

Automatic palletizing and management systems of aluminum cans through a robotic manipulator

Alfonso Falcone

Supervisor

Alessandro Rizzo

Co-Supervisor

Marco Valentini

A thesis presented for the Master degree of
Mechatronic Engineering



**Politecnico
di Torino**

DAUIN Department of Control and Computer Engineering

Politecnico di Torino

Italy, Turin

December 2020

Automatic palletizing and management systems of aluminum cans through a robotic manipulator

A full automatic multi-domain system

Alfonso Falcone

Abstract

Nowadays the industrial demand for robotic applications is growing like never before. The increasing need of automation and fulfillment of higher standards for production, precision and safety is one of the main reasons for such a demand. Robotic applications can fit in a lot of different industrial process, in fact, there exist many kinds of robots. In this project, the customer need is to obtain a full autonomous system, that is able to withdraw aluminum cans and palletizing them inside two chests of drawers. Although this seems a very simple task, the complexity behind it is fairly high, since the entire mechanical structure as well as the electrical and the programming logic ones have to be done from scratch. In order to meet these requirements, we have chosen to use an architecture with an external Plc that communicates with the robot PC, in order to make the manipulator work as just a muscle. We are confident that this architecture can be safer and more stable with respect to the one who uses just the robot PC and the embedded PLC simulator.

Contents

1	Hardware configuration	9
1.1	Introduction	9
1.2	Manipulator	14
1.3	PLC	17
1.4	End Effector - Tool design	18
1.4.1	Mechanical Failure Analysis	19
1.5	Distribution boards & data-sheets	23
1.5.1	Power	24
1.5.2	Signal	24
1.5.3	Terminal box	25
1.6	Pneumatic Circuit	26
1.7	Safety	28
1.7.1	SIL - Safety Integrator Level	28
1.7.2	Safety of human operators	28
1.7.3	Safety regions	29
2	Robotics basic concepts	30
2.1	Introduction	30
2.2	Basics	30
2.2.1	Elementary displacements	30
2.2.2	Cardan angles	33
2.2.3	Robot basic structure	33
2.2.4	Basic Terminology	34
2.2.5	Denavit-Hartenberg Convention	35
2.2.6	Direct Kinematics	37
2.2.7	Differential Kinematics	38
2.2.8	Singularity	39
2.2.9	Inverse kinematics	39
2.2.10	The Jacobian inverse technique	40
2.2.11	Heuristic methods	42
2.2.12	Trajectory Planning	42
2.2.13	Trajectory Planning Specifics	42
2.2.14	Trajectory Planning Algorithms	43
2.2.15	KUKA Motion Instructions	44
2.2.16	PTP : Point to Point	44
2.2.17	PTP_REL : Point to Point Relative	45
2.2.18	LIN : Linear	45
2.2.19	LIN_REL : Linear Relative	45

2.2.20	CIRC : Circular	45
2.2.21	CIRC_REL : Circular Relative	45
2.2.22	Spline	45
2.2.23	Spline	46
2.2.24	SPLINE	47
2.2.25	PTP_SPLINE ... ENDSPLINE	47
2.3	Safety in Modern Robotics	47
2.3.1	Tool Calibration	48
2.3.2	4-point method	49
2.3.3	XYZ Reference method	49
2.3.4	ABC 2-point	50
2.3.5	ABC World	50
2.3.6	Numeric input	50
2.3.7	Base Calibration	51
2.3.8	ABC 3-point	51
2.3.9	Configuring axis-specific workspace	51
2.3.10	Collision Detection	52
2.3.11	Safety Planes	54
2.3.12	Safety Barriers	54
2.3.13	Safety Hardware	54
2.3.14	Brake Test	57
2.4	Teach Pendant	58
2.4.1	Operating Modes	59
3	Omron PLC	61
3.1	OMRON	61
3.2	Computer Network	62
3.2.1	Local Area Network	62
3.2.2	Internet Protocol	62
3.2.3	Public IP addresses	63
3.2.4	IPV4	64
3.2.5	Subnet	64
3.3	Ethercat Protocol	64
3.3.1	Ethernet Communication	65
3.3.2	Add on-the-fly process data	65
3.3.3	Ethercat P: data & power supply on one cable	66
3.3.4	Distributed Clock for Precise Synchronization	66
3.3.5	Diagnostic and Localization Errors	67
3.3.6	High Availability Requirements	68
3.3.7	Communication Profiles	68
3.3.8	Transparent transmission of standard IT protocols	69
3.3.9	Services	69
3.4	Sysmac	69
3.5	Structure of the Code	72
3.5.1	Mission Concept	73
3.5.2	External PLC - Robot PC Communication	74

4	Coding	75
4.1	Sysmac	75
4.1.1	Ladder Main Program	75
4.1.2	Structured Code	78
4.2	WorkVisual	80
4.2.1	Files Extensions	80
4.3	Robot Program	81
4.3.1	Normal Operations	82
4.3.2	Special Operations	88
4.4	Full Missions Code	90
5	HMI Operator Panel	91
5.1	Common Uses of HMI	92
5.2	Premium HMI	92
5.2.1	Designed HMI	94
5.3	Home Page	95
5.3.1	Manual Operation Page	96
5.3.2	Setup Page	97
5.3.3	Alarm Pages	98
6	Conclusions	101
A	Appendix Title	103
A.1	Sysmac - Structured Language Code	103
A.2	Kuka - Missions Code	106
A.2.1	Mission 100	106
A.2.2	Mission 200	109
A.2.3	Mission 300	110
A.2.4	Mission 400	112
A.2.5	Mission 500	115
A.2.6	Mission 700	122
A.2.7	Mission 800	123
A.2.8	Mission 1000	125

List of Figures

1.1	Overview of the entire project	11
1.2	Main Required Functions	12
1.3	Full mechanical structure, particulars of drawer and rack	13
1.4	Technical Data written on the manipulator arm	14
1.5	Attachment flange for the design of a self designed tool	15
1.6	Nominal Weight vs Center of Mass Position	15
1.7	Range Space for the 5-th axis	16
1.8	PLC Omron NX102-9000	17
1.9	supplementary power supply NX-PF0630	17
1.10	Tool Interconnection Holes Mechanical Drawing	18
1.11	Fragment Of The Tool Mechanical Drawing	19
1.12	Displacement of a generic point P of a beam	20
1.13	Stress Strain Ductile Material graph	21
1.14	Static Analysis - Yield stress	22
1.15	Triangular mesh profile for static analysis purpose	22
1.16	First Reachability Analysis	23
1.17	Fragment of the Total element list	23
1.18	Fragment of the Power Zone	24
1.19	Fragment of the Signal Zone	25
1.20	Fragment of the Terminal box Zone	25
1.21	Air Treatment Group	26
1.22	Complete Pneumatic Circuit	27
1.23	Pneumatic Cylinder on the robot's tool	27
1.24	SIL - Safety Integrity Levels	28
1.25	Safety Planes activated when there is no drawer	29
2.1	Pure Translational motion	30
2.2	Euler Identity	31
2.3	Homogeneous Matrix structure	32
2.4	RPY Angles	33
2.5	Schematization of a robot - Links and Revolute Joints	34
2.6	Schematization of a robot - Open and closed chains, respectively on the left and right	34
2.7	Denavit–Hartenberg kinematic parameters	35
2.8	Kuka Kr220 schematization - Anthropomorphic arm with spherical wrist	37
2.9	Kuka Kr5, [2] singularities configurations	40

2.10	Inverse Kinematic Algorithms comparisons - NR vs NICCD - on two different manipulator type : Stanford Arm and WAM 7R	41
2.11	Trajectory Planning - 3 rd order polynomial trajectory - Trapezoidal velocity profile	44
2.12	Interpolation of continuous function $\sin(x)$ with various method . . .	46
2.13	Elementary Motion Types - from left to right P2P, LIN,CIRC	48
2.14	TCP Calibration	49
2.15	TCP Position Calibration - 4-point method	50
2.16	TCP Position Calibration - XYZ Reference method	50
2.17	TCP Orientation Calibration - ABC 2-point	51
2.18	TCP Orientation Calibration - ABC 3-point	52
2.19	Teach Pendant - Collision Detection Menu	53
2.20	Teach Pendant - Torque Monitoring Menu	53
2.21	Working Regions	54
2.22	Teach Pendant - Safety Plane Menu	55
2.23	Safety Barriers all around the manipulator	55
2.24	Safety Risk Zones	56
2.25	Safety Distance with respect to body parts	57
2.26	Brake Test - Faulty Brake	58
2.27	Teach Pendant - Front View	59
2.28	Teach Pendant - Rear View	59
2.29	Operating Mode and Safety functions	60
3.1	Ethercat Protocol - Communication, Power supply, Services and Synchronization	70
3.2	Sysmac - Topological View	71
3.3	Sysmac - Populated I/O Card	72
3.4	Algorithm Flow Chart	73
3.5	Fragment of I/O External PLC to Robot PC Mapping - Excel Sheet .	74
4.1	Plc Code - Ladder 1/5	75
4.2	Plc Code - Ladder 2/5	76
4.3	Plc Code - Ladder 3/5	77
4.4	Plc Code - Ladder 4/5	78
4.5	Plc Code - Ladder 5/5	79
4.7	Plc Code - Structured Language	79
4.6	Plc Main - Flow Chart	80
4.8	Robot missions set for normal Operations	82
4.9	Flow Chart Mission 100 - Home Position	83
4.10	Flow Chart Mission 200 - Withdraw Cans Approach	84
4.11	Flow Chart Mission 300 - Withdraw Cans	84
4.12	Flow Chart Mission 400 - Deposit Cans	85
4.13	Flow Chart Mission 500 - Close Drawer	86
4.14	Flow Chart Mission 600 - Cans Deposit	87
4.15	Robot missions set for special Operations	88
4.16	Flow Chart Mission 700 - Maintenance	88
4.17	Flow Chart Mission 800 - Quality Check Withdraw	89
4.18	Flow Chart Mission 1000 - Quality Check Deposit	89

5.1	HMI - Human-Machine Interface	91
5.2	Premium HMI IDE	93
5.3	HMI - Home Page	96
5.4	HMI - Manual Page	98
5.5	HMI - Setup Page	99
5.6	HMI - Alarm Page	100
A.1	Property of MVQuadro S.r.L	103
A.2	Property of MVQuadro S.r.L	106

Images owned by MVQuadro S.r.l. dissemination and reproduction prohibited, the company protects its rights by law

Chapter 1

Hardware configuration

1.1 Introduction

The following thesis work is the result of work in a company specialized in industrial automation and robotics, named "Mvquadro S.r.l.". For four months I was able to follow the project of building from scratch an automatic plant for the palletizing of aluminum lids. This project stems from the need of the customer, "Crown", which is a tin factory, to increase the level of automation of their company and invest in industry 4.0.

To this end, the company Crown has asked us to create from scratch the entire environment of the robotized island; so the racks, drawers, protective barriers, the base on which rests the robot itself, as well as electrical control panels, sensors, pneumatic system for opening and closing the robot gripper, the implementation of the software and graphics of the PLC and HMI operator panel were designed and programmed entirely by our company.

The robotized island consists of two racks on the left and right of the robot, which resides in the center, each containing 15 drawers, which were designed specifically for the housing of the batteries of aluminium lids that the company yet produces. This set of elements is fenced on all sides, except behind the racks, see fig.1.1, in order not to allow access to operators during normal operations of the robot. This is a very common practice, as the speeds and strength that an industrial robot can impress are remarkable and can easily cause injury or death in the event of a collision between robot and an operator.

But, since it is necessary for an operator to empty the rack when it is full, each rack has on board laser safety sensors, which detect the presence of operators in the rack area, if human presence is detected in this area, the robot program stops immediately to prevent collision.

The robot can be controlled by HMI operator panel or by physical control panel. Every alarm signal or every emergency event is reported on the operator panel, as well as the status code, that gives indications on the current state of operation of the robot.

The task of this robotized island is as follows, see fig.1.2:

- At the exit of a machine already present in the company are produced aluminium lids.
They are conveyed on a slide, which thanks to its inclination allows the stack of drawers to grow vertically.

Upon reaching a certain height, detected by a laser position sensor, decided by the operators and with the possibility of modification through the operator panel.

- The manipulator will pick up this battery and proceed, with the lids still held by the gripper, opening a drawer and palletizing the stack of lids inside the drawer.
- Once stacked the first stack continues so until it fills the entire drawer.
- Once the drawer is full, the robot has to close the full drawer and open an empty one.
When all the drawers in a rack are full, the robot stops working on that rack and starts working on the other, doing the same actions.
- When a rack is finished, an operator picks up the full racks and replaces them with empty ones.¹
- If both racks are occupied, the robot will stand still and wait.

When you buy a robot, you are not only buying a manipulator, but also the framework of the robot, which contains a particular computer that is able to communicate with the robot itself and in which you can simulate a PLC. Thus, in general it would be possible to use such a computer to create both the robot program and simulate the PLC behaviour to manage the signals coming from the field. However, the fundamental difference that there is between a PC and a PLC is the fact that a PLC is built specifically to perform operations sequentially respecting a certain time cycle. This guarantees a superior robustness compared to a traditional PC, but has the disadvantage of not being able to perform loops, such as while and for. That's why, since the robotized island has a considerable number of sensors and signals in input and output, we decided to use an external PLC and not the one emulated by the software of the robot.

We opted for this choice also because we can use the manipulator as if it was just a muscle at the service of the external PLC that is the mind.

The negative aspect of this choice, however, is the greater complexity of the general system and the longer time required to program the Plc and the robot PC and also to manage communications between them.

The gripper is designed to be able to open and close the drawers while at the same time holding the stacks of aluminium lids, the obvious advantage is the reduction of time cycle, that is the total time in which the robot completes all the operations necessary to complete the full program. In fact, this tool performs the functions of a double tool, avoiding automatic procedures of disassembling/assembling tools.

The electrical panel and the pneumatic circuit are simple elements that perform one the function of feeding the PLC Omron, its remote boards, the robot board and the safety modules and the other supplies the air needed to close and open of the robot tool.

¹Since the accuracy of the rack position is crucial, the racks are positioned with an objective positioning system, thanks to the presence of two corners at right angles with a proximity sensor that indicates that the rack position is the correct one. The same applies to drawers which are properly opened if proximity sensors at the corners of their limit switch detect their edges. As for the closure, instead, there are magnets that keep the drawers still and in position when the robot closes them. Look at fig.??.

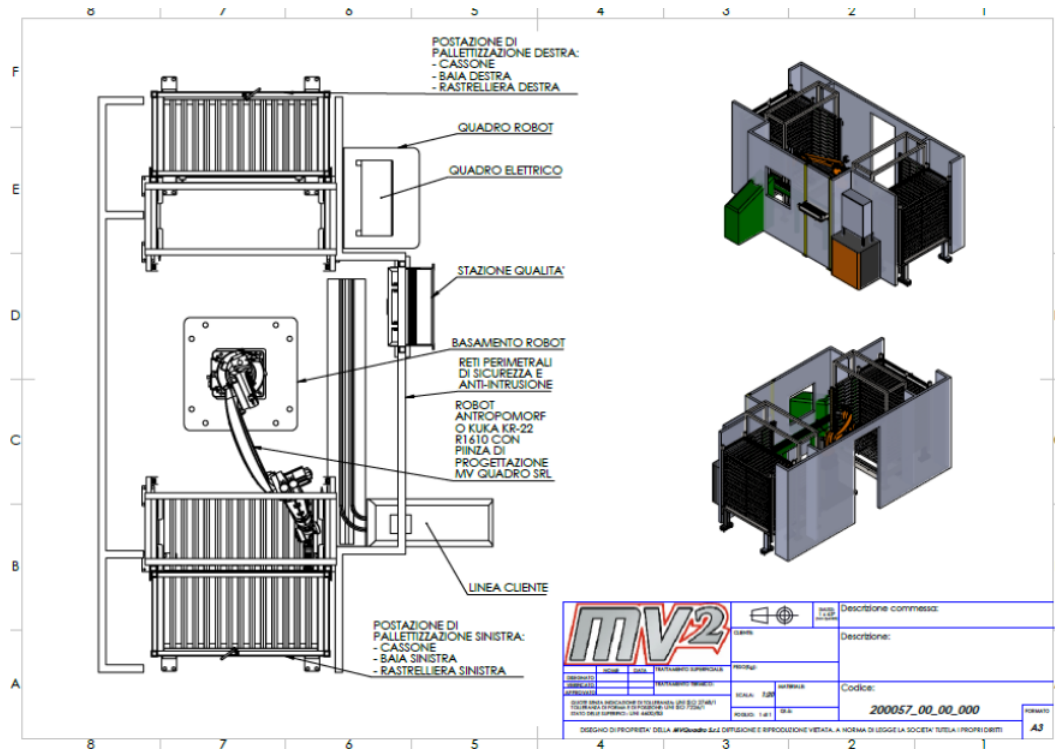


Figure 1.1: Overview of the entire project

The working process began first with the creation of a team of four people, consisting of a mechanical engineer, an electrical engineer and two automation engineers, including myself.

This division was necessary to define the tasks of each figure, in fact at the next stage each figure worked on a different area, in fact :

- The mechanical engineer has realized the design of the robot base, the racks, the drawers and the robot tool, providing the verification of the structural characteristics of the same through static analysis, as well as to design and validate the pneumatic circuit.
- The electrical engineer has provided for the drafting of the electrical panel, sizing the various components in accordance with the rules imposed by the state of the art, with particular attention to safety standards
- Automation engineers have taken care of the choice of the architecture of the program, deciding to divide the movements of the robot into categories and such categories in sub-programmes also called "missions".

As long as the electrical panel and the mechanical structures were not built, most of the work concerned the figures of mechanical and electrical engineers.

However, once the assembly phase of the robotic island: the connection of the electrical panels and the pneumatic circuit, etc. was finished, the programming work became the most consistent.

