accuracy, without requiring tuning by the programmer. The robot's motion is predictable and its performance high.
- **Safety:** The fulfillment of all relevant regulations and the certification of third-party inspectors worldwide improves the operator safety and is a leading benefit of the IRC5 controller.
- **Compatibility:** ABB's IRC5 controller is compatible with various types of main voltages and can handle a broad spectrum of environmental conditions. Additionally, the controller is also capable of communicating with other machines in a manufacturing environment, supporting most of the modern industrial networks for I/O.

- **Programmability:** All ABB robot systems are programmed with the flexible and high-level programming language named *RAPID*. Despite this language basic features and functionalities being easy to use, it allows for the creation of highly sophisticated solutions, supporting structured programs and advanced features.
- **Reliability:** The IRC5 is practically maintenance free and its quality ensures unmatched up-time. The built in diagnostic features help ensure fast recovery and production restarts when operations are interrupted on the factory floor.

The IRC5 Controller has two variants, the one used in this project is the Single Cabinet Controller, which is designed for high IP protection and full expandability, providing a protected environment for auxiliary equipment in the robot system. This controller is also capable of controlling up to four robots in a MultiMove setup, only requiring a compact drive module for each additional robot [28].

3.1.4 Laser Source

The laser source used in this project is the Rofin FL030 fiber laser. The fiber lasers of the FL series have great efficiency and service friendly, modular pumping units, which help reduce maintenance and operating costs. The selected laser source, available in figure 3.7, is high performance and suitable for laser cutting and laser welding applications [29].

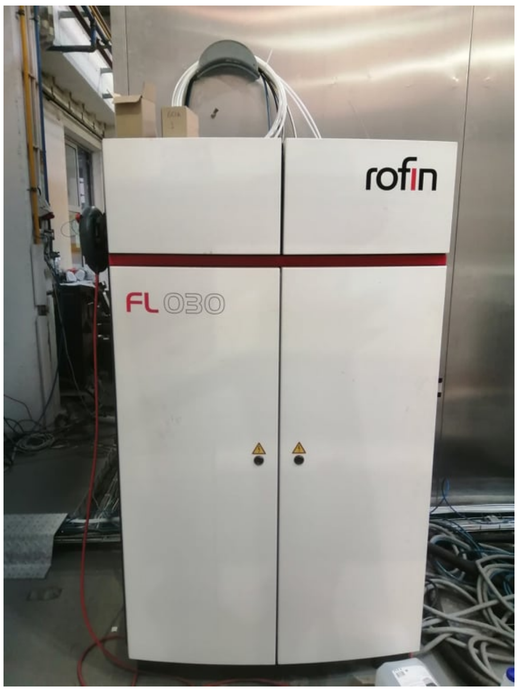


Figure 3.7: Rofin FL030 Fiber Laser [30]

According to the manufacturer, the main features of the selected laser source are [31] :

- High beam quality
- Modular and robust design resulting in highest industrial standards
- Operation of up to 4 optical fibers

- Maximum power output of 3000 W
- Fiber diameter of 1064 μm
- Handheld Controller Terminal with touch screen
- PC controlled and network-compatible
- High performance laser power supplies
- High electrical efficiency

It is also important to enlighten that for the laser source to function, several parameters have to be previously selected. After selecting the laser program and the laser type, which can be pulses or ramps, it is necessary to select the laser power and either the frequency or the duty cycle of the laser beam. Only two variables can be simultaneously manipulated during the procedures, and the laser power is considered essential; therefor, either the frequency or the duty cycle can be selected [31].

3.1.5 Laser Tool

The head used in this application is named "*Combi-head*", the model is *F2Y*, and it was developed by *LaserFact*. This head can execute 3D laser cutting and welding in an arbitrary sequence without retooling, resulting in [18]:

- reduced production time
- short and integrated process chains
- high machine utilization
- flexible and cost-efficient production of variants
- improved manufacturing accuracy, eliminating intermediate steps such as part handling, positioning and clamping

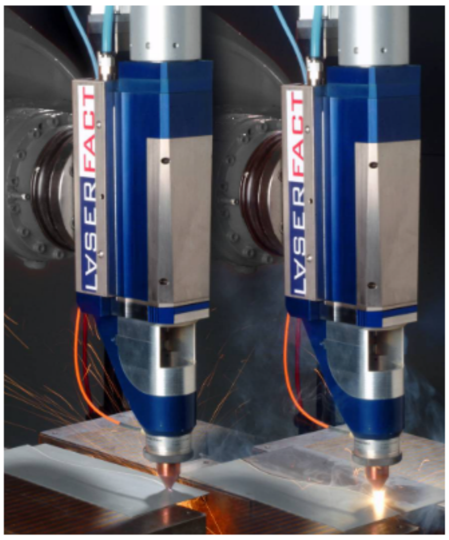


Figure 3.8: Combi-head F2Y laser cutting (left) and laser welding (right) of metal sheets [32]

The change of the processes is done automatically by changing the gas type, flow rate, focal and nozzle position, laser power and speed [32]. In addition, the *Combi-head* allows for the fabrication of complex metal components with a range of variants, which could not be produced cost-efficiently before. The product groups where the combination of both processes could be seen as an advantage are the following [18]:

- Complex component assemblies requiring multiple quick changes between welding and cutting operations
- Products with a wide range of variants
- Components that require exact positioning and orientation of the TCP between cut contours and weld beads.

Although it is possible to design a laser machine with the possibility of replacing heads for the cutting and welding operations, the increase in integration complexity, due to the two heads having to be mounted together and the necessity for a beam switch to be incorporated between the beam delivery system and the heads is cause for the use of a head that combines both processes [18]

Laser Cutting -Laser cutting will be done with a nozzle distance to the workpiece surface of approximately 0.5 to 1 mm [32].

Laser Welding -Laser welding will be done with a nozzle distance to the workpiece surface of approximately 5 to 6 mm. When setting the nozzle distance, either the z-axis of the laser system or an additionally axis of the Combi-Head can be used [32].

Combi-head Components

The Combi-head is formed by several components that perform the necessary tasks to achieve high quality results when using the laser system.

Autonomous Nozzle

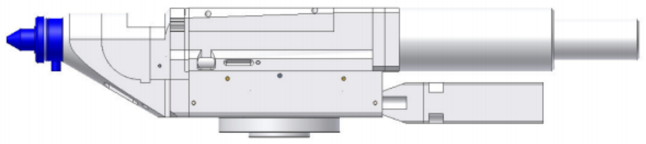


Figure 3.9: Autonomous Nozzle [32]

The design of the autonomous nozzle is based on a patented development of the Fraunhofer Institute for Laser technology. This nozzle can handle gas pressures up to 30 bars, gas flows between 20-30 l/min and is easily replaced. There is also no mechanical stress of optical components by high gas pressure and the nozzle position is defined by a capacitive distance control [32].

Crossjet Module

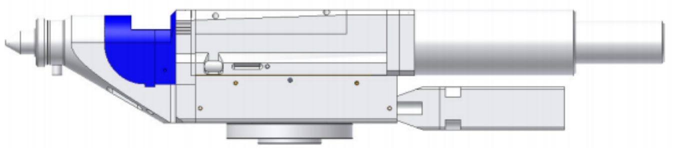


Figure 3.10: Crossjet Module [32]

The crossjet used in this tool has a special design using integrated suction device for stationary operation and multi-jet technology. This module is used to protect the optical elements against soiling by welding fume and impacts [32].

Protection Glass Module

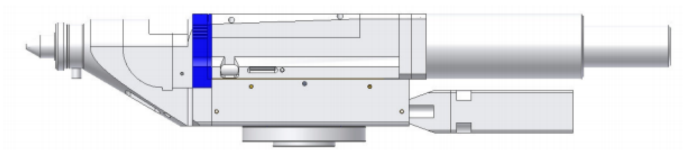


Figure 3.11: Protection Glass Module [32]

The protection glass is positioned in front of the focusing unit, which consists of a cheap glass disk that protects the expensive focusing unit. This module is also water cooled and the drawer design enables easy and quick replacement of the protection glass [32].

Quick Snap Focusing Optics Module

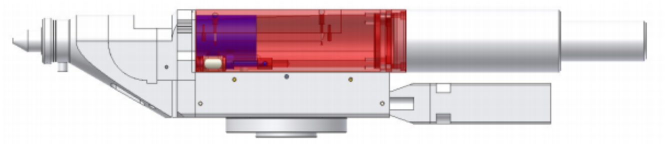


Figure 3.12: Quick Snap Focusing Optics Module [32]

The basic element design enables application of different focusing lengths, from 200mm to 300mm, without changing the TCP settings. The design of the focusing module allows a quick exchange and is developed for high laser power, with the standard version going up to 5 kW. It is also important to refer that the adjustment range for the focal position adaptation is 25 mm [32].

Lifting Drive Module

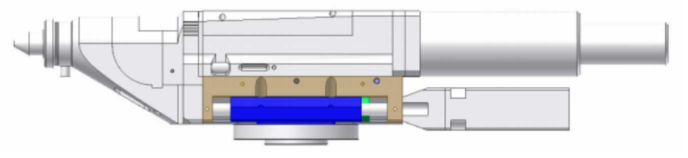


Figure 3.13: Lifting Drive Module [32]

This module has a high dynamic lifting drive and an internal supply of gas, water and status signals. It has an internal XY-adjustment for beam alignment and a ± 15mm stroke. The most crucial components of this module are the ball beared rails and drive.

Motor Drive

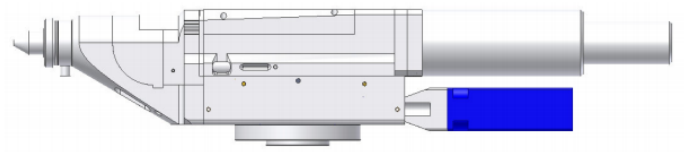


Figure 3.14: Motor Drive [32]

The motor used in this application, which is controlled by a digital motor controller, is a synchronic motor capable of 0.5g accelerations with brake.

Mouting Flange

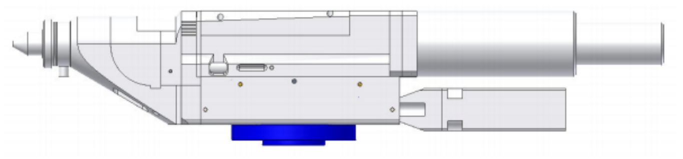


Figure 3.15: Mouting Flange [32]

The mounting flange is available for current assembling designs; however, it is possible to have a customized design.

Collision Protection Module

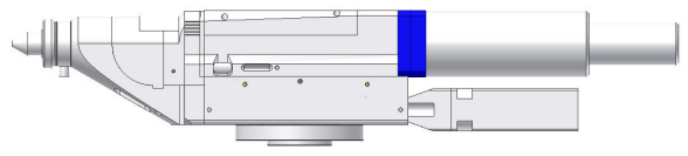


Figure 3.16: Collision Protection Module [32]

The collision protection module is available for current assembling designs; however, it is possible to have a customized design.

Collimation Module

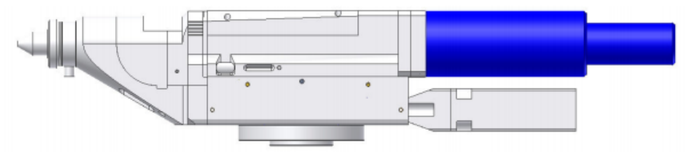


Figure 3.17: Collimation Module [32]

The collimation module adaption and assembly is possible for the current standard design, and the application specific adaption, with respect to laser and fiber typs, is also available.

A crucial component of the laser tool used in this application is the Adjust Box EG8110 Lasermatic Z, which can be seen in figure 3.18 . During laser beam cutting, deviations in the distance between nozzle and material surface, which can be caused by work-piece or position tolerances, may negatively affect the cutting result. Optimum cutting quality and cutting speed are only achieved when cutting nozzle and focus are positioned correctly in relation to the workpiece surface. Deviations of just a few tenths of a millimeter can lead to burr formation and negatively affect the cutting speed, the roughness of the cutting surfaces and kerf width [33].

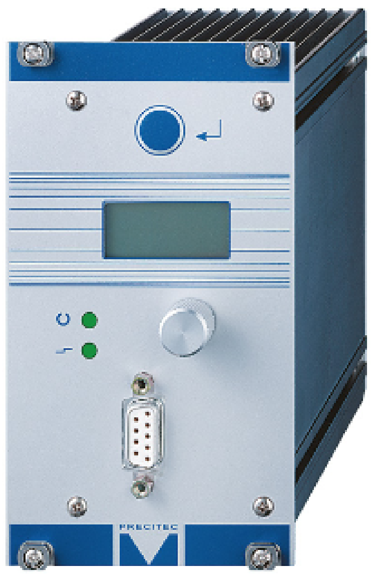
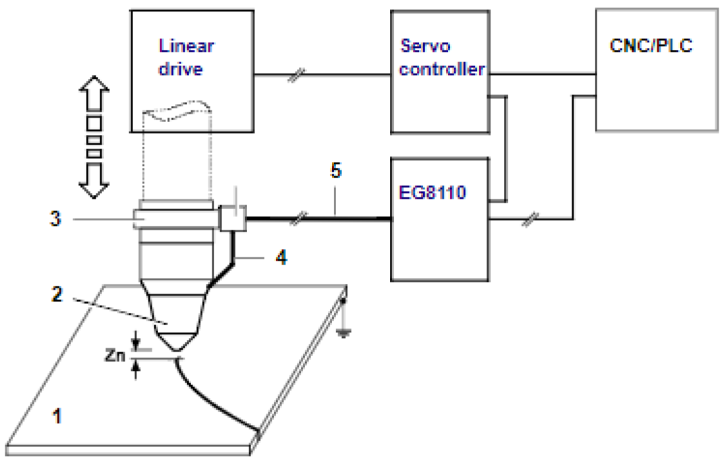


Figure 3.18: Adjust Box EG8110 Lasermatic Z [33]

The distance to the workpiece surface is detected using capacitive distance sensors in the combination head. The sensor signal is transmitted to the adjust box, which analyses the signal and outputs a signal that can be used to control a linear drive, as it can be seen in figure 3.19 [33].



1 **Workpiece**
2 **Sensor Lasermatic Z**
3 **Cutting head**
4 **Electrode cable**
5 **Sensor cable**

Figure 3.19: Integration of the distance controller for capacitive sensors [33]

3.1.6 PLC Beckhoff CX9020

The CX9020 is a compact, DIN rail-mountable Ethernet control system with 1GHz ARM Cortex – A8 CPU and the connections for the I/O systems are directly integrated into the CPU module. The unit offers automatic bus system identification (E-bus or K-bus) and independently switches in the corresponding mode. This controller consists of a CPU with two microSD card slots, the internal RAM and 128 kB NOVRAM as non-volatile memory. The basic configuration also includes two switched Ethernet RJ45 interfaces, four USB 2.0 interfaces and DVI-D interface [34].

It is possible to create a line topology without the need of additional Ethernet switches by using the preexisting RJ45 interfaces, which are connected to an internal switch.

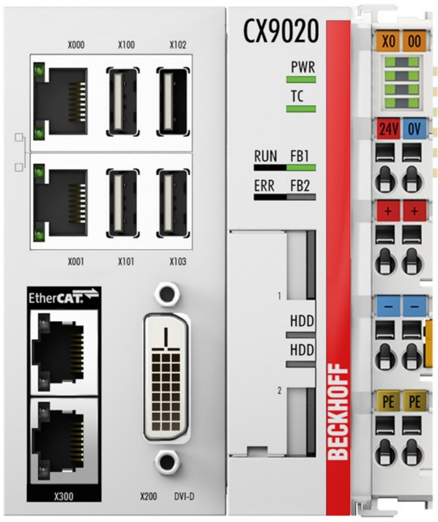


Figure 3.20: PLC Beckhoff CX9020 [34]

The TwinCAT automation software [35] transforms a CX9020 into a powerful PLC and motion control system that can be operated with or without visualization. The unit was ordered with fieldbus interface and the operating system is Microsoft Windows Embedded Compact 7 [36].

3.1.7 Gas Panel

In figure 3.21 it is possible to visualize all the components of the gas panel required to achieve a stable gas supply for the necessary procedures. However, it is important to enlighten the fact that the most relevant components of this installation are the proportional pressure control valve and proportional flow control valve identified in the figure 3.21 by the numbers 2 and 7, respectively. This control valves are responsible for precisely supplying the desired cutting and welding gases to the *Combi-head* nozzle.

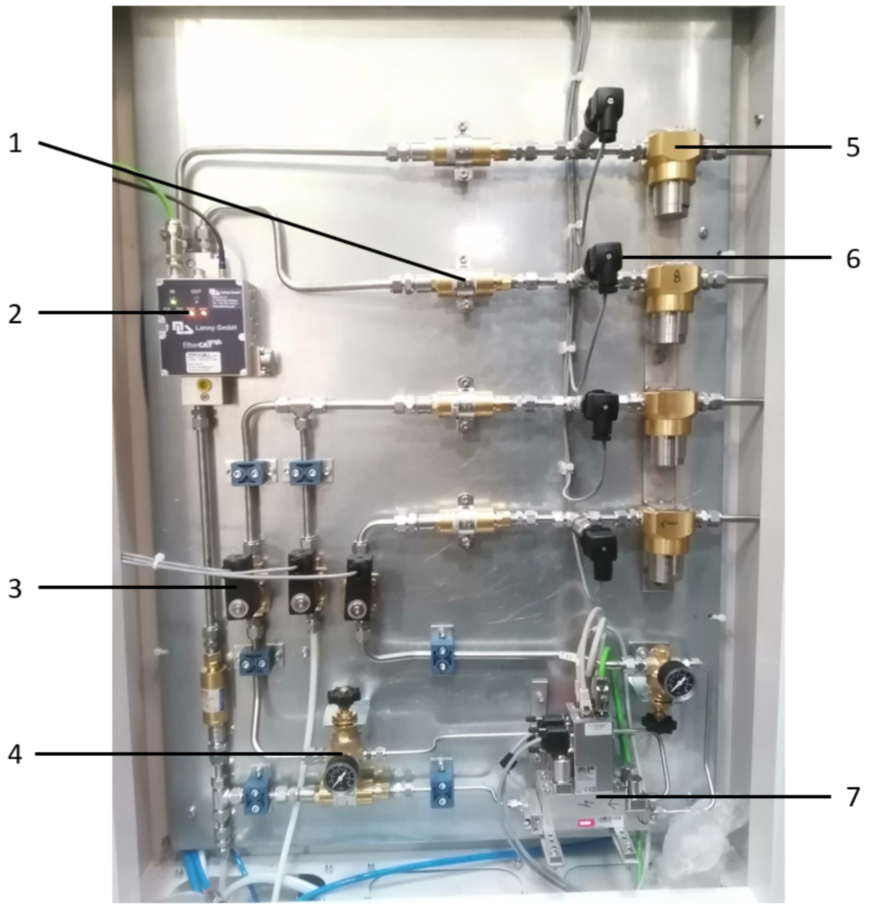


Figure 3.21: Installed gas panel for this application

Table 3.5: Main components of the Gas Panel

Number	Description
1	Return Valve - There is a total of 6 identical valves in the gas panel guaranteeing the flow of gases only moves one way
2	Proportional pressure control valve - Enables the selection of the output pressure for the cutting gases
3	Selector Valves -There is a total of 3 identical valves in the gas panel responsible for starting and stopping the gas supply
4	Manometer - Indicates the local gas pressure
5	Filter - There is a total of 4 identical filters for each gas type available
6	Pressure Control Valves - There is a total of 4 identical valves determining the gas pressure
7	Proportional flow control valve - Enables the selection of the output flow for the welding gases

3.1.8 Tool Changer

Although the robotic cell at the moment only has one tool available (*Combi-head*), the future plans include extending the number of tools to two, and as such a tool changer system is required. The system used was manufactured by Zimmer [37] and consists of a part attached to the robot, a part attached to the tool and a holder for the tool not being used.

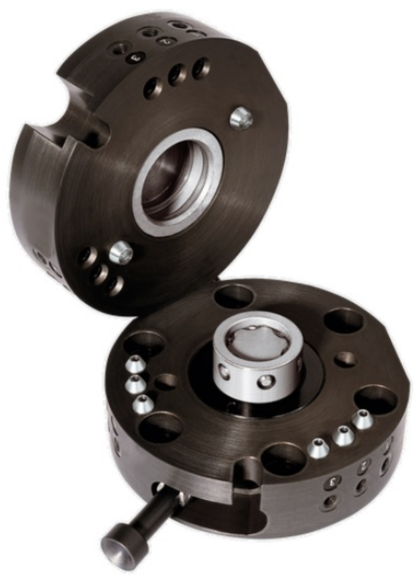
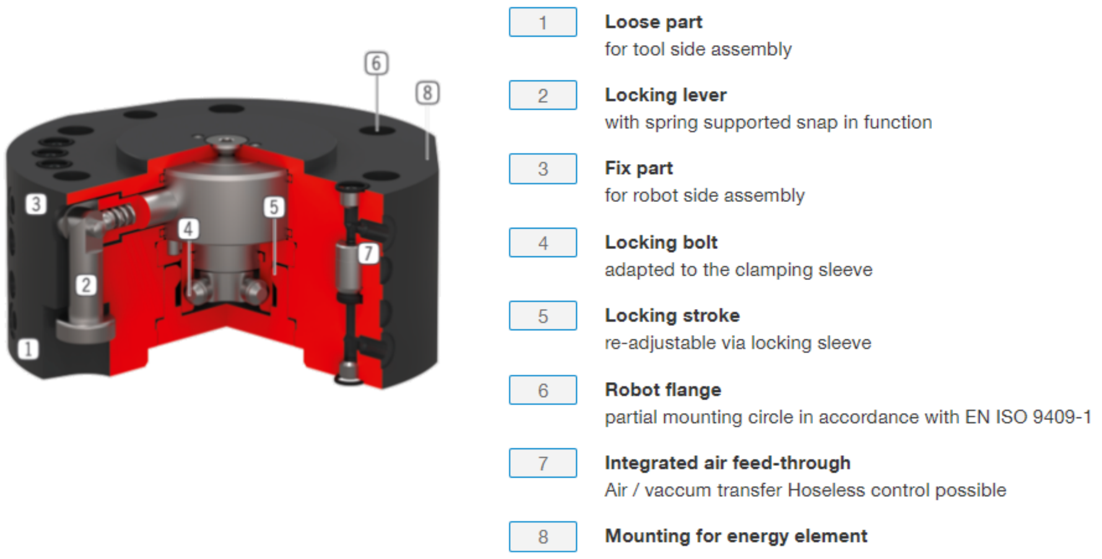


Figure 3.22: Zimmer tool changer *Series HWR* [38]

The selected variant of the tool changing system was the *Series HWR*, as it can be seen in figure 3.22, and according to Zimmer the main features of this system are [38]:

- Tool changing operation only lasting seconds, resulting in lower start up costs and a minimized downtime
- During the supply of the pneumatic actuators, additional data can be transmitted using optional energy elements, due to the integrated air transfer system
- The locking lever fully integrated into the housing makes it possible to replace tools without requiring any additional tools

Components:



1	**Loose part** for tool side assembly
2	**Locking lever** with spring supported snap in function
3	**Fix part** for robot side assembly
4	**Locking bolt** adapted to the clamping sleeve
5	**Locking stroke** re-adjustable via locking sleeve
6	**Robot flange** partial mounting circle in accordance with EN ISO 9409-1
7	**Integrated air feed-through** Air / vaccum transfer Hoseless control possible
8	**Mounting for energy element**

Figure 3.23: Components of the Zimmer tool changer *Series HWR* [38]

It is also important to enlighten the fact that during the welding and cutting procedures there is a possibility of the tool colliding with the workpiece. To prevent further damage to the tool, a crash protection system was also acquired from the same manufacturer of the tool changing system (Zimmer). The selected mechanism was the *CSR100*, available in figure 3.24, and according to Zimmer the main features of this device are the following [39]:

- Adjustable triggering sensitivity by controlling the air pressure
- The integrated sensors sharing a signal with the control system to trigger an emergency stop
- There is no automatic return ensuring that the operator inspects the machine and the reason for the emergency stop

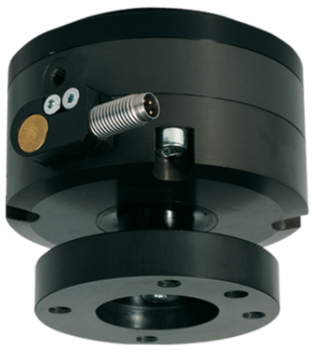


Figure 3.24: Zimmer Crash Protection *Series CSR100* [39]

The collision avoidance system is regularly used during the cutting operations, due to the cut parts not always being deposited in the correct location, as it can be seen in the fig. 3.25.

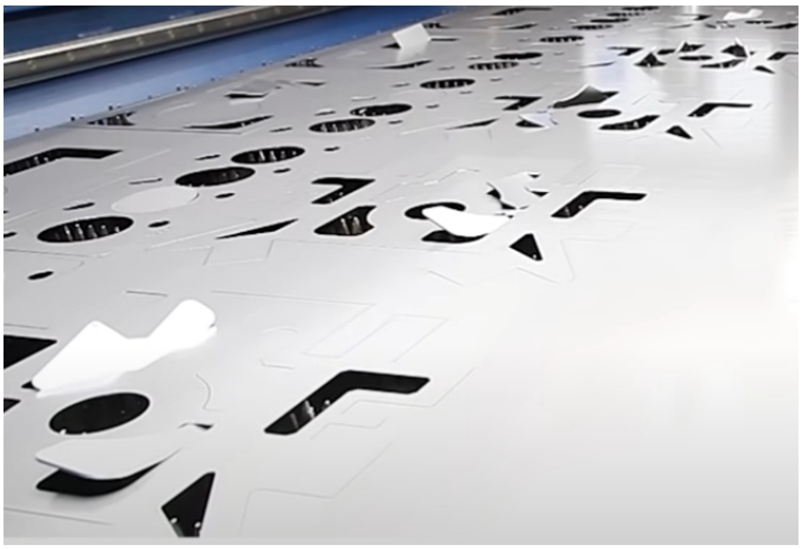


Figure 3.25: Common complication when laser cutting [40]

To conclude, the tool changing system is reliant on a directional valve pivoted by a digital signal defined in the robot controller, which changes value every tool changing cycle. The complete assembly of all the tool changing components can be seen in figure 3.26 .

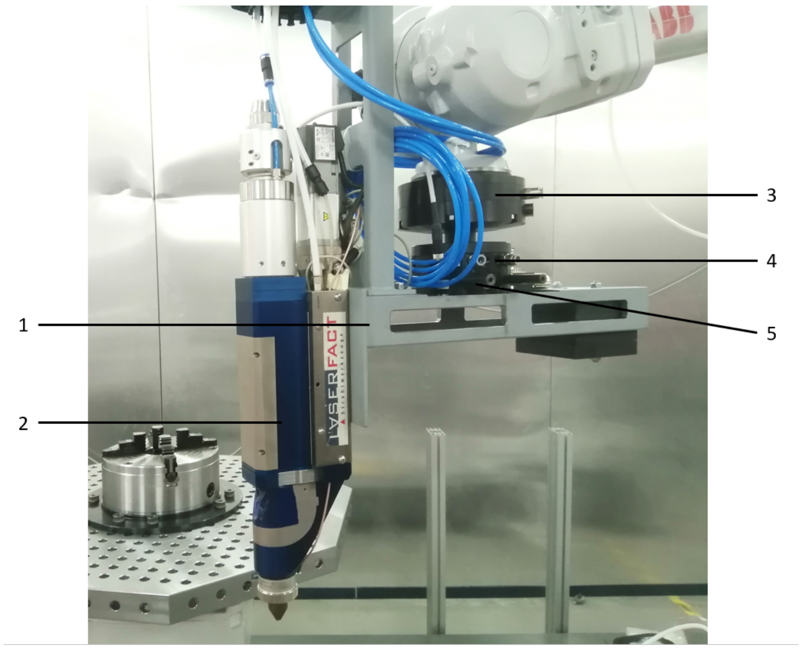


Figure 3.26: *Combi-head* and the equipment necessary for the tool changing procedures

Table 3.6: Main components of the Gas Panel

Number	Description
1	Support structure designed to increase the stability of the tool when placed on the holder
2	Laser tool *F2Y* manufactured by *LaserFact*
3	Crash detector manufactured by *Zimmer*
4	Male tool changer manufactured by *Zimmer*
5	Female tool changer manufactured by *Zimmer*

3.1.9 Laser Welding Table

The laser welding table used in this project is manufactured by Demmeler, with the reference D16 Okto 50 AF 1000 (PL16-01008-001). The table is positioned on top of the ABB positioner and is used to clamp the necessary components for the welding procedures. The welding table is available in figure 3.27 .

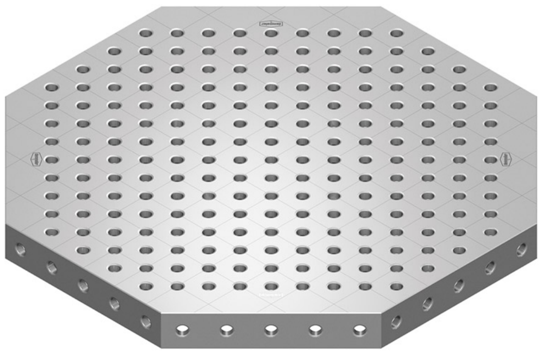


Figure 3.27: Welding Table manufactured by Demmeler with the reference PL16-01008-001 [41]

3.1.10 Clamping System

The clamping system used in this project to attach the metal tubes to the industrial positioner is manufactured by Röhm, with the reference PLATO ES 250-4 DIN 6351. The lathe chuck is positioned on top of the welding table and it can be seen in figure 3.28 .

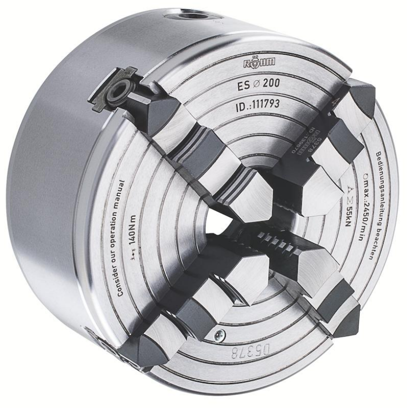


Figure 3.28: Lathe chuck manufactured by Röhm with the reference PLATO ES 250-4 DIN 6351 [42]

3.1.11 Energy Chain

The energy chain used to organize and protect the various cables necessary for the laser tool is manufactured by IGUS, with the reference triflex R TRCF (TRCF.65.200.0) - figure 3.29. This energy chain was chosen to be used by a multi-axial robotic system and protect the fiber which requires a minimum bend radius of 200 mm in dynamic movements. The length of this chain is 3800 mm and obtained with the help of *RobotStudio*.

Figure 3.29: Energy chain manufactured by IGUS with the reference TRCF.65 [43]

3.2 System Communication Architecture

In this sub-chapter it will be discussed the methods of data communication between each component of the robotic station, while reviewing the data being transferred. The ABB robot is sharing the robotic cell space with a KUKA robot capable 3D metal printing; hence, when starting a laser cutting/welding procedure it is necessary to request control of the laser system from the PLC associated with the KUKA robot.