The full system we designed has the following shape: During this period we faced several issues:

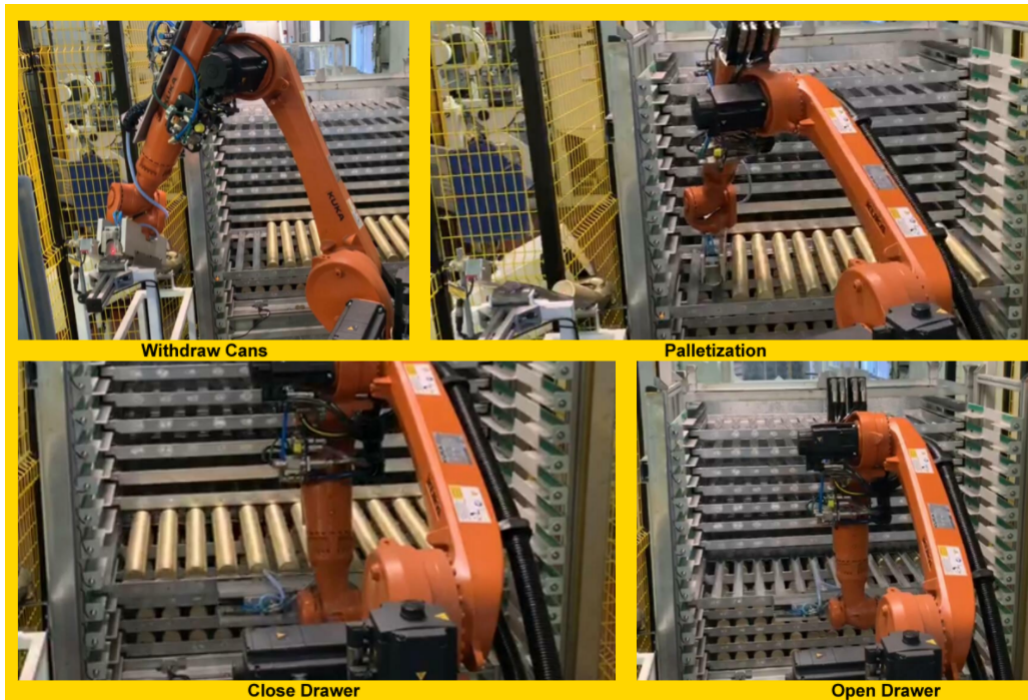


Figure 1.2: Main Required Functions

- Communication between devices with different types of Protocols, Ethercat and Profinet: this was due to the delays in shipments of Omron security modules, in fact we had to change safety devices and adapt to the more available Sick, but as said before they use the Profinet protocol instead of the Ethercat.
- Configuration of the safety positions of the robot: in practice, while the robot performs the movements necessary for the programmed functions, it happened that the power cables and tires twisted dangerously on the body of the robot with the risk of shearing. Therefore it was necessary to find intermediate positions between the configurations that the robot must necessarily reach, which we have defined "Safe" because they will certainly not create problems with the cables.
- Error Detection: In each programming project, errors occur, they can be obvious as an exchange of two positions, or not obvious as a variable that increases slightly at each cycle, which interferes with the accuracy of the robot, or an unexpected reset of the parameters via HMI, which starts the robot for a configuration completely different from the expected one. For these reasons it is always good to test your code at reduced speeds, even better without using the automatic command, but proceeding line by line of code.

At the end of this third phase of programming, the customer was shown the operation of the island and the changes they requested, which were all requests of little account, mainly interested in the increase of the speed of the robot.

Finally, we can say that although this seems a very simple task, the complexity behind it is fairly high.

This project contains several engineering figures, in fact there is need of mechanical engineers, electrical and software in order to create such a robotic system.

It is for this reason that it is interesting to go and see the design process element by element, going to capture the various aspects of this small robotized island.



Figure 1.3: Full mechanical structure, particulars of drawer and rack

1.2 Manipulator

The manipulator we have used in this project is a KUKA KR 22 R1610, according to the technical data, fig. 1.4, it has the following main characteristics:

1. Weight = 263 Kg
2. Payload = 22 Kg
3. Range = 1612 mm



Figure 1.4: Technical Data written on the manipulator arm

The complete list of all robot's parameter is available in the data-sheets provided by KUKA, but the most important one are the ones written above. In particular, we choose the robot basing on the load to be handled and on the available space, so the payload and range parameters are indispensable to choose the correct machine. Moreover we also need to know the robot's weight in order to secure it on a self designed base or to choose the proper fixing mechanisms.

This robot has 6 joints and each one of them has different characteristics, as shown in table 1.1. In the data-sheets, we also find the mechanical drawing of the attachment

Table 1.1: Range and Speed of each joint

Position [degrees]	Speed [$\frac{degrees}{s}$]
± 185	200
$-185/65$	175
$-138/175$	190
± 350	430
± 130	430
± 350	630

flange needed for the housing of a self-designed end effector, fig. 1.5. It is very important to understand, that a robot can sustain the maximum payload when the center of mass of the self designed end effector is as close as possible to the robot flange's one. In fact, as we can see from fig. 1.6, the more the distance between the position of the center of mass of the sole flange and the new center of mass position increases, the less weight the robot can sustain. Since the last joint rotation doesn't

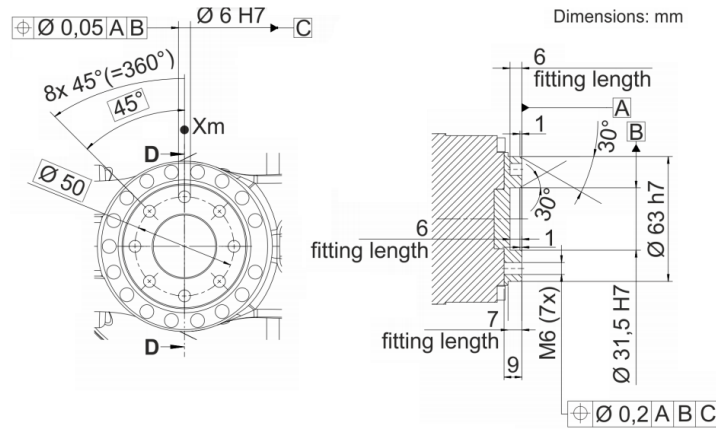


Figure 1.5: Attachment flange for the design of a self designed tool

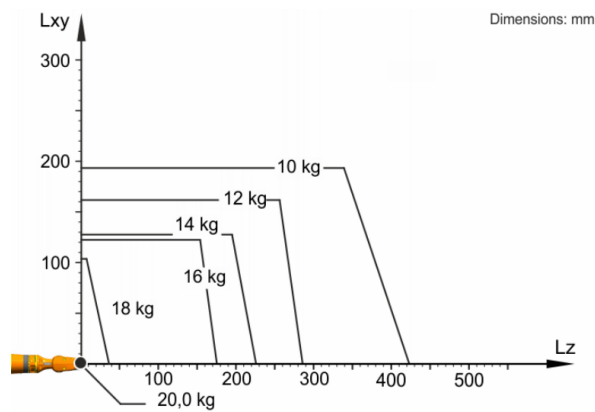


Figure 1.6: Nominal Weight vs Center of Mass Position

change the end effector tool center, in order to have a clear image of the robot full range it is sufficient to look at just the range of the 5th axis, as shown in the fig. 1.7

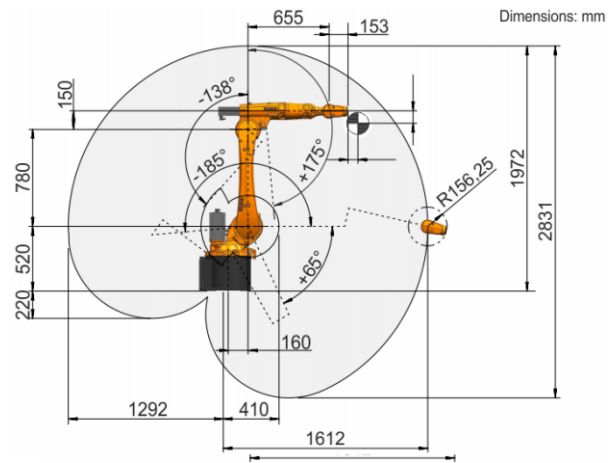


Figure 1.7: Range Space for the 5-th axis

1.3 PLC

A PLC or Programmable Logic Controller is the component responsible to the synchronization of all the tasks as well as the control unit of the system. It gives tasks to the manipulator and receives in input all the signals coming from the plant. The PLC that was chosen for this application is an Omron NX102-9000, fig. 1.8. Its characteristics are indicated here:

- Processing Time : 3.3ns LD / $\geq 70ns$ Maths instructions
- Program Capacity : 5 MB
- Memory capacity for variables : Retain attribute 1.5MB / No retain 32MB
- Number of data types : 1000
- Number of Ethernet Ip Ports : 2
- Maximum transmission distance between Ethernet switch and node : 100 m
- Communication protocol : Ethercat



Figure 1.8: PLC Omron NX102-9000

The PLC receives/send signals from the plant thanks to the following cards:

- 4 Digital Input 16 bit PNP 24V DC NX-ID5442
- 4 Digital Output 16 bit NX-OD5256
- 4 Analog Input 2 Word NX-AD2603

This cards are powered up by a supplementary power supply, fig.1.9, that is the NX-PF0630.



Figure 1.9: supplementary power supply NX-PF0630

1.4 End Effector - Tool design

It is possible to connect a self-designed tool to every robot, thanks to special holes placed on their flange. In the case of the KUKA kr 22, we find 8 holes. In order to be able to precisely connect the end effector to this holes, we have chosen to design seven holes which size's 6.6 mm and just one of 6 mm , fig.1.10. This is because, when the tool will be physically connected, it will always have some positions errors. That's why the tolerances to be chosen in this context should be all very stringent. But in order to save money and reach the same result, it is common practice to fabricate just one hole very precisely, so that the others will fit anyway into the robot's flange. The tool is made entirely of aluminum, as it is a resistant but light material and so its weight does not burden the robot. The part of the tool that grabs the lids is coated with a layer of rubber that provides the right coefficient of friction to grab the lids and also ensures that they are not damaged during the gripping.

The structure of the tool is preferably symmetrical in the x and y coordinates² in order to evenly distribute the weight of the tool with respect to the center of the robot flange, fig.1.11. Very important is to provide housing for the Safety Tool,

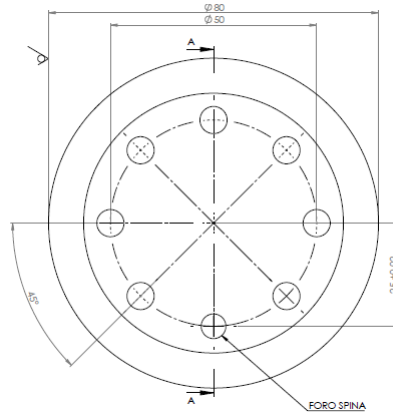


Figure 1.10: Tool Interconnection Holes Mechanical Drawing

which is an optional package sent by the robot constructor company, that is able to ensure that the designed tool is not substituted or compromised.

In fact, every time the robot is powered up, it will reach a given point in the 3D space. This point will make the safety tool touch a specific sensor that will enable robot's operations. If the sensor does not acknowledge its presence, the robot will not be able to start the program cycle. For what concerns the pneumatic part, the tool is equipped with six pneumatic cylinders, sized according to their strokes and also after a trial and error procedure.

Of course, there is the possibility to carry out more precise mathematical analyses, in order to better size the cylinders, but these analyses would cost more in terms of money and time, than a trial and error solution. In the context of this application, a total stroke of 10 mm was required and therefore we opted for cylinders that guaranteed this specific. Their number is sized according to the tool's weight. In particular the sum of the payloads of the n cylinders, P_{cyl} , calling W is the total

²The reference frame we are referring to is the same as the robot base frame, and it will be defined later.

1. The beam is a swiped plane section along its normal vector's direction
2. Load and constraints are concentrated at the beam ends
3. Stresses distribution are assumed to be uniform on the cross section
4. The beam material is linear,elastic,homogeneous and isotropic

Under these hypothesis, we can model the displacement of a point,fig. 1.12, according to the eq.1.3.

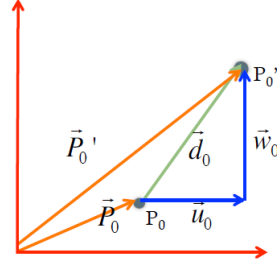


Figure 1.12: Displacement of a generic point P of a beam

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{Bmatrix} u_0 \\ v_0 \\ w_0 \end{Bmatrix} + \begin{Bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{Bmatrix} \begin{Bmatrix} r_x \\ r_y \\ r_z \end{Bmatrix} \quad (1.3)$$

This equation, describes the shear and strain stresses due to both rigid displacement and rotations of every point P of a beam. Since the material is assumed to be linear we can use the Generalized Hooks law for isotropic materials, eq.1.4.

$$\begin{cases} \epsilon_{xx} = \frac{\sigma_{xx}}{E} - \frac{\nu}{E} (\sigma_{yy} + \sigma_{zz} + \alpha \cdot \Delta T) \\ \epsilon_{yy} = \frac{\sigma_{yy}}{E} - \frac{\nu}{E} (\sigma_{xx} + \sigma_{zz} + \alpha \cdot \Delta T) \\ \epsilon_{zz} = \frac{\sigma_{zz}}{E} - \frac{\nu}{E} (\sigma_{xx} + \sigma_{yy} + \alpha \cdot \Delta T) \\ \gamma_{xy} = \frac{1}{G} \tau_{xy} \\ \gamma_{xz} = \frac{1}{G} \tau_{xz} \\ \gamma_{zy} = \frac{1}{G} \tau_{zy} \end{cases} \quad (1.4)$$

Whenever we apply an external load, because of the equilibrium principle, there must be some other kind of forces that tend to balance the external ones. Those forces are called *Internal Forces* or *Internal Actions*. In general, we can derive the expressions that link the stresses, strain and displacement with internal forces and geometrical characteristics of the beam. For example in the presence of displacements along the x axis, we have:

$$\sigma_{xx} = \frac{N}{A} \quad \epsilon_{xx} = \frac{N}{EA} \quad w(x) = \frac{N}{EA} x \quad (1.5)$$

For Bending moments we have:

$$\sigma_{xx} = \frac{M}{J_{zz}} y \quad (1.6)$$

For torques, we have:

$$\tau = \frac{M_t}{J_p} r \quad \theta = \frac{Cl}{GJ} \quad (1.7)$$

Clearly, every material has a different strain stress graph, but in general we can define three main working regions:

1. Linear: every deformation is reversible
2. Non linear: the deformations are not reversible
3. Rupture point (UTS³): crack nucleation

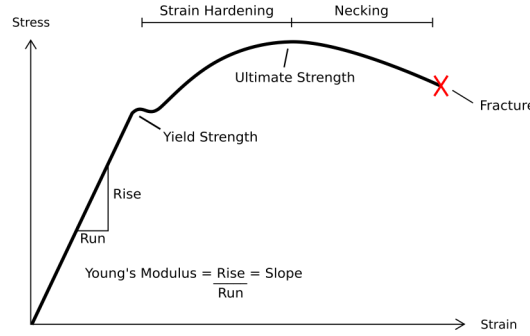


Figure 1.13: Stress Strain Ductile Material graph

When a body reaches the UTS point, it will experience crack nucleation and therefore permanent rupture. In order to be sure that the material we are working with will not fail under static stress and because of the high complexity of the mathematical model, we use to rely on the ideal equivalent stress σ_{id} . Thanks to this quantity we are basically comparing a multi axial system with a uni axial one, simplifying the calculus and obtaining a good approximation of the maximum and minimum stresses that an object undergo to. If we call σ_1 , the maximum principal stress and σ_3 the minimum one, in order to be sure that the body will not fail, it must be:

$$|\sigma_1 - \sigma_3| < R_{p02} \quad (1.8)$$

Where R_{p02} is a constant corresponding to the UTS limit. Since we are reducing a multi axial problem with a uni axial one, we need to define the ideal equivalent stress and this is usually done by choosing one of this three quantities:

$$\begin{aligned} \sigma_{id}^{galileo} &= \sigma_1 \\ \sigma_{id}^{tresca} &= |\sigma_1 - \sigma_3| \\ \sigma_{id}^{mises} &= \frac{1}{\sqrt{2}} \sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_1 - \sigma_3)^2 + (\sigma_3 - \sigma_2)^2} \end{aligned} \quad (1.9)$$

³Ultimate Tensile Stress

We have chosen the Von Mises equivalent ideal stress, because, computationally speaking, it is the fastest one, moreover the difference between Tresca and Von Mises result is at most 3%. To perform the static analysis, an object grid mesh was generated through a simulation software (Solid Works), to which the Von Mises criterion is applied systematically to every point inside the mesh. Then thanks to further interpolation of the results, the program outputs a total stress map, that is shown in fig. 1.14 As we can see from the following picture, fig. 1.15 the mesh has

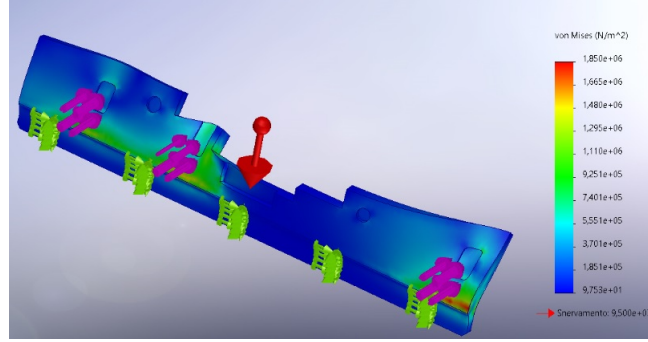


Figure 1.14: Static Analysis - Yield stress

a triangular profile. It is possible to use other geometric shapes as a mesh matrix, but the triangular one is often used because it is the one that is able to better approximate the surfaces of an object, because for three points lies one and only one plane. The precision of these meshes is excellent even in the presence of holes or discontinuities in the object of study. The objective of this analysis, fig.1.14 is

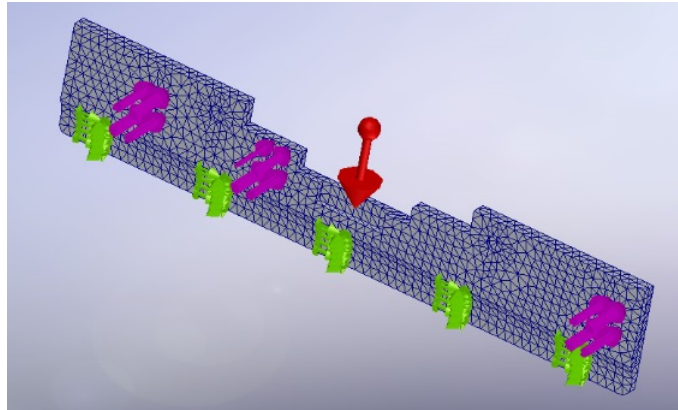


Figure 1.15: Triangular mesh profile for static analysis purpose

to determine whether the designed tool is robust and the highest stress has a lower modulus than the UTS point of the material used (in the case of aluminum it is worth $9.5 \cdot 10^7 \frac{N}{mm^2}$). During the analysis of the deformations, as we can see in the next figure, we want to ensure that the tool has a maximum deformation of less than some tenths of mm. In order to properly setup the tool and the robot, we also need to know its inertia matrix. This is, however, very difficult to calculate, as we need to use of dedicated software to obtain reliable estimates (e.g. Ansys). In addition, such simulations may take several days to produce a result. For this reason, the programmer proceeds through a trial and error approach, following the values of robot inertia, payload, linear and angular velocities reported on the robot

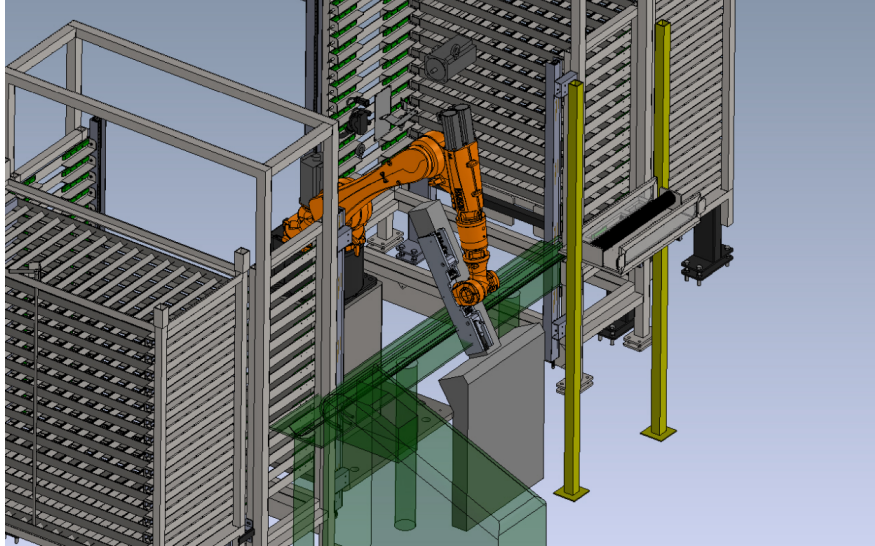


Figure 1.16: First Reachability Analysis

data sheet and also basing on the position of the tool center of mass, that can be obtained from a common cad software.