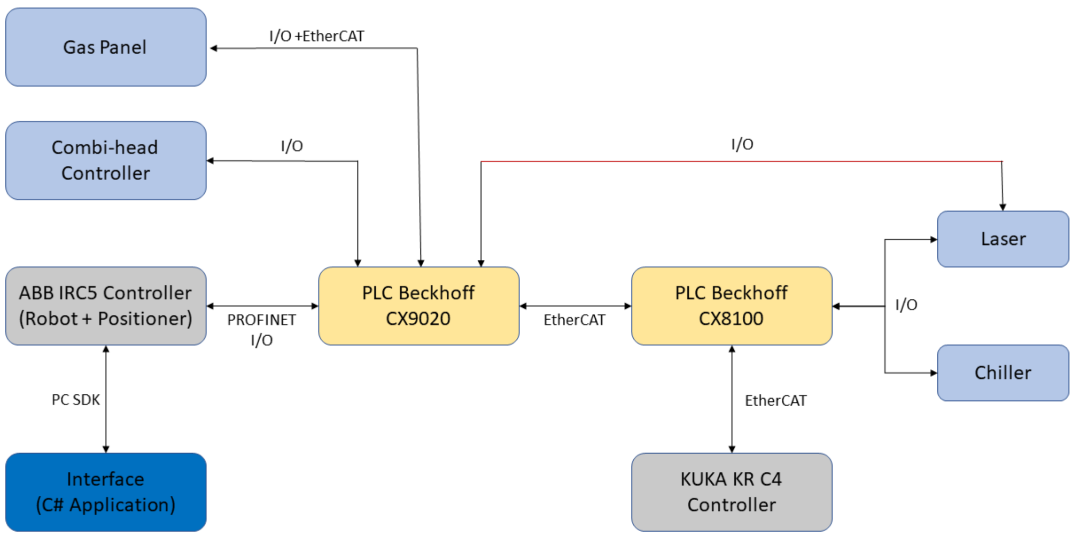


Figure 3.30: Scheme of the communication system between all of the components

3.2.1 Combi-head Controller & PLC Beckhoff CX9020

The method of data transfer between these two components is input/output cable connection. The sensor signal representing a process variable, such as the distance from the nozzle to the workpiece, is converted by an input I/O module and stored in the PLC memory.

The majority of the data being transferred between the Beckhoff CX9020 PLC and the *Combi-head* Controller is being used to determine the position of the tool in regard to workpiece and to alert for possible tool malfunctions. A detailed list of all the signals is available in the attachments in table A.1 .

3.2.2 Gas Panel & PLC Beckhoff CX9020

The gas panel is formed by pressure control valves, flow control valves and directional valves. The valves are communicating their values to the PLC using a *fieldbus* system with the objective of studying the influence of each process parameter in the procedure's results, so that the parameters can be optimized.

The parameters will be optimized to improve the overall quality of the pieces being produced; however, the ascertainment of the amount of gases being spent per procedure is also a major concern, since it helps to determine the price of a piece to study the economic viability of producing it. The process of minimizing the costs of a procedure can also involve changes to the piece's original design or the optimization of the robot's trajectories.

The data being transferred between these components is using not only input/output linking (I/O), similarly to the Combi-head Controller, but also using EtherCAT [44], which is a real time Ethernet-based fieldbus system designed for industrial automation. The EtherCAT system stands out due to its flexible topology, intuitive use and its high performance, which allows the creation of control systems that otherwise would not be possible.

The signals being transferred between the Beckhoff CX9020 PLC and the control valves in gas panel are available in the attachments in table A.3.

3.2.3 PLC Beckhoff CX9020 & PLC Beckhoff CX8100

As it was previously mentioned, the ABB robot used in this application is sharing the robotic cell space with a KUKA robot. They both use the same laser source and share some of the same safety mechanisms; hence, it is important highlight the data transferred between the PLCs associated with each robotic station. The communication between these two devices is done using EtherCAT and the scheme of the connection is available in figure 3.31 .

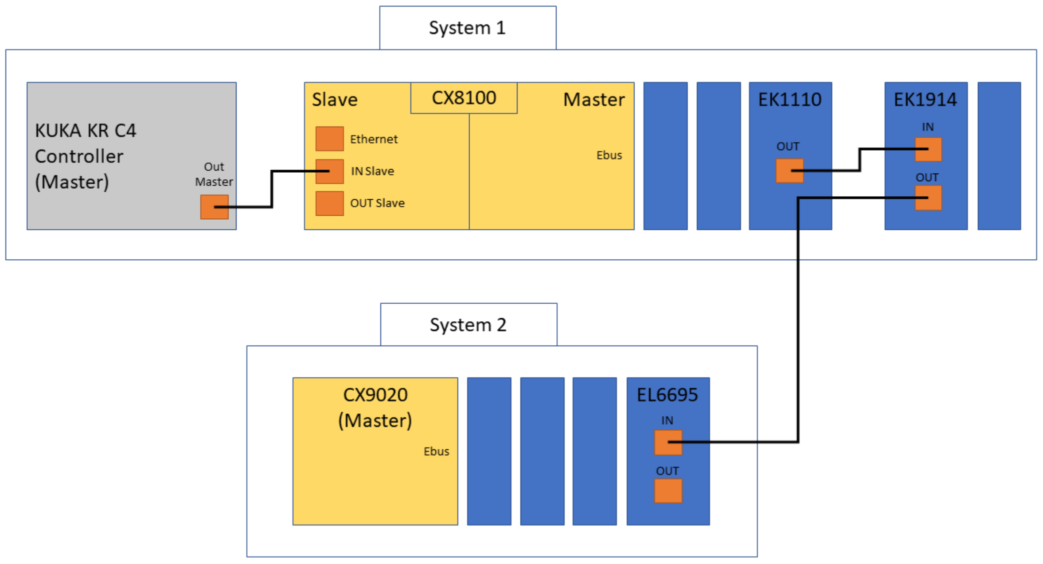


Figure 3.31: Scheme of the communication system between the two PLCs

The master of system 1 is the KUKA Controller and is connected to the PLC CX8100 [45] using EtherCAT. The CX8100 PLC is equipped with the EtherCAT extension EK1110 [46] placed at the end of the terminal segment, whose output is wired to the EK1914 coupler [47], which is extended with EL/ES terminals and is positioned inside the robotic cell. The EK1914 has safety parameters to adapt the functionality to the respective safety-oriented requirements with safety inputs and outputs being exchanged with TwinSAFE's Logic-capable component via FSoE.

The output of the EK1914 coupler is connected to the input of the EL6695 [48] bridge terminal mounted on the CX9020 PLC, enabling real-time data exchange between EtherCAT Terminals strands with different masters. The signals being shared between the two systems are available in the attachments in table A.4.

3.2.4 PLC Beckhoff CX9020 & ABB IRC5 Controller

The communication between these two components is using Profinet [49], which is an industry technical standard for data communication over Industrial Ethernet, designed for collecting data and controlling equipment in an industrial setup.

It is important to refer that the initial idea was to have the C# Application communicate directly with CX9020 PLC using Automation Device Specification (ADS) [50]; however, due to the limitation of the computer running the application needing TwinCAT3 software it was decided that all the PLC information needed for the display would first be transmitted to the IRC5 Controller and only then to the interface. Although this decision led to a significant larger number of signals in the robot controller, given the high-speed data transfer available when using Profinet there was not a problematic delay when communicating. The signals being shared between these two components are available in the attachments in table A.5.

3.2.5 Interface (C# Application) & ABB IRC5 Controller

The path taken to develop the interface for this robotic application was using ABB's PC Software Development Kit (PC SDK) [51], which is a software tool that enables programmers to develop customized interfaces for the IRC5 robot controller.

Since a well-designed user interface is capable of presenting relevant information and functionality at the right time, customized interfaces are clearly very desirable to the end-user of a robotic system. Tailored solutions are easier to operate and also optimize user's investment in automation; however, using PC SDK itself does not guarantee increased installation value. To achieve this increased installation value the application was developed with care and with a heavy emphasis on ease of use.

PC SDK uses Microsoft.NET and Microsoft Visual Studio; however, an important limitation is that ABB's support is only offered for Visual Basic and C#, letting the developer choose. Given the previous experiences developing applications using C# and establishing that this programming language is ideal for these situations, it was decided that the control application was going to be developed using C#.

The difference between the PC application and the FlexPendant application is that the PC application is a remote client, whereas the FlexPendant application is a local client. Remote clients do not have all the privileges of a local client. For example, both PC and FlexPendant applications can reset the program pointer and start RAPID execution, but for the PC SDK application to do this there are certain restrictions. Mastership of the Rapid domain must be requested by the application programmer and the IRC5 must be in automatic operating mode. On the other hand, the advantages of the remote client are the possibility to monitor and access several robot controllers from one location and the fact that the PC platform is less limited than the FlexPendant as regards the memory resources and processing power [52].

The signals and rapid variables used in the interface, as well as the mechanisms designed to interact with the robot, are described in the attachments in the *User Manual*.

3.3 Tool Changer Objectives

Before starting the development of a solution for the tool changer holder, it is first necessary to analyze the different objectives of this system and how they should be balanced. The first priority is to guarantee the stability of the tool in the holder, which means the tool must be stationary while positioned in the holder. Another important consideration is the type and tolerance of the robot movements to pick and place the tool, being preferable smaller movements and systems with larger dimensional tolerances. Finally, a vital concept to also consider is the amount of space the tool and the tool holder will require inside the robotic station, which must be minimized as much as possible.

3.4 Tool changer Holder Concepts

3.4.1 Vertical Holder

It was first analysed the holder proposed by *Zimmer* as a standard holder for this kind of tool changers. This type of holder is vertical, which means that the working end of the tool when placed in the holder will be vertically located. This solution can be seen in figure 3.32 .

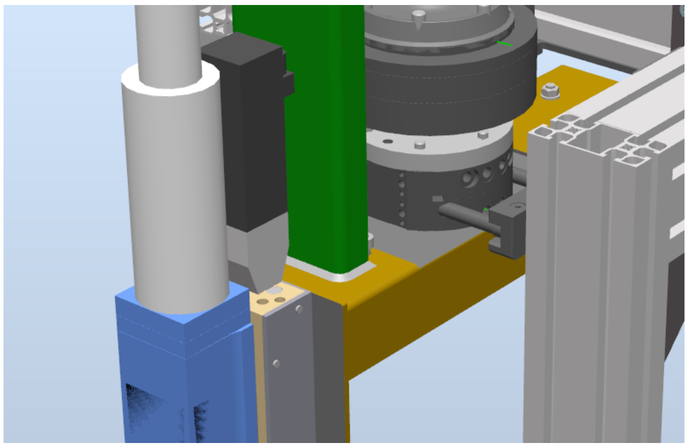


Figure 3.32: Display of the vertical holder with the *Combi-head* laser tool

The holder has two parallel grooves, identified in figure 3.33 by the number 3, and 4 pins attached to the female tool changer, identified in figure 3.33 by the number 2. As previously mentioned, the pins are attached to the female tool changer; however, in the figure 3.33 they are not, making it easier to understand how the pins slide into the groove. At the end of the grooves, there are 2 couplings, made of a flexible metal, where the pins will be fixed to stabilize the female tool changer. In figure 3.33 the couplings are identified by the number 1.

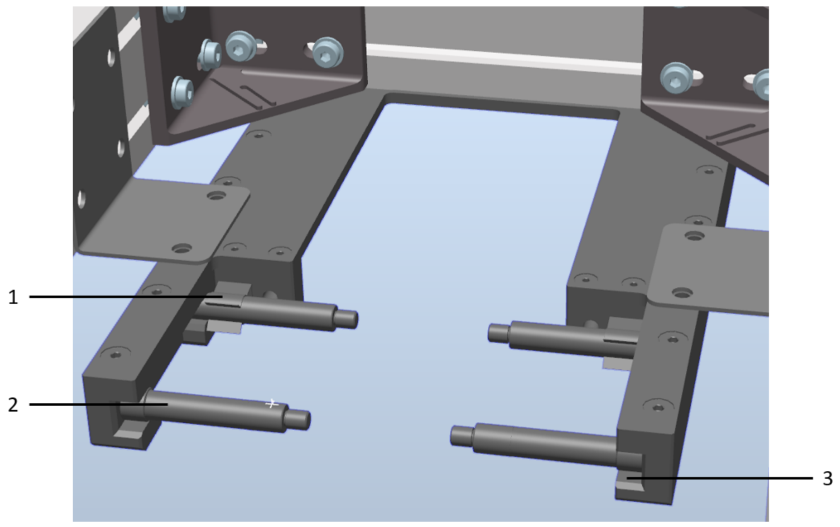


Figure 3.33: Components of the vertical tool holder

Although the tool is positioned vertically, the movement to place the tool in the holder will be horizontal, which means the robot will have to align, in the same horizontal plan, the female tool changer pins with the metal couplings. After the pins and couplings are aligned, it is then performed a slow linear movement in the x axis. In figure 3.34 it is possible to visualize how the tool would be placed in the holder.

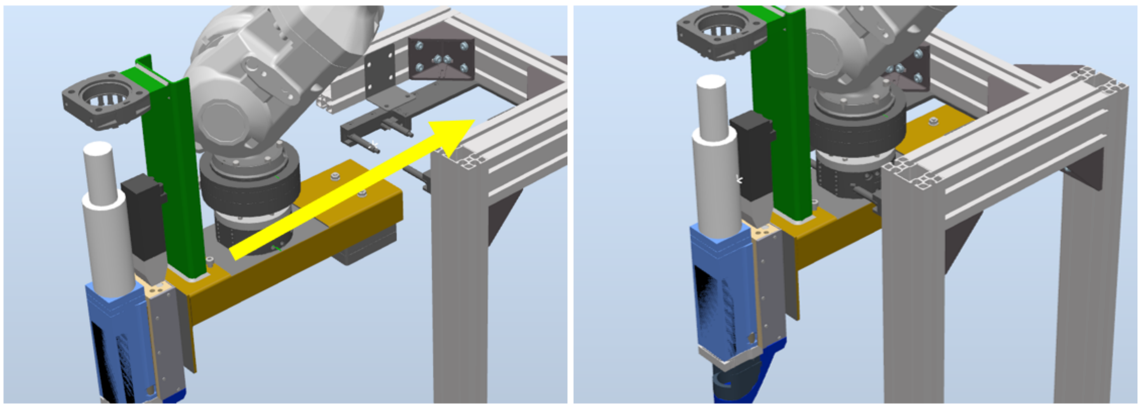


Figure 3.34: Horizontal movement to place the tool in the vertical holder

The advantages of using a vertical holder are:

- Superior stability of the tool, due to the existence of 4 holder points equally divided in space
- Programming does not raise concerns, since it requires no awkward robotic movements and most tools entailing a fairly small spectrum of movements

The disadvantages of using a vertical holder are:

- The space beneath a tool can not be utilized to store more tools, as the robot may collide with the standing tools
- Every tool must be stored side by side, occupying a significant amount of factory floor

3.4.2 Horizontal Holder

It was proposed an horizontal holder to achieve a more optimized occupation of the tool changing station space. This type of holder uses four pins connected to the female tool changer to grant stability to the tool. The main frame has a rectangular projection with a slightly larger gap than the distance between the pins of the female tool changer and also has four m6 holes, which are used to connect the holder to an eventual rack.

The tool changer will tightly slide inside the gap and due to the small difference between the gap in the frame and the gap between the pins the female tool changer will remain stuck in the frame and will not rotate, as it can be seen in figure 3.35

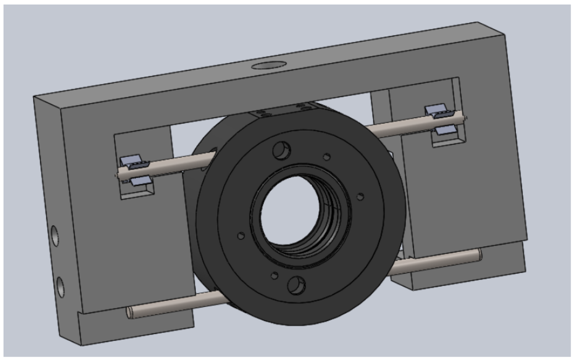


Figure 3.35: Display of the horizontal holder with the female tool changer

The advantages of using an horizontal holder are:

- Less space occupied, as the tools can be stored more efficiently.
- More flexibility when placing the tool, granting more options when deciding the position of each tool

The disadvantage of using a horizontal holder is the poor stability when storing an heavy tool, due to the possibility of the tool rotating or sliding.

The figure 3.36 illustrates the robotic movements to place the tool in the horizontal holder. The robot positions the tool in a location slightly more advanced then the location of the tool holder. Afterwards, the robot will move upwards, aligning the pins of the female tool changer with the metal couplings of the tool holder. When all the components are aligned, the robot will slowly move the female tool changer so that its pins can attach to the flexible metal couplings. Finally, after the pins are attached, the tool is released and the robot is free to return to its original position.

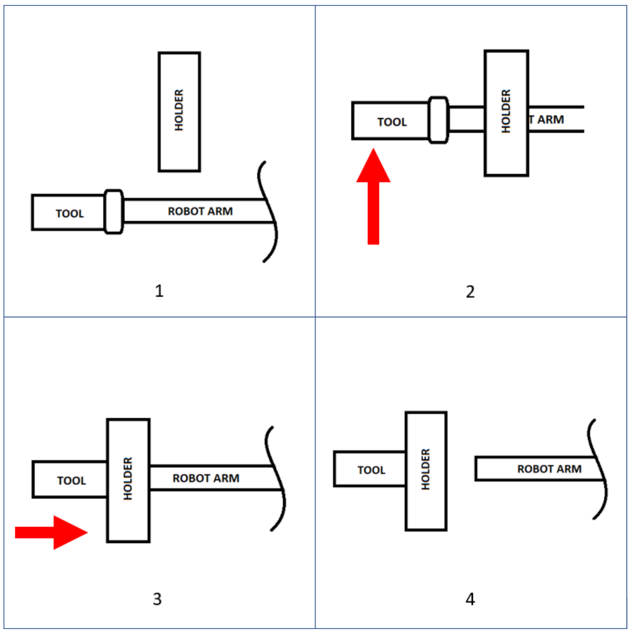


Figure 3.36: Representation of the robot movement to place a tool in the horizontal tool holder

3.5 Critical Analysis

In this sub-chapter, the different constructive solutions will be analysed and compared. The main guidelines when selecting the most adequate holder for our application are:

- Stability of the tool in the holder
- Type and complexity of the robot movements to reach the holder
- Space required when tool is placed in the holder

3.5.1 Tool Stabilization

The stabilization of the tool is vastly superior using the vertical holder, as there is no rotation of the tool and there is a good weigh distribution in the holder. It is important to enlighten the fact that the tool being used in this application (*Combi-head*) is significantly heavy, which might cause stability problems. To avoid future complications the selected tool holder should be vertical, since the horizontal tool holders are more appropriate for small/light tools and used to maximize the space of the robotic cell.

3.5.2 Movement Difficulties

The movement of the robot to reach the horizontal holder is notably shorter than the path to reach the vertical holder, resulting in the horizontal solution being superior to the vertical one. The vertical solution can result in complex movements and worst movement tolerance, especially if the holder is considerably elevated.

3.5.3 Space Occupation

The final aspect to consider is the space occupied by the tools when placed in the respective holders. The vertical solution has a high need for vertical space; however, due to the configuration of the robotic cell and positioning of the tool holder, it is preferable to use the vertical solution as there would not be enough space for the *Combi-head* when placed horizontally, due to the proximity of a cell wall, as it can be seen in figure 3.37 .

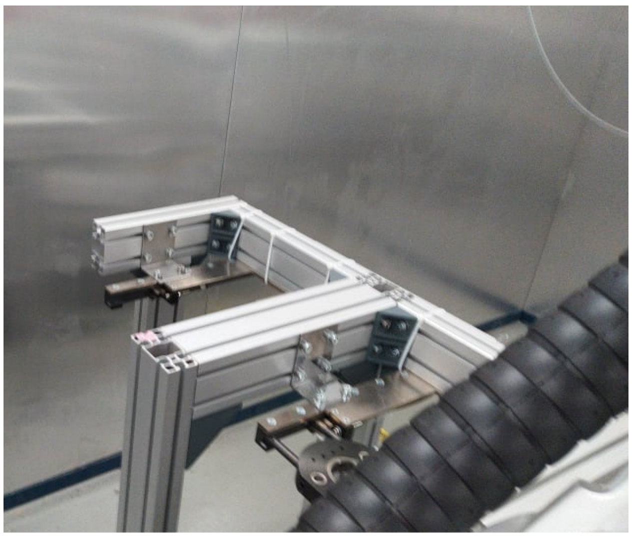


Figure 3.37: Space available for the tool changing station

To conclude, it is simple to understand that the best option for this project is the vertical tool holder. The only advantage of using the horizontal holder would be the optimized cell space occupation; however, only two tools are expected to be used in this station; therefor, the space occupied by the tools is not a crucial deciding factor in this project. Although the movements to reach the horizontal tool holder are shorter and simpler than the movements to reach the vertical tool holder, the tool stabilization is superior with the vertical tool holder, which is extremely important to guarantee the tool safety. Additionally, the current position of tool changer support beams is not ideal to assemble the horizontal tool changer.

In the forth chapter, the development of the control application and the *RAPID* program will be the focus. The steps necessary to create the virtual representation of the robotic cell will also be described in this chapter.

Chapter 4

Solution Development

In this chapter it will be discussed the development of the control application (C#) and the RAPID program used in this project to control the robot and the various components of the robotic cell. The software and methodologies used to develop these solutions will also be analysed.

Lastly, a detailed description of the virtual cell's assembly will be given to achieve a high quality solution.

4.1 Application Objectives

On the one hand, the tool changing capabilities should enable the pick and placing of the available tools at any given circumstance, with the selection of the desired tool being done with the C# application. The creation of the necessary robotic paths without risk of collision required the integrator to carefully pick the most optimized targets.

On the other hand, the laser processing capabilities of this system should allow the user to select sets of process parameters and robotic paths to start custom procedures. The laser system can be used to simply test process parameters with workpieces or to create custom parts with optimized parameters. The laser system must be able to perform laser cutting/welding procedures and achieve high quality results.

The requirements for the C# application can be seen in the following list:

- Maintenance Tab, where crucial information of the robot controller and the system could be analyzed, such as the IP address, the communication status of all the components, etc. .
- List of the relevant event logs, including all the messages from the robot controller and some information from the application, such as the start of the different procedures.
- Capacity to upload a rapid procedure into the the controller using the C# application.
- Method to select the process parameters for laser cutting and laser welding operations to create custom parts, either for 2D or 3D material processing.
- Move the robot's TCP using the C# application, by starting the jogging procedures or the procedures to move the robot to a useful target, named HOME positions.

- Display of the position and velocity of the robot's TCP.
- Display of the tool currently equipped to the robot.
- Method to test different process variables to determine the optimized conditions for various materials and workpieces.
- Method to stop all the robotic movements and reset all the necessary variables to prevent the system damages.
- Create reports of the testing procedures, referring the process parameters used, the date and time it started, the currently operator logged in the application and the type of procedure completed, which can either be laser cutting or laser welding.
- For the laser cutting operations of 2D parts it is necessary to have a procedure to edit the workobject associated to the cutting table, so that the robotic procedure can start from any given position of the table.

4.2 Tools and software

The development of the laser processing system required the use of different software. It was necessary to use *RobotStudio* to program the robot movements, it was required the use of *TwinCAT3* to program the PLC controlling all the various components of the robotic cell and to program the HMI it was used *VisualStudio (C#)*.

Figure 4.1: Flex Pendant [53]

The ABB programming language is *RAPID*, and it is used to program all of their robots. Despite this language being easy to use, it is a highly powerful; therefor it is necessary to explain some concepts inherent to this programming language.

As it was previously mentioned the software used by ABB is *RobotStudio*, which is used to program most robotic applications. This software includes a graphical station where the simulation of the robot movements is done. A series of add-ins that facilitate the programming and the integration of a real station is also made available in this software.

The creation of the graphical interface for this process required using *RobotStudio* and a C# application, while taking advantage of the manufacturer's software development tools (PC SDK). This tools facilitated the process of sharing information between the HMI and the robot controller, which was programmed using *RAPID*.

4.3 RAPID

RAPID is the language used for programming the different routines that were used in this project, therefore it is vital to understand the basic concepts intrinsic to this programming language and also comprehend how *RobotStudio* can be used to simplify the programming procedure.

First of all, it will be analyzed the main control mechanisms, which are the different types of routines. To understand the goal of each routine and the proper application is vital when creating a good quality solution to any robotic integration.

Throughout the programming of this solution, several robot movements were necessary and to program these movements it was essential to comprehend the meaning of the movement data used in this programming language, which will also be explored going forward.

Finally, it will be studied the different coordinate systems that *RAPID* uses to locate different objects and define the different robot movements.

4.3.1 Main Control Mechanisms

Procedures: The main routine used in *RAPID* is a procedure. It is a piece of code that runs sequentially, and does not return any value [54]. These functions are used to defined paths, which are fundamental to any *RAPID* program.

Functions: A function is a control mechanism that given an input returns a determined value, which is calculated the same way every time [54]. These functions are mostly used to avoid repeating the same expression several times.

Traps: A trap routine is a control mechanism used to handle interruptions. They are used as response to a given input, enabling the program to be immediately interrupted to perform a given task; however, this kind of routines cannot be directly called [54].

The 4 different ways to interrupt a *RAPID* program and start a trap routine are:

- Changing signals values, these signals can be of any type (digital,analog and group) and they can be inputs or outputs
- Occurrence of errors
- Cyclical interrupts, meaning the interrupt will be fired every set period of time
- Changing *RAPID* variables values

4.3.2 Movement Data

Paths: A path is a particular type of procedure that defines the way the robot moves between points. The use of *"Move"* instructions is required to define the movement of a robot from one position to the destination target.

To define a path it is necessary to choose the type of movement the robot will make. There are several types of movements available:

- **Linear Movement** (*"MoveL"*) - During a linear movement the joints of the robot rotate to allow a linear movement of the tool center point.
- **Joint Movement** (*"MoveJ"*) - During a joint movement, the path can either be linear or not linear.
- **Circular Movement** (*"MoveC"*) - During a circular movement, the path is calculated using two different points and the shape of the path is circular.

When defining a path it is also important to define the movement speed, which is represented by a (*"v"*) and a number in front of it. The value of the number corresponds to the maximum velocity of the TCP during that particular movement.

Another parameter to specify is the zone, which corresponds to the location where the robot stops moving towards the target inside the current move instruction and initiates the movement towards the target inside the next move instruction. This parameter is defined by the letter (*"z"*) and a number in front of it or a by using (*"fine"*) if the user wants the robot to reach precisely the destination target.

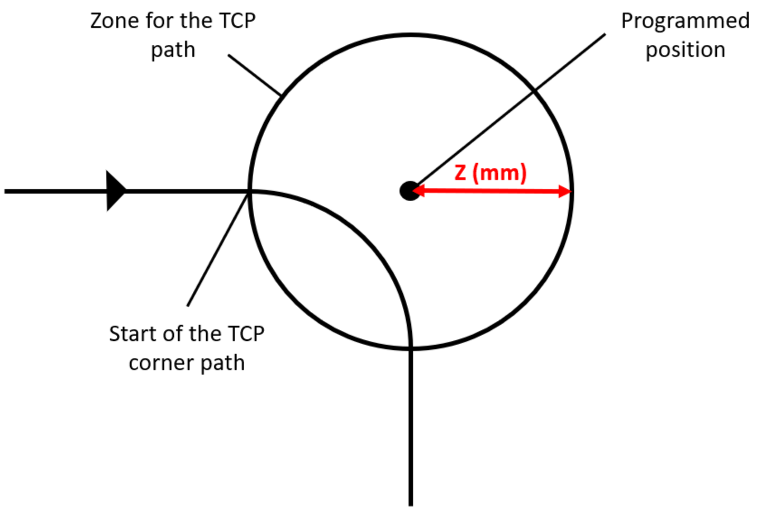


Figure 4.2: Influence of the zone parameter in a path movement

Lastly, it is necessary to characterize the tool equipped to the robot and the referential where the coordinates are defined, which can be seen in figure 4.3 .

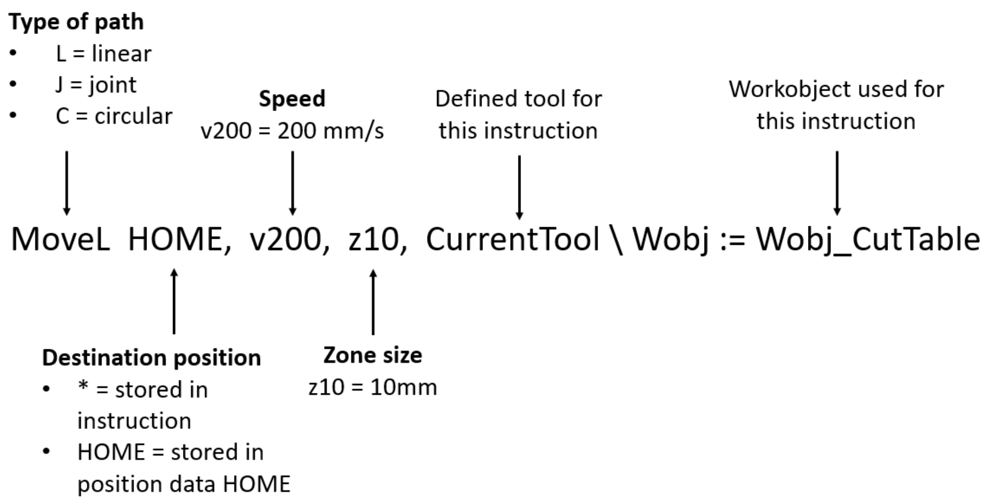

Type of path
• L = linear
• J = joint
• C = circular
Speed
v200 = 200 mm/s
Defined tool for this instruction
Workobject used for this instruction
MoveL HOME, v200, z10, CurrentTool \ Wobj := Wobj_CutTable
Destination position
• * = stored in instruction
• HOME = stored in position data HOME
Zone size
z10 = 10mm

Figure 4.3: Example of a *Move* instruction

In a procedure it is possible to define instructions for the robot controller at a given target, an example of this is the function (*"SetDO MoveXPos,1"*), which instructs the controller to change the value of the digital signal *"MoveXPos"* to true.

Targets: The destination point of a robot move instruction is a target, which are data elements that define a given position and orientation of the tool center point that the robot can visit. A target is defined using using a specific workobject and robot configuration.

The position is the spatial location within a coordinate system that the robot visits defined by the coordinates x, y and z. The orientation is the angle of the tool frame when in a specific position and is generally expressed in quaternions.

The configuration is defined by a vector containing the rotation of the robot's joints when reaching a target. Most targets can not be reached with every configuration; therefore, the user must choose a valid configuration. The targets can be defined using *RobotStudio*, in which it is possible to test different configurations and choose a valid one, whereas when using the *FlexPendant*, available in figure 4.1, it is not possible to verify the applicability of the robot's configurations automatically [54].

4.3.3 Coordinate System Representation

World Coordinate System: The world coordinate system (WCS) is the basic reference of the entire work station. This coordinate system has the highest hierarchy, which means that all other coordinate systems are referenced using it [54].

Workobjects: A workobject is a coordinate system used to describe the position of a workpiece and generally represents the physical location of an object in a workstation [54].

The workobject is composed by two frames, a user frame and an object frame. All programmed positions will be related to the object frame, which is defined according to the user frame. The user frame is defined using the world coordinate system as it can be seen in figure 4.4 .

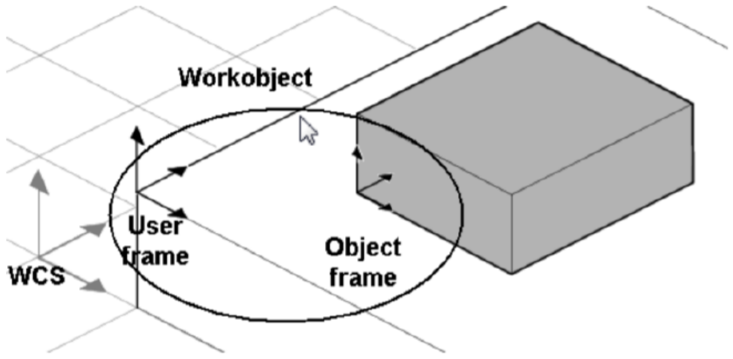


Figure 4.4: Workobject coordinate system [54]

This coordinate system is particularly useful due to its flexibility, as the reconfiguration is simplified by using two frames to define the coordinate system. When a re-calibration is required, it is only necessary to readjust the user frame, because all the targets that are defined to that workobject will be adjusted accordingly [54].

Tool Frame: A tool frame is a coordinate system bound to a given tool. This coordinate system has a orientation and a location, similarly to a target, and is used to simplify the programming of the robot movements.

The coordinate system is defined in the working end of the tool, for example the tool frame of the *Combi-head* is defined in the nozzle and the z axis is oriented to the target surface.

4.4 ABB RobotStudio

ABB RobotStudio is a program used to simulate robotic stations. This software allows the user to select a robot and other equipment from an impressive library and program these components accordingly. The programming can then be tested in a virtual representation of the real station, resulting in fewer mistakes when integrating the solution to the real world.

This software also allows for a more direct type of programming, since the creation of targets, paths, workobjects and other *RAPID* data is facilitated with the use of buttons and displays. It is possible to see in figure 4.5, a series of buttons that enable the creation of the different elements.

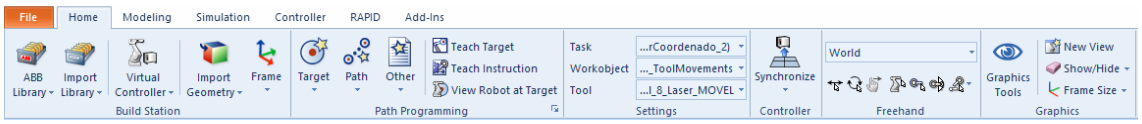


Figure 4.5: Example of the functions available with *RobotStudio* software

Smart components: A smart component is used to recreate a specific function from a tool or an object during the simulation of a procedure. An example of an application where the use of a smart components is required is a pick and place station, because the use of the *Attacher* and *Detacher* is essential to create the most authentic simulation of the real world station.

- Attacher: Connect two objects in the simulation, making them move as one.
- Detacher: Disconnect two objects in the simulation making them move independently.

There are several other uses for smart components, including sensors to detect objects inside the simulation and the capacity to convey movement to an object to simulate a conveyor belt, for example.

Parts: In RobotStudio, a part is an imported geometry that was built using CAD software. These geometries can have different purposes, such as creating an object for the robot to use, or alternatively creating an object for the robot to avoid.

The parts are essential to create tools, since it is possible to define a given geometry as a tool that the robot can use. The movement of certain tool components is also possible, such as closing and opening clamps of a gripping tool.

4.5 C# Application

This sub-chapter will consist of a brief description of the UI framework used to program this application, and it will be explored the functionalities and the architecture of the programmed C# application.

4.5.1 UI Framework

This application was programmed using Windows Presentation Foundation (WPF) which is a UI framework used to create desktop client applications. The WPF development platform supports a broad set of application development features, such as an application model, controls, graphics, layout, data binding, documents, and security [55].

The alternative of using WPF was using WinForms; however, the advantages of WPF in this situation outnumbered the benefits of using WinForms, as it can be seen in the following lists.

The advantages of using WPF instead of WinForms are:

- More recent; hence, more in tune with current standards
- More flexible, granting more functionalities without having to write or buy new controls
- Developers of third party controls are more likely to be using WPF because it is more current

- XAML [56], which is used to develop the graphical interface in WPF, makes it easy to create and edit the graphical user interface (GUI), and allows the work to be split between designer (XAML) and programmer (C#)
- Data binding in WPF allows to get a clean separation between data and layout
- Uses hardware acceleration for drawing the GUI, achieving better performance
- Allows the user to make interfaces for Windows applications and Web applications

The advantages of using WinForms when compared to WPF are:

- More mature and therefor more tried and tested
- Existence of a several third-party controls available to buy or get for free
- Simpler framework, resulting in developers needing less effort and time to learn WinForms

4.5.2 Content Display

After the control application start-up the *Authentication Window*, which is available in figure 4.6, will appear and require a password. The password is inserted and the login button is pressed, opening the *MainWindow*, which can be seen in figure 4.7. The *MainWindow*, where the actual control of the robot and process takes place, displays the name of the current application user, which is determined based on the inserted password.

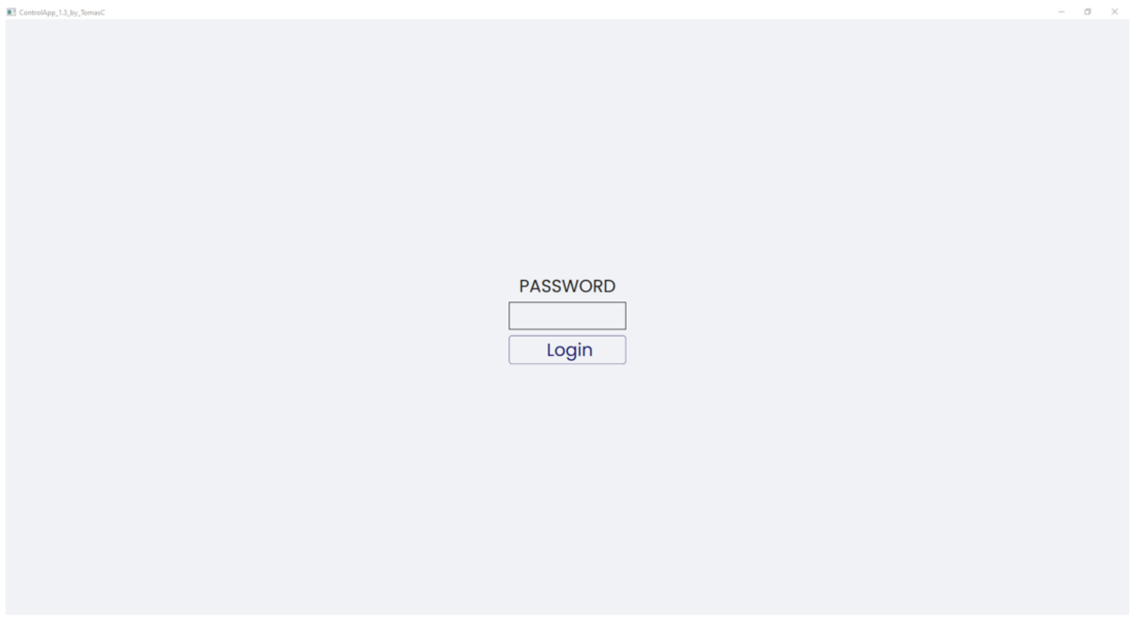
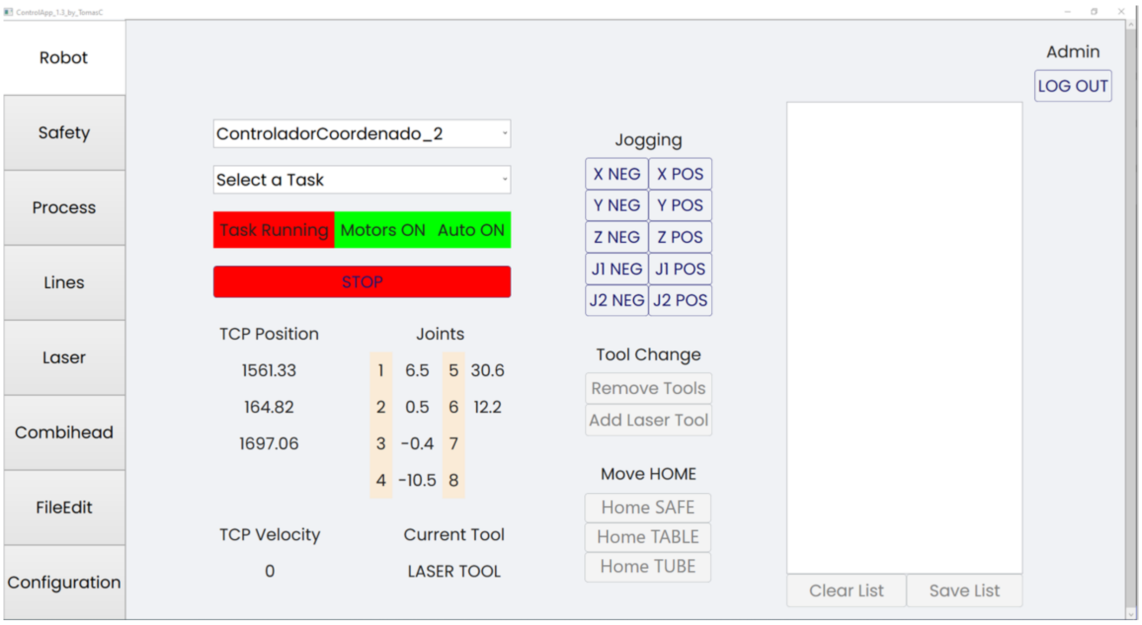


Figure 4.6: Display of the *Authentication Window*

The *MainWindow* has eight tabs, one for each crucial element of this application. The first tab header is *"Robot"*, and in this tab there are several controls available to interact with a robot controller. The selection of the controller and task used throughout the application will be done in this tab, since it is the first tab that opens when using the *MainWindow*. It is also possible to jog the robot and the positioner to the desired positions, while analysing the coordinates and velocity of the TCP. The values of the joints rotation is also displayed in this tab.

The *"Robot"* tab, available in figure 4.7, has the buttons responsible for starting the procedures to add and remove the available laser tool and a display of the currently equipped tool. Using the controls of this tab, it is also possible to start the procedures to move the robot's TCP to extremely useful positions to start common movements.

Finally, it was included an event log with all the messages from the robot controller and some messages from the C# application, to facilitate the detection of malfunctions.



ControladorCoordenado_2

Select a Task

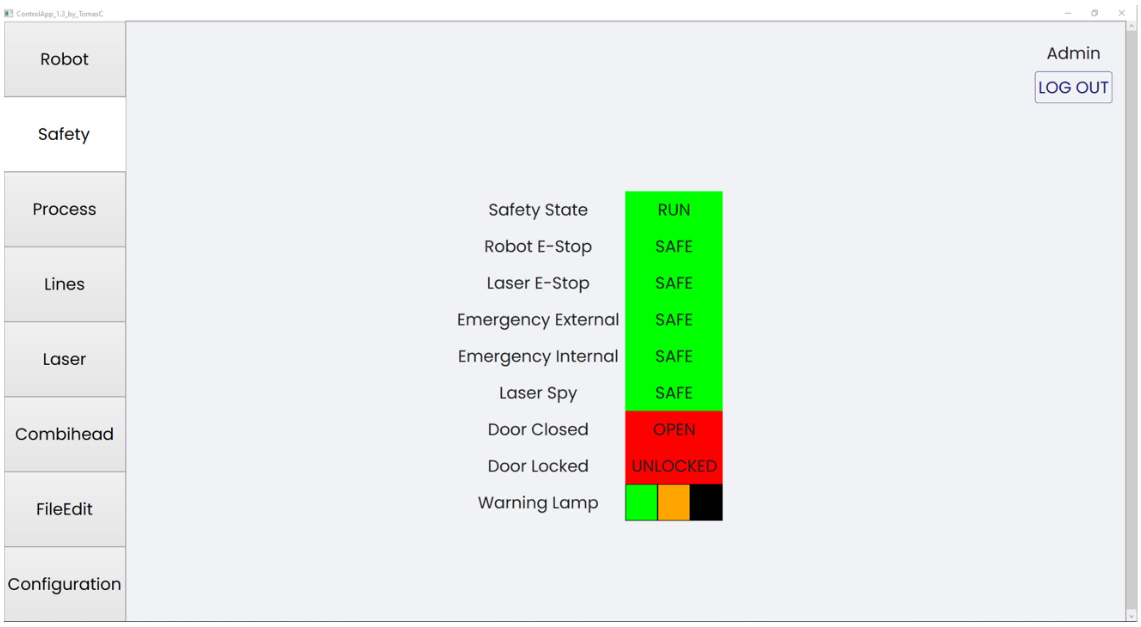
TCP Position	Joints			
1561.33	1	6.5	5	30.6
164.82	2	0.5	6	12.2
1697.06	3	-0.4	7	
	4	-10.5	8	

TCP Velocity: 0

Current Tool: LASER TOOL

Figure 4.7: Display of the *MainWindow* - *"Robot" Tab*

The second tab of the *MainWindow*, named *"Safety"*, is used to display the state of several safety features of the robotic cell. The decision to use the *"Safety"* tab as the second tab was obvious, as it reminds the user to verify all the safety systems before starting any laser procedure. The *"Safety"* tab is available in figure 4.8 .



Safety State	RUN
Robot E-Stop	SAFE
Laser E-Stop	SAFE
Emergency External	SAFE
Emergency Internal	SAFE
Laser Spy	SAFE
Door Closed	OPEN
Door Locked	UNLOCKED
Warning Lamp	🟩🟧⬛

Figure 4.8: Display of the *MainWindow* - *"Safety" Tab*

The third tab of the *MainWindow*, named *"Process"*, is used by the operator to create custom pieces. It is possible to import new *RobotStudio* modules, with different paths, to the robot controller and to select the process variables for the procedure, by choosing a CSV file containing the necessary data.

This tab also displays the values of all the process variables involved in the laser procedures to facilitate the process of optimizing the results. It was included a *"Stop"* button, which stops the robot movements and resets the values of the necessary process variables, guaranteeing the safety of the robotic station.

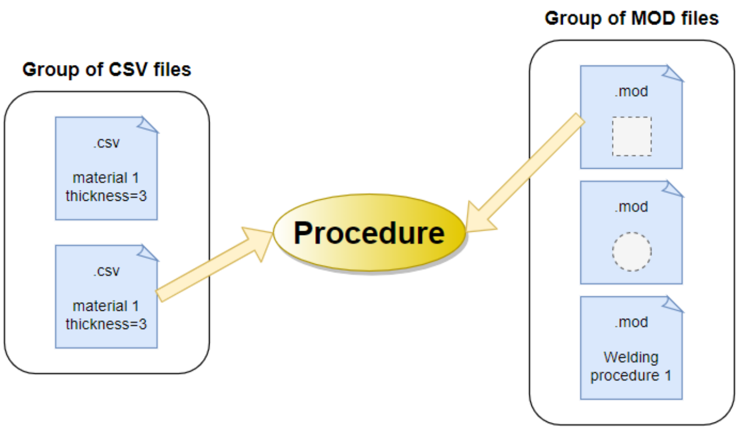
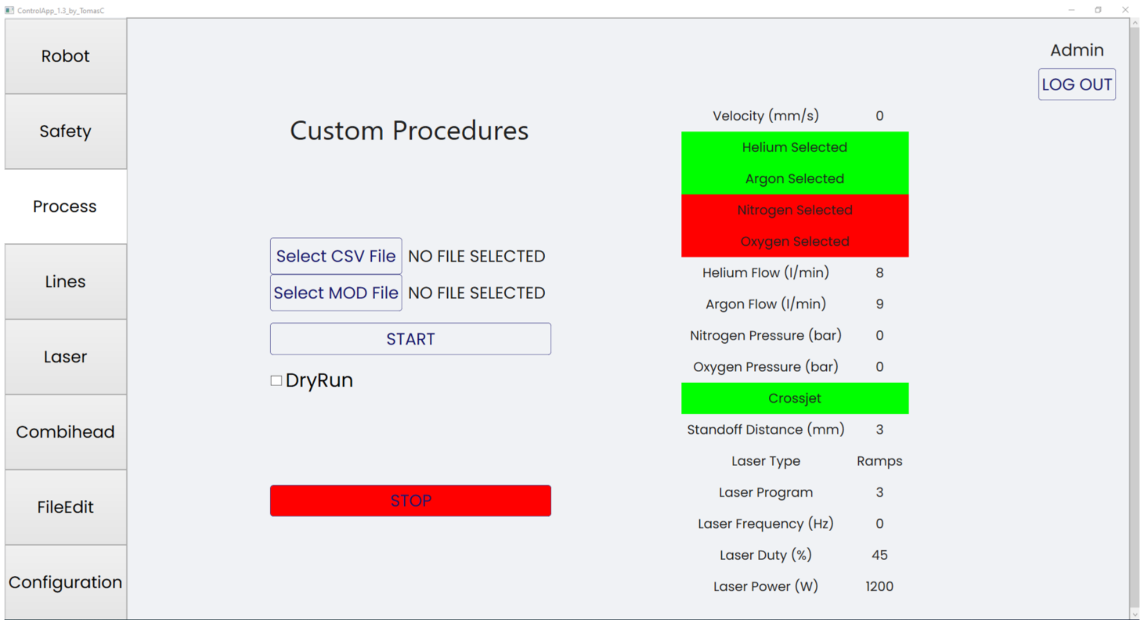


Figure 4.9: Methodology to create a custom piece using the *MainWindow* - *Process* tab

The purpose of this tab is for the operator to have the flexibility to use the same robot path with different CSV files contain the ideal process variables for a specific workpiece. For example, if the objective was to cut two identical squares from metal sheets with different thicknesses, the same robot path would be used, but different CSV files would be selected before starting each procedure. The *”Process”* tab is available in figure 4.10 .



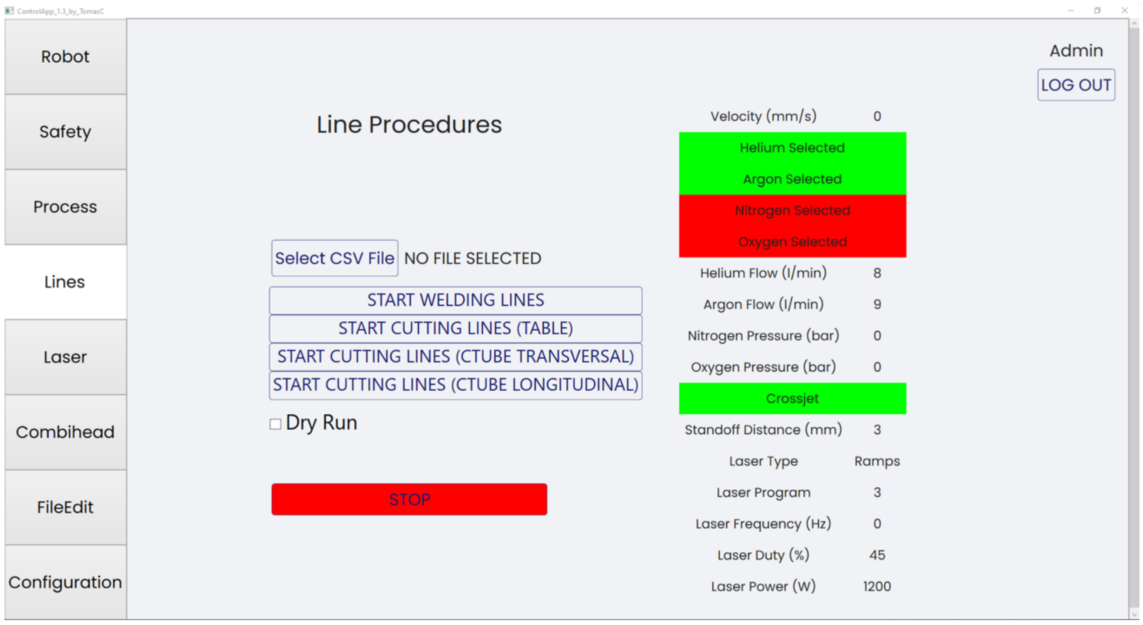
NO FILE SELECTED

NO FILE SELECTED

Velocity (mm/s)	0
Helium Selected	
Argon Selected	
Nitrogen Selected	
Oxygen Selected	
Helium Flow (l/min)	8
Argon Flow (l/min)	9
Nitrogen Pressure (bar)	0
Oxygen Pressure (bar)	0
Crossjet	
Standoff Distance (mm)	3
Laser Type	Ramps
Laser Program	3
Laser Frequency (Hz)	0
Laser Duty (%)	45
Laser Power (W)	1200

Figure 4.10: Display of the *MainWindow - ”Process” Tab*

The fourth tab of the *MainWindow*, named *“Lines”*, is used by the operator to test different process parameters to optimize the quality of the production. The testing requires the selection of a CSV file containing the number of lines and the process variables for each line. After the necessary file is selected one of the available procedures can be started, always resulting in the creation of several parallel lines. The process parameters can be monitored using this tab’s display, as it can be seen in figure 4.11 .

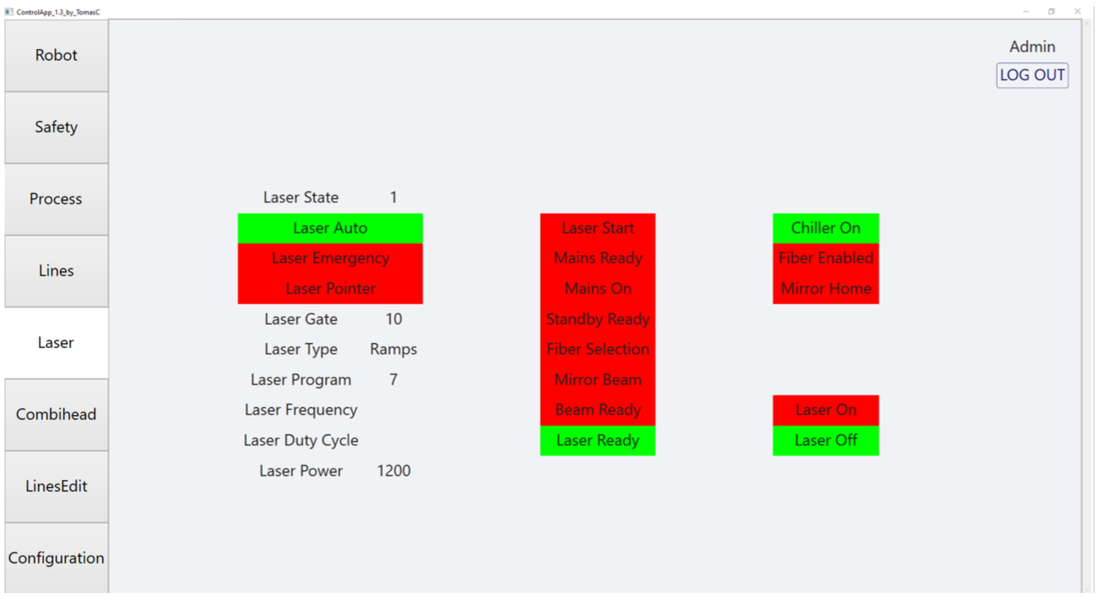


NO FILE SELECTED

Velocity (mm/s)	0
Helium Selected	
Argon Selected	
Nitrogen Selected	
Oxygen Selected	
Helium Flow (l/min)	8
Argon Flow (l/min)	9
Nitrogen Pressure (bar)	0
Oxygen Pressure (bar)	0
Crossjet	
Standoff Distance (mm)	3
Laser Type	Ramps
Laser Program	3
Laser Frequency (Hz)	0
Laser Duty (%)	45
Laser Power (W)	1200

Figure 4.11: Display of the *MainWindow - ”Lines” Tab*

The fifth tab of the *MainWindow*, named *“Laser”*, is used to monitor the state of several laser related variables and to activate the laser pointer, which enables a better positioning of the laser nozzle. In this tab it is also possible to analyze the startup sequence of the laser system, meaning that in case of malfunction it is easier to determine the origin of the problem, minimizing the downtime of the system. The *“Laser”* tab is available in figure 4.12 .



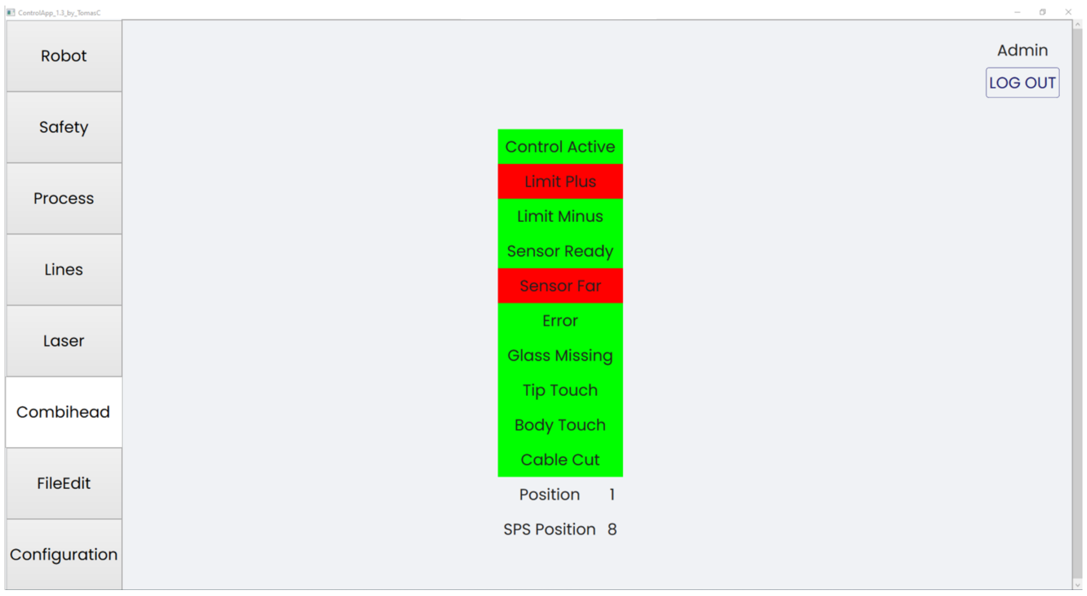
Laser State	1
🟩 Laser Auto	
🟥 Laser Emergency	
🟥 Laser Pointer	
Laser Gate	10
Laser Type	Ramps
Laser Program	7
Laser Frequency	
Laser Duty Cycle	
Laser Power	1200

- 🟥 Laser Start
- 🟥 Mains Ready
- 🟥 Mains On
- 🟥 Standby Ready
- 🟥 Fiber Selection
- 🟥 Mirror Beam
- 🟥 Beam Ready
- 🟩 Laser Ready

- 🟩 Chiller On
- 🟥 Fiber Enabled
- 🟥 Mirror Home
- 🟥 Laser On
- 🟩 Laser Off

Figure 4.12: Display of the *MainWindow* - *”Laser” Tab*

The sixth tab of the *MainWindow*, named *“Combihead”*, is used to display the state of several signals related to the used laser tool. This tab enables the detection of several tool problems, such as the collision of the tool tip and the body. Furthermore, it is possible to precisely determine the position of the *Combihead* in relation to the workpiece using the displayed values in this tab. The *“Combihead”* tab is available in figure 4.13 .



- 🟩 Control Active
- 🟥 Limit Plus
- 🟩 Limit Minus
- 🟩 Sensor Ready
- 🟥 Sensor Far
- 🟩 Error
- 🟩 Glass Missing
- 🟩 Tip Touch
- 🟩 Body Touch
- 🟩 Cable Cut

Position	1
SPS Position	8

Figure 4.13: Display of the *MainWindow* - *”Combihead” Tab*

The seventh tab of the *MainWindow*, named *"FileEdit"*, is used to edit and create the CSV files required for the testing procedures. After selecting a CSV file it is possible to analyse and edit the properties assigned to each line as well as add and remove lines. The edited version of the array of properties can be used to replace the original file or create a new CSV file. The *"FileEdit"* tab is available in figure 4.14 .

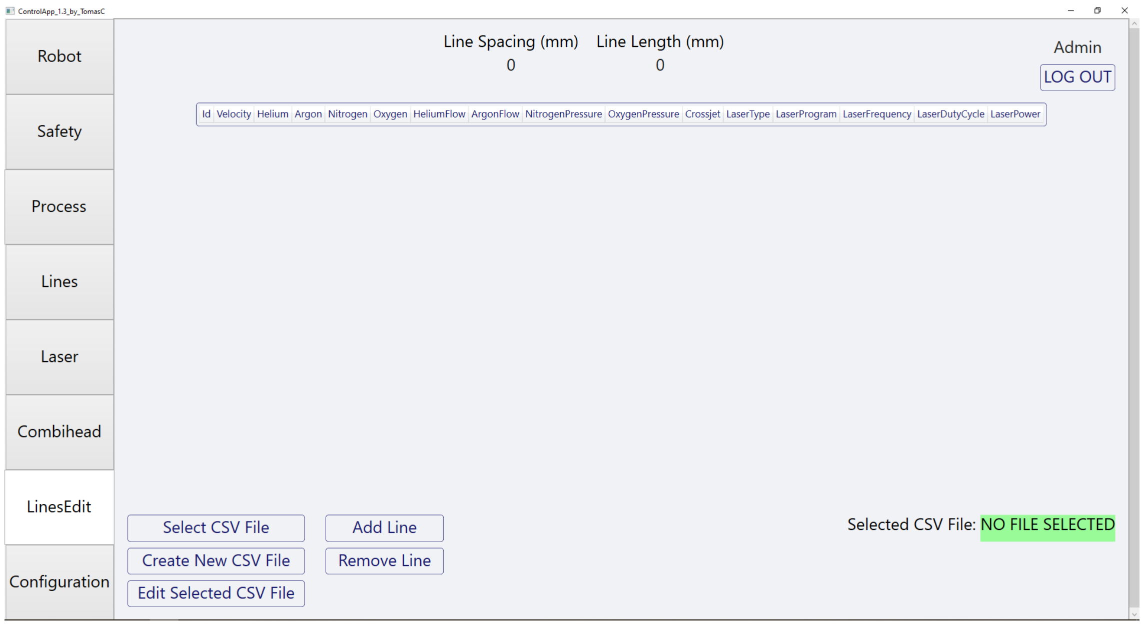
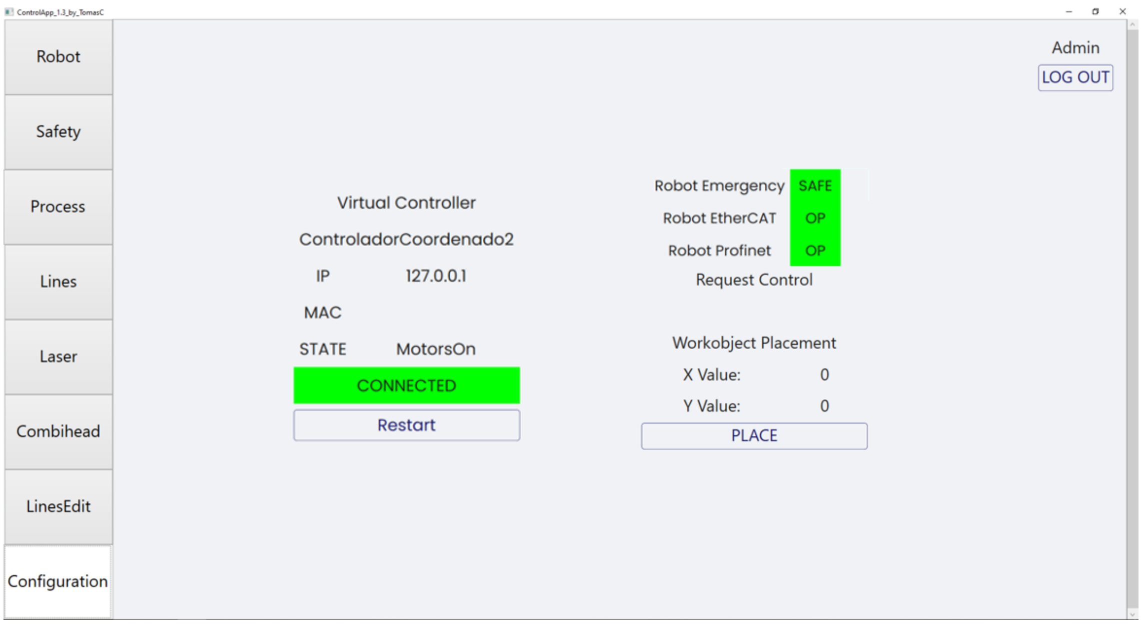


Figure 4.14: Display of the *MainWindow* - *"FileEdit" Tab*

The eighth tab of the *MainWindow*, named *"Configuration"*, is used to display vital information of the selected controller and to verify the state of communication between the components of the robotic cell. This tab is generally used to detect communication malfunctions and to request control of the laser system from the PLC assigned to the nearby robotic system. A useful functionality available in this tab is the ability to edit the position of the workobject associated with the cutting table, giving the user more flexibility when choosing the starting point of the cutting path in the available metal sheet. The restart (*warmstart*) of the selected controller is also possible using this tab's *"Restart"* button. The *"Configuration"* tab is available in the figure 4.15 .