1.5 Distribution boards & data-sheets

When designing an electric panel, the first thing one wonder is how many electrical, mechanical, pneumatic devices need to be driven and how much power should we provide. So, once we have a list, fig. 1.17, of all the necessary components, we begin to size the circuit according to the needs of the plant. In our case, we have a three-phase power supply for the robot electrical panel and a single-phase power supply (obtained from one phase of the previous one) to power up both the PLC and the signal devices. It's a good practice to separate the framework into two areas:

- power zone, where there are larger voltages and currents that are dangerous for operator safety
- signal part, where there are smaller voltages and currents who represent a minor problem to solve

The electric panel is therefore designed as shown in the figure:

Nome/Item	Tipo/Type	Descrizione/Description	Costruttore/Mark	Quadro/Board/Fg/Sh	Qta/Uhj
9A1	2702975	PSI-ukhPSIB-CT-CH-EN-1-B	PHOENIX CONTACT	+005	8
	2702973	PSI-ukhPSIB-CT-C-CT	PHOENIX CONTACT	+005	9
			PHOENIX CONTACT PHOENIX CONTACT		1
9A2	2702975	PSI-ukhPSIB-CT-CH-EN-1-B	PHOENIX CONTACT	+005	9
	2702973	PSI-ukhPSIB-CT-C-CT	PHOENIX CONTACT	+005	1
			PHOENIX CONTACT PHOENIX CONTACT		1
A1.0	NX102-9000	Controller di automazione della macchina	Omron	+005	4
A1.1	NX-FF1x30	Modulo silenzioso supplementare I/O	Omron	+005	4
A1.2	NX-DS4x2	Modulo I/O ingressi digitali PNP 2x V DC	Omron	+005	4
A1.3	NX-DS5154	Modulo I/O uscite digitali PNP 2x V DC	Omron	+005	4
A1.4	NX-AD7x153	Modulo I/O ingressi analogici 0/5 V (single-ended input)	Omron	+005	4
AS.0	NX-FF1x30	Modulo silenzioso supplementare I/O	Omron	+005	5
AS.1	NX-SDB810	Modulo I/O ingressi di sicurezza PNP 2x V DC	Omron	+005	5
AS.2	NX-SDB810	Modulo I/O ingressi di sicurezza PNP 2x V DC	Omron	+005	5
AS.3	NX-SDB810	Modulo I/O uscite alla corrente di sicurezza PNP 2x V DC	Omron	+005	5
AS.4	NX-SDB810	Modulo I/O uscite alla corrente di sicurezza PNP 2x V DC	Omron	+005	5
AS.6	NX-SDB810	Modulo I/O uscite di sicurezza PNP 2x V DC	Omron	+005	5
BFL.14	EFB-4P21	Servono relè a transistor - Circuito c.a. 230V a riflettore polarizzato 0,1-4m regolo: PNP impulso luce/bolo com/NC	Omron	+005	16
BFL.15	E3N-B15	Accessori sensori - Catartografi 1/1 3x59 9 mm	Omron		1
	5A8389	SAC-4P-10.0-PLC/10278 Cavo sensore	Phoenix Contact		1
	E3N-B15	Servono relè a transistor - Circuito c.a. 230V a riflettore polarizzato 0,1-4m regolo: PNP impulso luce/bolo com/NC	Omron	+005	16
	5A8389	Accessori sensori - Catartografi 1/1 3x59 9 mm	Omron		1
		SAC-4P-10.0-PLC/10278 Cavo sensore	Phoenix Contact		1

Figure 1.17: Fragment of the Total element list

1.5.1 Power

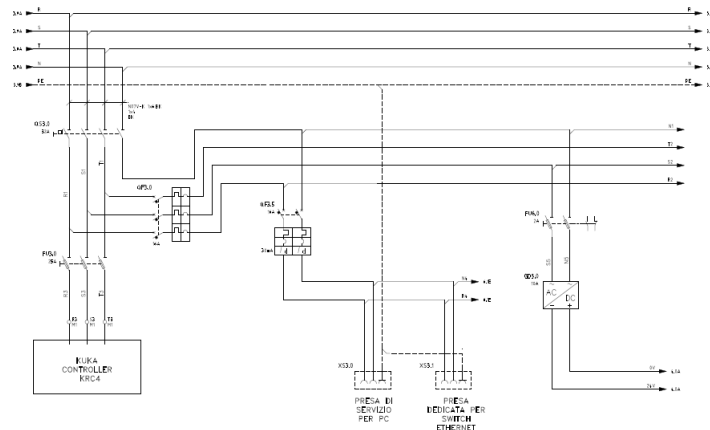


Figure 1.18: Fragment of the Power Zone

At the top, there's the power zone, that provides power to the electric panel of the robot. In it we find :

- Fuses
- Differential switches
- Isolator

Since the power part, fig.1.18, is the one that puts the safety of people more at risk, it is necessary to calculate an estimate of the powers in play and use graphs, in order to properly size the differential switches and the size of the fuses to be adopted. The role of fuses and that of a differential switch is exactly the same, the fundamental difference lies in the price, so often in the electrical panels are installed more fuses than the differential switches. Conversely, fuses only dissect the circuit when the internal metal conductor, connecting the two ends, fuses. So every time a fuse dissects the circuit it has to be replaced. The isolator, instead, is a purely mechanical contractor and therefore does not intervene in the event of over-current or over-voltage. Its purpose is to make it possible for an operator to cut power to a part of the circuit in order to carry out maintenance operations.

1.5.2 Signal

Below, fig.1.19, we find the PLC's UPS power supply, the PLC itself, with its remote boards and SICK safety modules. The UPS module is designed to ensure continuous operation even in the event of a power failure, while the safety modules are special devices that are required by IEC standards. Their task is to continuously monitoring the status of the circuit and dissecting it in the event of failure. The problem we faced when designing the framework, was the different network interface that the PLC and the SICK module had. In fact, the former uses an Ethercat protocol, while the latter uses a Profinet protocol. In a previous version of the same application, the company had used as security modules of the same Omron brand, but because of shipment delay caused by the pandemic, we had to search for another solution. With this choice, there is no direct connection between PLC and safety modules,

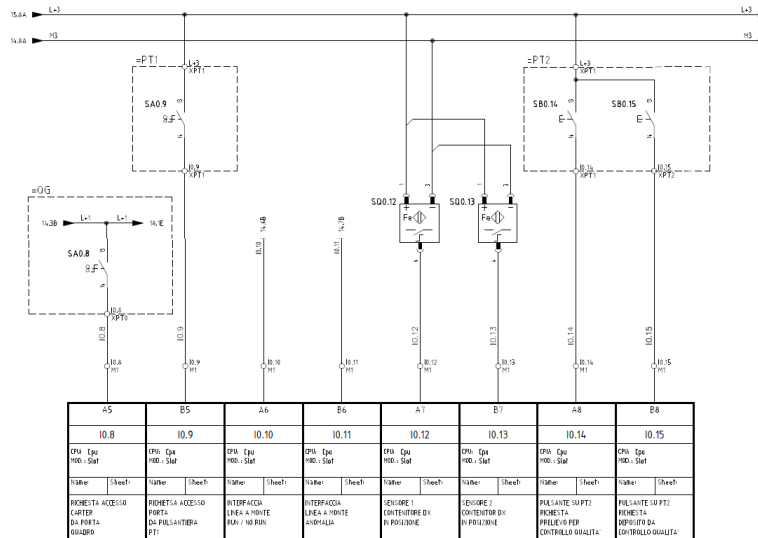


Figure 1.19: Fragment of the Signal Zone

but on the contrary the safety modules are completely independent from the PLC and are connected to the same devices to which the PLC is connected. In order to ensure both:

- effective operation of the security
- possibility of the PLC to read data

1.5.3 Terminal box

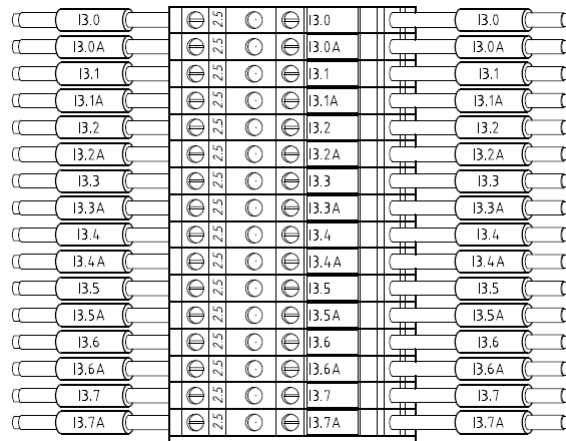


Figure 1.20: Fragment of the Terminal box Zone

In the lowest part of the electrical panel, we find the terminal block, fig.1.20, that it is the electrical interface connecting the PLC panel to the robot and to the signals coming from the field. Finally, the electrical classification of the system is TT, in fact all the masses connected to the plant share the same earth terminal. This is necessary to prevent that in occurrence of electrical failures, the foreign devices masses become active parts and therefore cause of indirect contacts. The electrical panel of the robot is an integral part of the manipulator and therefore does not have

to be designed. Inside there are all the necessary components to be able to control the brush-less motors of each joint of the robot, as well as the processors able to control it, through algorithms of inverse kinematics, calculation of the trajectory, safety management and much more.

1.6 Pneumatic Circuit

For what concerns the pneumatic circuit, its objective is to make the gripper be able to open and close. Because of this, we have used six pneumatic cylinders, who are able to exert the right amount of force in order to accomplish this operations. In order to supply and treat the air needed for the correct functioning of the cylinders, we designed an input circuit with the following elements:

1. Regulator-Filter Ref. PNEUMAX T171BEMAD is used to filter liquid or solid particles and regulate the pressure of compressed air
2. Compressed Air Lubricators REF. PNEUMAX T171BL it is installed in a compressed air line to lubricate pneumatic tools.
3. Electric Shut-Off Valve T171BVEB2 Automated shut-off valve with electric controls and actuator
4. Progressive starter T171BAP The task of the air intake is to smooth the transition from low pressures levels to highest ones, in order to not damage the pneumatic equipment.
5. Electric Shut-Off Valve T171BVEB2 Automated shut-off valve with electric controls and actuator

The complete air treatment group is shown in fig. [1.21](#).

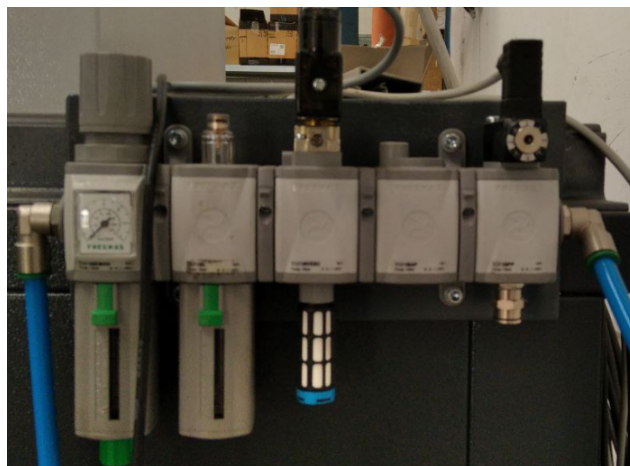


Figure 1.21: Air Treatment Group

For what concerns the cylinders, we have used the following ones:

1. DPDM-6-5-P-PA

Who's specifics are shown in table [1.2](#). The complete pneumatic circuit is shown in fig. [1.22](#). Finally, the pneumatic circuit for the gripping and releasing is shown in fig. [1.23](#)

Table 1.2: Characteristics of the DPDM-6-5-P-PA cylinders

Parameter	Value
Stroke	5 mm
Piston Diameter	6 mm
Cushioning	Elastic cushioning rings/plates at both ends
Operating pressure	2.5 ÷ 8 bar
Theoretical force (in/out-stroke)	9 N

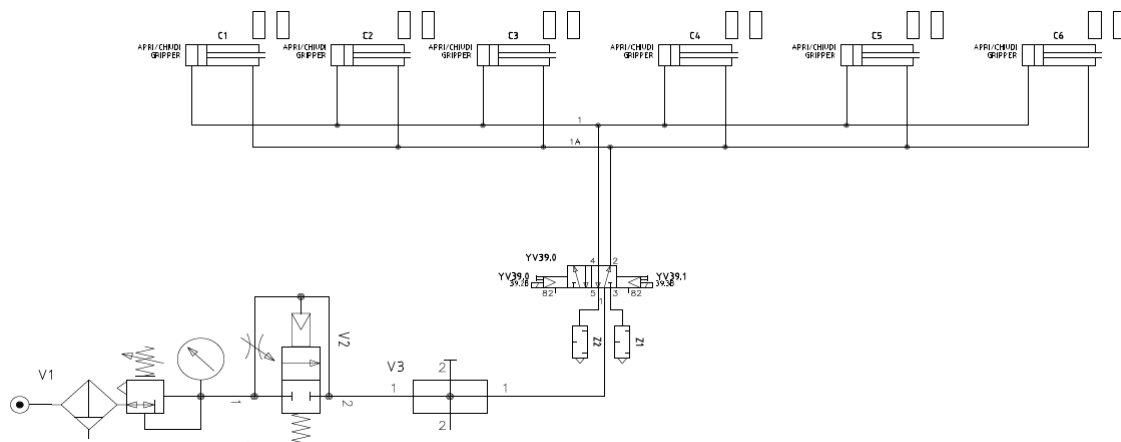


Figure 1.22: Complete Pneumatic Circuit

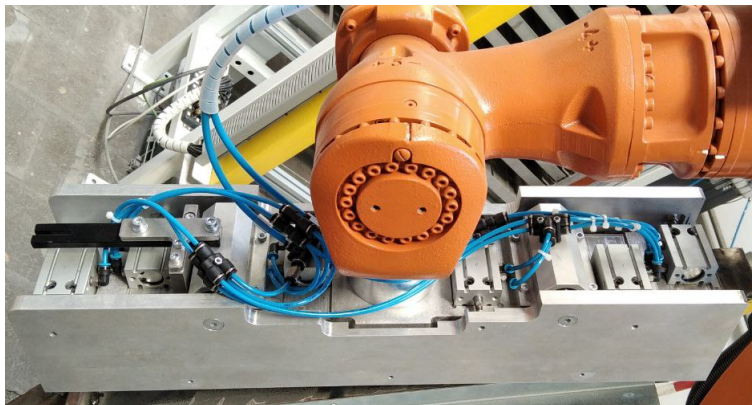


Figure 1.23: Pneumatic Cylinder on the robot's tool

1.7 Safety

1.7.1 SIL - Safety Integrator Level

IEC 61508, [3], is an international standard published by the International Electrotechnical Commission consisting of methods on how to apply, design, deploy and maintain automatic protection systems called safety-related systems.

It is titled Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES). IEC 61508 is a basic functional safety standard applicable to all industries. It defines functional safety as:

“Part of the overall safety relating to the EUC (Equipment Under Control) and the EUC control system which depends on the correct functioning of the E/E/PE safety-related systems, other technology safety-related systems and external risk reduction facilities.”

The calculation of the probability of failure on demand (PFD)[9] is a common engineering task when designing an interlock in compliance with IEC 61511. The calculation of the PFD is often done using approximate equations defined in the ISA TR84.00.02 technical report. The PFD corresponds to one of four safety integrity levels (SILs), where each level possesses a PFD that is one order of magnitude less than the next. There are several methods used to assign a SIL. These are normally used in combination, and may include:

1. Risk matrices
2. Risk graphs
3. Layers of protection analysis (LOPA)

Safety integrity level	Probability of failure on demand (PFD _{avg})	Risk reduction factor (ÅR)
4	$10^{-4} > \text{PFD}_{\text{avg}} > 10^{-5}$	10,000 ÅR < 100,000
3	$10^{-3} > \text{PFD}_{\text{avg}} > 10^{-4}$	1,000 ÅR < 10,000
2	$10^{-2} > \text{PFD}_{\text{avg}} > 10^{-3}$	100 ÅR < 1,000
1	$10^{-1} > \text{PFD}_{\text{avg}} > 10^{-2}$	10 ÅR < 100

Figure 1.24: SIL - Safety Integrity Levels

1.7.2 Safety of human operators

In order to prevent electrical shocks and damages to circuit's components, there exist some normative⁴, that define general rules to adopt for the design of an electrical panel.

In particular, when an operator is working on an electrical circuit, it is possible to define two main types of contacts:

- Direct contact: when the operator touches a live wire, or any other conductive material, who is not isolated.

⁴The most important are the: IEC 61140, IEC 60364, EN 60204-1, CEI 64-8

- Indirect contact: when the operator touches a part of the electric circuit, who is normally not conductive, but in case of faults, it can be seat of current flowing and/or have a not null voltage potential.

1.7.3 Safety regions

Industrial manipulator can reach very high speeds and accelerations and because also of their consistent weight (e.g. a Kuka KR22 R1610 weights 263Kg) they represent a risk for operator's safety. For this reason it is needed to design special strategies that are able to minimize the risk of human/robot collisions. So, it is very common to use barriers in order to prevent operators from getting too close to the robot, but it is also very common to use software solutions, like defining work zones. For example, we can subdivide the whole working region of the manipulator as :

- Working Zone: the robot can reach full speed since no collisions human/robot can happen
- Safe Zone: the robot can either stop or slow down in order to not cause damage to operators

In our case we have defined three safety zones, where the robot has to stop immediately. In fact, when there is a missing drawer, it is possible for an operator to enter inside the robotized area and be very close to the manipulator. In this case, as soon as the manipulator enters in that region it must stop. In order to restart the cycle, one has to acknowledge the alarm and then press the start cycle button present in the electrical panel cabinet. A representation of the safety planes is shown in the [fig.1.25](#)

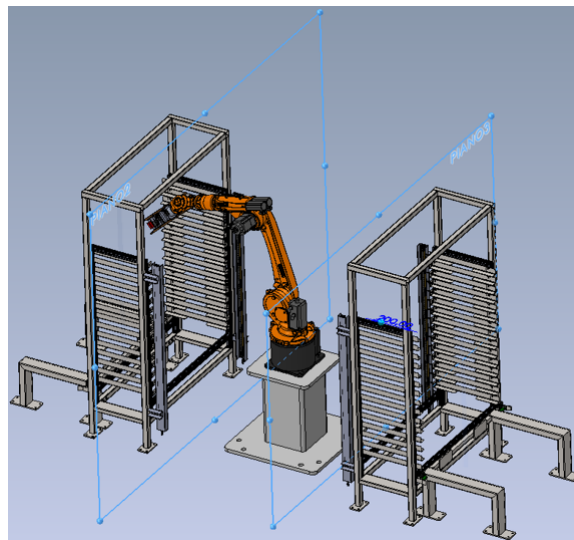


Figure 1.25: Safety Planes activated when there is no drawer

Chapter 2

Robotics basic concepts

2.1 Introduction

In order to properly describe a robot motion it is needed to obtain a mathematical model who can simulate any desired trajectory. Since a robot is a device who can move in the 3-D space, we need a model which is able to describe positions, velocities and accelerations in the 3D space. Moreover, because the robot joints can also rotate, we need to define three parameters that give us insight about the robot's orientation in the 3D space. Summing up our needs, we need three variables for the position description and other three for describing the orientation. Let's start considering the problem of determining the position of an object in a three dimensional space.

2.2 Basics

2.2.1 Elementary displacements

Each point in the 3D space has three coordinates and can be subject to translations, rotations or roto-translations. When a body undergoes a translation its coordinate only change in the same translation direction. Meaning that in order to describe this simple motion is enough to get the direction of movement and sum this vector to each of the body points. As shown in fig.2.1, we only need to know three variables. For what concerns pure rotation, in order to describe how this operation affects

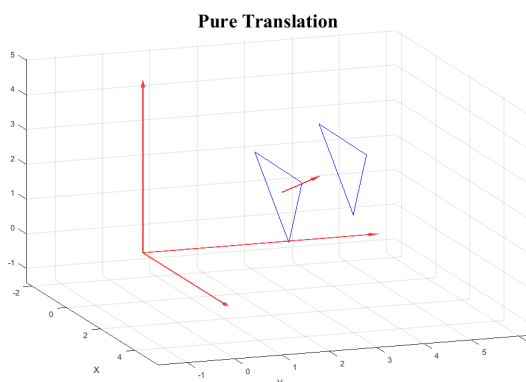


Figure 2.1: Pure Translational motion

a body position, we need to consider both the axis of rotation and the quantity about which we rotate around it. So we need three variable to get the axis direction and one variable, which describe the amount of rotation to be accomplished. But once we find the axis direction, in what way will we rotate around it? Clockwise or counterclockwise? It is a convention to choose always as positive rotation side the counterclockwise one, according to the right hand rule. Intuitively, when a body

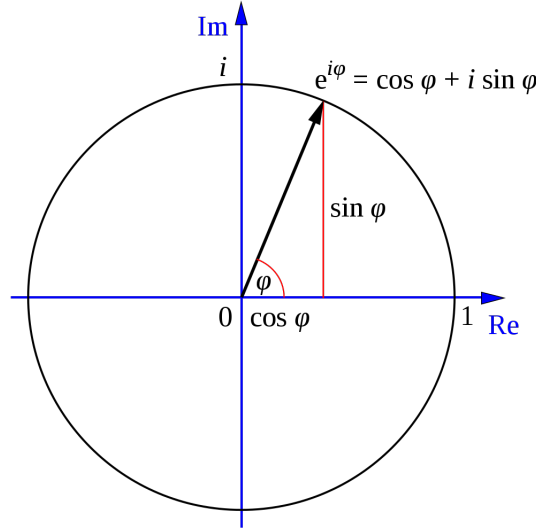


Figure 2.2: Euler Identity

rotates it follow a circular shape motion, so every one of its point will keep the same distance from the center of rotation. Moreover, if we rotate with respect to some axis, all the point which lies on the axis itself won't rotate, so, mathematically, we just need to consider the problem of a plane which rotates with respect to a axes perpendicular to itself, which makes this problem kind of bi-dimensional. Consider the equation of a circle

$$x^2 + y^2 = r^2 \quad (2.1)$$

This equation preserves the distance of each point of the circumference with respect to the circle's center. Let's see this equation in another way :

$$\begin{cases} X = r \cos(\theta + \phi) \\ Y = r \sin(\theta + \phi) \end{cases} \quad (2.2)$$

where X,Y are the new coordinates obtained from rotation. By noticing that

$$\begin{cases} x = r \cos(\phi) \\ y = r \sin(\phi) \end{cases} \quad (2.3)$$

Applying the cosine and sine sum formula

$$\begin{cases} X = r (\cos(\theta)\cos(\phi) - \sin(\theta)\sin(\phi)) \\ Y = r (\sin(\theta)\cos(\phi) + \cos(\theta)\sin(\phi)) \end{cases} \quad (2.4)$$

we can derive the following system

$$\begin{cases} X = x\cos(\theta) - y\sin(\theta) \\ Y = x\sin(\theta) + y\cos(\theta) \end{cases} \quad (2.5)$$

which can be written in matrix form, as in eq.2.6

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.6)$$

The last equation is known as the elementary rotation around the z axis. The other two elementary rotation matrices can be found in a similar way and all of them are written in eq.2.7.

$$\begin{aligned} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ 0 & 1 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \end{aligned} \quad (2.7)$$

Lastly rototranslational movement can be obtained summing up both this elementary displacements. In particular in order to fully describe the position of a point which undergoes a rototranslational motion we use homogeneous coordinates and we represent the position as a 4D vector. Moreover we embed the position and the rotation in a matrix whose called homogeneous matrix which structure is indicated in fig.2.3. A rigid body B can be represented by a reference frame RB associated

$$\mathbf{T}_b^a \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{R}_b^a & \mathbf{t}_b^a \\ \mathbf{0}^T & 1 \end{bmatrix}$$

$$\mathbf{0}^T \stackrel{\text{def}}{=} [0 \ 0 \ 0]$$

Figure 2.3: Homogeneous Matrix structure

to it, called "body frame". We call pose of a rigid body the set of parameters that uniquely define its position and orientation (attitude) in R3. The pose can be obtained from the homogeneous transformation T0B The body pose is formally defined as

$$p(t) := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \quad (2.8)$$

Where x is a geometrical vector, while α cannot be considered a vector since vector operations are meaningless.

2.2.2 Cardan angles

Each rotation matrix R can be built by the composition of three elementary rotations

$$R = Rot(u_1, a_1)Rot(u_2, a_2)Rot(u_3, a_3) \quad (2.9)$$

There are twelve admissible combinations of rotations that form the set of Cardan

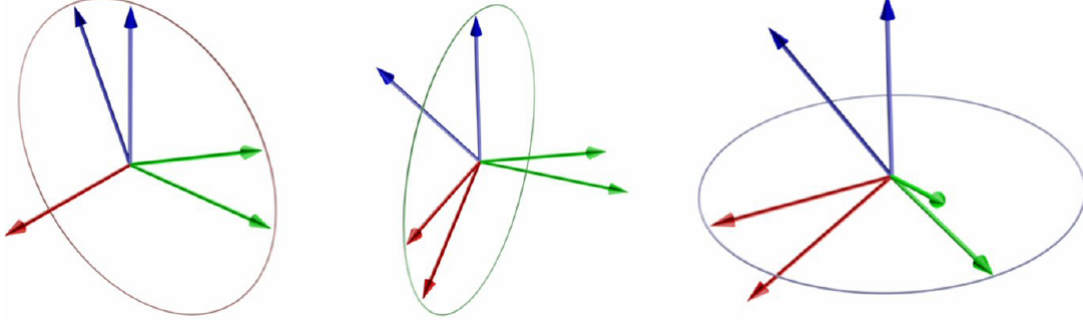


Figure 2.4: RPY Angles

angles.

Among the Cardan angles, one of the most common used are the Roll, Pitch, Yaw (RPY) angles $(q_x; q_y; q_z)$, see eq.?? also known as Tait-Bryant angles, defined as follows

$$R(q_x; q_y; q_z) = R(\hat{k}; q_z)R(\hat{j}; q_y)R(\hat{i}; q_x) \quad (2.10)$$

$$\begin{bmatrix} c(\theta_y)c(\theta_z) & -c(\theta_y)s(\theta_z) & s(\theta_y) \\ s(\theta_x)s(\theta_y)c(\theta_z) + c(\theta_x)s(\theta_z) & -s(\theta_x)s(\theta_y)s(\theta_z) + c(\theta_x)c(\theta_z) & -s(\theta_x)c(\theta_y) \\ -c(\theta_x)s(\theta_y)c(\theta_z) + s(\theta_x)s(\theta_z) & c(\theta_x)s(\theta_y)s(\theta_z) + s(\theta_x)c(\theta_z) & c(\theta_x)c(\theta_y) \end{bmatrix} \quad (2.11)$$

2.2.3 Robot basic structure

Every robot can be seen as modeled, as in fig.2.5, with the help of:

- Links/arms are idealized geometrical bars connecting two or more joints
- Joints are idealized physical components allowing a relative motion between the attached links
- Joints allow a single “degree of motion” (DOM) between the connected links
- Joints may be of two types:
 1. Revolute (or rotational) joints, they allow a rotation between the connected links
 2. Prismatic (or translation) joints, they allow a translation between the connected links
- Other types are possible

The robot joints are moved by actuators and when they are not, they are called passive joint. Robots structures can be further classified, fig.2.6, as:

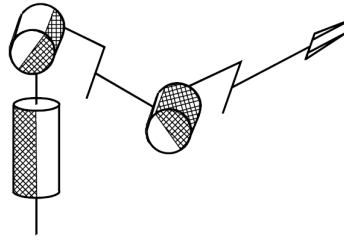


Figure 2.5: Schematization of a robot - Links and Revolute Joints

- Open chains: when there is only one link between any two joints. The Kuka kr 2210 has the three-like structure
- Closed chains: when there is more than one link between two joints.

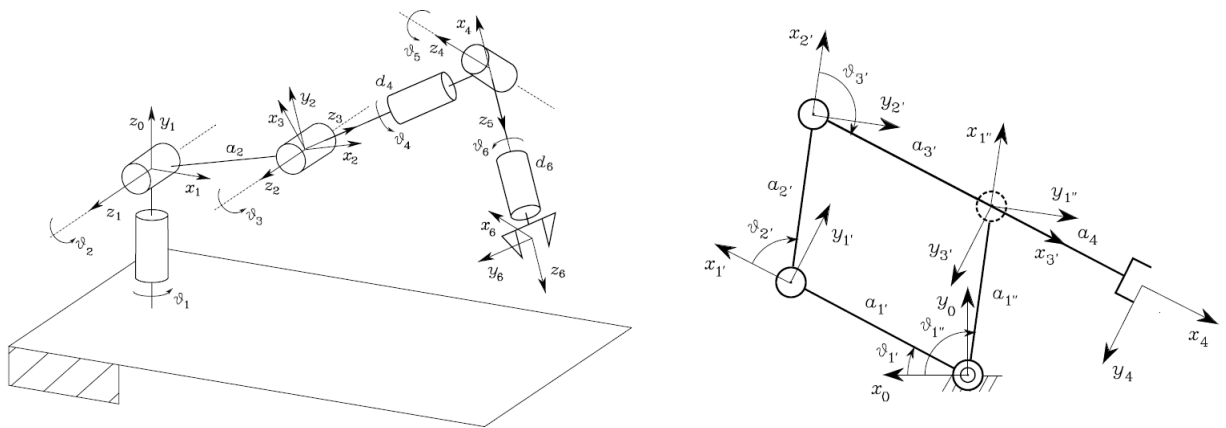


Figure 2.6: Schematization of a robot - Open and closed chains, respectively on the left and right

2.2.4 Basic Terminology

Let's introduce some basic terminology that will be useful throughout this reading.

- The following terms mean the same thing: *End effector*, *gripper*, *hand*, *end tool*. They identify last link ending structure, that is able to perform the required task or can hold a tool.
- With TCP we refer to the ideal point on the end effector that the robot software moves through space. It has an associated reference frame and moves in a 3D cartesian/euclidean space called Task Space.
- The Task space is the subset of the cartesian space that can be reached by the TCP
- The Joint Space is the mathematical structure whose elements are the joint values.

Kinematics deals with the study of four functions (called kinematic functions or KFs) that mathematically transform joint variables into cartesian variables and vice versa :

1. Direct Position KF: from joint space variables to task space pose
2. Inverse Position KF: from task space pose to joint space variables
3. Direct Velocity KF: from joint space velocities to task space velocities
4. Inverse Velocity KF: from task space velocities to joint space velocities

The first step to approach kinematics is to fix a reference frame (RF) on each robot arm

- In general, to move from a RF to the following RF, 6 parameters are required
- A number of conventions were introduced to reduce the number of parameters and to find a common way to describe the relative position of reference frames
- Denavit-Hartenberg conventions were introduced in 1955 and are still widely used in industry

2.2.5 Denavit-Hartenberg Convention

In order to represent the position of a n-link manipulator, we need to consider its homogeneous transformation matrix, that relates the joints angular position and the geometric manipulator characteristics, with the end effector tool position in space. This matrix can be easily found thanks to the use of the Denavit-Hartenberg(DH) rules. The latter are meant to simplify the computation of homogeneous matrices. In particular, the rules are the following ones: With reference to fig.2.7, let axis i

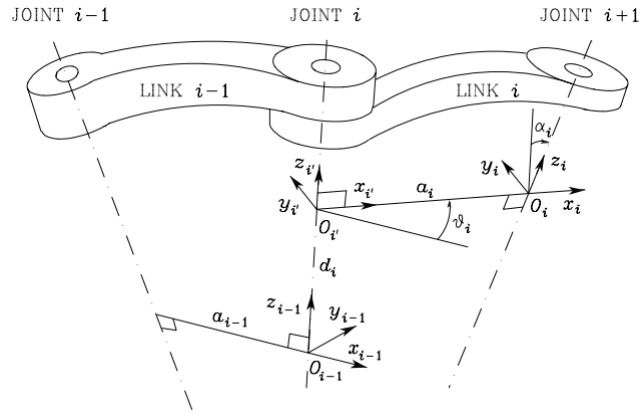


Figure 2.7: Denavit-Hartenberg kinematic parameters

denote the joint axis connecting Link $i - 1$ to Link i ; the DH convention is adopted to define link Frame i :

1. Choose axis z_i along the axis of Joint $i + 1$.
2. Locate the origin O_i at the intersection of axis z_i with the common normal to axes z_{i-1} and z_i . Also, locate O_i at the intersection of the common normal with axis z_{i-1} .
3. Choose axis x_i along the common normal to axes z_{i-1} and z_i with direction from Joint i to Joint $i + 1$.
4. Choose axis y_i so as to complete a right-handed frame.

The Denavit–Hartenberg convention gives a non unique definition of the link frame in the following cases:

1. For Frame 0, only the direction of axis z_0 is specified; then O_0 and x_0 can be arbitrarily chosen.
2. For Frame n , since there is no Joint $n + 1$, z_n is not uniquely defined while x_n has to be normal to axis z_{n-1} .

Typically, Joint n is revolute, and thus z_n is to be aligned with the direction of z_{n-1} . The common normal between two lines is the line containing the minimum distance segment between the two lines.

1. When two consecutive axes are parallel, the common normal between them is not uniquely defined.
2. When two consecutive axes intersect, the direction of x_i is arbitrary.
3. When Joint i is prismatic, the direction of z_{i-1} is arbitrary.

Once the link frames have been established, the position and orientation of Frame i with respect to Frame $i - 1$ are completely specified by the following parameters:

1. a_i distance between O_i and O_{i-1}
2. d_i coordinate of O_i along z_{i-1}
3. α_i angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise,
4. θ_i angle between axes x_{i-1} and x_i about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.