Virtual Controller
ControladorCoordenado2
IP 127.0.0.1
MAC
STATE MotorsOn
CONNECTED

Robot Emergency SAFE
Robot EtherCAT OP
Robot Profinet OP
Request Control

Workobject Placement
X Value: 0
Y Value: 0

Figure 4.15: Display of the *MainWindow - "Configuration" Tab*

To better comprehend the functionalities of each tab it is suggested to read the *User Manual* available in the attachements, which explains extensively the purpose of every component in each tab.

4.5.3 Program Structure

The creation of the control application was inspired by the concepts of modularity and flexibility, consequently the existence of distinct elements, such as models, viewmodels and views, were essential to achieve a well structured application, as it can be seen in figure 4.16. This sub-chapter will discuss the purpose of the different components of the C# application.

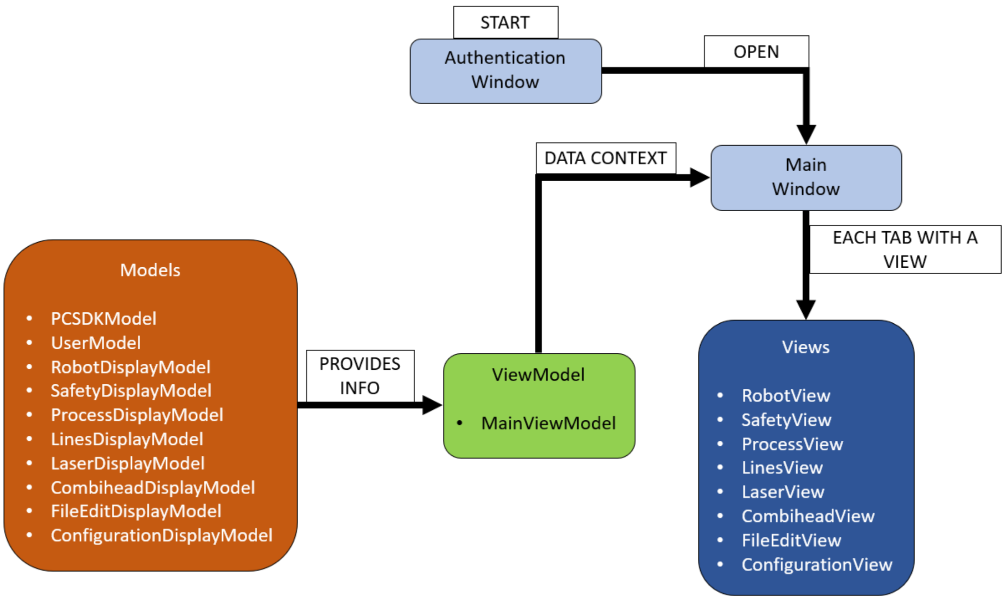


Figure 4.16: Architecture of the C# application

Models

- **PCSDK Model -** This model was created with the purpose to encapsulates all the properties and methods that required the use of ABB's PC SDK functionalities. The use of this model is essential for the interchangeability of robot software in this application, meaning that if this app were to be used by a different brand of industrial robots, most of its components would still be useful.

  Some of the components found in this class are:

  - List of all the controllers available in the network
  - List of all the digital signals necessary in this application
  - Method to start RAPID procedures by moving the program pointer and starting the TASK
  - Method to restart the currently selected controller

- **User Model -** The model is exclusively used to hold the name of the current application user.
- **File Model -** This model has all the directories and titles of the different files being used in this application as well as some methods responsible for editing the files and their directory.

  Some of the components found in this class are:

  - Path to the CSV file used in the testing procedures
  - Title of the CSV file used in the testing procedures
  - Path of the CSV file used in the various functions
  - Title of the CSV file used in the various functions
  - Path to the selected MOD File
  - Title of the selected MOD File
  - Method to determine a new path for the created reports

  The following models are used to define the brushes and strings that are bound to the labels in each tab.

  - Robot Display Model
  - Safety Display Model
  - Process Display Model
  - Lines Display Model
  - Laser Display Model
  - Combihead Display Model
  - FileEdit Display Model
  - Configuration Display Model

ViewModels

MainViewModel - The view model used in this application uses all the information from the declared models, as it can be seen in figure 4.17. This view model is then used as the data context for the *MainWindow*, as it can be seen in figure 4.18, making the binding process possible.

```
public MainViewModel (string ActiveUser)
{
.   user = new UserModel();
.   file = new FileModel();
.   robot_dm = new RobotDisplayModel();
.   safety_dm = new SafetyDisplayModel();
.   process_dm = new ProcessDisplayModel();
.   lines_dm = new LinesDisplayModel();
.   laser_dm = new LaserDisplayModel();
.   combihead_dm = new CombiheadDisplayModel();
.   fileedit_dm = new FileEditDisplayModel();
.   configuration_dm = new ConfigurationDisplayModel();
.   control = new PCSDKModel (user, file, robot_dm, safety_dm, process_dm,
.   lines_dm, laser_dm, combihead_dm, fileedit_dm, configuration_dm);
.
.   user.CurrentUser=ActiveUser;
}
```

Figure 4.17: Usage of all the objects available in the *MainViewModel*

```
public MainWindow (string CurrentUser)
{
.   InitializeComponents();
.   DataContext= new MainViewModel(CurrentUser);
}
```

Figure 4.18: *MainViewModel* as the data context of the *MainWindow*

Views

The different tabs of the *MainWindow* are all displaying different *User Controls*. The *User Controls* are programmed using *XAML* and as such several controls need to be bound to the desired properties, as it can be seen in figure 4.19. The buttons, combo boxes and other interactive controls need to use the *ICommand* interface to be able to call methods from the existing models and viewmodels, as it can be seen in figure 4.20.

```
    <Label Content = "{Binding robot_dm.TCPxaxis}"\>
```

Figure 4.19: Example of binding the content of a label with a property in a model

```
<Button Grid.Row = "4" Content = "J2 NEG">
.   <i:Interaction.Triggers>
.   .   <i:EventTrigger EventName ="PreviewMouseLeftButtonDown">
.   .   .   <i:InvokeCommandAction Command ="{Binding JogCommand}"
.   .   .   CommandParameter ="j2neg"/>
.   .   </i:EventTrigger>
.   .   <i:EventTrigger EventName ="PreviewMouseLeftButtonUp">
.   .   .   <i:InvokeCommandAction Command ="{Binding StopJogCommand}"/>
.   .   </i:EventTrigger>
.   .   <i:EventTrigger EventName ="MouseLeave">
.   .   .   <i:InvokeCommandAction Command ="{Binding StopJogCommand}"/>
.   .   </i:EventTrigger>
.   </i:Interaction.Triggers>
</Button>
```

Figure 4.20: Example of button commands (iCommand)

4.6 Virtual Station

The creation of the virtual representation of the real station was done in *RobotStudio*. This virtual copy is both graphical, due to a visual depiction of the workstation, and true to reality, as it could be directly applied to a real-world station without requiring a great deal of work.

As it was previously mentioned the robot used is the IRB 4600 40Kg 2,55m and the positioner used is the IRBP A-500, which are available in the libraries provided by Robot-Studio. It is only necessary to select the desired components and an appropriate controller to create a base station, which can be seen in fig.4.21.

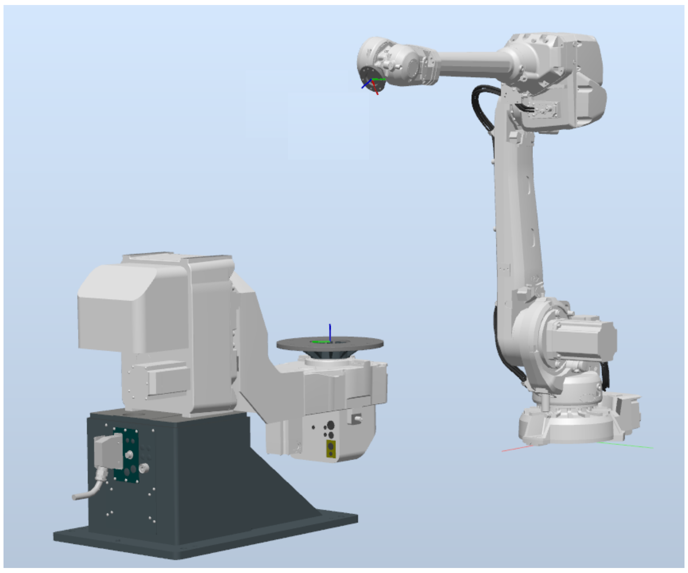


Figure 4.21: Station with ABB IRB 4600 and the IRPA-500

Once the station has been created, it is necessary to import and attach all the tool changing system components to the designated parts of the robotic cell. The male tool changer is permanently attached to the industrial robot's arm and the female tool changer is permanently attached to the laser tool. The male tool changer can also be attached to the female tool changer whenever it is necessary, equipping the robot with a tool. To import these components it is first necessary to get a 3D model of the tool and the tool changing system, which were made available for download by the respective manufacturers. After the models have been downloaded they can be directly imported to the station using the adequate file type *(.SAT)*. In fig.4.22 it is possible to see the male tool changer attached to the robot.

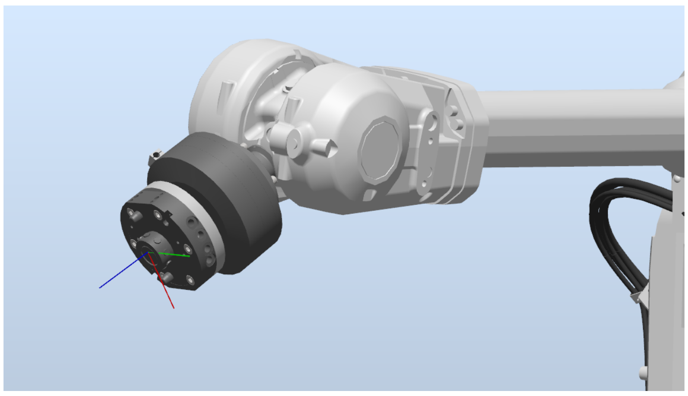


Figure 4.22: Imported male tool changer equipped to the robot

After the tool and the tool changing system are imported, it is necessary to define the frames that will be the tool center point of the male tool changer and the laser tool. For the male tool changer the TCP is defined in the center of the flange that goes into the female tool changer to perform the connection of the two components. The z axis of the male tool changer frame is parallel to the flange. The origin of the *Combihead* associated frame is defined in the tool tip and the z axis of this frame is parallel to the laser beam.

Once the geometry of the tool and the tool changing system have been imported and the necessary frames are defined, it is possible to compile the laser tool. By selecting *RobotStudio's* "*Create a Tool*" option and selecting the necessary geometry and frame, it is possible to create a tool (*tooldata*). However, it is also necessary to define the mass and center of gravity of the tool, which is usually done later by using the procedures already available in the robot controller to calibrate the tool.

Subsequently, it is necessary to import and position the other components of the robotic cell, including the support structure for the robot, the racks to place the tools, the cutting table, the duct holding the necessary cables, the structure to support the duct and other crucial components necessary to achieve a reliable simulation environment. All the mentioned components are fairly easy to include in the virtual environment, only requiring the respective CAD files and the precise positioning to match the real world station. The virtual station after all these components have been inserted can be seen in figure 4.23 .

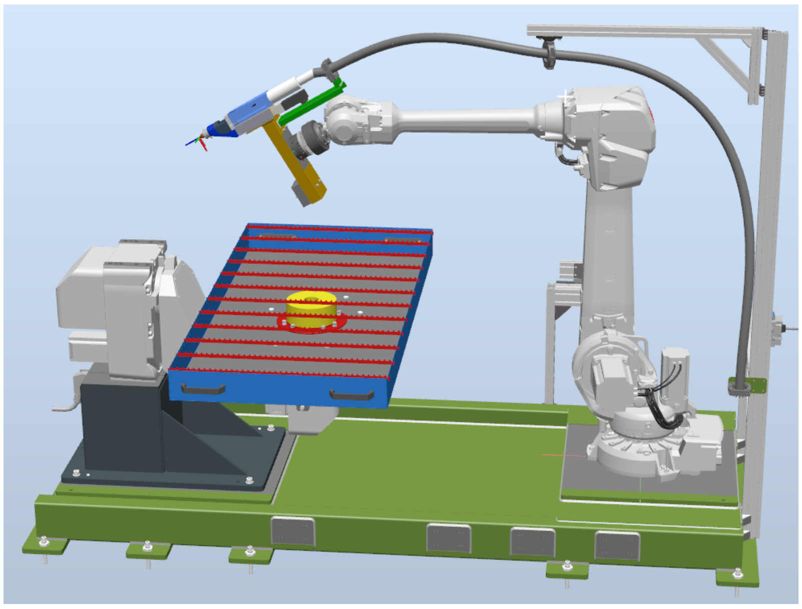


Figure 4.23: Complete virtual station

When the station is completed it is necessary to use smart components for the tool changing simulations, which requires the edition of the "*Station Logic*". To simulate the picking of the laser tool is was used an "*Attacher*", which binds the laser tool and the male tool changer. To separate the components a *Detacher* was used, which unbinds the laser tool and the male tool changer.

These smart components only work with a particular combination of male tool changer and tool, which means that to simulate the picking and placing of a new tool it is necessary to add a new "*Attacher*" and a new "*Detacher*" to the "*Station Logic*" to simulate the functions of the tool changing system.

4.7 *RAPID* Program Structure

The IRC5 controller used in this project has been equipped with several software options, including *Multitasking*, which grants the controller the ability to run several tasks simultaneously. In this project, the use of two task is ideal, as one task is responsible for the robot's movements, the other task is responsible for keeping track of the TCP position, the TCP velocity and the robotic joint's values. In the figure 4.24 it is explained the architecture of the *RAPID* program used in this project.

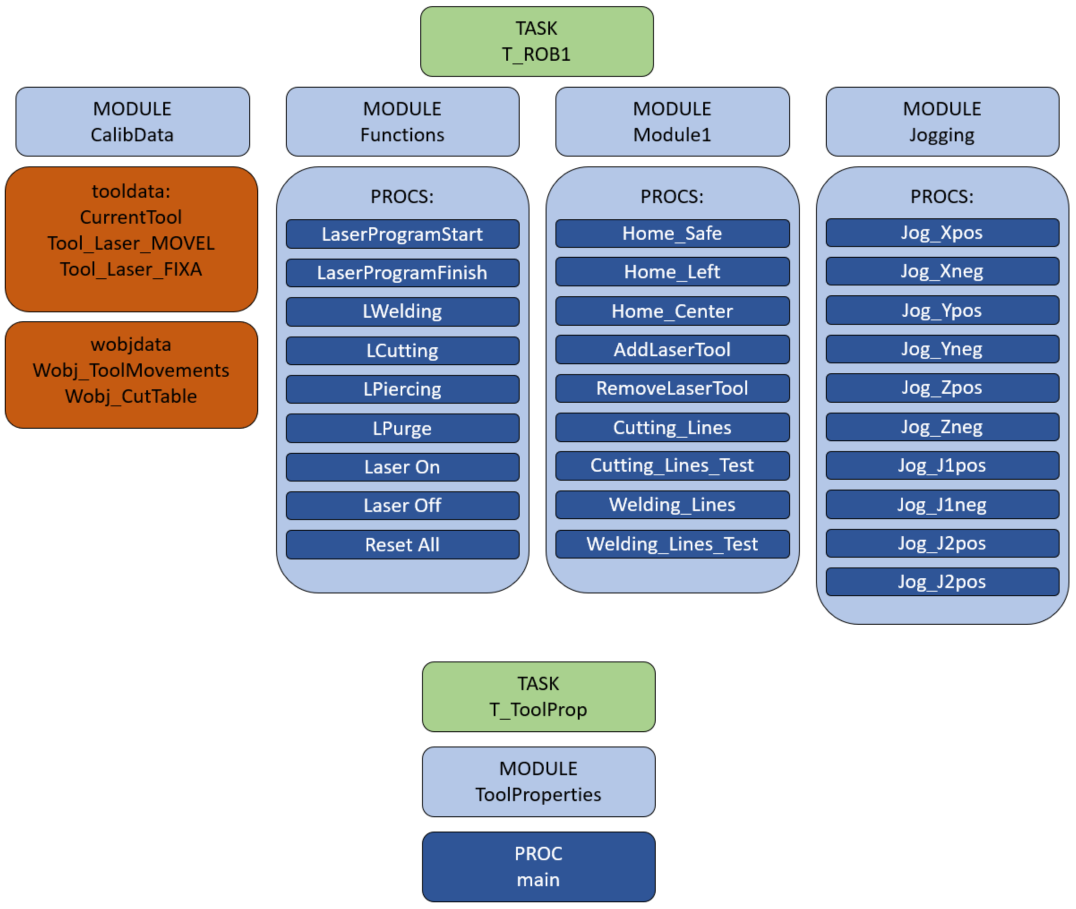


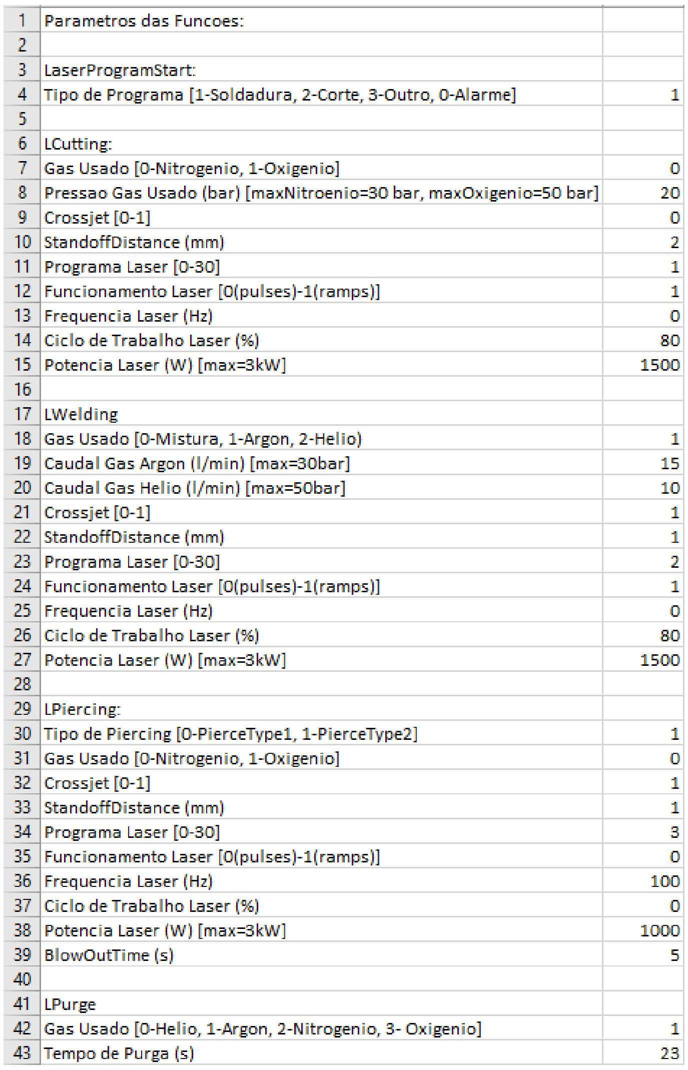
Figure 4.24: Elements of the *RAPID* program structure

TASK T_ROB1

MODULE *CalibData* - This module was used to declare the *tooldata* and the *wobjdata* necessary for the robot to complete all the necessary movements.

MODULE *Functions* - This module was used to declare several functions capable of controlling the various components of the robotic cell, including the laser system, the *Combihead* and the gas panel. Some examples of these functions are *LCutting* and *ResetAll*

PROC *LCutting* - Responsible for defining and activating several process parameters necessary for the cutting procedure, such as the gas used, the gas pressure, the distance from the nozzle to the workpiece and others. The parameters necessary in this functions and several others, including *StartLaser*, *LWelding*, *LPiercing* and *LPurge*, are kept in a CSV file, which must be selected prior to the start of the procedure. The template for the mentioned CSV file can be seen in figure 4.25 .



1	Parametros das Funcoes:	
2		
3	LaserProgramStart:	
4	Tipo de Programa [1-Soldadura, 2-Corte, 3-Outro, 0-Alarme]	1
5		
6	LCutting:	
7	Gas Usado [0-Nitrogenio, 1-Oxigenio]	0
8	Pressao Gas Usado (bar) [maxNitroenio=30 bar, maxOxigenio=50 bar]	20
9	Crossjet [0-1]	0
10	StandoffDistance (mm)	2
11	Programa Laser [0-30]	1
12	Funcionamento Laser [0(pulses)-1(ramps)]	1
13	Frequencia Laser (Hz)	0
14	Ciclo de Trabalho Laser (%)	80
15	Potencia Laser (W) [max=3kW]	1500
16		
17	LWelding	
18	Gas Usado [0-Mistura, 1-Argon, 2-Helio)	1
19	Caudal Gas Argon (l/min) [max=30bar]	15
20	Caudal Gas Helio (l/min) [max=50bar]	10
21	Crossjet [0-1]	1
22	StandoffDistance (mm)	1
23	Programa Laser [0-30]	2
24	Funcionamento Laser [0(pulses)-1(ramps)]	1
25	Frequencia Laser (Hz)	0
26	Ciclo de Trabalho Laser (%)	80
27	Potencia Laser (W) [max=3kW]	1500
28		
29	LPiercing:	
30	Tipo de Piercing [0-PierceType1, 1-PierceType2]	1
31	Gas Usado [0-Nitrogenio, 1-Oxigenio]	0
32	Crossjet [0-1]	1
33	StandoffDistance (mm)	1
34	Programa Laser [0-30]	3
35	Funcionamento Laser [0(pulses)-1(ramps)]	0
36	Frequencia Laser (Hz)	100
37	Ciclo de Trabalho Laser (%)	0
38	Potencia Laser (W) [max=3kW]	1000
39	BlowOutTime (s)	5
40		
41	LPurge	
42	Gas Usado [0-Helio, 1-Argon, 2-Nitrogenio, 3- Oxigenio]	1
43	Tempo de Purga (s)	23

Figure 4.25: Example of a CSV file with the process parameters necessary to start a custom procedure

The methodology to activate the necessary process parameters whenever a function is called is available in the figure 4.26 . The button to start a custom *RAPID* procedure is pressed in the C# application. The selected procedure is a laser cutting operation, which will require the use of the *LCutting* function. When the *LCutting* function is started the digital signal *"OnCuttingStart"* is set to true. The C# application will detect this variation and get the values of the necessary process parameters from the previously selected CSV file and assign them to several *RAPID* variables. After the respective *RAPID* variables have been assigned a value, the program pointer will continue to move though the function and the controller signals, which are responsible for the activation and deactivation of the process parameters, will be assigned the values of the respective *RAPID* variables. When all the required signals have been activated it is necessary to guarantee that the gases have reached the tool nozzle before turning on the laser beam. Therefore, it is necessary to wait for the digital signal *"GasReady"* to be set to true, which indicates that the procedure is being supplied with the necessary gases.

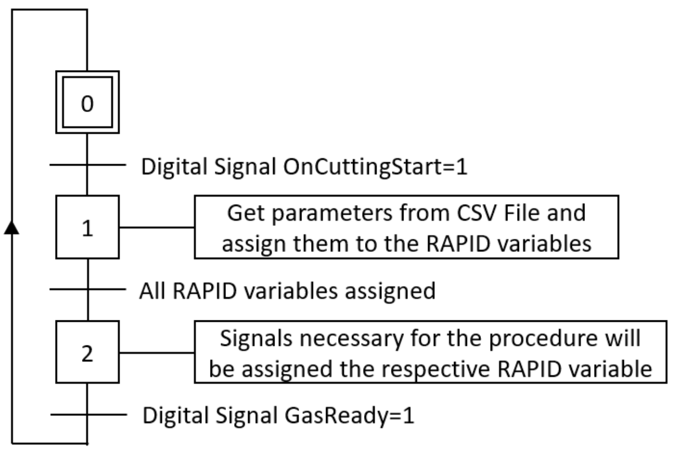

0
Digital Signal OnCuttingStart=1
1
Get parameters from CSV File and assign them to the RAPID variables
All RAPID variables assigned
2
Signals necessary for the procedure will be assigned the respective RAPID variable
Digital Signal GasReady=1

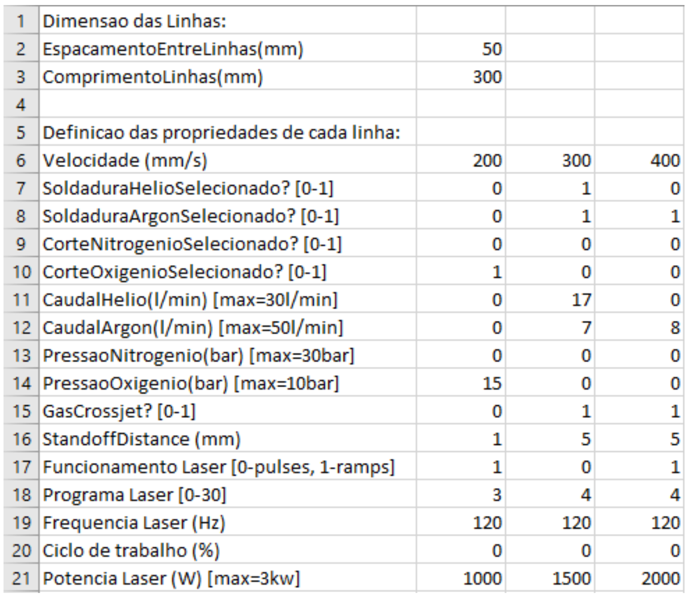
Figure 4.26: Grafcet of the activation of the process parameters previously selected in the CSV file for the *LCutting* function

PROC *ResetAll* - The purpose of this functions is to simultaneously reset all the necessary signals to guarantee the safety of the robotic station. The reset of signals in this function stop all the gasses from flowing and prevents the the laser system from firing the laser beam. The *RAPID* variables are also reset to prevent possible complications during procedure's restarts.

MODULE *Module1* - This module is used to declare several essential procedures to command the robot movements. The procedures consist of attaching and detaching the laser tool to the robotic arm, moving the robot's TCP to one of the three *HOME* positions and starting one of the testing procedures (lines), where different procedure parameters can be tested in laser cutting/welding.

An important procedure to analyse is the *"Cutting_Lines_H"*, which is a cutting procedure performed on a metal sheet laid out on the cutting table. Several parallel lines with different process parameters are created and analyzed, with the purpose of optimizing the quality of the results. The parameters are kept in a CSV file, which is selected prior to start of the procedure. Other procedures in this module where the parameters must be defined in a CSV file are *"Welding_Lines"*, Cutting_Lines_CTubes_L() and Cutting_Lines_CTubes_T().

PROC *Cutting_Lines_H* - All the parameters necessary for this procedure are defined in a CSV file, which is available in figure 4.27. The parameters include the length of the lines being created and the spacing between them. The number of rows being done is determined by the number of rows of properties defined in the CSV file, if there are three rows of properties then three lines will be cut into the metal sheet. At the end of the first line a report is created defining the starting time of the procedure, the current user of the control application and the process parameters being used. Every subsequent line of the same procedure will simply add a new array of the current process variables to the report. The methodology to test different process parameters by creating short parallel lines is explained in the figure 4.28 .



1	Dimensao das Linhas:			
2	EspacamentoEntreLinhas(mm)	50		
3	ComprimentoLinhas(mm)	300		
4				
5	Definicao das propriedades de cada linha:			
6	Velocidade (mm/s)	200	300	400
7	SoldaduraHelioSelecionado? [0-1]	0	1	0
8	SoldaduraArgonSelecionado? [0-1]	0	1	1
9	CorteNitrogenioSelecionado? [0-1]	0	0	0
10	CorteOxigenioSelecionado? [0-1]	1	0	0
11	CaudalHelio(l/min) [max=30l/min]	0	17	0
12	CaudalArgon(l/min) [max=50l/min]	0	7	8
13	PressaoNitrogenio(bar) [max=30bar]	0	0	0
14	PressaoOxigenio(bar) [max=10bar]	15	0	0
15	GasCrossjet? [0-1]	0	1	1
16	StandoffDistance (mm)	1	5	5
17	Funcionamento Laser [0-pulses, 1-ramps]	1	0	1
18	Programa Laser [0-30]	3	4	4
19	Frequencia Laser (Hz)	120	120	120
20	Ciclo de trabalho (%)	0	0	0
21	Potencia Laser (W) [max=3kw]	1000	1500	2000

Figure 4.27: Example of a CSV file with the process parameters and the lines properties necessary to start a testing procedure

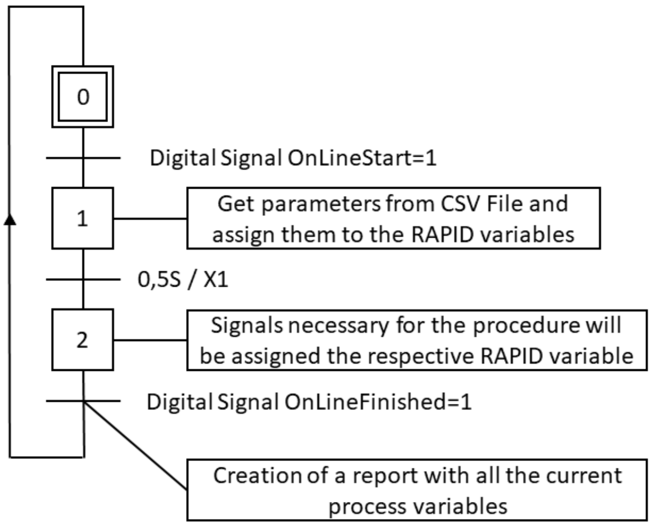

0
Digital Signal OnLineStart=1
1
Get parameters from CSV File and assign them to the RAPID variables
0,5S / X1
2
Signals necessary for the procedure will be assigned the respective RAPID variable
Digital Signal OnLineFinished=1
Creation of a report with all the current process variables

Figure 4.28: Grafcet of the testing procedures (lines)

MODULE *Jogging* - This module's purpose is to define all the procedures responsible for the robot and positioner jogging. When a jogging procedure is started a designated digital signal will also be set to true and will remain true until the left mouse button is let go. While the designated digital signal is set to true, the robot will continue the desired movement; however, when the digital signal is reset the robot will immediately stop.