Two of the four parameters (a_i and α_i) are always constant and depend only on the geometry of connection between consecutive joints established by Link i . Of the remaining two parameters, only one is variable depending on the type of joint that connects Link $i - 1$ to Link i . In particular:

1. if Joint i is revolute the variable is θ_i ,
2. if Joint i is prismatic the variable is d_i

At this point, it is possible to express the coordinate transformation between Frame i and Frame $i - 1$ according to the following steps:

1. Choose a frame aligned with Frame $i - 1$.
2. Translate the chosen frame by d_i along axis z_{i-1} and rotate it by θ_i about axis z_{i-1} ; this sequence aligns the current frame with Frame i and is described by the homogeneous transformation matrix

$$A_{i-1}^i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.12)$$

Translate the frame aligned with Frame i by a_i along axis x_i and rotate it by α_i about axis x_i ; this sequence aligns the current frame with Frame i and is described by the homogeneous transformation matrix

$$A_{i-1}^i = \begin{bmatrix} A_i = 1 & 0 & 0 & ai \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.13)$$

The resulting coordinate transformation is obtained by post-multiplication of the single transformations as

$$A_{i-1}^i(q_i) = A_{i-1}^i A_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.14)$$

The transformation matrix from Frame i to Frame $i - 1$ is a function only of the joint variable q_i , that is, θ_i for a revolute joint or d_i for a prismatic one. The Kuka kr 2210 is a 6 dof manipulator, composed by 6 revolute joints, so by applying the DH convention, we get the parameters in table 2.8

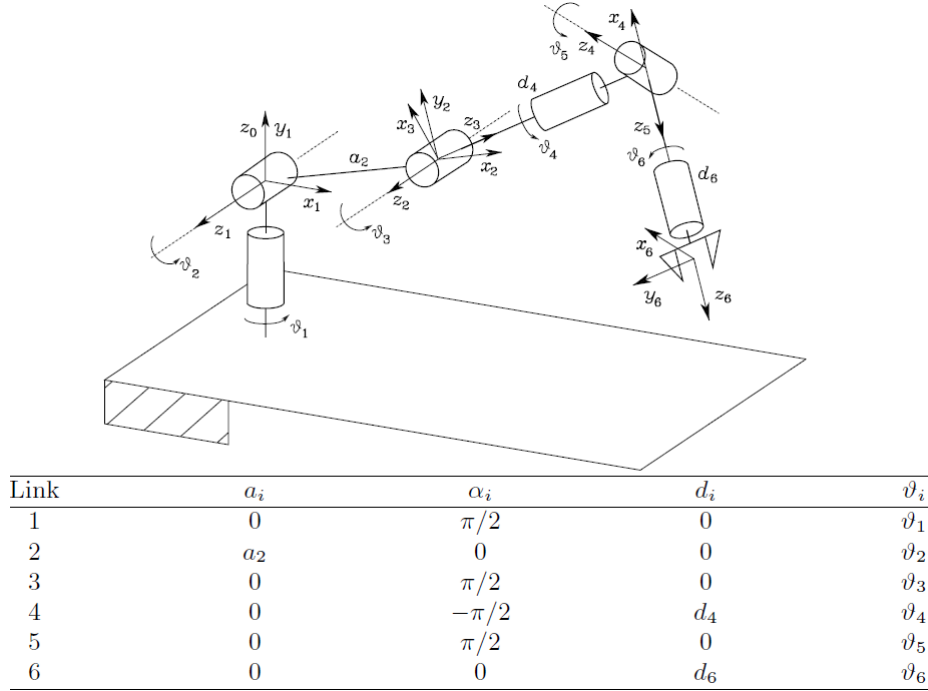


Figure 2.8: Kuka Kr220 schematization - Anthropomorphic arm with spherical wrist

2.2.6 Direct Kinematics

Computation of the direct kinematics function leads to expressing the position and orientation of the end-effector frame as:

$$p_0^6 = \begin{bmatrix} a_2 c_1 c_2 + d_4 c_1 s_{23} + d_6 c_1 (c_{23} c_4 s_5 + s_{23} c_5) + s_1 s_4 s_5 \\ a_2 s_1 c_2 + d_4 s_1 s_{23} + d_6 s_1 (c_{23} c_4 s_5 + s_{23} c_5) - c_1 s_4 s_5 \\ a_2 s_2 - d_4 c_{23} + d_6 (s_{23} c_4 s_5 - c_{23} c_5) \end{bmatrix} \quad (2.15)$$

and the orientation

$$n_0^6 = \begin{bmatrix} c_1 \text{left}(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) + s_1(s_4c_5c_6 + c_4s_6) \\ s_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) - c_1(s_4c_5c_6 + c_4s_6) \\ s_{23}(c_4c_5c_6 - s_4s_6) + c_{23}s_5c_6 \end{bmatrix} \quad (2.16)$$

$$s_0^6 = \begin{bmatrix} c_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) + s_1(-s_4c_5s_6 + c_4c_6) \\ s_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) - c_1(-s_4c_5s_6 + c_4c_6) \\ -s_{23}(c_4c_5s_6 + s_4c_6) - c_{23}s_5s_6 \end{bmatrix} \quad (2.17)$$

$$a_0^6 = \begin{bmatrix} c_1(c_{23}c_4s_5 + s_{23}c_5) + s_1s_4s_5 \\ s_1(c_{23}c_4s_5 + s_{23}c_5) - c_1s_4s_5 \\ s_{23}c_4s_5 - c_{23}c_5 \end{bmatrix} \quad (2.18)$$

2.2.7 Differential Kinematics

The goal of the differential kinematics is to find the relationship between the joint velocities and the end-effector linear and angular velocities. In order to express the end-effector linear velocity \dot{p}_e and angular velocity ω_e as a function of the joint velocities \dot{q} . As will be seen afterwards, the sought relations are both linear in the joint velocities, i.e.,

$$\dot{p}_e = J_P(q)\dot{q}, \quad \omega_e = J_O(q)\dot{q} \quad (2.19)$$

Here, J_P is the $(3 \times n)$ matrix relating the contribution of the joint velocities \dot{q} to the end-effector linear velocity \dot{p}_e , while J_O is the $(3 \times n)$ matrix relating the contribution of the joint velocities \dot{q} to the end-effector angular velocity ω_e . In compact form :

$$v_e = \begin{bmatrix} \dot{p}_e \\ \omega_e = J(q)\dot{q} \end{bmatrix} \quad (2.20)$$

which represents the manipulator differential kinematics equation. The $(6 \times n)$ matrix J is the manipulator *geometric* Jacobian

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix} \quad (2.21)$$

which in general is a function of the joint variables. In summary, the Jacobian can be partitioned into the (3×1) column vectors j_{P_i} and j_{O_i} as:

$$J = \begin{bmatrix} j_{P1} & \dots & j_{Pn} \\ j_{O1} & \dots & j_{On} \end{bmatrix} \quad (2.22)$$

where:

$$\begin{bmatrix} j_{P_i} \\ j_{O_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ \vec{0} \end{bmatrix} & \text{for a prismatic joint} \\ \begin{bmatrix} z_{i-1} \wedge (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{for a revolute joint} \end{cases} \quad (2.23)$$

The expressions in eq.2.23 allow Jacobian computation in a simple, systematic way on the basis of direct kinematics relations. Finally, notice that the Jacobian matrix depends on the frame in which the end-effector velocity is expressed. The above

equations allow computation of the geometric Jacobian with respect to the base frame. The Jacobian in the differential kinematics equation of a manipulator defines a linear mapping

$$v_e = J(q)\dot{q} \quad (2.24)$$

between the vector \dot{q} of joint velocities and the vector $v_e = [\dot{p}^T \ \omega_e^T]^T$ of end-effector velocity.

2.2.8 Singularity

The Jacobian is, in general, a function of the configuration q ; those configurations at which J is rank-deficient are termed kinematic singularities, fig. 2.9. To find the singularities of a manipulator is of great interest for the following reasons:

- Singularities represent configurations at which mobility of the structure is reduced, i.e., it is not possible to impose an arbitrary motion to the end-effector.
- When the structure is at a singularity, infinite solutions to the inverse kinematics problem may exist.
- In the neighborhood of a singularity, small velocities in the operational space may cause large velocities in the joint space.

Singularities can be further classified into:

- Boundary singularities that occur when the manipulator is either outstretched or retracted. It may be understood that these singularities do not represent a true drawback, since they can be avoided on condition that the manipulator is not driven to the boundaries of its reachable workspace.
- Internal singularities that occur inside the reachable workspace and are generally caused by the alignment of two or more axes of motion, or else by the attainment of particular end-effector configurations. Unlike the above, these singularities constitute a serious problem, as they can be encountered anywhere in the reachable workspace for a planned path in the operational space.

2.2.9 Inverse kinematics

Inverse kinematics is the mathematical process of calculating the variable joint parameters needed to place the end of a kinematic chain, such as a robot manipulator or animation character's skeleton, in a given position and orientation relative to the start of the chain. Given joint parameters, the position and orientation of the chain's end, e.g. the hand of the character or robot, can typically be calculated directly using multiple applications of trigonometric formulas, a process known as forward kinematics. However, the reverse operation is, in general, much more challenging. There are many methods of modelling and solving inverse kinematics problems. The most flexible of these methods typically rely on iterative optimization to seek out an approximate solution, due to the difficulty of inverting the forward kinematics equation and the possibility of an empty solution space. The core idea behind several of these methods is to model the forward kinematics equation using a Taylor series expansion, which can be simpler to invert and solve than the original system.

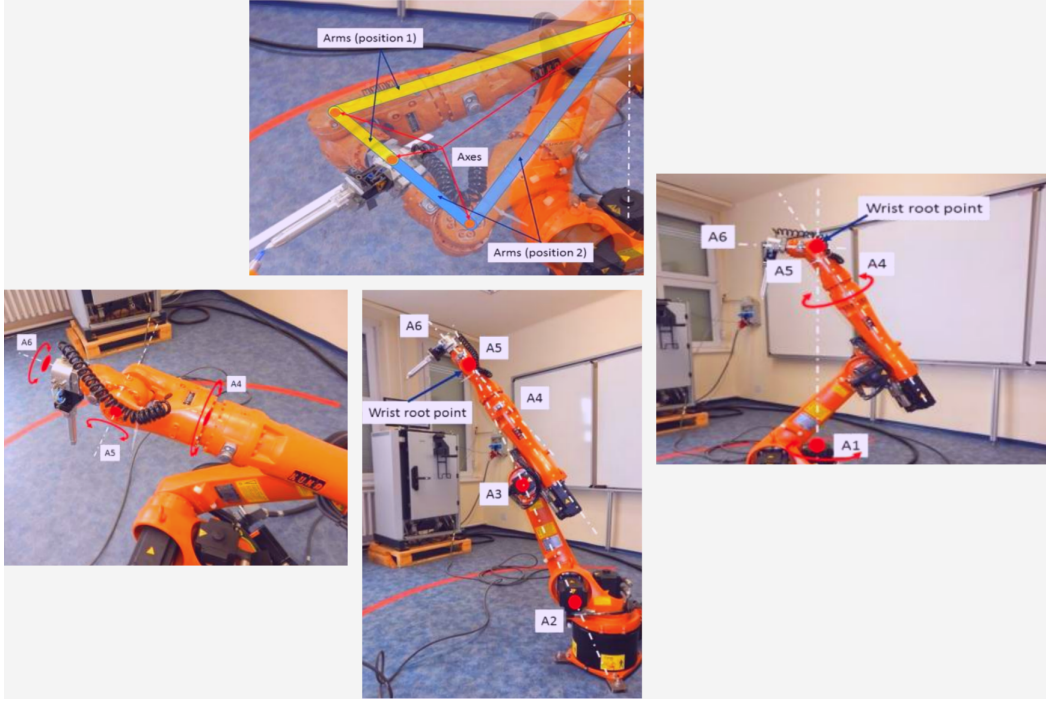


Figure 2.9: Kuka Kr5, [2] singularities configurations

2.2.10 The Jacobian inverse technique

The Jacobian inverse technique is a simple yet effective way of implementing inverse kinematics. Let there be m variables that govern the forward-kinematics equation, i.e. the position function. These variables may be joint angles, lengths, or other arbitrary real values. If the IK system lives in a 3-D space, the position function can be viewed as a mapping

$$p(x) : \mathbb{R}^m \rightarrow \mathbb{R}^3 \quad (2.25)$$

Let $p_0 = p(x_0)$ give the initial position of the system, and

$$p_1 = p(x_0 + \Delta x) \quad (2.26)$$

be the goal position of the system. The Jacobian inverse technique iteratively computes an estimate of Δx that minimizes the error given by

$$\|p(x_0 + \Delta x_{estimate}) - p_1\| \quad (2.27)$$

For small Δx , the series expansion of the position function gives

$$p(x_1) \approx p(x_0) + J_p(x_0)\Delta x \quad (2.28)$$

where $J_p(x_0)$ is the $(3 \times m)$ Jacobian matrix of the position function at x_0 .

Note that the (i, k) -th entry of the Jacobian matrix can be approximated numerically

$$\frac{\partial p_i}{\partial x_k} \approx \frac{p_i(x_{0,k} + h) - p_i(x_0)}{h} \quad (2.29)$$

where $p_i(x)$ gives the i -th component of the position function, $x_{0,k} + h$ is simply x_0 with a small delta added to its k -th component, and h is a reasonably small positive

value. Taking the *Moore-Penrose pseudo inverse* of the Jacobian (computable using a singular value decomposition) and re-arranging terms results in:

$$\Delta x \approx J_p^+(x_0)\Delta p \quad (2.30)$$

where:

$$\Delta p = p(x_0 + \Delta x) - p(x_0) \quad (2.31)$$

Applying the inverse Jacobian method once will result in a very rough estimate of the desired Δx . A line search should be used to scale this Δx to an acceptable value. The estimate for Δx can be improved via the following algorithm (known as the Newton–Raphson method):

$$\Delta x_{k+1} = J_p^+(x_k)\Delta p_k \quad (2.32)$$

Once some Δx has caused the error to drop close to zero, the algorithm should terminate. Existing methods based on the Hessian matrix of the system have been reported to converge to desired Δx values using fewer iterations, though, in some cases more computational resources.

The Newton–Raphson method (NR) is one of the oldest approach used to solve inverse kinematics problem. There is still a lot of research going on in this field, see for example the paper [10], in this work, the author have developed a different inverse kinematic algorithm starting from NR, that is called *Newton-improved cyclic coordinate descent* (NICCD) and they have shown that their method is more accurate, see fig.2.10, robust and generalizable.

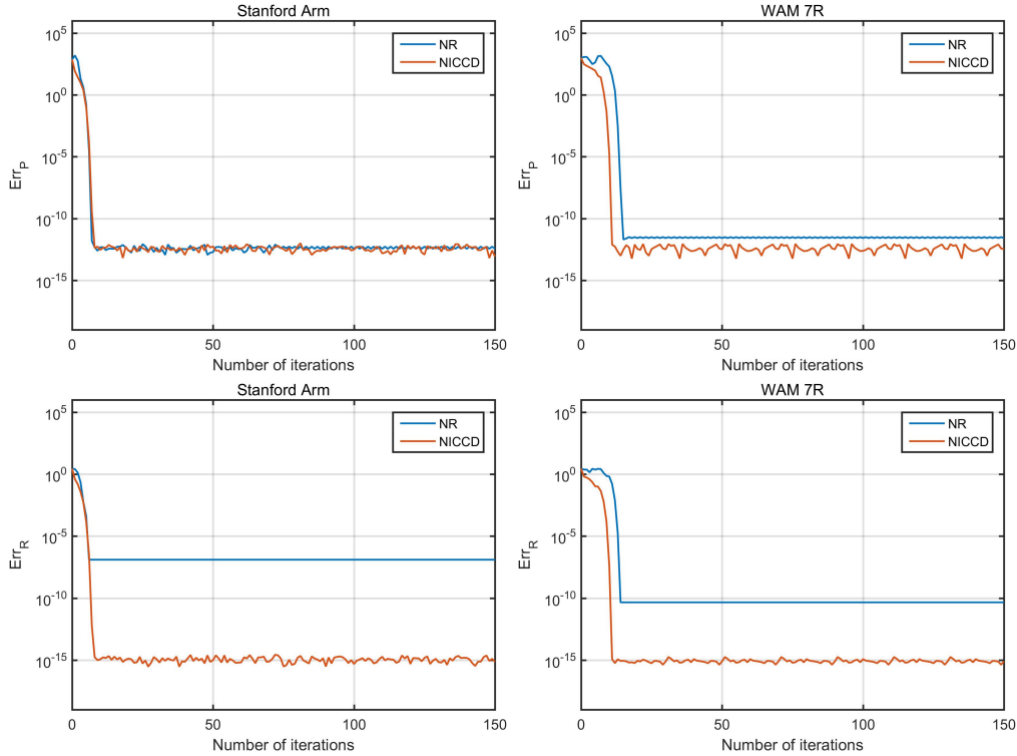


Figure 2.10: Inverse Kinematic Algorithms comparisons - NR vs NICCD - on two different manipulator type : Stanford Arm and WAM 7R

2.2.11 Heuristic methods

The Inverse Kinematics problem can also be approximated using heuristic methods. These methods perform simple, iterative operations to gradually lead to an approximation of the solution. The heuristic algorithms have low computational cost (return the final pose very quickly), and usually support joint constraints.

2.2.12 Trajectory Planning

One of the crucial ingredients in robotics is the trajectory and motion planning. Those tools are meant to make us able to give commands to the manipulator in order to make it move to a point a series of points or a trajectory in the space. It is now needed to explain the difference between path and trajectory.

- A path denotes the locus of points in the joint space, or in the operational space, which the manipulator has to follow in the execution of the assigned motion; a path is then a pure geometric description of motion.
- a trajectory is a path on which a timing law is specified, for instance in terms of velocities and/or accelerations at each point.

Moreover, when talking about motion planning, we mean planning motions for robots to move from point A to point B (such as for mobile robots, etc.) or pose A to pose B (such as for manipulators, etc.). In order to do so, a number of constraints need to be taken into account: collision avoidance, joint limits, velocity/acceleration limits, jerk limits, dynamic balance, torque bounds, and many more. In this sense, not only the the robots is considered but also its environment. When talking about trajectory generation, the scope can be narrower than that of motion planning. Often time, in trajectory generation, people really focus on generating a trajectories with joint limits, velocity and acceleration constraints.

The geometric path can be defined in the work-space or in the joint-space. Usually, it is expressed in a parametric form as:

$$\begin{cases} p = p(s) & \text{work-space} \\ q = q(\sigma) & \text{joint-space} \end{cases} \quad (2.33)$$

The parameter $s(\sigma)$ is defined as a function of time, and in this manner the motion law $s = s(t)$, ($\sigma = \sigma(t)$) is obtained. In the joint space, geometric paths are obtained by assigning initial/final and/or intermediate values for the joint variables, along with the desired motion law. Usually, polynomial functions a of proper degree n are employed:

$$s(t) = a_0 + a_1t + a_2t^2 + \dots + a_nt^n \quad (2.34)$$

In this way, a “smooth” interpolation of the points defining the geometric path is achieved.

2.2.13 Trajectory Planning Specifics

Input data to an algorithm for trajectory planning are:

- data defining on the path (points),

- geometrical constraints on the path (e.g. obstacles),
- constraints on the mechanical dynamics
- constraints due to the actuation system

Output data are the trajectory in the joint- or work-space, given as a sequence (in time) of the acceleration, velocity and position values:

$$a(kT), v(kT), p(kT) \quad k = 0, \dots, N \quad (2.35)$$

being T a proper time interval defining the instants in which the trajectory is computed (and converted in the joint space) and sent to each actuator. We can compute a trajectory planning in both work or joint space:

- Work-space trajectories allow to consider directly possible constraints on the path that are more difficult to take into consideration in the joint space.
- Joint space trajectories are computationally simpler and allow to consider problems due to singular configurations, actuation redundancy, velocity/acceleration constraints.

2.2.14 Trajectory Planning Algorithms

Trajectories are specified by defining some characteristic points directly assigned by some specifications or by defining desired configurations in the work-space, which are then converted in the joint space using the inverse kinematic model, [7]. The algorithm that computes a function $q(t)$ interpolating the given points is characterized by the following features:

- trajectories must be computationally efficient
- the position and velocity profiles (at least) must be continuous functions of time
- undesired effects (such as non regular curvatures) must be minimized or completely avoided.

In the most simple cases, a trajectory is specified by assigning initial and final conditions on: time, position, velocity, acceleration. Then, the problem is to determine a function $q = q(t)$ or $q = q(\sigma)$, $\sigma = \sigma(t)$ so that those conditions are satisfied. This is a boundary condition problem, that can be easily solved by considering polynomial functions such as:

$$q(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_n t^n \quad (2.36)$$

The degree $n = 3, 5, \dots$ of the polynomial depends on the number of boundary conditions that must be verified and on the desired *smoothness* of the trajectory. Among many other combinations, a possible approach for planning a trajectory is to use linear segments joined with parabolic blends. In the linear tract, the velocity is constant while, in the parabolic blends, it is a linear function of time: trapezoidal profiles, typical of this type of trajectory, are then obtained. In trapezoidal trajectories, fig.2.11, the duration is divided into three parts:

1. in the first part, a constant acceleration is applied, then the velocity is linear and the position a parabolic function of time
2. in the second, the acceleration is null, the velocity is constant and the position is linear in time
3. in the last part a (negative) acceleration is applied, then the velocity is a negative ramp and the position a parabolic function.

Usually, the acceleration and the deceleration phases have the same duration $t_a = t_d$. Therefore, symmetric profiles, with respect to a central instant $\frac{t_f - t_i}{2}$, are obtained. The trajectory is computed according to the following equations:

$$q(t) = \begin{cases} q_i + \frac{1}{2}\ddot{q}_c^2 & 0 \leq t \leq t_c \\ q_i + \ddot{q}_c t_c(t - \frac{t_c}{2}) & t_c < t \leq t_f - t_c \\ q_f - \frac{1}{2}\ddot{q}_c(t_f - t)^2 & t_f - t_c < t \leq t_f \end{cases} \quad (2.37)$$

Other functions can be obtained by properly composing segments defined with polynomial functions of different degree (piece-wise, polynomial functions).

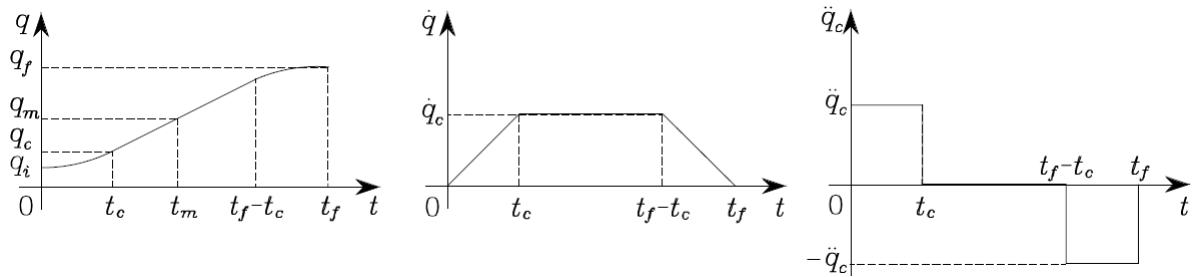


Figure 2.11: Trajectory Planning - 3 - rd order polynomial trajectory - Trapezoidal velocity profile

2.2.15 KUKA Motion Instructions

In this section, we are going to describe the practical implementation of motion planning, in a coding point of view. Therefore, we will present the elementary moving instructions, fig.2.13, given by the KUKA Kr2210 manual and provide a little context for those type of motions.

2.2.16 PTP : Point to Point

This directive executes a point-to-point motion to the end point. The coordinates of the end point are absolute. Example 1 End point specified in Cartesian coordinates. PTP X 12.3,Y 100.0,Z 50,A 9.2,B 50,C 0,S 'B010',T 'B1010' Example 2 End point specified in axis-specific coordinates. The end point is approximated. PTP A1 10,A2 -80.6,A3 -50,A4 0,A5 14.2, A6 0 C _ PTP Example 3 End point specified with only 2 components. For the rest of the components, PTP Z 500,X 123.6

2.2.17 PTP_REL : Point to Point Relative

This directive executes a point-to-point motion to the end point, but the coordinates of the end point are relative to the current position. Example 1 Axis 2 is moved 30 degrees in a negative direction. None of the other axes moves. PTP_REL A2 -30
Example 2 The robot moves 100 mm in the X direction and 200 mm in the negative Z direction from the current position. PTP_REL X 100,Z -200

2.2.18 LIN : Linear

This directive executes a linear motion to the end point. The coordinates of the end point are absolute. Example End point with two components. For the rest of the components, the controller takes the values of the previous position. LIN Z 500,X 123.6

2.2.19 LIN_REL : Linear Relative

This directive executes a linear motion to the end point. The coordinates of the end point are relative to the current position.

2.2.20 CIRC : Circular

This directive executes a circular motion. An auxiliary point and an end point must be specified in order for the controller to be able to calculate the circular motion. The coordinates of the auxiliary point and end point are absolute. Example The end point of the circular motion is defined by a circular angle of 260°. The end point is approximated. CIRC X 5,Y 0, Z 9.2,X 12.3,Y 0,Z -5.3,A 9.2,B -5,C 20

2.2.21 CIRC_REL : Circular Relative

This directive executes a circular motion. An auxiliary point and an end point must be specified in order for the controller to be able to calculate the circular motion. The coordinates of the auxiliary point and end point are relative to the current position. Example The end point of the circular motion is defined by a circular angle of 500°. The end point is approximated. CIRC_REL X 100,Y 3.2,Z -20,Y 50,CA 500 C -VEL

2.2.22 Spline

In mathematics, a spline, [fig.2.12](#) is a special function defined piece-wise by polynomials. In interpolating problems, spline interpolation is often preferred to polynomial interpolation because it yields similar results, even when using low-degree polynomials, while avoiding Runge's phenomenon¹ for higher degrees. Splines are popular curves because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve

¹Runge's phenomenon is a problem of oscillation at the edges of an interval that occurs when using polynomial interpolation with polynomials of high degree over a set of equispaced interpolation points.

fitting and interactive curve design. The term spline comes from the flexible spline devices used by shipbuilders and draftsmen to draw smooth shapes

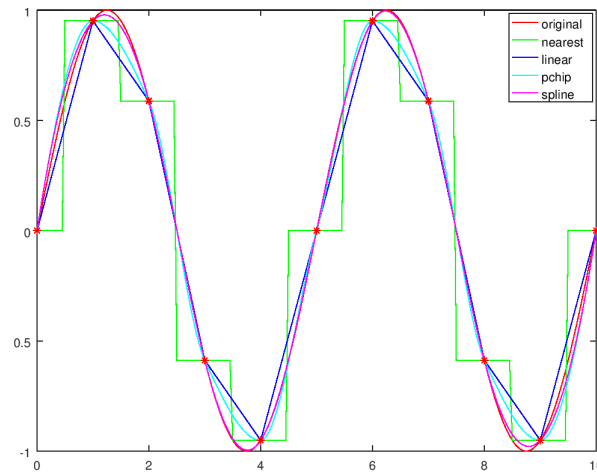


Figure 2.12: Interpolation of continuous function $\sin(x)$ with various method

2.2.23 Spline

Spline is a motion type that is suitable for particularly complex, curved paths. Such paths can generally also be generated using approximated LIN and CIRC motions, but spline nonetheless has advantages. The most versatile spline motion is the spline block. A spline block is used to group together several motions as an overall motion. The spline block is planned and executed by the robot controller as a single motion block. The motions that may be included in a spline block are called spline segments. They are taught separately. A CP spline block can contain SPL, SLIN and SCIRC segments. A PTP spline block can contain SPTP segments. In addition to spline blocks, individual spline motions can also be programmed: SLIN, SCIRC and SPTP. Advantages of spline blocks:

- The path is defined by means of points that are located on the path. The desired path can be generated easily.
- The programmed velocity is maintained better than with conventional motion types.
- There are few cases in which the velocity is reduced. Furthermore, special constant velocity ranges can be defined in CP spline blocks.
- The path always remains the same, irrespective of the override setting, velocity or acceleration.
- Circles and tight radii are executed with great precision.

Disadvantages of LIN/CIRC:

- The path is defined by means of approximated points that are not located on the path. The approximate positioning ranges are difficult to predict. Generating the desired path is complicated and time-consuming.

- In many cases, the velocity may be reduced in a manner that is difficult to predict, e.g. in the approximate positioning ranges and near points that are situated close together.
- The path changes if approximate positioning is not possible, e.g. for time reasons.
- The path changes in accordance with the override setting, velocity or acceleration.

2.2.24 SPLINE

`SPLINE ... ENDSPLINE` defines a CP spline block. A CP spline block may contain: `SLIN`, `SCIRC` and `SPL` segments (number limited only by the memory capacity)

- `PATH` trigger
- 1 time block (`TIME _BLOCK`) or 1 constant velocity range (`CONST_VEL`)

A spline block must not include any other instructions, e.g. variable assignments or logic statements. A spline block does not trigger an advance run stop.

2.2.25 PTP_SPLINE ... ENDSPLINE

`PTP _SPLINE, ENDSPLINE` defines a PTP spline block. A PTP spline block may contain:

- `PATH` trigger
- 1 time block (`TIME _BLOCK`) or 1 constant velocity range (`CONST_VEL`)
- `SPTP` segments (number limited only by the memory capacity)

A spline block must not include any other instructions, e.g. variable assignments or logic statements. A spline block does not trigger an advance run stop.

2.3 Safety in Modern Robotics

As already pointed out, a manipulator can exert large forces and torques that can be dangerous when a human operator is around. So in order to avoid robot human collisions, it is necessary to create some kind of barriers, that can be either software made (working ones) or hardware (physical barriers). A simple way to introduce motion constraint is to avoid certain configurations by means of custom Software limit switch. There are 2 ways of modifying the software limit switches:

- Enter the desired values manually.
- automatically adapt the limit switches to one or more programs.

The robot controller determines the minimum and maximum axis positions occurring in the program. These values can then be set as software limit switches.

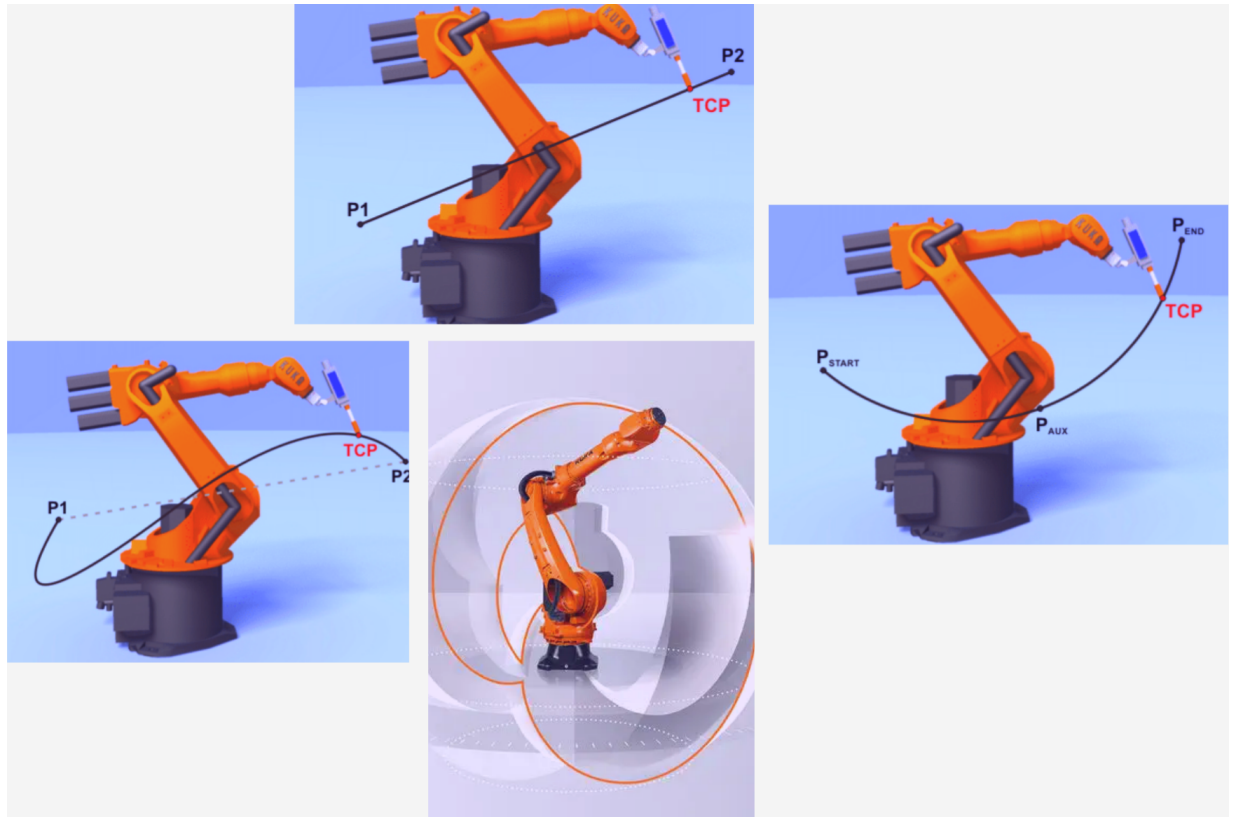


Figure 2.13: Elementary Motion Types - from left to right P2P, LIN, CIRC

2.3.1 Tool Calibration

Robot calibration is the process of identifying certain parameters in the kinematic structure of an industrial robot, such as the relative position of robot links, the manipulator's inertia matrix, etc. The calibration can be classified in three different ways:

1. Level-1 calibration, also known as *mastering* only models differences between actual and reported joint displacement values
2. Level-2 calibration, also known as *kinematic calibration*, concerns the entire geometric robot calibration which includes angle offsets and joint lengths
3. Level-3 calibration, also called a *non-kinematic calibration*, models errors other than geometric defaults such as stiffness, joint compliance, and friction

Often Level-1 and Level-2 calibration are sufficient for most practical needs. During tool calibration, the user assigns a Cartesian coordinate system (TOOL coordinate system) to the tool mounted on the mounting flange. The TOOL coordinate system has its origin at a user-defined point. This is called the TCP (Tool Center Point). The TCP is generally situated at the working point of the tool. Advantages of tool calibration:

1. The tool can be moved in a straight line in the tool direction.
2. The tool can be rotated about the TCP without changing the position of the TCP.

3. In program mode: The programmed velocity is maintained at the TCP along the path.

A maximum of 16 TOOL coordinate systems can be saved inside the variable `TOOL_DATA[1...16]`. This structure store the following data :

1. X, Y, Z : Origin of the TOOL coordinate system relative to the FLANGE coordinate system
2. A, B, C : Orientation of the TOOL coordinate system relative to the FLANGE coordinate system

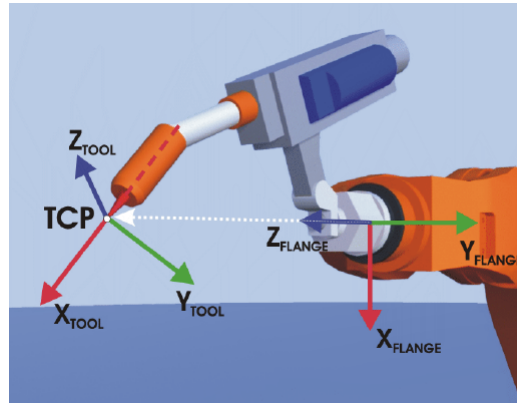


Figure 2.14: TCP Calibration

In order to define the origin of the TOOL coordinate system, the following methods are available:

1. XYZ 4-point
2. XYZ Reference

2.3.2 4-point method

The TCP of the tool to be calibrated is moved to a reference point from 4 different directions, fig.2.15. The reference point can be freely selected. The robot controller calculates the TCP from the different flange positions.

2.3.3 XYZ Reference method

In the case of the XYZ Reference method, fig.2.16, a new tool is calibrated with a tool that has already been calibrated. The robot controller compares the flange positions and calculates the TCP of the new tool. In order to define the tool's orientation, the following methods are available:

1. ABC 2-point
2. ABC World
3. Numeric input
4. Base Calibration
5. ABC 3-point

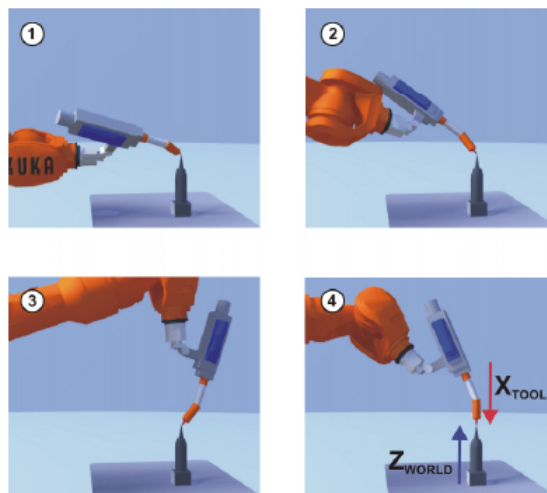


Figure 2.15: TCP Position Calibration - 4-point method

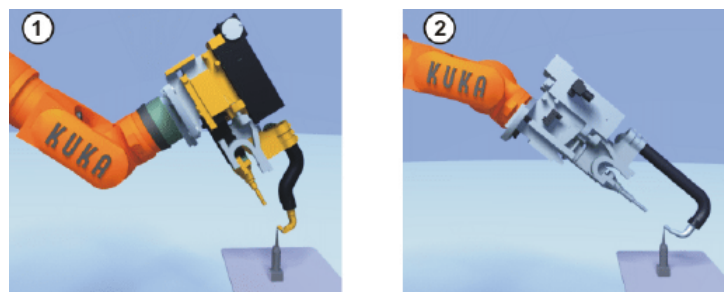


Figure 2.16: TCP Position Calibration - XYZ Reference method

2.3.4 ABC 2-point

The axes of the TOOL coordinate system are aligned parallel to the axes of the WORLD coordinate system, fig.2.17. This communicates the orientation of the TOOL coordinate system to the robot controller. There are 2 variants of this method:

1. 5D: Only the tool direction is communicated to the robot controller. By default, the tool direction is the X axis. The directions of the other axes are defined by the system and cannot be detected easily by the user. Area of application: e.g. MIG/MAG welding, laser cutting or water jet cutting
2. 6D: The directions of all 3 axes are communicated to the robot controller. Area of application: e.g. for weld guns, grippers or adhesive nozzles

2.3.5 ABC World

The axes of the TOOL coordinate system are communicated to the robot controller by moving to a point on the X axis and a point in the XY plane. This method is used if it is necessary to define the axis directions with particular precision.