When one of the buttons, from the C# application, is pressed and quickly released the robot will move two millimetres in the desired direction; however, the positioner will rotate the desired joint 5 degrees. The methodology to jog of the robotic components is available in figure 4.29

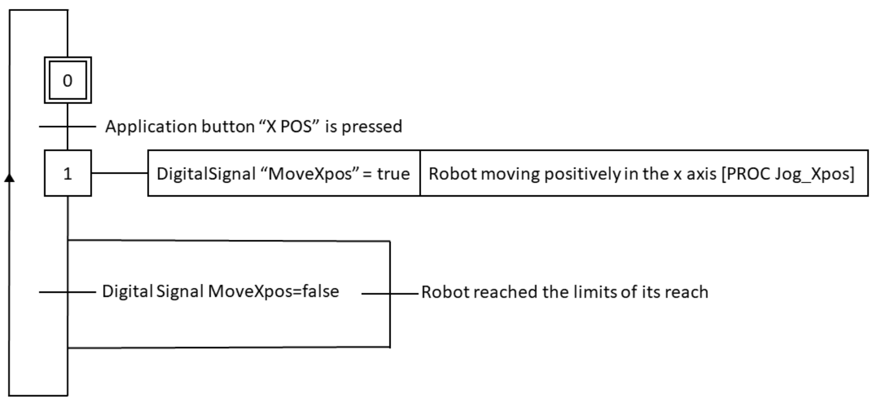

0
Application button "X POS" is pressed
1
DigitalSignal "MoveXpos" = true
Robot moving positively in the x axis [PROC Jog_Xpos]
Digital Signal MoveXpos=false
Robot reached the limits of its reach

Figure 4.29: Grafcet of the robot jogging procedures

TASK T_ToolProp

MODULE *ToolProperties* - Each task of the robot controller only has one program pointer, which means only one procedure can run simultaneously inside the same task. The task *T_ROB1* is responsible for robot movements and the control of the various elements of the robotic station. The task *T_ToolProp* runs simultaneously with the task *T_ROB1* and its purpose is to continuously determine, at a high frequency, the position and the velocity of the robot's TCP as well as the values of the robotic components joints. The information from this module is used in several *RAPID* variables and displayed in the C# application.

In the next chapter it will be discussed the results from the initial testing procedures. The problems that were encountered will be identified and the solutions will be reported.

Chapter 5

Results

This chapter will discuss the results from the initial testing procedures, including the problems encountered and their respective solutions. It is extremely important to enlighten that the quality of the following test's results is not optimized; instead, the values of the process variables are chosen to simply guarantee that the metal sheet is cut.

5.1 Laser Test

The laser test aims to verify if the laser system is working as expected. A procedure was created to select the laser program and power and fire the laser; however, the selected laser program limits the time the laser is turned on to 30 ms to prevent malfunctions that would damage the robotic station. The laser nozzle is aimed at a small piece of paper that has a small hole after the procedure was completed, as it can be seen in figure 5.1, indicating the laser has fired. Given the fact that the laser worked without complications this test was considered a success and granted the opportunity to perform more complex procedures.

LaserPower=200W LaserProgram=7

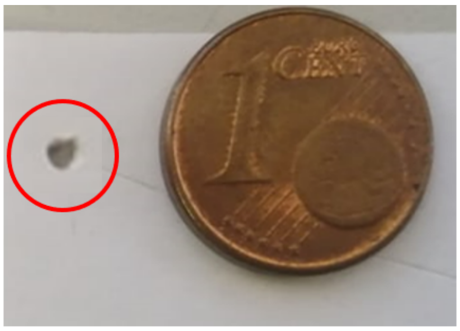


Figure 5.1: Results from the laser system test with a 30ms discharge

5.2 Path Test

The first path test consists of turning on the laser during a robotic path, that can be seen in figure 5.2, creating a laser cut in a metal sheet.

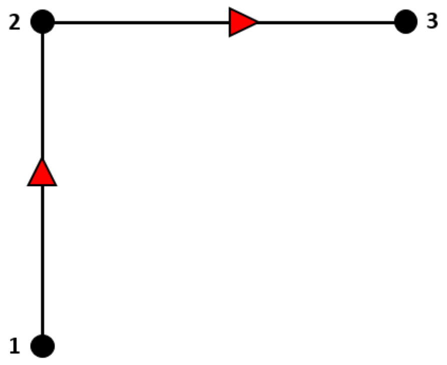


Figure 5.2: Path selected for the first path test - 90º curve trajectory

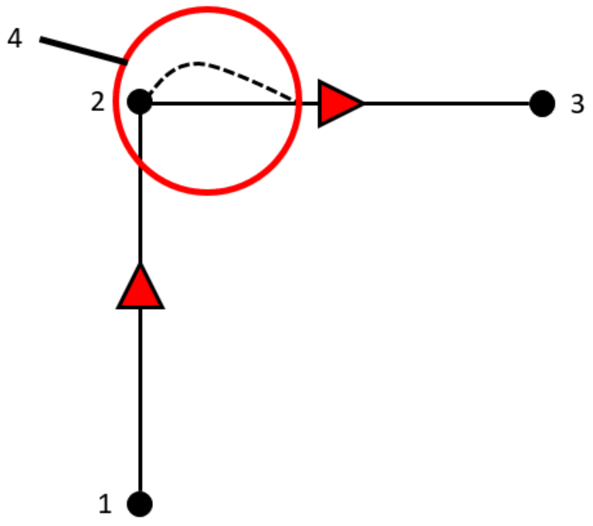


Figure 5.3: Results from the first path test - 90º curve trajectory

Analysing a scheme of the testing results, which can be seen in figure 5.3, it is possible to verify that a change of direction is causing a significant divergence between the robot movement and the programmed trajectory. The divergence is identified by the number 4 in figure 5.3 and its cause is the miscalibration of the the *Combi-head*. After successfully calibrating the laser tool it was possible to attain far better results, as it can be seen in the following tests.

The second path test consists of turning on the laser while the robot performs a zig zag pattern, which can be seen in figure 5.4. The purpose of this test is to verify if the proper calibration of the laser tool solved the discrepancy between the programmed path and the real trajectory in corners. Analysing the results, available in figure 5.5, it is possible to conclude that the proper calibration of the laser tool minimizes undesirable vibrations of the laser tool, leading to higher quality results when changing direction.

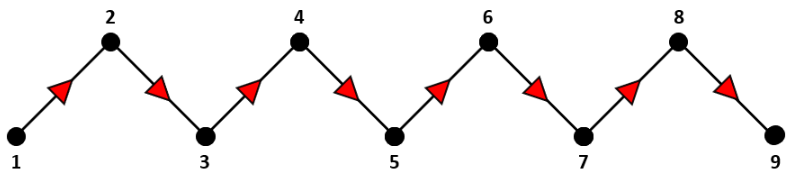


Figure 5.4: Path selected for the second path test - zig zag trajectory

LaserPower=300W LaserProgram=7 TCPspeed=20mm/s

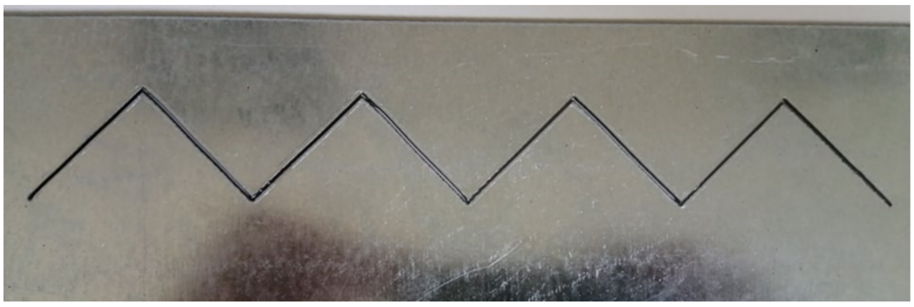
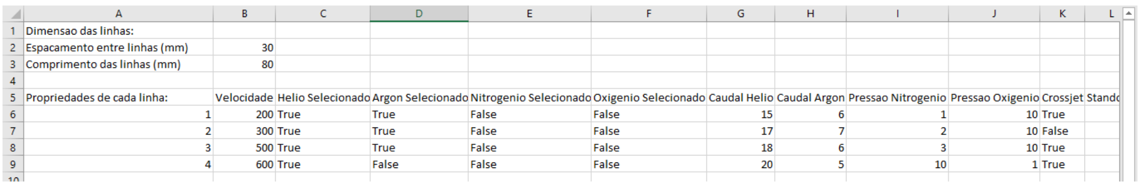


Figure 5.5: Results from the secong path test - zig zag trajectory

5.3 Line Test for Process Parametrization

The line test focuses on the creation of parallel cuts using the methodology to vary process parameters to determine the optimized parameters and achieve higher quality results. The test consists in positioning the tool perpendicular and close to the cutting table, selecting the CSV file available in figure 5.6 and starting the adequate procedure to create parallel lines on a flat surface. The test results are available in figure 5.7 .



	A	B	C	D	E	F	G	H	I	J	K	L
1	Dimensao das linhas:											
2	Espacamento entre linhas (mm)	30										
3	Comprimento das linhas (mm)	80										
4												
5	Propriedades de cada linha:	Velocidade	Helio Selecionado	Argon Selecionado	Nitrogenio Selecionado	Oxigenio Selecionado	Caudal Helio	Caudal Argon	Pressao Nitrogenio	Pressao Oxigenio	Crossjet	Standc
6	1	200	True	True	False	False	15	6	1	10	True	
7	2	300	True	True	False	False	17	7	2	10	False	
8	3	500	True	True	False	False	18	6	3	10	True	
9	4	600	True	False	False	False	20	5	10	1	True	

Figure 5.6: Data of the CSV file selected for the first line test

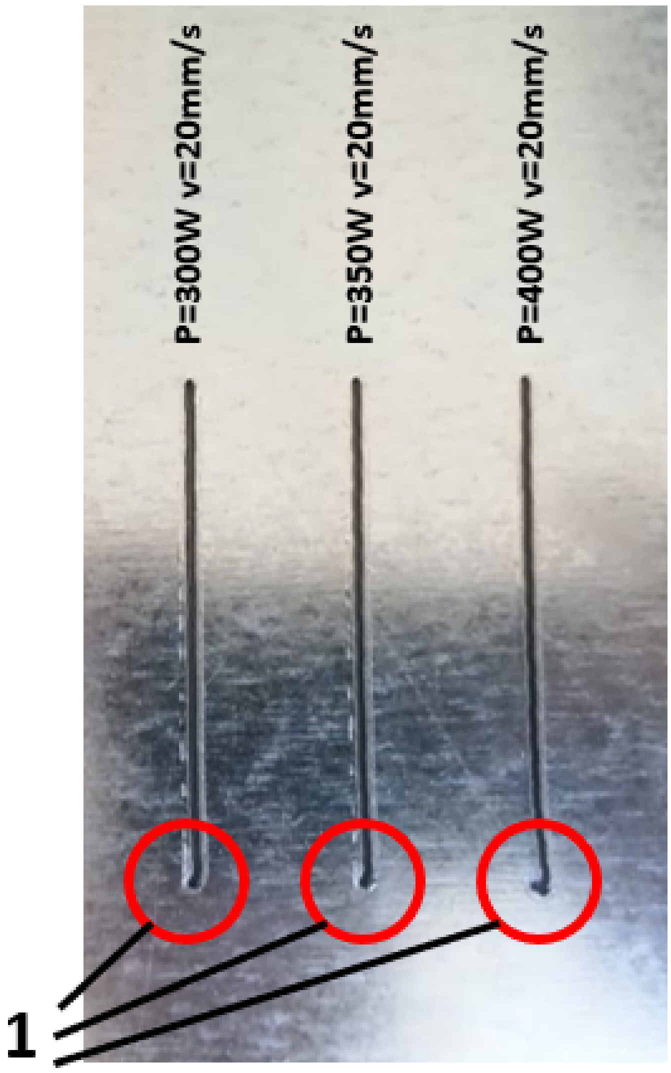


Figure 5.7: Results from the first line test - horizontal lines

Analysing the results from the first line test, it is clear that every line has an initial burr, that can be identified by the number 1 in figure 5.7. A hypothesis for this problem is that the gear of the number one joint of the industrial robot is changing direction abruptly when the lines start, causing noticeable inconsistencies when following the programmed path. To test this hypothesis a new path was created that only turns on the laser after the gear of the number one joint is already rotating in the desirable direction. A comparison of the two paths used in this testing procedures can be seen in figure 5.8 .

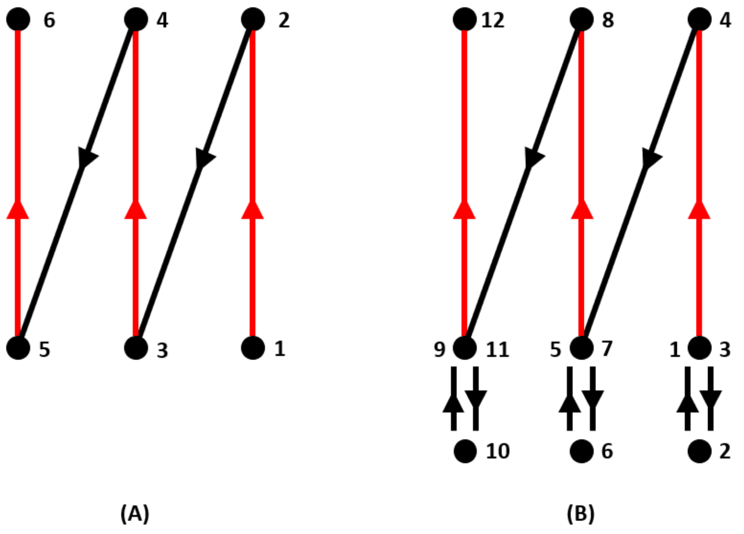


Figure 5.8: Comparison between the old (A) and the new (B) path to test laser cutting properties on the cutting table

The results from the second line test with the new path are available in figure 5.9 and it is possible to establish that the burr at the beginning of each line disappeared. Given the results from this test it is possible to determine that the backlash resulting from the abrupt inversion of the number one joint causes significant irregularities and must be taken into consideration when creating future laser procedures.

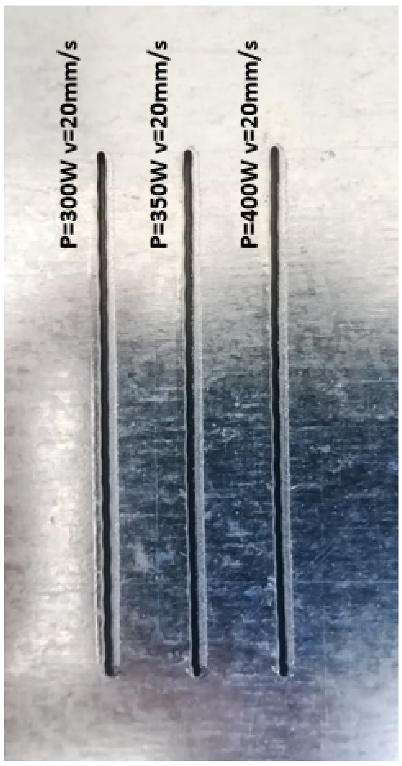


Figure 5.9: Results from the second line test - horizontal lines with new path

The third line test aims to analyse the methodology of creating parallel lines in a circular tube. There are two procedures to cut parallel lines in a circular tube, in the first one, the lines created are parallel to the axis of the tube (longitudinal lines) and in the second one the lines created are perpendicular to the axis of the tube (transversal lines). The procedure that creates the longitudinal lines was the one being tested and the results are available in figure 5.10 .

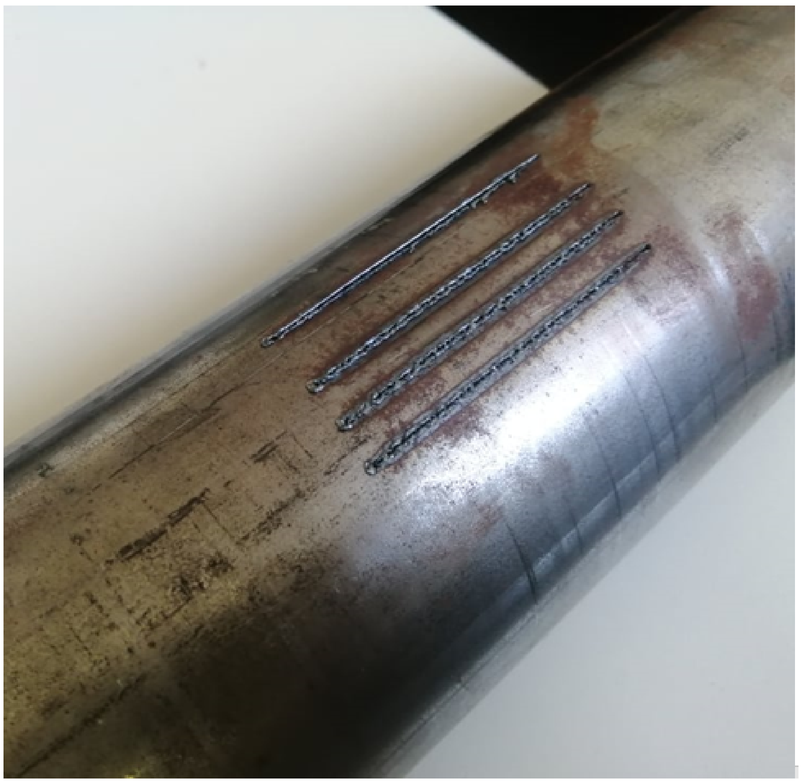


Figure 5.10: Results from the third line test - Laser cutting of a circular tube

Examining the results from the third line test, it is possible to verify that the low laser power, approximately 350 W, used on all the previous tests was insufficient to penetrate the walls of the cylindrical tube, which are noticeably thicker than the metal sheet previously used. Although the tube walls were not completely cut during this procedure, the test was considered a success because the lines were correctly placed and an increase of laser power would certainly solve the penetration issue.

5.4 Part Test

The part test aims to explore the methodology of importing a custom procedure to the robot controller and starting it using the C# application. The module containing all the targets, functions and move instructions necessary to create the custom part, available in figure 5.11, was developed using *RobotStudio*, with the virtual station. The MOD file created and the CSV file necessary for the functions are selected in the C# application and the desired procedure is started. The results of this test can be seen in figure 5.12.

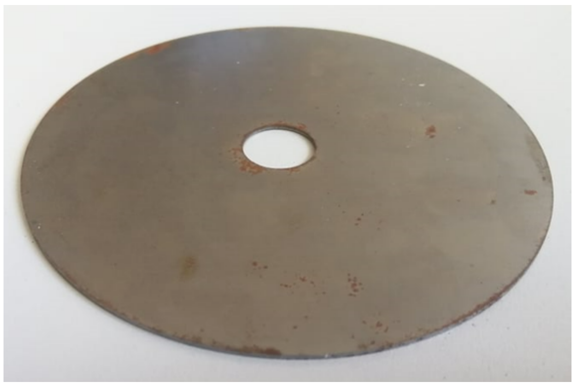


Figure 5.11: Proposed part to recreate using the laser system

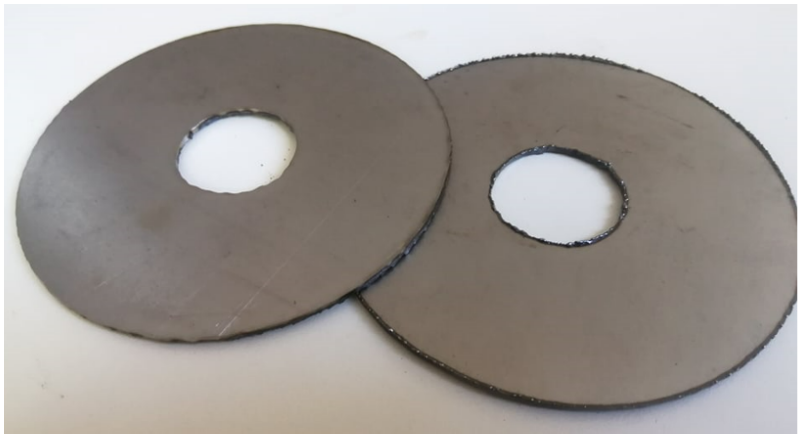


Figure 5.12: Results from the part test - Laser cutting of a custom part

Analysing the results from the part test, it is possible to verify that final product is identical to the proposed part. The quality of the final product is not ideal; however, with more testing the process parameters can be optimized and parts with far greater quality can be produced using this system. It is important to enlighten that the possibility of editing the workobject associated to the cutting table was also tested during these procedures. The figure 5.13 represents the position and orientation of the workobject associated to the cutting table; however, if it is necessary to modify the origin of this workobject for the programmed path to have a different starting point it is only necessary to use the C# application. The C# application will start a procedure that replaces the x and y value of the workobject origin with the current x and y value of the TCP. In figure 5.14 the number 1 identifies the position of the workobject used to program the path and the number 2 and 3 identify the position of the edited workobject and their respective paths.

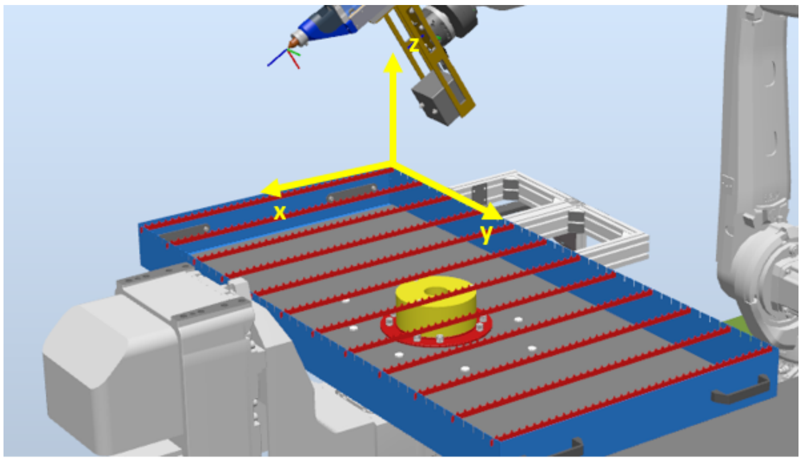


Figure 5.13: Position and orientation of the workobject used to program paths in the virtual station

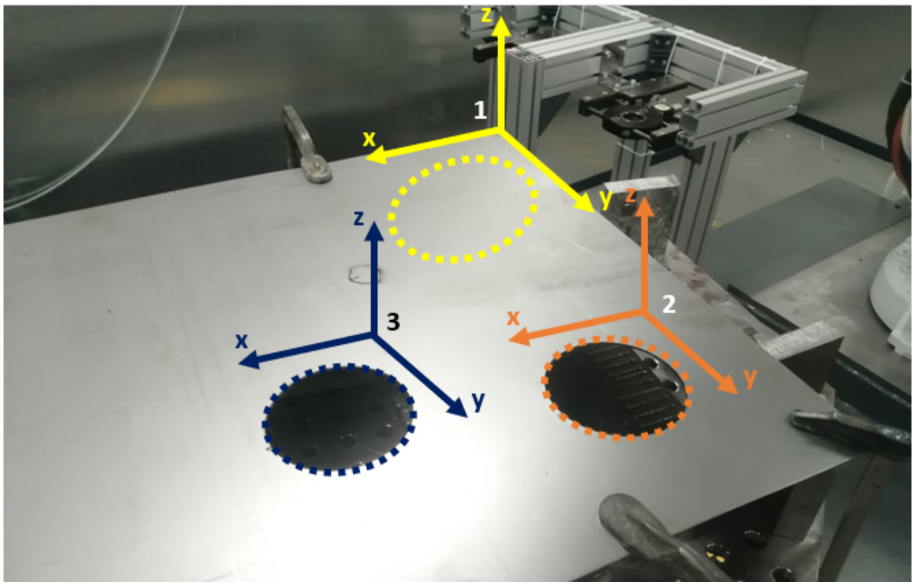


Figure 5.14: Position and orientation of the workobjects used to create the custom pieces

The next chapter will consist of the conclusions attained from this project and the future work suggested to achieve a system with better performance and overall higher quality.

Chapter 6

Conclusion

6.1 Conclusions

The dissertation reports the development of a virtual production cell, that simulates the paths generated for the desired workpieces, and a control application to manage the robot and the rest of the equipment. With an accurately built digital twin setup it is expected a great correlation between the simulation environment and the real system, enabling engineers to better prepare and reduce the development time.

To generate the paths for the industrial robot several software options were considered, including *DXF2GCODE*, *fusion360*, *Deepnest*, *RobotStudio* and others. Most options were either too expensive or simply not optimized for laser processing; therefore, it was decided that the trajectories were initially going to be created using *RobotStudio* basic functionalities, such as "AutoPath". It is possible to import a 3D representation of an object to the virtual environment and create the necessary paths to reproduce that custom part. Since different workpieces will require different sets of properties, a human machine interface (HMI) was designed to provide the user with a higher level of control over the entire process, facilitating the detection of problems.

The proposed solutions are compatible with the virtual environments of ABB simulation software, indicating the alignment of the development of this robotic cell with most of the currently developed laser processing stations. One of the most important aspects of a material laser processing system using an industrial robot is the path planning, which is not ideal in this station, since most trajectories will not be automatically generated. Ideally, the desired part would be selected and a trajectory would be automatically generated, taking into consideration all aspects of material laser processing, including piercing and hole cutting order, minimizing the work needed to create a new procedure.

The development of the control application has meaningfully impacted the flexibility of this system, since it allowed for the selection of the robotic path and the process parameters separately, which can be very useful in a variety of situations. The C# application has several other functionalities, including the constant monitoring of key variables, the remote start of fundamental procedures and more; however, the possibility of rapidly selecting a set process parameters and starting a testing procedure is extremely useful to determine the optimized conditions for each workpiece, which is expected to occur multiple times with this project.

6.2 Future Work

While working on this project there were several objectives set out to achieve, and while most of them were reached, some of them, due to a lack of time or knowledge, were not completed. In the following list various useful tasks will be registered for the future of this project.

- Creation of a tab, in the C# application, to edit the CSV files containing the process parameters for the functions used in the custom procedures. This tab would look and work similarly to the tab responsible for editing the CSV files necessary for the testing procedures.
- Parameterization of a workpiece, to achieve the best possible results when using a workpiece with identical properties (material, thickness, etc.).
- Sharing of the system's crucial variables with a Siemens's IIoT to display the information in a dashboard, granting a better understanding of the entire system.
- Testing of more complex cutting/welding procedures.
- Complete all the necessary steps to add a new tool to this robotic cell. It is also important to enlighten that the new tool would also need to be added and properly configured in the virtual environment.
- Creation of a database to keep the results from all the testing procedures to organize the parameters based on material type, supplier and geometry.
- Addition of the necessary sensors to acquire real-time data to better analyse the process parameters.

References

[1] P Hourd, M Robinson, L Bolton, K Cichomska, and J Baldwin. The factory of the future: A study for the Government Office for Science. 2013.

[2] Christopher Müller. Agenda. Technical report.

[3] Mohsen Moghaddam and Shimon Y. Nof. The collaborative factory of the future. *International Journal of Computer Integrated Manufacturing*, 30(1):23–43, jan 2017. ISSN 13623052. doi: 10.1080/0951192X.2015.1066034.

[4] Horizon europe-work programme 2021-2022 digital, industry and space. URL `https://ec.europa.eu/info/funding-tenders/opportunities/docs/2021-2027/horizon/wp-call/2021-2022/wp-7-digital-industry-and-space_horizon-2021-2022_en.pdf`. (visited 3/9/2021).

[5] Mike Shafto. Draft modeling, sinulation, information technology processing roadmap technology area 11. 2010. URL `https://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf`. (visited 3/9/2021).

[6] Digital twin – a key software component of Industry 4.0 | ABB, . URL `https://new.abb.com/news/detail/11242/digital-twin-a-key-software-component-of-industry-40`. (visited 9/6/2021).

[7] Digital twin applications - What is new | ABB, . URL `https://new.abb.com/control-systems/features/digital-twin-applications`. (visited 9/6/2021).

[8] Roland Rosen, Georg Von Wichert, George Lo, and Kurt D. Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. In *IFAC-PapersOnLine*, volume 28, pages 567 – 572. Elsevier, may 2015. doi: 10.1016/j.ifacol.2015.06.141.

[9] Daniel Fernando. Utilização do software robotstudio na programação off-line de células robóticas de quinagem. 2016. URL `https://repositorio-aberto.up.pt/bitstream/10216/85116/2/139402.pdf`. (visited 3/9/2021).

[10] Digital Twin | Digital Twin | Siemens Global. URL `https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/digital-twin.html`. (visited 15/6/2021).

[11] SOLO - ElectraMeccanica. URL `https://electrameccanica.com/product/solo-reservation/`. (visited 16/6/2021).

[12] SINUMERIK ONE Media Center | SINUMERIK ONE | Siemens Global. URL `https://new.siemens.com/global/en/products/automation/systems/sinumerik-one/sinumerik-one-media-center.html`. (visited 17/6/2021).

[13] What a Digital Twin can do – and what not – KUKA BLOG, . URL https://www.blog.kuka.com/2020/07/23/what-a-digital-twin-can-do-and-what-not/?lang=en. (visited 16/6/2021).

[14] KUKA.Sim – simulation software | KUKA AG, . URL https://www.kuka.com/pt-pt/produtos-servi{ç}os/sistemas-de-rob{ô}/software/planejamento-proje{ç}{~{a}}o-service-seguran{ç}a/kuka_sim. (visited 16/6/2021).

[15] What's the difference between offline programming and simulation? - robodk blog. URL https://robodk.com/blog/difference-simulation-offline-programming/. (visited 3/9/2021).

[16] Carlos Ye Zhu. A novel multi-brand robotic software interface for industrial additive manufacturing cells. 2020. URL https://www.emerald.com/insight/0143-991X.htm. (visited 3/9/2021).

[17] Frank Schneider Dr. Dirk Petring. One machine does it all for laser beam welding and cutting. 2009. URL https://www.researchgate.net/publication/291810544_One_Machine_Does_It_All_for_Laser_Beam_Welding_and_Cutting. (visited 9/6/2021).

[18] Dr. Dirk Petring. Laser cutting and welding with one tool. *The Fabricator*, 2015. URL https://www.thefabricator.com/thefabricator/article/lasercutting/laser-cutting-and-welding-with-one-tool. (visited 3/9/2021).

[19] Complex 3D laser cutting | PRECITEC, . URL https://www.precitec.com/laser-cutting/products/laser-cutting-head/solid-cutter/. (visited 17/6/2021).

[20] Focusing optics YW30/YW52 Laser welding | PRECITEC, . URL https://www.precitec.com/laser-welding/products/processing-heads/yw30/. (visited 17/6/2021).

[21] China robotic arm fiber laser 3d cutting machine factory and manufacturers | goldenlaser. URL https://www.goldenfiberlaser.com/robotic-arm-fiber-laser-3d-cutting-machine.html. (visited 3/9/2021).

[22] Abb Robotics. Product specification - IRB 4600. Technical report, 2009. URL https://library.e.abb.com/public/1548bd38cd0b455b8bc37de07bcb319f/3HAC032885%20PS%20IRB%204600-en.pdf. (visited 9/6/2021).

[23] O. Safety and H. Administration. OSHA Technical Manual. Technical report.

[24] D. Taslakova. Positioning accuracy and repeatability of a class of technological robots. *PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS*, pages 99 – 105, 1997.

[25] IRB 4600 | ABB Robotics - Industrial Robots | ABB Robotics, . URL https://new.abb.com/products/robotics/industrial-robots/irb-4600. (visited 10/6/2021).

[26] ABB Robotics. Product specification IRBP D2009. page 202, 2009. URL https://search.abb.com/library/Download.aspx?DocumentID=3HAC038208-001&LanguageCode=en&DocumentPartId=&Action=Launch.

[27] IRBP A - ABB Robotics - Workpiece Positioners (Robot Equipment and Accessories | ABB Robotics), . URL https://new.abb.com/products/robotics/application-equipment-and-accessories/workpiece-positioners/irbp-a. (visited 10/6/2021).

[28] ABB Robotics. IRC5 Industrial Robot Controller, data sheet, PDF, 2019. URL https://search.abb.com/library/Download.aspx?DocumentID=ROB0295EN&LanguageCode=en&DocumentPartId=&Action=Launch. (visited 10/6/2021).

[29] Leader in laser solutions and photonics technology | coherent. URL https://www.coherent.com/. (visited 10/6/2021).

[30] Laser Cutting | Rofin FL Series. URL https://www.medicalsearch.com.au/laser-cutting-rofin-fl-series/p/138589. (visited 10/6/2021).

[31] Rofin Coherent. Combi-head f2y for alternating laser beam welding and cutting in any order, operating instructions.

[32] LaserFact. Rofin fl030 fiber laser, operating manual.

[33] Precitec KG. *Adjust Box EG8110 Lasermatic Z | Operating Instructions.* Precitec, 2011.

[34] Beckhoff. CX2020 | Basic CPU module, 2017. URL https://www.beckhoff.com/english.asp?embedded_pc/cx2020.htm. (visited 11/6/2021).

[35] Beckhoff. TExxxx | TwinCAT 3 Engineering | Beckhoff USA, . URL https://www.beckhoff.com/en-us/products/automation/twincat/te1xxx-twincat-3-engineering/?gclid=CjwKCAjwqvyFBhB7EiwAER786c0SA9GGBriyP4gpSXCYMoLm2crqxyE5K1g0HPcV7aaXf_GSHAMH4RoC9d0QAvD_BwE. (visited 11/6/2021).

[36] Windows Embedded Compact 7 | Microsoft. URL http://www.microsoft.com/windowsembedded/en-us/windows-embedded-compact-7.aspx. (visited 11/6/2021).

[37] Zimmer Group. Home - Zimmer Group, . URL https://www.zimmer-group.com/en/. (visited 11/6/2021).

[38] Zimmer Group. Series HWR - Zimmer Group, . URL https://www.zimmer-group.com/en/technologies-components/components/handling-technology/tool-changer/manual/series-hwr. (visited 11/6/2021).

[39] Zimmer Group. CSR100 - Zimmer Group, . URL https://www.zimmer-group.com/en/technologies-components/components/handling-technology/collision-protection/csr/products/csr100. (visited 11/6/2021).

[40] TRUMPFtube. TRUMPF laser cutting: Smart collision prevention - For worry-free production - YouTube, 2014. URL https://www.youtube.com/watch?v=SG8h1Ykf1lc&ab_channel=TRUMPFtube. (visited 11/6/2021).

[41] URL https://www.demmeler.com/fileadmin/user_upload/6-Downloads/EN/DEMMELER_3D_Welding_Tables.pdf. (visited 3/9/2021).

[42] Es, size 250, 4 jaws, din 6351, form a, steel body - röhm eshop. URL https://eshop247.roehm.biz/DE-en/es-size-250-4-jaws-din-6351-form-a-steel-body-788.html?tab=product-variants-tab. (visited 3/9/2021).

[43] triflex® r trcf series. URL https://www.igus.com/product/20124?artNr=TRCF-65-100-0. (visited 3/9/2021).

[44] EtherCAT. EtherCAT Technology Group. URL http://www.ethercat.org/. (visited 11/6/2021).

[45] Beckhoff. CX8100 | Embedded PC series (compact controller) | Beckhoff Worldwide, . URL https://www.beckhoff.com/en-en/products/ipc/embedded-pcs/cx8100-arm-cortex-a9/. (visited 11/6/2021).

[46] Beckhoff. EK1110 | EtherCAT extension | Beckhoff Worldwide, . URL https://www.beckhoff.com/en-en/products/i-o/ethercat-terminals/ek1xxx-bk1xx0-ethercat-coupler/ek1110.html. (visited 11/6/2021).

[47] Beckhoff. EK1914 | EtherCAT Coupler with integrated digital standard and safety I/Os | Beckhoff USA, . URL https://www.beckhoff.com/en-us/products/automation/twinsafe/twinsafe-hardware/ek1914.html. (visited 11/6/2021).

[48] Beckhoff. EL6695 | EtherCAT bridge terminal | Beckhoff USA, . URL https://www.beckhoff.com/en-us/products/i-o/ethercat-terminals/el6xxx-communication/el6695.html?gclid=CjwKCAjwqcKFBhAhEiwAfEr7zdQrVM8z7HguintrMkKLXyPKbh-q6aiY4Kwg33LfAmeznqEv6sOMqxoC8HQQA
BwE. (visited 11/6/2021).

[49] PROFINET. URL https://www.profibus.com/technology/profinet. (visited 11/6/2021).

[50] Beckhoff. Automation device specification (ads)| beckhoff usa, . URL https://infosys.beckhoff.com/english.php?content=../content/1033/tcadscommon/html/tcadscommon_intro.htm&id=. (visited 11/6/2021).

[51] Abb Robotics. ABB-Developer Center, . URL https://developercenter.robotstudio.com/pc-sdk. (visited 11/6/2021).

[52] Abb Robotics. ABB Robotics Application manual PC SDK. Technical report, 2010.

[53] Abb Robotics. 3HAC028357-001 | ABB, . URL https://new.abb.com/products/3HAC028357-001/3hac028357-001. (visited 12/6/2021).

[54] P.Abreu. *Manual de utilização robotstudio*. FEUP, 2018.

[55] What is WPF? - Visual Studio | Microsoft Docs. URL https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019. (visited 13/6/2021).

[56] XAML overview - WPF .NET | Microsoft Docs. URL https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-5.0. (visited 13/6/2021).

Appendix A

Tables of shared variables

Table A.1: Data transferred between the *Combi-head* controller and the PLC Beckhoff CX9020

Variable Name	IO Address	Variable Type	Description
Combihead_ControlActive	I	BOOL	*Combihead* control active
Combihead_Position_Reached	I	BOOL	*Combihead* has reached the desired position
Combihead_SoftwareLimit_Plus	I	BOOL	Software limit plus has been reached
Combihead_SoftwareLimit_Minus	I	BOOL	Software limit minus has been reached
Combihead_Control_Control	O	BOOL	Activate *Combihead* control
Combihead_Error	I	BOOL	Error occurred with the *Combihead*
Combihead_Reset	O	BOOL	Error reset for the *Combihead*
Combihead_Fast_Up	O	BOOL	*Combihead* quickly moves up
Combihead_Fast_Down	O	BOOL	*Combihead* quickly moves down
Combihead_Middle_Position	O	BOOL	*Combihead* moves to the middle position
Combihead_Control_Auto	O	BOOL	*Combihead* control in automatic mode
Combihead_Inc_Plus	O	BOOL	Position increment for the *Combihead*
Combihead_Inc_Minus	O	BOOL	Position decrement for the *Combihead*
Combihead_Set_Position	O	BOOL	Set the desired position of the *Combihead*

Combihead_Set_Distance	O	BOOL	Set the distance to the desired position of the *Combihead*
Combihead_Distance_Actual	I	WORD	Distance traveled by the *Combihead* to reach the desired position
Combihead_Position_Actual	I	WORD	Current position of the *Combihead*
Combihead_Distance_Setpoint	O	WORD	Distance to reach the desired position

Table A.2: Data transferred between the distance controller for capacitive sensors and the PLC Beckhoff CX9020

Variable Name	IO Address	Variable Type	Description
LaserSensor_Far	I	BOOL	Laserhead Sensor outside measuring range signal
LaserSensor_TipTouch	I	BOOL	Laserhead Sensor collision/nozzle lost
LaserSensor_CableCut	I	BOOL	Laserhead Sensor or electrode cable cut signal
LaserSensor_Ready	I	BOOL	Laserhead Sensor ready status
LaserSensor_BodyTouch	I	BOOL	Short curcuit between laserhead sensor and workpiece signal (collision)
LaserSensor_PosReached	I	BOOL	Laserhead Sensor successfull calibration/missing nozzle
LaserSensor_Standoff_Bit2	O	BOOL	Laserhead sensor signal for selecting standoff distance (bit2)
LaserSensor_Standoff_Bit1	O	BOOL	Laserhead sensor signal for selecting standoff distance (bit1)
LaserSensor_Standoff_Bit0	O	BOOL	Laserhead sensor signal for selecting standoff distance (bit0)
LaserSensor_Char_Bit3	O	BOOL	Laserhead sensor selection of the characteristic curve (bit3)
LaserSensor_Char_Bit2	O	BOOL	Laserhead sensor selection of the characteristic curve (bit2)
LaserSensor_Char_Bit1	O	BOOL	Laserhead sensor selection of the characteristic curve (bit1)
LaserSensor_Char_Bit0	O	BOOL	Laserhead sensor selection of the characteristic curve (bit0)
LaserSensor_Distance_IN	I	BOOL	Laserhead sensor linear distance value (0-10V)
LaserSensor_Distance_mm	I	BOOL	Laserhead sensor distance, in mm

Table A.3: Data transferred between the Gas Panel and the PLC Beckhoff CX9020

Variable Name	IO Address	Variable Type	Description
WeldingGasSelector_He	O	BOOL	Enable welding gas valve: Helium
WeldingGasSelector_Ar	O	BOOL	Enable welding gas valve: Argon
CuttingGasSelector_N2	O	BOOL	Enable cutting gas valve: N2
CuttingGasSelector_O2	O	BOOL	Enable cutting gas valve: O2
GasCrossjet_YP	O	BOOL	Laser crossjet valve enable
Helium_Pressure	O	BOOL	Work gas pressure: Helium
Argon_Pressure	O	BOOL	Work gas pressure: Argon
Nitrogen_Pressure	O	BOOL	Work gas pressure: Nitrogen
Oxygen_Pressure	O	BOOL	Work gas pressure: Oxygen
LaserAttacher	O	BOOL	Enables the mechanism to attach the tool
LaserDetacher	O	BOOL	Enables the mechanism to detach the tool

Table A.4: Data transferred between the PLC Beckhoff CX9020 and the PLC Beckhoff CX8100

Variable Name	IO Address	Variable Type	Description
Chiller_On	O	BOOL	Activates the chiller
Laser_Fault_Reset	O	BOOL	Reset the fault in the laser source
Laser_Gate	O	INT	Define the value of the laser gate
Laser_Mains_On	O	BOOL	Activate laser mains
Laser_Ramp_Selection	O	BOOL	Laser ramp has been selected
Laser_Spy_Start/Reset	O	BOOL	Start/Restart *LaserSpy* system
Laser_Standby_On	O	BOOL	Change laser status to standby
Laser_Pointer	O	BOOL	Red laser pointer light enabled
Potencia_Laser_Out	O	WORD	Laser power regulation
Program_selection	O	BYTE	Selected Laser Program
Chiller_On	I	BOOL	Chiller activated
Compressed_Air	I	BOOL	Compressed air available
External_Chiller_on	I	BOOL	External chiller activated
Failure_Beam_Switch_Safety	I	BOOL	Beam Switch failure
Failure_Laser_Safety	I	BOOL	Laser Safety Failure
Fiber_Mirror_in_Beam	I	BOOL	Fiber mirror in the beam
Fiber_Mirror_in_Home	I	BOOL	Fiber mirror in the *home* position
Fiber_enabled	I	BOOL	Fiber enabled
Lase_Beam_Ready	I	BOOL	Laser beam ready
Laser_Estop	I	BOOL	Laser in an emergency stop
Laser_Fault_Reset	I	BOOL	Laser error successfully reset
Laser_General_Fault	I	BOOL	Laser in general fault
Laser_Job_End	I	BOOL	Laser job finished status
Laser_Mode_Auto	I	BOOL	Laser AUTO mode status
Laser_On	I	BOOL	Laser turned ON status

Variable Name	IO Address	Variable Type	Description
Laser_Program_End	I	BOOL	Laser program finished status
Laser_Regulation_Reserve_ok	I	BOOL	Laser reserve regulation OK status
Laser_Safe	I	BOOL	Laser in the safe status
Laser_Spy_Safe	I	BOOL	*LaserSpy* in safe mode
Laser_Standby_Ready	I	BOOL	Laser ready to start STANDBY mode, Stage 2
Laser_System_Error	I	BOOL	Laser system in error status
Laser_System_Ready	I	BOOL	Laser system is ready
Laser_Warning	I	BOOL	Laser system warning
Mains_on_Safety	I	BOOL	Laser mains in safety status
Ponteiro_Laser	I	BOOL	Red laser pointer is active
Potencia_laser_feedback	I	WORD	Current Laser Power

Table A.5: Data transferred between the ABB IRC5 Controller and the PLC Beckhoff CX9020

Variable Name	IO Address	Variable Type	Description
StandoffDistance	O	GO	Desired distance from the nozzle to the Workpiece
HeadPosition	I	GI	Current position of the *Combihead*
SPS_Position	I	GI	Desired position of the *Combihead*
GlassMissing	I	DI	Glass is missing from the *Combihead*
SensorReady	I	DI	*Combihead* sensor is ready
HeadControlActive	I	DI	*Combihead* control is active
HeadError	I	DI	Error occured with the *Combihead*
BodyTouch	I	DI	*Combihead* body has collided
TipTouch	I	DI	*Combihead* nozzle tip has collided
CableCut	I	DI	*Combihead* cable has been cut
SensorFar	I	DI	Nozzle distance to the workpiece has been exceeded
LimitPlus	I	DI	*Combihead* upper limit has been reached
LimitMinus	I	DI	*Combihead* lower limit has been reached
ArgonFlow	O	GO	Desired flow value for the Argon flow control valve
HeliumFlow	O	GO	Desired flow value for the Helium flow control valve
NitrogenPressure	O	GO	Desired pressure value for the Nitrogen pressure control valve
OxygenPressure	O	GO	Desired pressure value for the Oxygen pressure control valve

Variable Name	IO Address	Variable Type	Description
WeldingArgonSelected	O	DO	Activate the flow of argon through the nitrogen flow control valve
WeldingHeliumSelected	O	DO	Activate the flow of helium through the helium flow control valve
CuttingNitrogenSelected	O	DO	Activate the flow of nitrogen through the nitrogen pressure control valve
CuttingOxygenSelected	O	DO	Activate the flow of oxygen though the oxygen pressure control valve
GasCrossjet	O	DO	Activate Gas crossjet
LaserAttacher	O	DO	Activate the mechanism to attach the laser tool to the robot's arm
LaserDetacher	O	DO	Activate the mechanism to detach the laser tool from the robot's arm
GasReady	I	DI	The necessary gases have reached the nozzle of the laser tool
LaserEmergency	I	DI	Laser system in emergency status
LaserAuto	I	DI	Laser system in Automatic mode
GeneralFault	I	DI	Laser system in General Fault status
LaserBeamReady	I	DI	Laser Beam is ready
LaserControlMode	I	DI	Laser system in Control mode
LaserMains	I	DI	Value of the laser mains
LaserState	I	DI	Status of the Laser
FiberSelection	I	DI	Fiber has been selected
FiberEnabled	I	DI	Fiber has been enabled
Mirror	I	DI	Laser mirror has been activated

Variable Name	IO Address	Variable Type	Description
ProgramDone	I	DI	Laser program has finished
StandbyReady	I	DI	Laser in standby status
Warning	I	DI	Laser system warning
LaserType	O	DO	Determine the laser functioning mode
LaserPointer	O	DO	Activates Red Laser Pointer
LaserProgram	O	GO	Determines the laser program used in the procedure
LaserDuty	O	GO	Determines the laser signal duty cycle
LaserFrequency	O	GO	Determines the laser signal frequency
LaserPower	O	GO	Determines the laser power
LaserGate	I	GI	Value of the laser gate
SafetyState	I	DI	Safety features of the robotic station are not triggered
LaserEStop	I	DI	Laser emergency detected
LaserSpy	I	DI	*LaserSpy* has detected a problem
EmergInternal	I	DI	Emergency detected inside the robotic cell
EmergExternal	I	DI	Emergency detected outside the robotic cell
DoorClosed	I	DI	Robotic cell door is closed
DoorLocked	I	DI	Robotic cell door is open
WarningLamp_G	I	DI	Green warning lamp is active
WarningLamp_O	I	DI	Orange warning lamp is active
WarningLamp_R	I	DI	Red warning lamp is active
O_MotorsOn	O	DO	Robot electric motors are turned ON
O_AutoOn	O	DO	Controller is in automatic mode

Variable Name	IO Address	Variable Type	Description
O_TaskExecuting	O	DI	Controller tasks are running
O_SimMode	O	DO	Controller in simulation mode
O_EmergencyStop	O	DO	Robot emergency was detected
O_TCPSpeed	O	AO	Velocity of the tool center point

Appendix B

User Manual

This manual should be read by the system operator and will consist of a detailed description of the purpose of every component in each tab. It will also carefully describe the process to test a set of process parameters, as well as the process to import and start a custom procedure.

ROBOT TAB

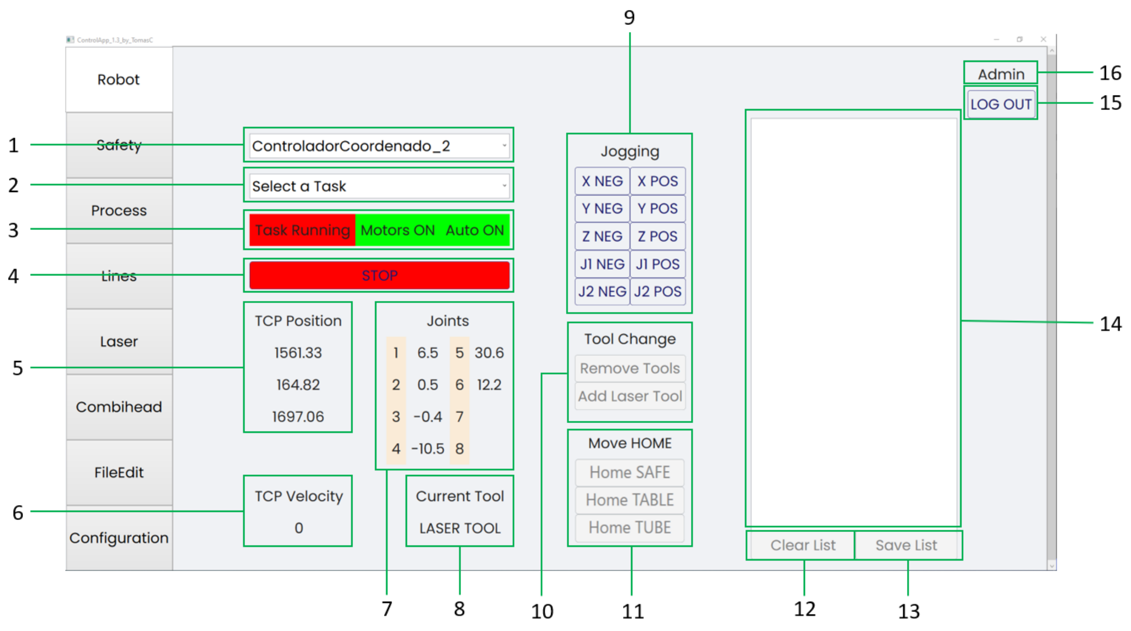


Figure B.1: Components of the *MainWindow - "Robot" Tab*

(1) Combo Box Controllers - When this combo box is dropped down a search for all the ABB controllers, real and virtual, in the network is made. All the controllers found are then designated as the item source of this combo box.

If a controller is selected then all the digital, analog and group signals necessary for the display and control of the application are saved to their respective lists, so that a signal changed event can be detected. The controller properties are also saved and displayed in the *"Configuration"* tab.

(2) Combo Box Tasks - After the controller is selected all the tasks of the selected controller are designated as the item source of this combo box.

When a task is selected several functionalities of the application are unlocked, including the tool changing buttons, preventing possible application errors or unintentional robot movements.

(3) Robot Display Labels - Displays critical information about the controller current state.

- **Task Running Label -** The background of this label changes colour based on the value of the digital signal *"O_TaskExecuting"*.

Table B.1: Behaviour of the "Task Running Label"

"O_TaskExecuting" Value	Background Colour	Description
True	green	Controller tasks are running
False	red	Controller tasks are not running

- **Motors On Label -** The background of this label changes colour based on the value of the digital signal *"O_MotorsOn"*.

Table B.2: Behaviour of the "Motors On Label"

"O_MotorsOn" Value	Background Colour	Description
True	green	Robots's eletric motors are turned ON
False	red	Robot's eletric motors are turned OFF

- **Auto On Label -** The background of this label changes colour based on the value of the digital signal *"O_AutoOn"*.

Table B.3: Behaviour of the "Auto On Label"

"O_AutoOn" Value	Background Colour	Description
True	green	Controller is in automatic mode
False	red	Controller is not in automatic mode

(4) Stop Button - The click of this button will stop the robot in its current position and deactivate all the process gasses and laser variables. The current path of the robot is not stored and resumed after the stop, instead the procedure needs to be restarted.

(5) TCP Position Label - This section is formed by three labels, each one representing the position of the robot's TCP in one the of the cartesian axis of the referential with the origin located at the base of the robot. The position and orientation of the used referential is available in figure B.2 .

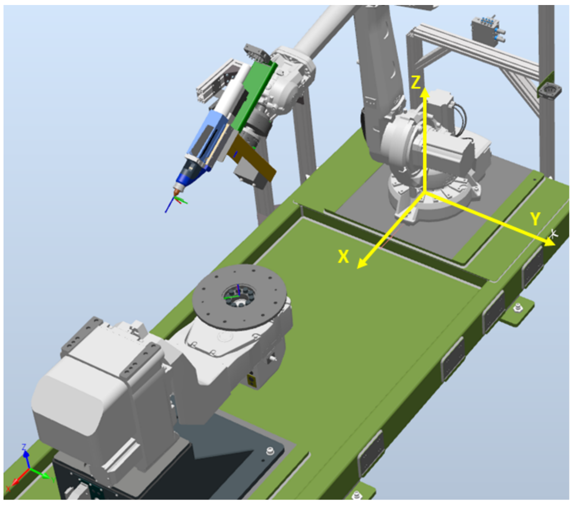


Figure B.2: Location and orientation of the WCS

(6) TCP Velocity Label - Represents the value of the analog signal *"O_TCPSpeed"*, which is associated with the system output *"TCP Speed"* of the robot controller.

(7) Joints Labels - This section is formed by eight labels, from one to six the labels represent the rotation of the robot's axis motors and the last two labels represent the rotation of the positioner motors.

(8) Current Tool Label - Displays the current tool equipped to the robot. The application determines the current tool equipped to the robot by analysing the *RAPID tooldata "CurrentTool"*.

The value of *"CurrentTool"* changes every time a tool is attached or detached from the robot arm using TRAP routines. The mentioned trap routines are used as following:

- User presses a tool changing button in the C# application
- Variable is set to true during the *RAPID* program
- Trap routine started
- Trap routine finished
- Variable is set to false

This behavior can be summarized using the grafcet available in figure B.3 .

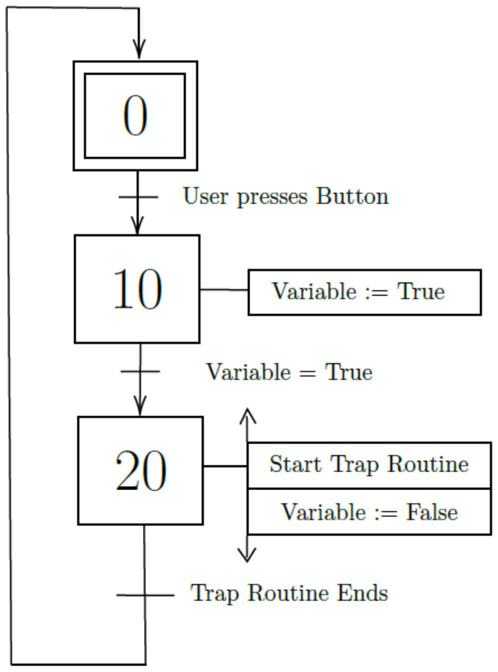


Figure B.3: Trap Routine Grafcet

If the *tooldata "CurrentTool"* is equal to the *tooldata "Tool_Laser_MOVEL"*, then the label will display "LASER TOOL". However, if the *tooldata "CurrentTool"* is equal to the *tooldata "Tool_Laser_FIXA"*, then the label will display "NO TOOL". In case the variable *"CurrentTool"* is different from both the *tooldata "Tool_Laser_MOVEL"* and the *tooldata "Tool_Laser_FIXA"* a problem has occurred; therefore, the label will display "TOOL PROBLEM".

(9) Jogging Buttons - The function of this buttons is to slowly move the robot TCP to the desired position inside the robot's working space. Additionally, some buttons are used to rotate the positioner's axes.

For the buttons responsible for linearly jogging the robot using cartesian axes, if the button is pressed and quickly released then the robot TCP will only move 2mm in the desired direction; however, if the button is held down the robot will continue to move in the selected direction until the button is released.

If the buttons responsible for jogging the positioner are pressed a five degrees rotation of the selected axis will occur every second until the button is released. The delay of one second was used to avoid the "short movements" controller error.

(10) Tool Change Buttons - The function of the button *"Remove Tools"* is to start the robot controller's procedure to remove the current tool equipped, whereas the function of the button *"Add Laser Tool"* is to start the robot controller's procedure to attach the laser tool (*Combi-head*) to the robot's arm.

(11) Move HOME Buttons - The function of the button "Home SAFE" is to call the robotic procedure to move the TCP to the HOME SAFE position. When the robot reaches this target the majority of the joint's values will be 0º, with the exception of the fifth joint, which will be 90º. The joints of the positioner will also revolve to reach 0º, as it can be seen in figure B.4 .

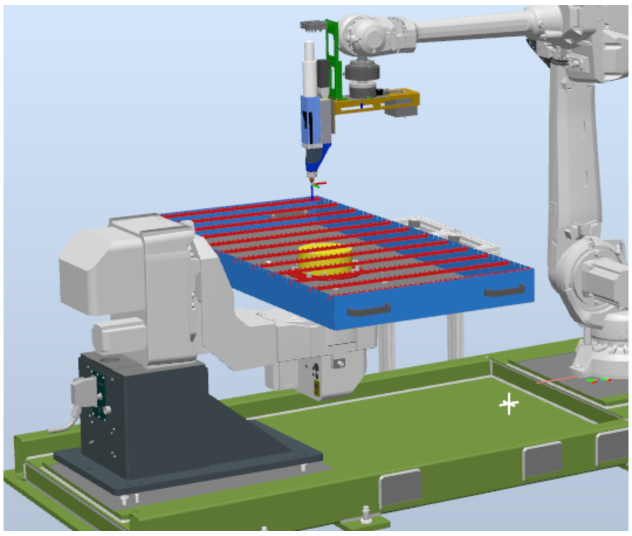


Figure B.4: Illustration of the Home SAFE position

The function of the button "Home TABLE" is to start the robotic procedure to move the TCP to the HOME TABLE position. The purpose of this target is to get the laser tool's nozzle perpendicular to the cutting table, which is crucial when starting laser cutting procedures, and in the proximity of the desired cutting position. The figure B.5 represents the robot's pose when reaching the HOME TABLE position.

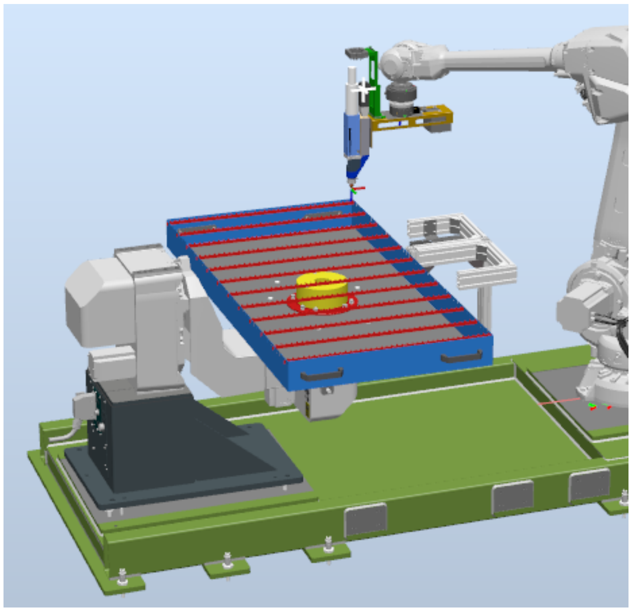


Figure B.5: Illustration of the Home TABLE position

The function of the button “Home TUBE” is to start the robotic procedure to move the TCP to the HOME TUBE position. The purpose of this target is to get the laser tool’s nozzle perpendicular to a tube’s surface; hence, it is necessary to rotate the robot’s fifth joint to 90º and the positioner’s first joint to 90º, as it can be seen in figure B.6.

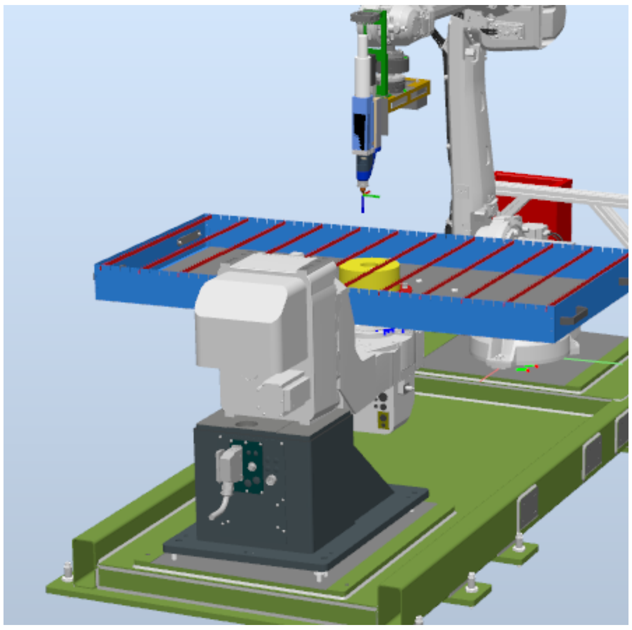


Figure B.6: Illustration of the Home TUBE position

(12) Clear List Button - The click of this button will clear the list of log messages from the control application’s memory.

(13) Save List Button - The click of this button will open a dialog box for the user to choose a file directory and save the list of log messages in a text file format.

(14) Log Messages List Box - The list box displays all the messages from the robot controller and some messages relevant to the control application, such as the start of different procedures.

(15) Log Out Button - The click of this button will open the *Authentication Window* and clear all the saved information

(16) Current User Label - Displays the name of the user currently logged in the control application.

SAFETY TAB

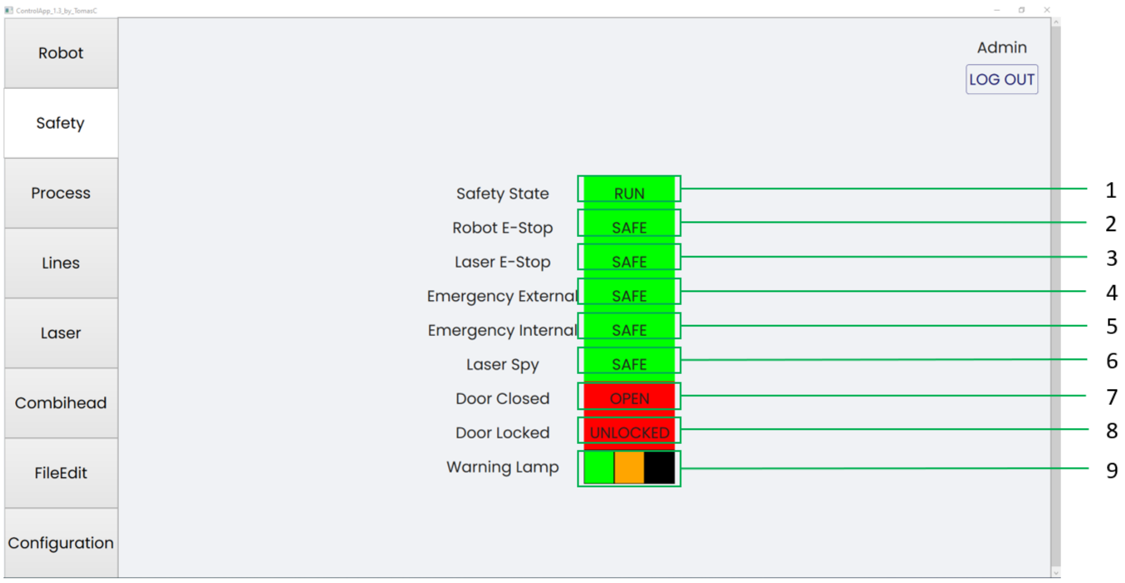


Figure B.7: Components of the *MainWindow* - *"Safety" Tab*

(1) Safety State Label - The content and background colour of this label change based on the value of the digital signal *"SafetyState"*.