2.3.6 Numeric input

The tool data can be entered manually. Possible sources of data:

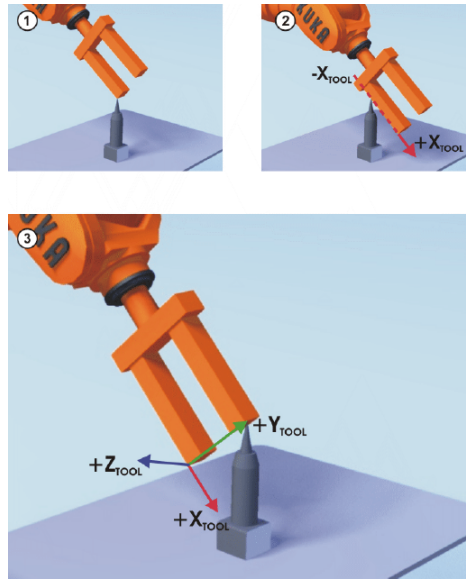


Figure 2.17: TCP Orientation Calibration - ABC 2-point

1. CAD
2. Externally calibrated tool
3. Tool manufacturer specifications

2.3.7 Base Calibration

During base calibration, the user assigns a Cartesian coordinate system (BASE coordinate system) to a work surface or the work-piece. The BASE coordinate system has its origin at a user-defined point. Advantages of base calibration:

1. The TCP can be jogged along the edges of the work surface or work-piece.
2. Points can be taught relative to the base. If it is necessary to offset the base, e.g. because the work surface has been offset, the points move with it and do not need to be retaught.

A maximum of 32 BASE coordinate systems can be saved inside the variable `BASE_DATA[1...32]`.

2.3.8 ABC 3-point

The robot moves to the origin and two further points of the new base. These three points define the new base, [fig.2.18](#).

2.3.9 Configuring axis-specific workspace

Description Axis-specific workspace can be used to restricted yet further the areas defined by the software limit switches in order to protect the robot, tool or work-piece.

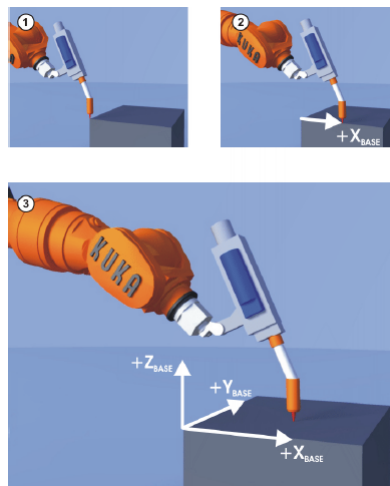


Figure 2.18: TCP Orientation Calibration - ABC 3-point

2.3.10 Collision Detection

If the robot collides with an object, the robot controller increases the axis torques in order to overcome the resistance. This can result in damage to the robot, tool or other objects. Collision detection reduces the risk of such damage. It monitors the axis torques. If these deviate from a specified tolerance range, the following reactions are triggered:

1. The robot stops with a STOP 1.
2. The robot controller calls the program `tm_useraction`. This is located in the Program folder and contains the HALT statement. Alternatively, the user can program other reactions in the program `tm_useraction`.

The robot controller automatically calculates the tolerance range. A program must generally be executed 2 or 3 times before the robot controller has calculated a practicable tolerance range. The user can define an offset via the user interface for the tolerance range calculated by the robot controller. There are though some major Restrictions:

1. Collision detection is not possible for HOME positions and other global positions.
2. Collision detection is not possible for external axes.
3. Collision detection is not possible during backward motion.
4. Collision detection is not possible in T1 mode.
5. High axis torques arise when the stationary robot starts to move. For this reason, the axis torques are not monitored in the starting phase (approx. 700 ms).
6. The collision detection function reacts much less sensitively for the first 2 or 3 program executions after the program override value has been modified. Thereafter, the robot controller has adapted the tolerance range to the new program override.

Defining an offset for the tolerance range

An offset for the torque and for the impact can be defined for the tolerance range, fig.2.19,. The lower the offset, the more sensitive the reaction of the collision detection. The higher the offset, the less sensitive the reaction of the collision detection. The torque is effective if the robot meets a continuous resistance.

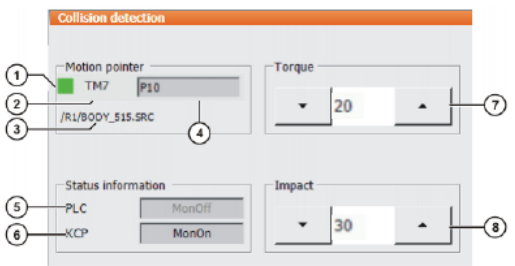
Examples:

- The robot collides with a wall and pushes against the wall.
- The robot collides with a container. The robot pushes against the container and moves it.

Impact: The impact is effective if the robot meets a brief resistance.

Example:

1. The robot collides with a panel which is sent flying by the impact.



Item	Description
1	Indicates the status of the current motion: <ul style="list-style-type: none"> Red: the current motion is not monitored. Green: the current motion is monitored. Orange: an arrow key has been pressed in the Torque or Impact box. The window remains focused on the motion and the offset can be modified. The change can then be applied by pressing Save. Pixelated: a program must generally be executed 2 or 3 times before the robot controller has calculated a practicable tolerance range. This display is pixelated during this learning phase.
2	Number of the TMx variable The robot controller creates a TMx variable for each motion block in which the parameter Collision detection is set to TRUE. TMx contains all the values for the tolerance range of this motion block. If 2 motion blocks refer to the same point Px, the robot controller creates 2 TMx variables.
3	Path and name of the selected program
4	Point name
5	This box is only active in "Automatic External" mode. It appears gray in all other modes. MonOn: collision detection has been activated by the PLC. If collision detection is activated by the PLC, the PLC sends the input signal <code>sTQM_SPSACTIVE</code> to the robot controller. The robot controller responds with the output signal <code>sTQM_SPSSTATUS</code> . The signals are defined in the file <code>\$config.dat</code> . Note: Collision detection is only active in Automatic External mode if both the PLC box and the KCP box show the entry MonOn .
6	MonOn: collision detection has been activated from the KCP. Note: Collision detection is only active in Automatic External mode if both the PLC box and the KCP box show the entry MonOn .
7	Offset for the torque. The lower the offset, the more sensitive the reaction of the collision detection. Default value: 20. If an arrow key is pressed, the window remains focused on the motion and the offset can be modified. The change can then be applied by pressing Save . <ul style="list-style-type: none"> N.A.: the option Collision detection in the inline form is set to FALSE for this motion.
8	Offset for the impact. The lower the offset, the more sensitive the reaction of the collision detection. Default value: 30. If an arrow key is pressed, the window remains focused on the motion and the offset can be modified. The change can then be applied by pressing Save . <ul style="list-style-type: none"> N.A.: the option Collision detection in the inline form is set to FALSE for this motion.

Figure 2.19: Teach Pendant - Collision Detection Menu

Torque monitoring

The maximum torque deviation, fig.2.20, that has occurred can be determined as a percentage by means of the system variable `TORQ_DIFF[...]`.

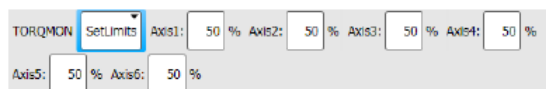


Figure 2.20: Teach Pendant - Torque Monitoring Menu

2.3.11 Safety Planes

Another important tool used for safety reasons, is the definition of work zones, safety zones and stopping zones, thanks to the creation of planes inside the Workspace, see fig.2.21. In this way we are able to define different types of behavior depending in which area the robot is working in. For example, if we are sure that in a specific portion of space, there are no obstacle as well as no people, the robot can move at full speed without any problem. Instead if we know that in some regions, there can be an operator who is close enough to robot, we may want to wither slow down the robot or stop it immediately. Thanks to the introduction of these planes, we are able to teach the robot where it can go fast, where it has to slow down and where it must stop. In order to create a safety plane, we can use the Teach Pendant as

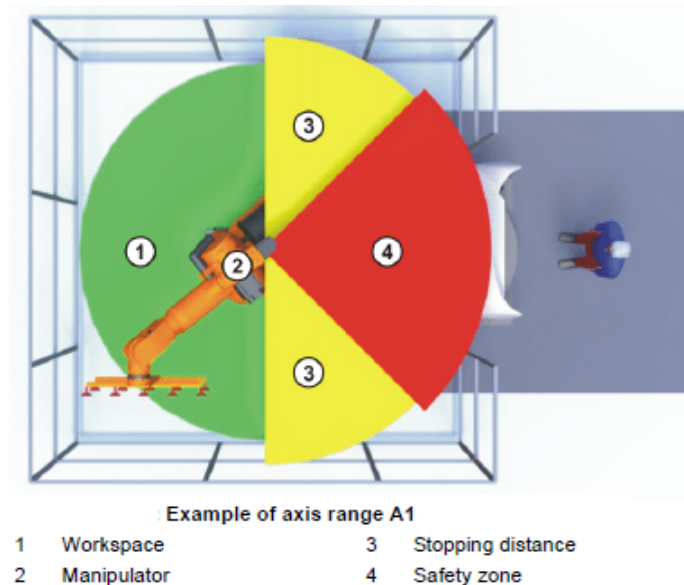


Figure 2.21: Working Regions

shown in fig.2.22

2.3.12 Safety Barriers

In general, robotic structures are provided with a series of physical barriers, that prevents the operator entering inside the robot's working zone as well as protecting them from malfunctioning of the manipulator. This safety systems need to be designed specifically for each project.

2.3.13 Safety Hardware

It is necessary to identify the safety distances necessary to reduce to zero the possibility of risk by the operator. The reference standard is EN ISO 120218:2-2011 which transposes the machinery directive to which must be added the standard EN ISO 13857 which shows the values, tabulated, of the distances to be respected for safety. It starts from the standard EN ISO 13857 that allows us to define the minimum distances to be respected, and then make an analysis of the risks due to the actual working area of the Robot (DCS).

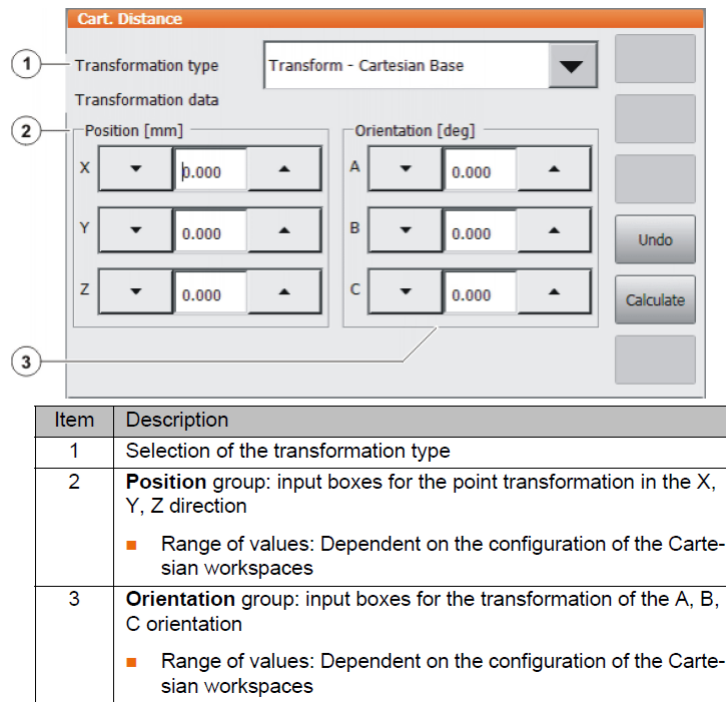


Figure 2.22: Teach Pendant - Safety Plane Menu

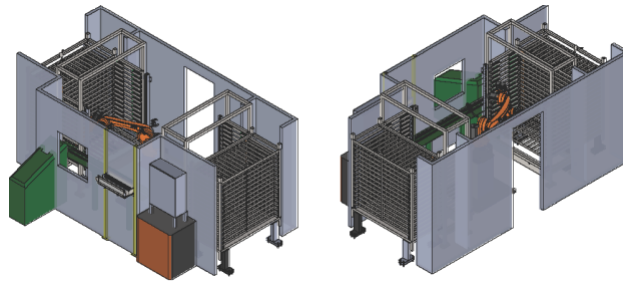


Figure 2.23: Safety Barriers all around the manipulator

1. a = height of the point nearest to the Danger Zone
2. b = height of protective structure
3. c = horizontal safety distance of the Danger Zone
4. 1 = Danger Zone

If regular openings exist, it is necessary to evaluate the size of this shape and to adapt it according to the reachability of the limb. In case it is not possible to limit the movement of the limb or in case the distance of the Danger Zone is not necessary for the reduction of the risk, special shelters will be made that prevent the limb from being able to reach the area. With regard to the provisions provided by EN ISO 10218-2:2011, it is necessary to first define the operating spaces of the robot. The operating space is that part of the area needed for the robot to perform all the functions required by the work program. The confined space is the maximum work area reached by the robot, dictated by both operational constraints and constructive constraints. The protected space is that area protected by fences where the operator can be present for a limited time, with the robot switched off or set, according to the standard, with a speed lower than $250 \frac{mm}{s}$. The design shall be done bearing in

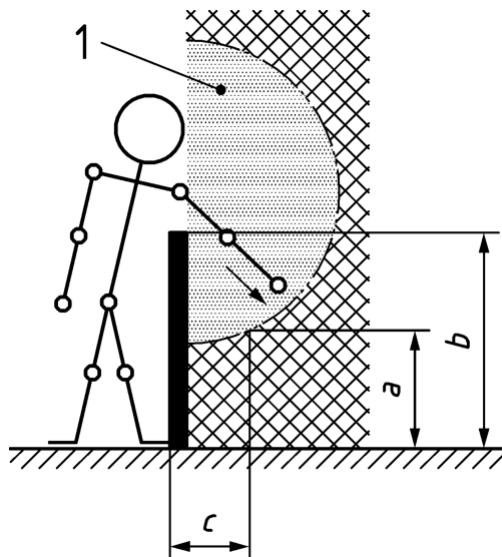


Figure 2.24: Safety Risk Zones

mind that the operator must always be outside the protected area if the robot is being executed. If the system is running automatically then:

1. access to the protected area, when the automatic cycle is active, shall cause all equipment likely to cause hazardous situations to stop
2. Automatic cycle selection shall not cancel or reset any protective stop or emergency stop
3. The automatic cycle can only be started from outside the protected area
4. Automatic cycle start-up shall be possible when all protective devices are active
5. Personnel must be protected from rebooting the robotic system when they are inside the protected space

For the last point, the following considerations may be made:

1. the reboot can be done by two people, introducing two remote reset buttons, outside of the fences, and in places where you can see everything that happens within the work area
2. The reboot can be performed via a timed pre-reset button inside the workspace and an actual reset button outside the workspace. Non-compliance with timing inhibits the restoration of the island.

If the system is running manually then:

1. local control shall be performed by a single portable control panel
2. the speed must not exceed a standard limit value (250 mm/s). It must be possible to select lower speeds

Mobile control panels shall comply with the standards set out in EN ISO 10218:1. Maintenance must be carried out when the installation is stationary and the operator must be given specific access to the areas concerned. If maintenance should

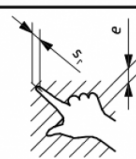
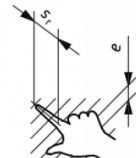
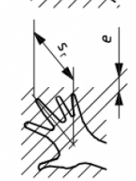
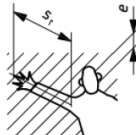
Part of body	Illustration	Opening	Safety distance, s_r		
			Slot	Square	Round
Fingertip		$e \leq 4$	≥ 2	≥ 2	≥ 2
		$4 < e \leq 6$	≥ 10	≥ 5	≥ 5
Finger up to knuckle joint		$6 < e \leq 8$	≥ 20	≥ 15	≥ 5
		$8 < e \leq 10$	≥ 80	≥ 25	≥ 20
		$10 < e \leq 12$	≥ 100	≥ 80	≥ 80
		$12 < e \leq 20$	≥ 120	≥ 120	≥ 120
Hand		$20 < e \leq 30$	$\geq 850^a$	≥ 120	≥ 120
Arm up to junction with shoulder		$30 < e \leq 40$	≥ 850	≥ 200	≥ 120
		$40 < e \leq 120$	≥ 850	≥ 850	≥ 850

Figure 2.25: Safety Distance with respect to body parts

occur when the plant is in motion, it is necessary to inhibit access to the working area of the robot with special systems, such as protections or capacitive platforms. It is important to point out that in the design of the robotized island it must always be prevented that electromagnetic interference causes problems of interface and operation such as to make the robot act independently, or differently from the function for which it was designed. Every robot on the island must have its own locking button and its own emergency button.

In our case, even if the entire working region is closed with physical barriers, there's still the possibility to get closer to the robot. This is because the customer need to take off the full chest of drawers and then carrying it back the empty one again. So, during this operations the operator can be dangerously close to the robot. So in order to avoid collisions, we had to place a laser sensor, that is able to perceive if a person is entering close to this area and consequently send a signal to the PLC, that will immediately stop the robot. This types of laser sensor are very accurate and can be tuned to acknowledge fingers, arms or people presence, fig.2.25.