Table B.4: Behaviour of the "Safety State Label"

"SafetyState" Value	Background Colour	Content	Description
True	green	SAFE	Robotic cell is safe
False	red	NOT SAFE	Robot cell is not safe

(2) Robot E-Stop Label - The content and background colour of this label change based on the value of the digital signal *"O_EmergencyStop"*.

Table B.5: Behaviour of the "Robot E-Stop Label"

"O_EmergencyStop" Value	Background Colour	Content	Description
True	red	NOT SAFE	Robot emergency not detected
False	green	SAFE	Robot emergency detected

(3) Laser E-Stop Label - The content and background colour of this label change based on the value of the digital signal *"LaserEStop"*.

Table B.6: Behaviour of the "Laser E-Stop Label"

"LaserEStop" Value	Background Colour	Content	Description
True	red	NOT SAFE	Laser emergency detected
False	green	SAFE	Laser emergency not detected

(4) Emergency External Label - The content and background colour of this label change based on the value of the digital signal *"EmergExternal"*.

Table B.7: Behaviour of the "Emergency External Label"

"EmergExternal" Value	Background Colour	Content	Description
True	red	NOT SAFE	Emergency detected outside the robotic cell
False	green	SAFE	No emergency detected outside the robotic cell

(5) Emergency Internal Label - The content and background colour of this label change based on the value of the digital signal *"EmergInternal"*.

Table B.8: Behaviour of the "Emergency Internal Label"

"EmergInternal" Value	Background Colour	Content	Description
True	red	NOT SAFE	Emergency detected inside the robotic cell
False	green	SAFE	No emergency detected inside the robotic cell

(6) Laser Spy Label - The content and background colour of this label change based on the value of the digital signal *"LaserSpy"*.

Table B.9: Behaviour of the "Laser Spy Label"

"LaserSpy" Value	Background Colour	Content	Description
True	red	NOT SAFE	LaserSpy detected a problem
False	green	SAFE	LaserSpy did not detect a problem

(7) Door Closed Label - The content and background colour of this label change based on the value of the digital signal *"DoorClosed"*.

Table B.10: Behaviour of the "Door Closed Label"

"DoorClosed" Value	Background Colour	Content	Description
True	green	CLOSED	Robotic cell door is closed
False	red	OPEN	Robotic cell door is open

(8) Door Locked Label - The content and background colour of this label change based on the value of the digital signal *"DoorLocked"*.

Table B.11: Behaviour of the "Door Locked Label"

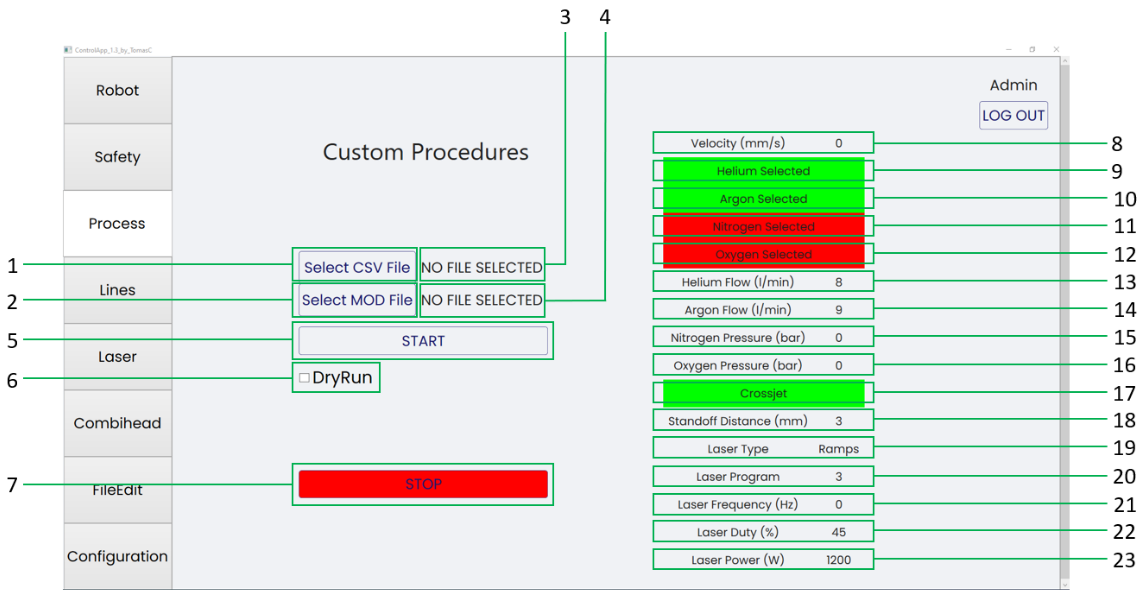
"DoorLocked" Value	Background Colour	Content	Description
True	green	LOCKED	Robotic cell door is locked
False	red	UNLOCKED	Robotic cell door is unlocked

(9) Warning Lamp Labels - This section is formed by three labels, whose content and background colour change based on the value of the digital signals that control the warning lamps of the robotic cell (*"WarningLamp_G"*, *"WarningLamp_O"*, *"WarningLamp_R"*).

Table B.12: Behaviour of the "Warning Lamp Labels"

Signal	Value	Background Colour	Description
WarningLamp_G	True	green	Green warning lamp ON
	False	red	Green warning lamp OFF
WarningLamp_O	True	orange	Orange warning lamp ON
	False	black	Orange warning lamp OFF
WarningLamp_R	True	red	Red warning lamp ON
	False	black	Red warning lamp OFF

PROCESS TAB



Velocity (mm/s) 0
Helium Selected
Argon Selected
Nitrogen Selected
Oxygen Selected
Helium Flow (l/min) 8
Argon Flow (l/min) 9
Nitrogen Pressure (bar) 0
Oxygen Pressure (bar) 0
Crossjet
Standoff Distance (mm) 3
Laser Type Ramps
Laser Program 3
Laser Frequency (Hz) 0
Laser Duty (%) 45
Laser Power (W) 1200

Figure B.8: Components of the *MainWindow - "Process" Tab*

(1) Select CSV File Button - The click of this button will open a dialog box for the user to select the CSV file with the process variables necessary for the planned procedure.

(2) Select MOD File Button - The click of this button will open a dialog box for the user to select the MOD File that contains the desired RAPID procedure, which will then be added to the robot controller. In addition to including all the move instructions the RAPID procedure will also call several functions to control the various components of the robotic cell, such as the laser source and the gas panel.

(3) Selected CSV File Label - Displays the currently selected CSV File for the planned custom procedure.

(4) Selected Mod File Label - Displays the currently selected MOD file.

(5) Start Button - The click of this button will start the most recently added *RAPID* custom procedure to the robot controller.

(6) Dry Run Checkbox - If checked the custom procedures will not activate process gasses and laser variables, which is extremely useful when testing new robotic trajectories.

(7) Stop Button - The click of this button will stop the robot in its current position and deactivate all the process gasses and laser variables. The current path of the robot is not stored and resumed after the stop, instead the procedure needs to be restarted.

(8) Velocity Label - The content of this label is equal to the value of the analog signal *"O_TCPSpeed"*, which represents the current TCP velocity of the robotic arm.

(9) Helium Selected Label - The background colour of this label changes based on the value of the digital signal *“WeldingHeliumSelected”*.

Table B.13: Behaviour of the ”Helium Selected Label”

”WeldingHeliumSelected” Value	Background Colour	Description
True	green	Helium valve activated
False	red	Helium valve deactivated

(10) Argon Selected Label - The background colour of this label changes based on the value of the digital signal *“WeldingArgonSelected”*.

Table B.14: Behaviour of the ”Argon Selected Label”

”WeldingArgonSelected” Value	Background Colour	Description
True	green	Argon valve activated
False	red	Argon valve deactivated

(11) Nitrogen Selected Label - The background colour of this label changes based on the value of the digital signal *“CuttingNitrogenSelected”*.

Table B.15: Behaviour of the ”Nitrogen Selected Label”

”CuttingNitrogenSelected” Value	Background Colour	Description
True	green	Nitrogen valve activated
False	red	Nitrogen valve deactivated

(12) Oxygen Selected Label - The background colour of this label changes based on the value of the digital signal *“CuttingOxygenSelected”*.

Table B.16: Behaviour of the ”Oxygen Selected Label”

”CuttingOxygenSelected” Value	Background Colour	Description
True	green	Oxygen valve activated
False	red	Oxygen valve deactivated

(13) Helium Flow Label - The content of this label matches the value of the group signal *“HeliumFlow”*. This label represents the current helium flow being used in the procedure.

(14) Argon Flow Label - The content of this label matches the value of the group signal *“ArgonFlow”*. This label represents the current argon flow being used in the procedure.

(15) Nitrogen Pressure Label - The content of this label matches the value of the group signal *“NitrogenPressure”*. This label represents the current nitrogen pressure being used in the procedure.

(16) Oxygen Pressure Label - The content of this label matches the value of the group signal *“OxygenPressure”*. This label represents the current oxygen pressure being used in the procedure.

(17) Crossjet Label - The background colour of this label changes based on the value of the digital signal *“GasCrossjet”*.

Table B.17: Behaviour of the ”Crossjet Label”

”GasCrossjet” Value	Background Colour	Description
True	green	Crossjet activated
False	red	Crossjet deactivated

(18) Standoff Distance Label - The content of this label matches the value of the group signal *“StandoffDistance”*. This label represents the current distance from the tool aperture to the work-piece.

(19) Laser Type Label - The content of this label changes based on the value of the digital signal *“LaserType”*.

Table B.18: Behaviour of the ”Laser Type Label”

”LaserType” Value	Content	Description
True	Ramps	Laser Type in RAMPS mode
False	Pulses	Laser Type in PULSES mode

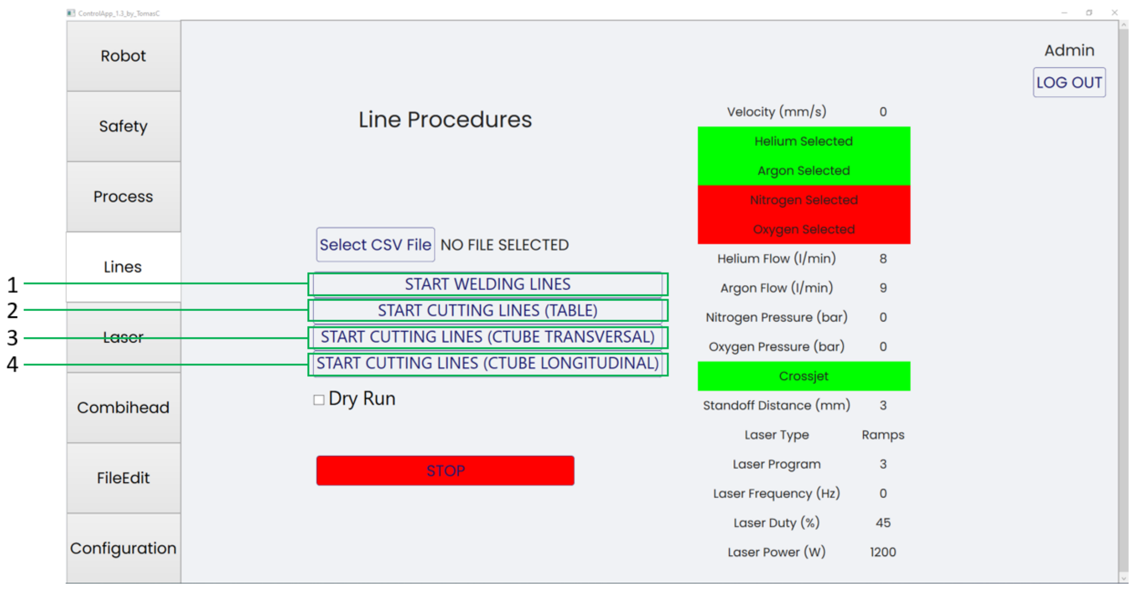
(20) Laser Program Label - The content of this label matches the value of the group signal *“LaserProgram”*. This label represents the current laser program being used in the procedure.

(21) Laser Frequency Label - The content of this label matches the value of the group signal *“LaserFrequency”*. This label represents the current laser frequency being used in the procedure

(22) Laser Duty Cycle Label - The content of this label matches the value of the group signal *“LaserDuty”*. This label represents the current laser duty cycle being used in the procedure.

(23) Laser Power Label - The content of this label matches the value of the group signal *“LaserPower”*. This label represents the current laser power being used in the procedure.

LINES TAB



NO FILE SELECTED

Velocity (mm/s)	0
Helium Selected	
Argon Selected	
Nitrogen Selected	
Oxygen Selected	
Helium Flow (l/min)	8
Argon Flow (l/min)	9
Nitrogen Pressure (bar)	0
Oxygen Pressure (bar)	0
Crossjet	
Standoff Distance (mm)	3
Laser Type	Ramps
Laser Program	3
Laser Frequency (Hz)	0
Laser Duty (%)	45
Laser Power (W)	1200

Figure B.9: Components of the *MainWindow* - *"Lines" Tab*

Although this tab is very similar to the *"Process"* tab, there are a few differences worth mentioning.

(1) Start Welding Lines Button – The click of this button will start the robot controller procedure to create the welding lines (*"Welding_Lines"*).

If the DryRun checkbox is selected the button click will instead start the robot controller's procedure to test the trajectories of the welding lines without process gasses and laser variables (*"Welding_Lines_Test"*)

(2) Start Cutting Lines (Table) Button – The click of this button will start the robot controller's procedure to create the cutting lines on the horizontal cutting table (*"Cutting_Lines_H"*).

If the DryRun checkbox is selected the button click will instead call the robot controller's procedure to test the trajectories of the cutting lines without process gasses and laser variables (*"Cutting_Lines_H_Test"*).

(3) Start Cutting Lines (Table) Button – The click of this button will start the robot controller's procedure to create the transversal cutting lines on a circular tube (*"Cutting_Lines_CT"*).

If the DryRun checkbox is selected the button click will instead call the robot controller's procedure to test the trajectories of the cutting lines without process gasses and laser variables (*"Cutting_Lines_CT_Test"*).

(4) Start Cutting Lines (Table) Button – The click of this button will start the robot controller's procedure to create the longitudinal cutting lines on a circular tube (*"Cutting_Lines_CL"*).

If the DryRun checkbox is selected the button click will instead call the robot controller's procedure to test the trajectories of the cutting lines without process gasses and laser variables (*"Cutting_Lines_CL_Test"*).

LASER TAB

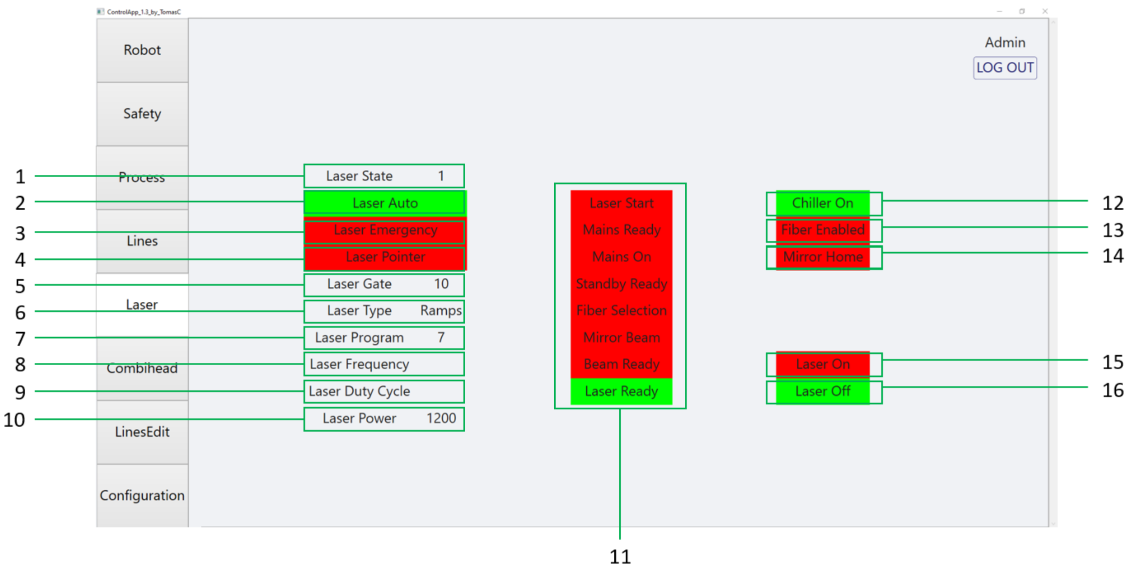


Figure B.10: Components of the *MainWindow* - *"Laser" Tab*

(1) Laser State Label - The content of this label matches the value of the group signal *"LaserState"*. This label represents the current state of the laser source.

(2) Laser Auto Label - The background colour of this label changes based on the value of the digital signal *"LaserAuto"*.

Table B.19: Behaviour of the "Laser Auto Label"

"LaserAuto" Value	Background Colour	Description
True	green	Laser is in auto mode
False	red	Laser is in manual mode

(3) Laser Emergency Label - The background colour of this label changes based on the value of the digital signal *“LaserEStop”*.

Table B.20: Behaviour of the ”Laser Emergency Label”

”LaserEStop” Value	Background Colour	Description
True	red	Laser is in emergency mode
False	green	Laser is not in emergency mode

(4) Laser Pointer Label - The background colour of this label changes based on the value of the digital signal *“LaserPointer”*. A double click of this label will activate the laser pointer.

Table B.21: Behaviour of the ”Laser Pointer Label”

”LaserPointer” Value	Background Colour	Description
True	green	Laser pointer is active
False	red	Laser pointer is not active

(5) Laser Gate Label - The content of this label matches the value of the group signal *“LaserGate”*. This label represents the current value of the laser gate.

(6) Laser Type Label - The content of this label changes based on the value of the digital signal *“LaserType”*.

Table B.22: Behaviour of the ”Laser Type Label”

”LaserType” Value	Content	Description
True	Ramps	Laser Type in RAMPS mode
False	Pulses	Laser Type in PULSES mode

(7) Laser Program Label - The content of this label matches the value of the group signal *“LaserProgram”*. This label represents the current laser program.

(8) Laser Frequency Label - The content of this label matches the value of the group signal *“LaserFrequency”*. This label represents the current laser frequency.

(9) Laser Duty Cycle Label - The content of this label matches the value of the group signal *“LaserDuty”*. This label represents the current laser duty cycle.

(10) Laser Power Label - The content of this label matches the value of the group signal *“LaserPower”*. This label represents the current laser power.

(11) Laser Startup Sequence Labels - This section is formed by eight labels, corresponding to the startup sequence of the laser source. The background colour of each label changes to green if the corresponding phase of the startup sequence has been reached. If the *Laser Ready Label* is green the laser is ready to start the procedure; however, if there is a problem with the startup it is simple to identify the current phase of the sequence, making it easier to determine the cause of the problem.

- **Laser Start Label -** The background colour of this label changes based on the value of the digital signal *"LaserStart"*.

Table B.23: Behaviour of the "Laser Start Label"

"LaserStart" Value	Background Colour	Description
True	green	*Laser Start* startup phase has been reached
False	red	*Laser Start* startup phase has not been reached

- **Mains Ready Label -** The background colour of this label changes based on the value of the digital signal *"MainsReady"*.

Table B.24: Behaviour of the "Mains Ready Label"

"MainsReady" Value	Background Colour	Description
True	green	*Mains Ready* startup phase has been reached
False	red	*Mains Ready* startup phase has not been reached

- **Mains On Label -** The background colour of this label changes based on the value of the digital signal *"MainsOn"*.

Table B.25: Behaviour of the "Mains On Label"

"MainsOn" Value	Background Colour	Description
True	green	*Mains On* startup phase has been reached
False	red	*Mains On* startup phase has not been reached

- **Standby Ready Label -** The background colour of this label changes based on the value of the digital signal *"StandbyReady"*.

Table B.26: Behaviour of the "Standby Ready Label"

"StandbyReady" Value	Background Colour	Description
True	green	*Standby Ready* startup phase has been reached
False	red	*Standby Ready* startup phase has not been reached

- **Fiber Selection Label -** The background colour of this label changes based on the value of the digital signal *"FiberSelection"*.

Table B.27: Behaviour of the "Fiber Selection Label"

"FiberSelection" Value	Background Colour	Description
True	green	*Fiber Selection* startup phase has been reached
False	red	*Fiber Selection* startup phase has not been reached

- **Mirror Beam Label -** The background colour of this label changes based on the value of the digital signal *"MirrorBeam"*.

Table B.28: Behaviour of the "Mirror Beam Label"

"MirrorBeam" Value	Background Colour	Description
True	green	*Mirror Beam* startup phase has been reached
False	red	*Mirror Beam* startup phase has not been reached

- **Beam Ready Label -** The background colour of this label changes based on the value of the digital signal *"BeamReady"*.

Table B.29: Behaviour of the "Beam Ready Label"

"BeamReady" Value	Background Colour	Description
True	green	*Beam Ready* startup phase has been reached
False	red	*Beam Ready* startup phase has not been reached

- **Laser Ready Label -** The background colour of this label changes based on the value of the digital signal *“LaserReady”*.

Table B.30: Behaviour of the ”Laser Ready Label”

”LaserReady” Value	Background Colour	Description
True	green	*Laser Ready* startup phase has been reached
False	red	*Laser Ready* startup phase has not been reached

(12) Chiller On Label - The background colour of this label changes based on the value of the digital signal *“ChillerOn”*.

Table B.31: Behaviour of the ”Chiller On Label”

”ChillerOn” Value	Background Colour	Description
True	green	Chiller is activated
False	red	Chiller is deactivated

(13) Fiber Enabled Label - The background colour of this label changes based on the value of the digital signal *“FiberEnabled”*.

Table B.32: Behaviour of the ”Fiber Enabled Label”

”FiberEnabled” Value	Background Colour	Description
True	green	Laser fiber is enabled
False	red	Laser fiber is deactivated

(14) Mirror Home Label - The background colour of this label changes based on the value of the digital signal *“MirrorHome”*.

Table B.33: Behaviour of the ”Mirror Home Label”

”MirrorHome” Value	Background Colour	Description
True	green	Mirror in home position
False	red	Mirror not in home position

(15) Laser On Label - The background colour of this label changes based on the value of the digital signal *“LaserOn”*.

Table B.34: Behaviour of the ”Laser On Label”

”LaserOn” Value	Background Colour	Description
True	green	Laser is firing
False	red	Laser is not firing

(16) Laser Off Label - The background colour of this label changes based on the value of the digital signal *"LaserOff"*.

Table B.35: Behaviour of the "Laser Off Label"

"LaserOff" Value	Background Colour	Description
True	green	Laser can not be fired
False	red	Laser can be fired

COMBIHEAD TAB

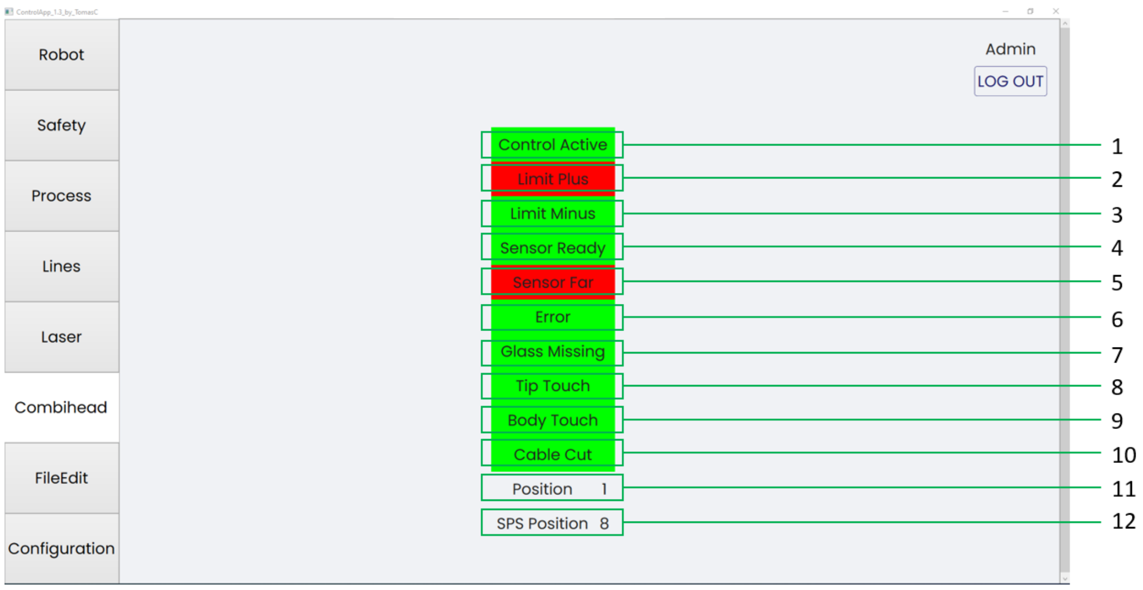


Figure B.11: Components of the *MainWindow - "Combihead" Tab*

(1) Control Active Label - The background colour of this label changes based on the value of the digital signal *"HeadControlActive"*.

Table B.36: Behaviour of the "Control Active Label"

"HeadControlActive" Value	Background Colour	Description
True	green	Combi-head control is active
False	red	Combi-head control is not active

(2) Limit Plus Label - The background colour of this label changes based on the value of the digital signal *“LimitPlus”*.

Table B.37: Behaviour of the ”Limit Plus Label”

”LimitPlus” Value	Background Colour	Description
True	green	Tool has reached the upper limit
False	red	Tool has not reached the upper limit

(3) Limit Minus Label - The background colour of this label changes based on the value of the digital signal *“LimitMinus”*.

Table B.38: Behaviour of the ”Limit Minus Label”

”LimitMinus” Value	Background Colour	Description
True	green	Tool has reached the lower limit
False	red	Tool has not reached the lower limit

(4) Sensor Ready Label - The background colour of this label changes based on the value of the digital signal *“SensorReady”*.

Table B.39: Behaviour of the ”Sensor Ready Label”

”SensorReady” Value	Background Colour	Description
True	green	Tool sensor is ready
False	red	Tool sensor is not ready

(5) Sensor Far Label - The background colour of this label changes based on the value of the digital signal *“SensorFar”*.

Table B.40: Behaviour of the ”Sensor Far Label”

”SensorFar” Value	Background Colour	Description
True	green	Tool is too far from the work-piece
False	red	Tool is not too far from the work-piece

(6) Error Label - The background colour of this label changes based on the value of the digital signal *"HeadError"*.

Table B.41: Behaviour of the "Error Label"

"HeadError" Value	Background Colour	Description
True	red	Error detected with the tool
False	green	No error detected with the tool

(7) Glass Missing Label - The background colour of this label changes based on the value of the digital signal *"GlassMissing"*.

Table B.42: Behaviour of the "Glass Missing Label"

"GlassMissing" Value	Background Colour	Description
True	red	Glass is missing
False	green	Glass is not missing

(8) Tip Touch Label - The background colour of this label changes based on the value of the digital signal *"TipTouch"*.

Table B.43: Behaviour of the "Tip Touch Label"

"TipTouch" Value	Background Colour	Description
True	red	Tool tip suffered a collision
False	green	Tool tip did not suffer a collision

(9) Body Touch Label - The background colour of this label changes based on the value of the digital signal *"BodyTouch"*.

Table B.44: Behaviour of the "Body Touch Label"

"BodyTouch" Value	Background Colour	Description
True	red	Tool Body suffered a collision
False	green	Tool Body did not suffer a collision

(10) Cable Cut Label - The background colour of this label changes based on the value of the digital signal *"CableCut"*.

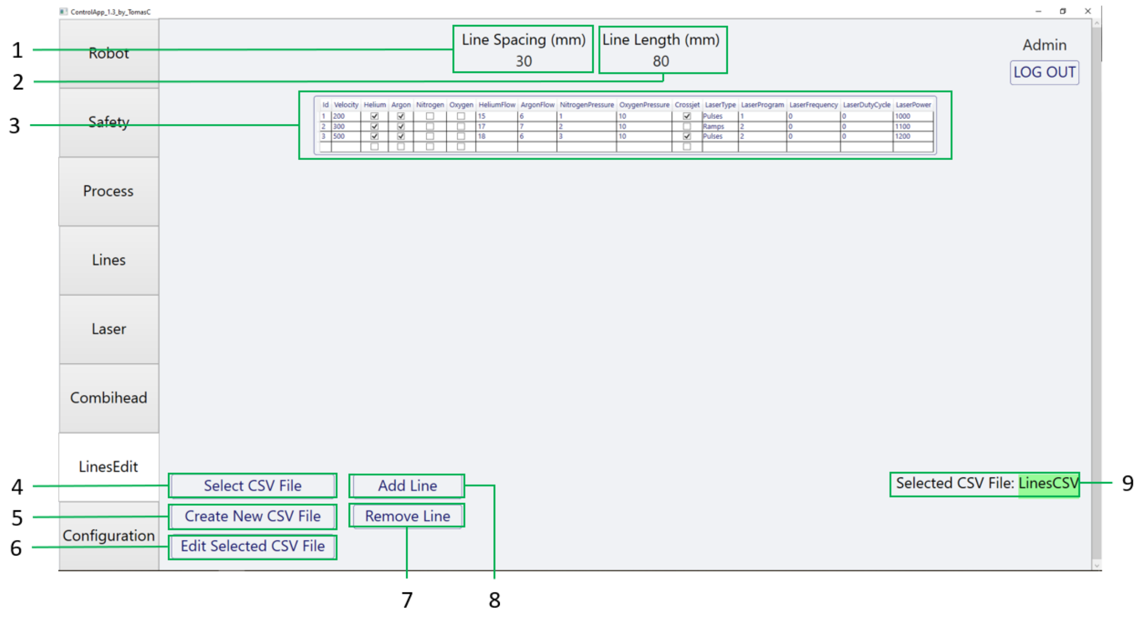
Table B.45: Behaviour of the "Cable Cut Label"

"CableCut" Value	Background Colour	Description
True	red	A tool cable is cut
False	green	No tool cable is cut

(11) Position Label - The content of this label matches the value of the group signal *"HeadPosition"*. This label represents the current distance to the work-piece.

(12) SPS Position Label - The content of this label matches the value of the group signal *"SPS_Position"*. This label represents the distance to the tool's desired location.

FILE EDIT TAB



Line Spacing (mm) 30

Line Length (mm) 80

Id	Velocity	Helium	Argon	Nitrogen	Oxygen	HeliumFlow	ArgonFlow	NitrogenPressure	OxygenPressure	Crossjet	LaserType	LaserProgram	LaserFrequency	LaserDutyCycle	LaserPower
1	200	✓	✓	☐	☐	15	6	1	10	✓	Pulses	1	0	0	1000
2	300	✓	✓	☐	☐	17	7	2	10	☐	Ramps	2	0	0	1100
3	500	✓	✓	☐	☐	18	6	3	10	✓	Pulses	2	0	0	1200
		☐	☐	☐	☐					☐					

Selected CSV File: LinesCSV

Figure B.12: Components of the *MainWindow* - *"FileEdit" Tab*

(1) Line Spacing Label - The content of this label represents the distance between the parallel lines in the testing procedures.

(2) Line Length Label - The content of this label represents the length of the parallel lines in the testing procedures.

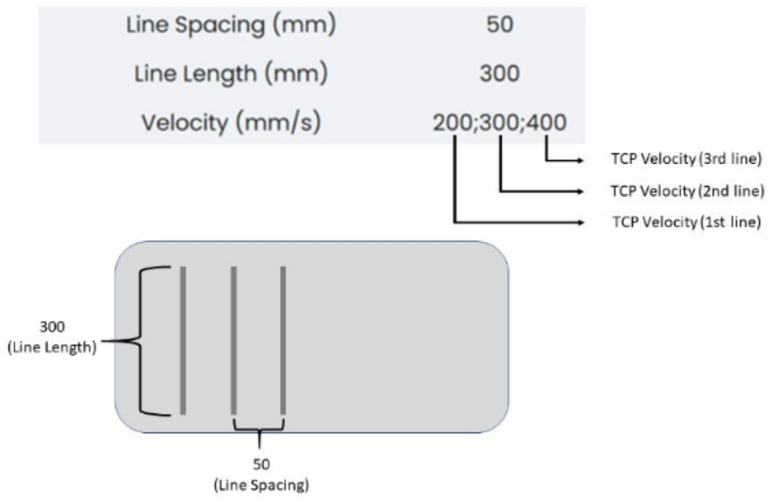


Figure B.13: Example of a set of variables from the CSV file

(3) Process Variables Grid - This section consists of a list of process variables for each line in the testing procedures.

(4) Select CSV File Button - The click of this button will open a dialog box for the user to select the CSV file to be displayed and/or edited in this tab.

(5) Create New CSV File Button - The click of this button will open a dialog box for the user to select a directory of a new CSV File containing the information for a testing procedure.

(6) Edit Selected CSV File Button - The click of this button will change the information of the selected CSV file and replace the original values with the content from the *Process Variables Grid*.

(9) Selected CSV File Label - Displays the currently selected CSV file for the testing procedures.

CONFIGURATION TAB

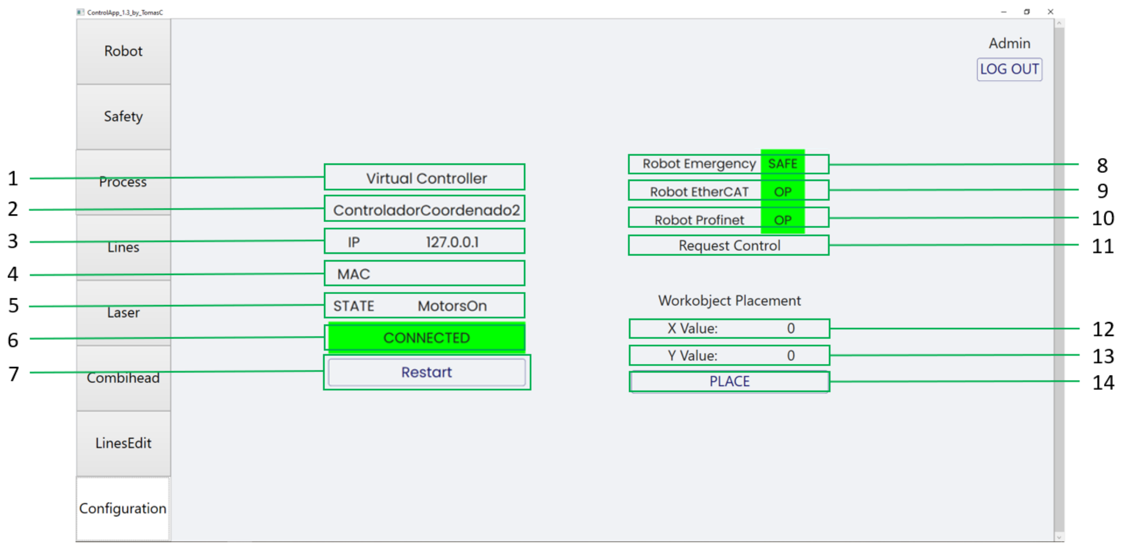


Figure B.14: Components of the *MainWindow* - *"Configuration" Tab*

(1) Controller Type Label - Displays the type of controller that was selected in the application, it can either be "Virtual Controller" or "Real Controller".

(2) Controller Name Label - Displays the name attributed to the controller that was selected in the application

(3) Controller IP Label - Displays the IP address of the controller that was selected in the application, if the selected controller is virtual then the IP is 127.0.0.1.

(4) Controller Mac Address Label - Displays the mac address of the controller that was selected in the application.

(5) Controller State Label - Displays the current state of the controller that was selected in the application.

(6) Controller Connection Label - Displays the connection status of the selected controller.

Table B.46: Behaviour of the "Controller Connection Label"

Status	Background Colour	Content
Controller Connected	green	CONNECTED
Controller Disconnected	red	DISCONNECTED

(7) Controller Restart Button - The click of this button will restart (warmstart) the selected controller.

(8) Robot Emergency Label - The content and background colour of this label change based on the value of the digital signal *"O_EmergencyStop"*. Displays the information about the robot emergency status.

Table B.47: Behaviour of the "Robot Emergency Label"

"O_EmergencyStop" Value	Background Colour	Content	Description
True	red	NOT SAFE	Robot emergency detected
False	green	SAFE	Robot emergency not detected

(9) Robot EtherCAT Label - Displays the information about the status of the PLC *EtherCAT* connection.

Table B.48: Behaviour of the "Robot EtherCAT Label"

Status	Background Colour	Content
EtherCAT is transferring data	green	OP
EtherCAT is not transferring data	red	OFF

(10) Robot Profinet Label - Displays the information about the status of the PLC *Profinet* connection.

Table B.49: Behaviour of the "Robot Profinet Label"

Status	Background Colour	Content
Profinet is transferring data	green	OP
Profinet is not transferring data	red	OFF

(11) Request Control Label - The content and background of this label changes colour based on the value of the digital signal *"RequestControl"*. A double click of this label will set the digital signal *"RequestControl"* to true.

Table B.50: Behaviour of the "Request Control Label"

"RequestControl" Value	Background Colour	Content	Description
True	green	TRUE	Control has been requested
False	red	FALSE	Control has not been requested

(12) Wobj *X* Value Label - Displays the distance, in the x axis, between the origin of the original workobject associated with the cutting table and the edited version of that workobject.

(13) Wobj *Y* Value Label - Displays the distance, in the y axis, between the origin of the original workobject associated with the cutting table and the edited version of that workobject.

(14) Place Button - The click of this button will start the robot controller's procedure to edit the position of the workobject associated to the cutting table. The new edited position of the workobject will be determined by the current position of the tool center point, which means that to edit the workobject it is first necessary to move the robot's TCP to the new desired position. The height of the cutting table will remain constant; hence, it is not necessary to edit the height of the workobject.

INSTRUCTIONS TO TEST PROCESS PARAMETERS:

The method to test different process parameters only involves a CSV file containing the length of the lines, the spacing between them and the set of process parameters for each line. The number of parallel lines to create is directly proportional to number of sets of process variables. It is expected that the application user has the CSV file ready to be selected before starting the following list of instructions.

1. Insert the application password in the *Authentication Window*

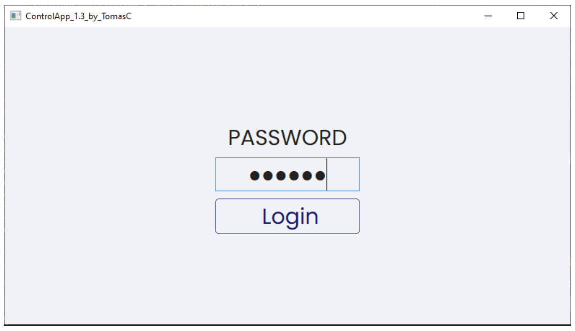


Figure B.15: Password entered in the *Authentication Window*

2. Select the controller in the *MainWindow - "Robot"* tab
3. Select the task in the *MainWindow - "Robot"* tab

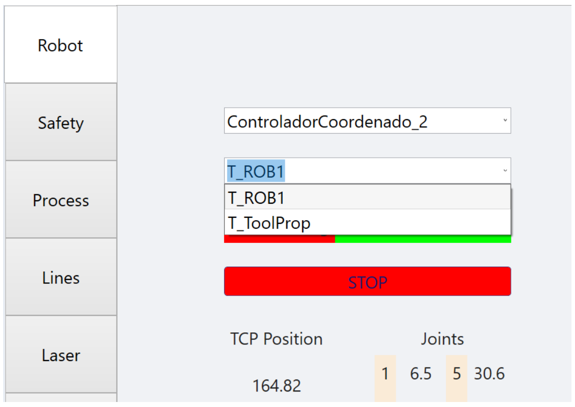
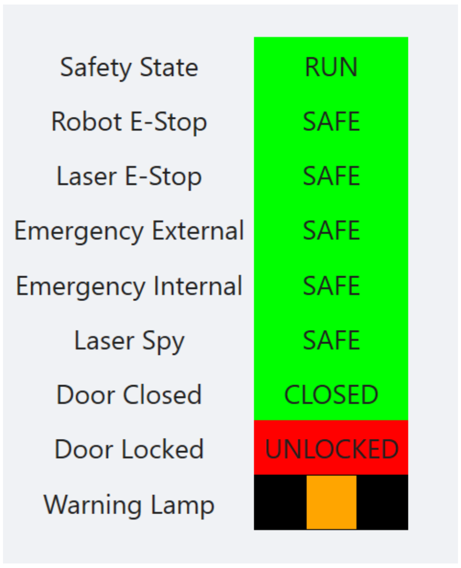


Figure B.16: Selection of the Controller and the Task in the *MainWindow - Robot* tab

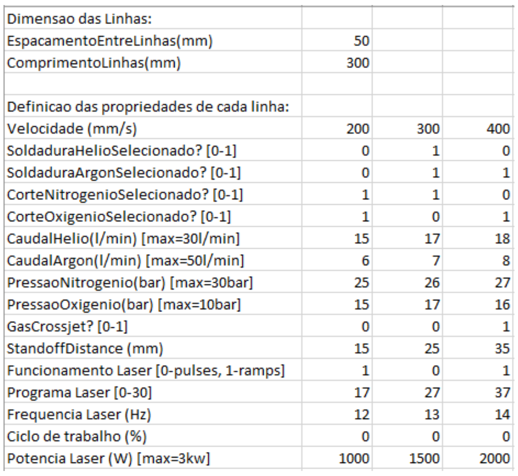
4. Position the tool center point in the desired starting position, using the *"Jogging"* functionalities and the *"Move HOME"* functionalities from the *MainWindow-"Robot"* tab
5. Verify the system's safety mechanisms using the *MainWindow - "Safety"* tab



Safety State	RUN
Robot E-Stop	SAFE
Laser E-Stop	SAFE
Emergency External	SAFE
Emergency Internal	SAFE
Laser Spy	SAFE
Door Closed	CLOSED
Door Locked	UNLOCKED
Warning Lamp	

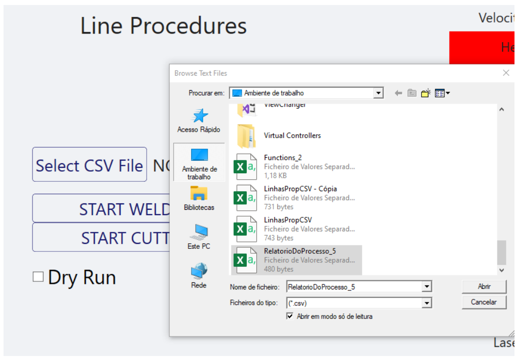
Figure B.17: State of the safety mechanisms of the robotic cell

6. Select the CSV file with the desired set of process parameters in the *Main Window - "Lines"* tab



Dimensao das Linhas:			
EspacamentoEntreLinhas(mm)	50		
ComprimentoLinhas(mm)	300		
Definicao das propriedades de cada linha:			
Velocidade (mm/s)	200	300	400
SoldaduraHelioSelecionado? [0-1]	0	1	0
SoldaduraArgonSelecionado? [0-1]	0	1	1
CorteNitrogenioSelecionado? [0-1]	1	1	0
CorteOxigenioSelecionado? [0-1]	1	0	1
CaudalHelio(l/min) [max=30l/min]	15	17	18
CaudalArgon(l/min) [max=50l/min]	6	7	8
PressaoNitrogenio(bar) [max=30bar]	25	26	27
PressaoOxigenio(bar) [max=10bar]	15	17	16
GasCrossjet? [0-1]	0	0	1
StandoffDistance (mm)	15	25	35
Funcionamento Laser [0-pulses, 1-ramps]	1	0	1
Programa Laser [0-30]	17	27	37
Frequencia Laser (Hz)	12	13	14
Ciclo de trabalho (%)	0	0	0
Potencia Laser (W) [max=3kw]	1000	1500	2000

(a)



(b)

Figure B.18: (a) Example of a CSV file containing the process parameters necessary to run the testing procedure; (b) Selection of the CSV file in the *Main Window - Lines* Tab

7. Check the *"Dyr Run"* checkbox to test the robot movements and the reach of the robot's arm

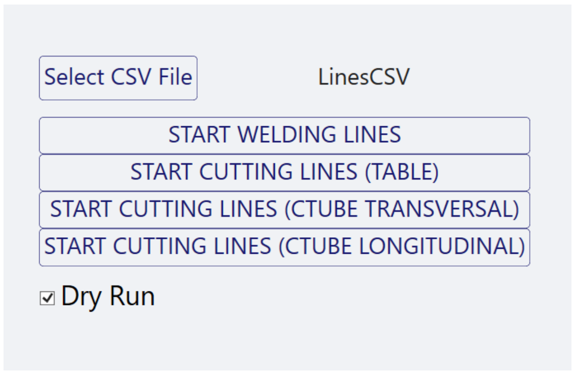
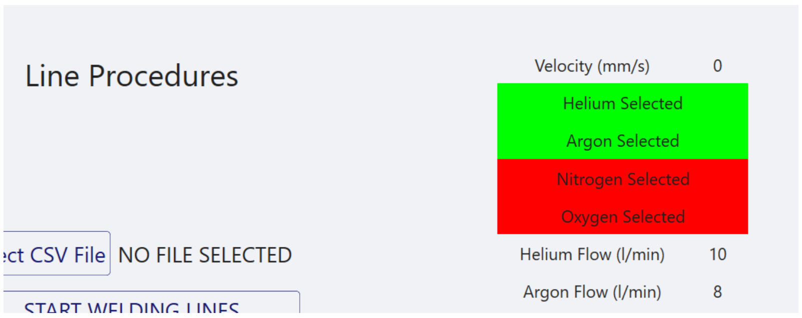

Select CSV File
LinesCSV
START WELDING LINES
START CUTTING LINES (TABLE)
START CUTTING LINES (CTUBE TRANSVERSAL)
START CUTTING LINES (CTUBE LONGITUDINAL)
☑ Dry Run

Figure B.19: Selection of the *Dry Run* checkbox

8. If the robot has the reach for every line defined in the CSV file and no other problem is detected, the user should uncheck the *"Dry Run"* checkbox and start the desired procedure by pressing the respective button.

9. Throughout the robot procedure it is advisable to check the values of the process parameters and the status of the other equipment in robotic station, such as the laser source and the *Combi-head* . If a problem is detected it is possible to stop the robot movement and shut off the gas and laser variables using the *"STOP"* button

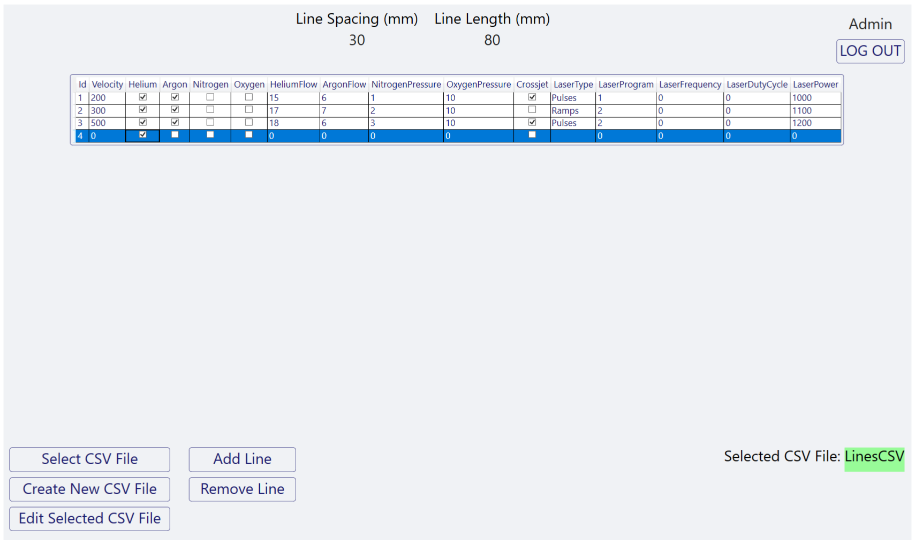

Line Procedures

ct CSV File NO FILE SELECTED

Velocity (mm/s)	0
Helium Selected	
Argon Selected	
Nitrogen Selected	
Oxygen Selected	
Helium Flow (l/min)	10
Argon Flow (l/min)	8

Figure B.20: Display of the current process parameters in the *MainWindow - Lines* tab

10. When the testing procedure is finished if the user decides to use a slightly different set of process parameters for a new set of lines it is possible to edit or create a new CSV file using the *MainWindow - "FileEdit"* tab.


Line Spacing (mm) 30

Line Length (mm) 80

Id	Velocity	Helium	Argon	Nitrogen	Oxygen	HeliumFlow	ArgonFlow	NitrogenPressure	OxygenPressure	Crossjet	LaserType	LaserProgram	LaserFrequency	LaserDutyCycle	LaserPower
1	200	☑	☑	☐	☐	15	6	1	10	☑	Pulses	1	0	0	1000
2	300	☑	☑	☐	☐	17	7	2	10	☐	Ramps	2	0	0	1100
3	500	☑	☑	☐	☐	18	6	3	10	☑	Pulses	2	0	0	1200
4	0	☑	☐	☐	☐	0	0	0	0	☐		0	0	0	0

Selected CSV File: LinesCSV

Figure B.21: Example of editing a CSV file to test a slightly a different set of process parameters

INSTRUCTIONS TO CREATE A CUSTOM PIECE:

The method to create a custom piece involves a CSV file and a MOD file. The MOD file contains an ABB module with the desired path and the necessary functions to control all the components of the station. The path of the tool center point can be generated using several software tools; however, in this project it is only used *RobotStudio* basic functionalities, such as the Auto-Path. The CSV file contains the process parameters for each function defined in the *RAPID* procedure.

The advantage of this methodology is that the same robotic trajectory can be used for different work-pieces, with different thicknesses and materials, by just selecting a different CSV file. It is expected that the user has the CSV file ready to be selected before starting the following list of instructions.

1. Using CAD software, create the virtual representation of the desired custom piece
2. Import the CAD file (.SAT) into the virtual station in *RobotStudio*
3. Program the necessary paths and insert the required functions to create the desired procedure. The new procedure must be created in an independent ABB module, named *"CustomProcedure"*, and saved as a MOD file to be used by the C# application.
4. Insert the application password in the *Authentication Window*

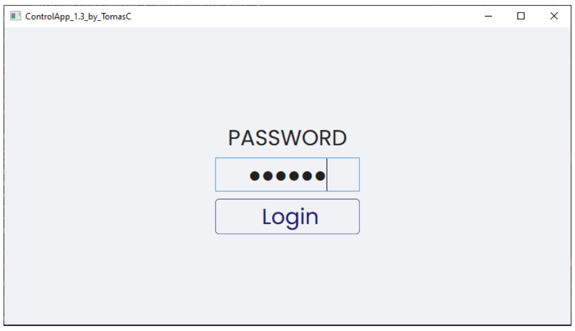


Figure B.22: Password entered in the *Authentication Window*

5. Select the controller in the *MainWindow* - *"Robot"* tab
6. Select the task in the *MainWindow* - *"Robot"* tab

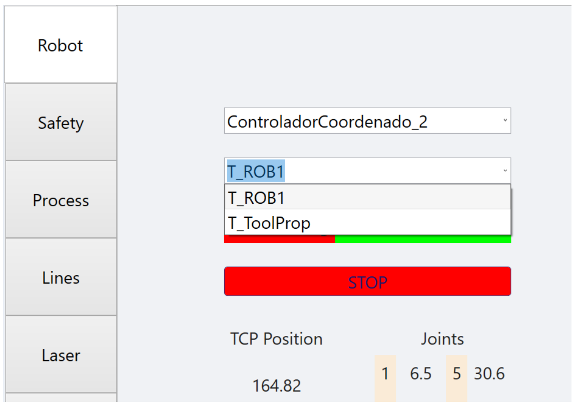
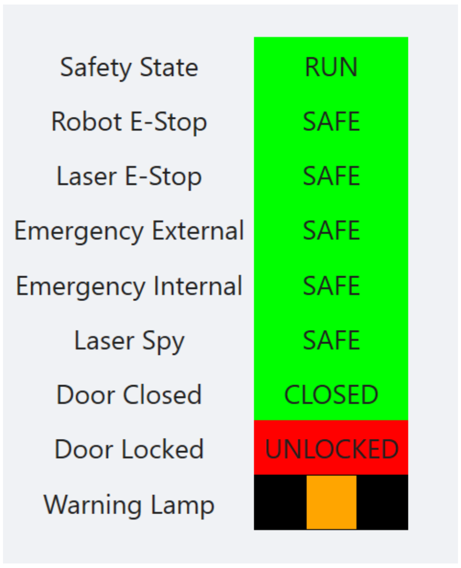


Figure B.23: Selection of the Controller and the Task in the *MainWindow* - *Robot* tab

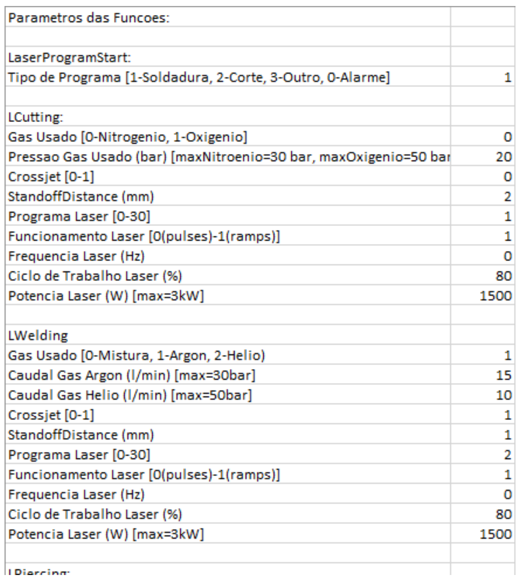
7. When using the cutting table, if the starting point of the procedure, created in the virtual station, is not ideal for the metal sheet being used, it is possible to alter the position of the workobject associated to the cutting table and consequently shift the starting position. To edit the mentioned workobject it is necessary to use the features available in *MainWindow* - *"Configuration"* tab.
8. Move the laser tool to a safe position, using the *"Jogging"* functionalities and the *"Move HOME"* functionalities from the *MainWindow* - *"Robot"* tab
9. Verify the system safety features using the *MainWindow* - *"Safety"* tab



Safety State	RUN
Robot E-Stop	SAFE
Laser E-Stop	SAFE
Emergency External	SAFE
Emergency Internal	SAFE
Laser Spy	SAFE
Door Closed	CLOSED
Door Locked	UNLOCKED
Warning Lamp	⬛🟧⬛

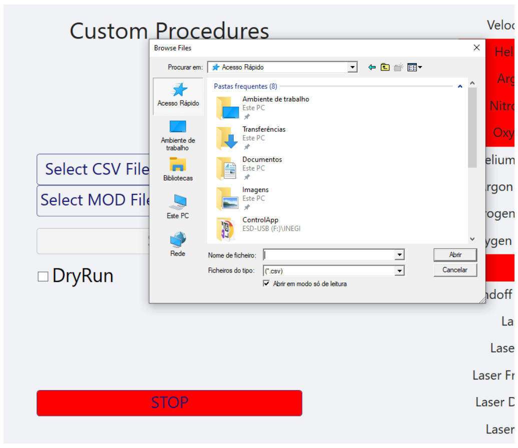
Figure B.24: State of the safety mechanisms of the robotic cell

10. Select the CSV file with the desired set of process parameters for each function in the *MainWindow* - *"Process"* tab



Parametros das Funcoes:	
LaserProgramStart:	
Tipo de Programa [1-Soldadura, 2-Corte, 3-Outro, 0-Alarme]	1
LCutting:	
Gas Usado [0-Nitrogenio, 1-Oxigenio]	0
Pressao Gas Usado (bar) [maxNitroenio=30 bar, maxOxigenio=50 bar	20
Crossjet [0-1]	0
StandoffDistance (mm)	2
Programa Laser [0-30]	1
Funcionamento Laser [0(pulses)-1(ramps)]	1
Frequencia Laser (Hz)	0
Ciclo de Trabalho Laser (%)	80
Potencia Laser (W) [max=3kW]	1500
LWelding	
Gas Usado [0-Mistura, 1-Argon, 2-Helio)	1
Caudal Gas Argon (l/min) [max=30bar]	15
Caudal Gas Helio (l/min) [max=50bar]	10
Crossjet [0-1]	1
StandoffDistance (mm)	1
Programa Laser [0-30]	2
Funcionamento Laser [0(pulses)-1(ramps)]	1
Frequencia Laser (Hz)	0
Ciclo de Trabalho Laser (%)	80
Potencia Laser (W) [max=3kW]	1500
LPiercing:	

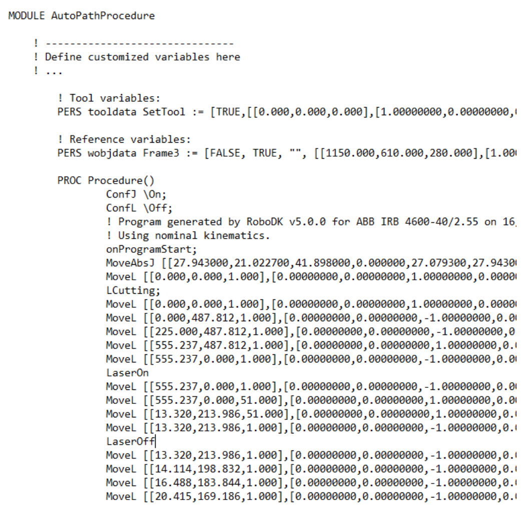
(a)



(b)

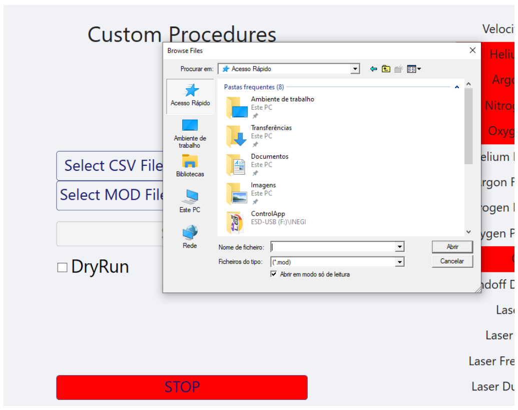
Figure B.25: (a) Example of a CSV file containing the process parameters necessary for the functions in the custom procedure; (b) Selection of the CSV file in the *MainWindow* - *Process* Tab

11. Select the MOD file containing the procedure to create the desired custom piece in the *MainWindow* - *"Process"* tab



```
MODULE AutoPathProcedure

    ! ------------------------------
    ! Define customized variables here
    ! ...

        ! Tool variables:
        PERS tooldata SetTool := [TRUE,[[0.000,0.000,0.000],[1.00000000,0.00000000,

        ! Reference variables:
        PERS wobjdata Frame3 := [FALSE, TRUE, "", [[1150.000,610.000,280.000],[1.00

        PROC Procedure()
                ConfJ \On;
                ConfL \Off;
                ! Program generated by RoboDK v5.0.0 for ABB IRB 4600-40/2.55 on 16
                ! Using nominal kinematics.
                onProgramStart;
                MoveAbsJ [[27.943000,21.022700,41.898000,0.000000,27.079300,27.9430
                MoveL [[0.000,0.000,1.000],[0.00000000,0.00000000,1.00000000,0.0000
                LCutting;
                MoveL [[0.000,0.000,1.000],[0.00000000,0.00000000,1.00000000,0.0000
                MoveL [[0.000,487.812,1.000],[0.00000000,0.00000000,-1.00000000,0.0
                MoveL [[225.000,487.812,1.000],[0.00000000,0.00000000,-1.00000000,0
                MoveL [[555.237,487.812,1.000],[0.00000000,0.00000000,1.00000000,0.
                MoveL [[555.237,0.000,1.000],[0.00000000,0.00000000,-1.00000000,0.0
                LaserOn
                MoveL [[555.237,0.000,1.000],[0.00000000,0.00000000,-1.00000000,0.0
                MoveL [[555.237,0.000,51.000],[0.00000000,0.00000000,1.00000000,0.0
                MoveL [[13.320,213.986,51.000],[0.00000000,0.00000000,1.00000000,0.
                MoveL [[13.320,213.986,1.000],[0.00000000,0.00000000,-1.00000000,0.
                LaserOff
                MoveL [[13.320,213.986,1.000],[0.00000000,0.00000000,-1.00000000,0.
                MoveL [[14.114,198.832,1.000],[0.00000000,0.00000000,-1.00000000,0.
                MoveL [[16.488,183.844,1.000],[0.00000000,0.00000000,-1.00000000,0.
                MoveL [[20.415,169.186,1.000],[0.00000000,0.00000000,-1.00000000,0.
```

(a)



(b)

Figure B.26: (a) Example of a MOD file containing the procedure and the functions to create a custom piece; (b) Selection of the MOD file in the *MainWindow - Process* Tab

12. Check the *"Dry Run"* checkbox to test the robot movements and verify if there are any unexpected collisions

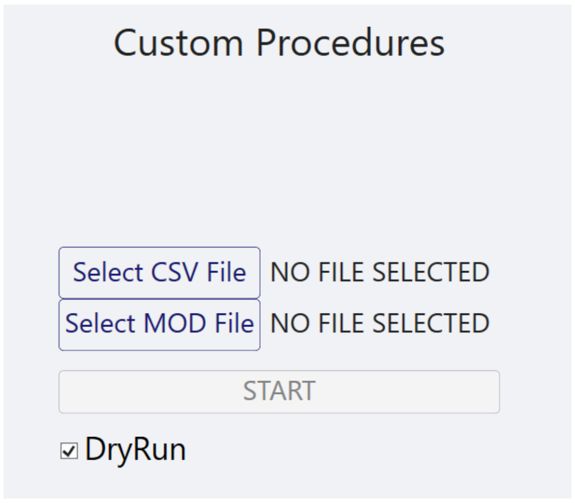


Figure B.27: Selection of the *Dry Run* checkbox in the *MainWindow - Process* tab

13. If no problem is detected the user should uncheck the *"Dry Run"* checkbox and start the desired procedure by pressing the "START" button in the *MainWindow - "Process"* tab

14. Throughout the robot procedure it is advisable to check the values of the process parameters and the status of the other equipment in the robotic station, such as the laser source and the *Combi-head* . If a problem is detected it is possible to stop the robot movement and shut off the gas and laser variables using the *"STOP"* button

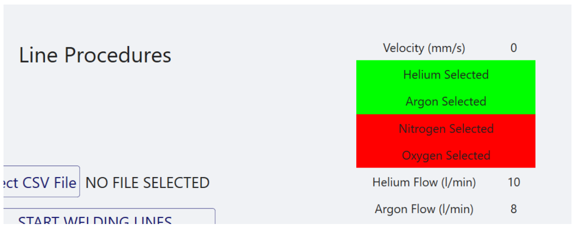


Figure B.28: Display of the current process parameters in the *MainWindow - Process* tab