2.3.14 Brake Test

Each robot axis has a holding brake integrated into the motor. The brake test checks every axis at low speed and at the current temperature to see if the braking torque is sufficiently high, i.e. whether it exceeds a certain minimum value. The minimum value for the individual axes is stored in the machine data. (The brake test does not calculate the absolute value of the braking torque.) The brake test is deactivated by default. Exception: If the KUKA.Safe Operation or KUKA.SafeRangeMonitoring option is installed, the brake test is activated by default. The user can activate and deactivate the brake test. The cycle time is 46 h. It is deemed to have elapsed when

the drives have been under servo-control for a total of 46 h. The robot controller then requests a brake test and generates the following message: Brake test required. The robot can be moved for another 2 hours. It then stops and the robot controller generates the following acknowledgment message: Cyclical check for brake test request not made. Once the message has been acknowledged, the robot can be moved for another 2 hours. The brake test checks all brakes one after the other.

- The robot accelerates to a defined velocity. (The velocity cannot be influenced by the user.)
- Once the robot has reached the velocity, the brake is applied and the result for this braking operation is displayed in the message window.
- If a brake has been identified as being defective, the brake test can be repeated for confirmation or the robot can be moved to the parking position.

If a brake has reached the wear limit, the robot controller indicates this by means of a message. A worn brake will soon be identified as defective. Until then, the robot can be moved without restrictions. If a brake has been identified as being defective, the drives remain under servo-control for 2 hours following the start of the brake test (=monitoring time). The robot controller then switches the drives off.

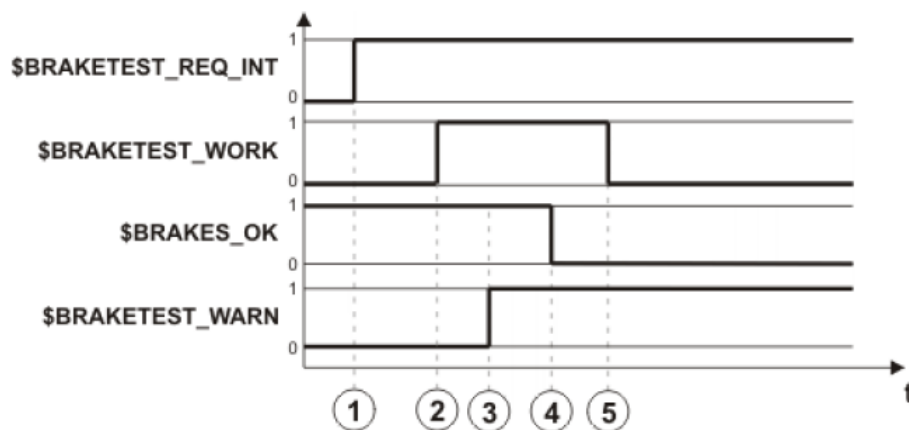


Figure 2.26: Brake Test - Faulty Brake

2.4 Teach Pendant

A teach pendant device is needed to control an industrial robot remotely. The device allows its controller to work with robots without the need for tethering to a fixed terminal. Teach pendants offer a variety of settings to control robots and are also utilized to design new capabilities and features. Within the robotics repair industry, technicians not only repair the units themselves but use the device to test robotic equipment. The teach pendant is an essential component for industrial robots and utilized for application use, along with the repair and refurbishment process.

Thanks to the teach pendant, we can basically control everything that it's related with the manipulator. In fact, we can

1. Write an executable program and run it

2. Create safety planes, calibrating tool, setting custom limit switches, etc.
3. Check the I/O signals that the robot exchange with the rest of the world
4. Monitoring speed, torques, current, etc.
5. Import/export program files, update robot firmware, etc.
6. Move the robot manually
7. Change operating modes

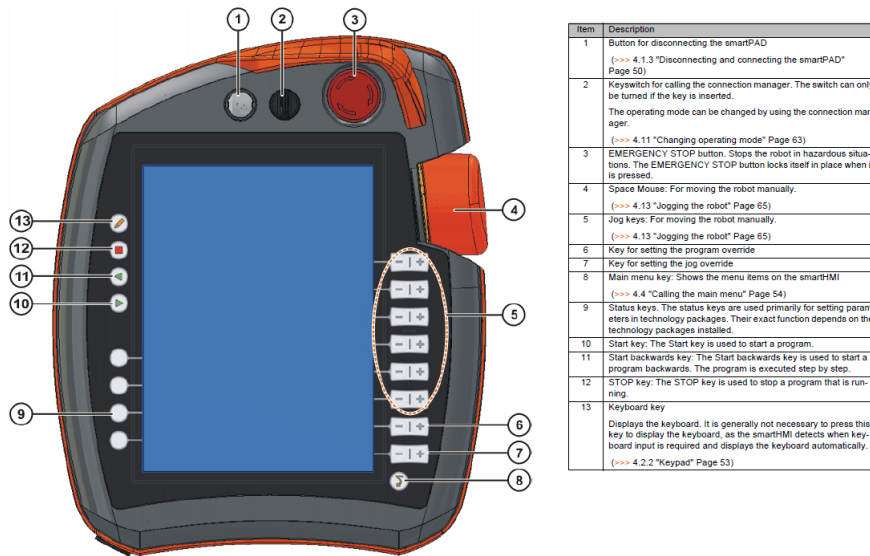


Figure 2.27: Teach Pendant - Front View

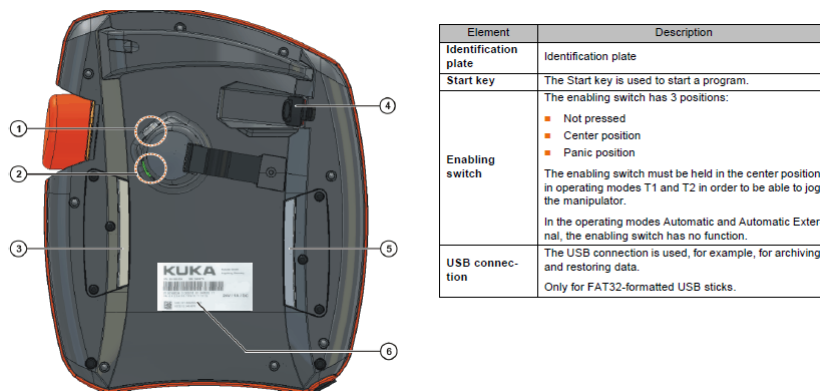


Figure 2.28: Teach Pendant - Rear View

2.4.1 Operating Modes

The industrial robot can be operated in the following modes:

1. Manual Reduced Velocity (T1)
2. Manual High Velocity (T2)

3. Automatic (AUT)
4. Automatic External (AUT EXT)
5. CRR

In addition, the table in fig.2.29 indicates the operating modes in which the safety functions are active.

Safety functions	T1, CRR	T2	AUT	AUT EXT
Operator safety	-	-	active	active
EMERGENCY STOP device	active	active	active	active
Enabling device	active	active	-	-
Reduced velocity during program verification	active	-	-	-
Jog mode	active	active	-	-
Software limit switches	active	active	active	active

Figure 2.29: Operating Mode and Safety functions

Chapter 3

Omron PLC

3.1 OMRON

OMRON was founded in Japan in 1933 by Kazuma Tateisi as an electric manufacturing company. Since the beginning OMRON has presented itself as an innovative industry leader in automation, sensing and control technology. Since then, many everyday products and services were invented and introduced to society. OMRON developed the world's first automated traffic signal and the world's first unmanned train station, also OMRON was the first with face recognition technology for mobile phones.

OMRON's primary business is the manufacturing and sales of automation components, equipment and systems, but it is generally known by the public for medical devices.

Omron deals with:

1. Industrial automation: industrial robots, sensors, switches, industrial cameras, safety components, relays, control components, electric power monitoring equipment, power supplies and PLCs
2. Electronic components: relays, switches, connectors, micro sensing devices, MEMS sensors, image sensing technologies,
3. Social systems: access control systems (building entry systems), road management systems, traffic signal controllers, security/surveillance cameras, automated ticket gates, ticket vending machines, fare adjustment machines
4. Healthcare: Personal use: blood pressure monitors, digital thermometers, body composition monitors, pedometers, nebulizers
5. Healthcare : Professional use: blood pressure monitors, non-invasive vascular monitors, portable ECGs, patient monitors
6. Power distribution and controls for drilling rigs
7. Environmental solutions[buzzword]
8. Electronic controls and automation for detention center systems

3.2 Computer Network

A computer network is a set of computers sharing resources located on or provided by network nodes. The computers use common communication protocols over digital interconnections to communicate with each other. These interconnections are made up of telecommunication network technologies, based on physically wired, optical, and wireless radio-frequency methods that may be arranged in a variety of network topologies. The nodes of a computer network may include personal computers, servers, networking hardware, or other specialised or general-purpose hosts. They are identified by network addresses, and may have hostnames. Hostnames serve as memorable labels for the nodes, rarely changed after initial assignment. Network addresses serve for locating and identifying the nodes by communication protocols such as the Internet Protocol. Computer networks may be classified by many criteria, including the transmission medium used to carry signals, bandwidth, communications protocols to organize network traffic, the network size, the topology, traffic control mechanism, and organizational intent. Computer networks support many applications and services, such as access to the World Wide Web, digital video, digital audio, shared use of application and storage servers, printers, and fax machines, and use of email and instant messaging applications.

3.2.1 Local Area Network

LAN stands for local area network. A network is a group of two or more connected computers, and a LAN is a network contained within a small geographic area, usually within the same building. Home WiFi networks and small business networks are common examples of LANs. LANs can also be fairly large, although if they take up multiple buildings, it is usually more accurate to classify them as wide area networks (WAN) or metropolitan area networks (MAN). Not all LANs connect to the Internet. In fact, LANs predate the Internet: the first LANs were used in businesses in the late 1970s. (These old LANs used network protocols that are no longer in use today.) The only requirement for setting up a LAN is that the connected devices are able to exchange data. This usually requires a piece of networking equipment for packet switching, such as a network switch. Today, even non-Internet-connected LANs use the same networking protocols that are used on the Internet (such as IP).

3.2.2 Internet Protocol

An IP address is a unique address that identifies a device on the internet or a local network. IP stands for "Internet Protocol," which is the set of rules governing the format of data sent via the internet or local network. In essence, IP addresses are the identifier that allows information to be sent between devices on a network: they contain location information and make devices accessible for communication. The internet needs a way to differentiate between different computers, routers, and websites. IP addresses provide a way of doing so and form an essential part of how the internet works. An IP address is a string of numbers separated by periods. IP addresses are expressed as a set of four numbers — an example address might be 192.158.1.38. Each number in the set can range from 0 to 255. So, the full IP addressing range goes from 0.0.0.0 to 255.255.255.255. Internet Protocol works the same way as any other language, by communicating using set guidelines to pass

information. All devices find, send, and exchange information with other connected devices using this protocol. By speaking the same language, any computer in any location can talk to one another. The use of IP addresses typically happens behind the scenes. The process works like this:

1. Your device indirectly connects to the internet by connecting at first to a network connected to the internet, which then grants your device access to the internet.
2. When you are at home, that network will probably be your Internet Service Provider (ISP). At work, it will be your company network.
3. Your IP address is assigned to your device by your ISP.
4. Your internet activity goes through the ISP, and they route it back to you, using your IP address. Since they are giving you access to the internet, it is their role to assign an IP address to your device.

However, your IP address can change.

3.2.3 Public IP addresses

Public IP addresses come in two forms – dynamic and static.

1. Dynamic IP addresses: they change automatically and regularly. ISPs buy a large pool of IP addresses and assign them automatically to their customers. Periodically, they re-assign them and put the older IP addresses back into the pool to be used for other customers. The rationale for this approach is to generate cost savings for the ISP. Automating the regular movement of IP addresses means they don't have to carry out specific actions to re-establish a customer's IP address if they move home, for example. There are security benefits, too, because a changing IP address makes it harder for criminals to hack into your network interface.
2. Static IP addresses: in contrast to dynamic IP addresses, static addresses remain consistent. Once the network assigns an IP address, it remains the same. Most individuals and businesses do not need a static IP address, but for businesses that plan to host their own server, it is crucial to have one. This is because a static IP address ensures that websites and email addresses tied to it will have a consistent IP address — vital if you want other devices to be able to find them consistently on the web.

Within an IP network, each interface connected to the physical network is assigned a unique address. The IP address is assigned to the interface itself (such as a network card) and not to the host, because the host is connected to the network. A router, for example, has several interfaces and each requires an IP address. The connection protocols, at level 2 of the ISO/OSI model, direct the computers using the MAC address. When on a local network you use IP, each computer must also be assigned an IP address, to allow it to communicate with computers outside of its local network. The correspondence between IP address and MAC address is managed through the ARP protocol, which allows to know the MAC address of a given computer its IP address through a distributed query. Assigning an IP address to a computer can be manual, or automated by protocols such as DHCP

3.2.4 IPV4

Internet Protocol version 4 (IPv4) is the fourth version of the Internet Protocol (IP). It is one of the core protocols of standards-based internetworking methods in the Internet and other packet-switched networks. The first header field in an IP packet is the four-bit version field. For IPv4, this is always equal to 4. The IPv4 header is variable in size due to the optional 14th field (options). The IHL field contains the size of the IPv4 header, it has 4 bits that specify the number of 32-bit words in the header. The minimum value for this field is 5,[30] which indicates a length of 20 bytes. As a 4-bit field, the maximum value is 15, this means that the maximum size of the IPv4 header is 60 bytes.

3.2.5 Subnet

A subnetwork or subnet is a logical subdivision of an IP network. The practice of dividing a network into two or more networks is called subnetting. Devices that belong to the same subnet are addressed with an identical most-significant bit-group in their IP addresses. This results in the logical division of an IP address into two fields: the network number or routing prefix and the rest field or host identifier. The rest field is an identifier for a specific host or network interface. For IPv4, a network may also be characterized by its subnet mask or netmask, which is the bitmask that when applied by a bitwise AND operation to any IP address in the network, yields the routing prefix. Subnet masks are also expressed in dot-decimal notation like an address. For example, 255.255.255.0 is the subnet mask for the prefix 198.51.100.0/24. Traffic is exchanged between subnetworks through routers when the routing prefixes of the source address and the destination address differ. A router serves as a logical or physical boundary between the subnets. The benefits of subnetting an existing network vary with each deployment scenario. In the address allocation architecture of the Internet using CIDR and in large organizations, it is necessary to allocate address space efficiently. Subnetting may also enhance routing efficiency, or have advantages in network management when subnetworks are administratively controlled by different entities in a larger organization. Subnets may be arranged logically in a hierarchical architecture, partitioning an organization's network address space into a tree-like routing structure, or other structures such as meshes.

3.3 Ethercat Protocol

The EtherCAT protocol is a communication protocol based on the industrial Ethernet one, but with some modifications. The Ethercat master sends a telegram through all the nodes, fig. 3.1. Each Ethercat slave reads the output data for it and writes those produced by it in the frame "on-the-fly", while the latter spreads to the next nodes. The delay suffered by the frame is equal to the only physical crossing time of the slave. The last node in a segment or full line redirects the message to the master using full-duplex Ethernet communication. The actual rate of use of telegrams rises to over 90%, and thanks to full-duplex communication the theoretical data flow is even higher than 100 Mbit/s. The Ethercat master is the only node in the network that can actively send an Ethercat frame; all other nodes

do is forward those frames downstream. This principle prevents delays of variable duration and guarantees deterministic performance. The master uses a standard Media Access Controller (MAC) without any dedicated communication processor. This allows users to implement a master device on any hardware platform with a network port, regardless of the operating system or application software used. Ethercat slave devices integrate a so-called Ethercat Slave Controller (ESC) capable of processing on-the-fly frames and in a purely hardware way, which makes network performance predictable and independent of the particular implementation of slave devices. Ethercat uses standard Ethernet frames. Ethercat frames are identified by Ether-type 0x88A4. Being Ethercat optimized for few cyclical process data, you can avoid the use of bulky software stacks such as TCP/IP or UDP/IP.

3.3.1 Ethernet Communication

To enable classic Ethernet communication between nodes, TCP/IP telegrams can be tunneled through an acyclic channel without impacting the deterministic data exchange, fig.3.1. During the boot phase, the master configures and maps the cyclic process data in the slaves. Each slave can exchange variable amounts of data, from one bit to a few bytes, or even kilobytes of data. Each Ethercat frame contains one or more Datagrams. The Datagram header indicates what type of access the master device requires:

Read, write or read+write

Access to a slave server via direct addressing, or access to multiple slaves via logical addressing (implicit addressing).

Logical addressing is used for the exchange of cyclic process data. Each Datagram addresses a specific subset of the network process image, whose maximum overall size can reach 4 Gbytes. During the initial configuration, each slave is assigned a specific location in that global address space. Slaves allocated in the same range can be addressed by the same Datagram. Since Datagrams contain all the information related to data access, the master can decide when and to which data to access. For example, the master can use fast cycle times to access servo data, and slower cycle times to sample I/O. This simplifies the operation of the master compared to other field buses, in which the data of each node must be read individually, ordered by a processor, and copied to memory.

In Ethercat, the master only has to fill a single Ethercat frame with new output data, and send the frame to the MAC controller via Direct Memory Access (DMA). When a frame with new input data is received by the MAC, the master can always transfer the frame via DMA to the controller's memory - all without the CPU having to actively perform any copying operation. In addition to cyclical data, additional Datagrams can be used for asynchronous or event-based communication.

3.3.2 Add on-the-fly process data

In addition to logical addressing, the master can access a slave based on its position in the network. This method is used at the initial configuration stage to determine the network topology and compare it with the expected one.

After verifying the network configuration, the master can assign each node a precon-

figured address and communicate with the node itself through that fixed address. This allows you to access individual devices even if the network topology is changed, as in Hot Connect applications. There are two solutions for slave to slave communication. A slave can send data directly to another node located downstream in the network. Since Ethercat telegrams are processed only in the forward path, this type of direct communication depends on network topology and is particularly suitable for fixed machine architectures (e.g. printing or packaging machines). In contrast, a fully flexible slave to slave communication passes through the master and requires two communication cycles (not necessarily two controller cycles). Thanks to Ethercat's excellent performance, this type of slave-to-slave communication is faster than other field bus technologies.

3.3.3 Ethercat P: data & power supply on one cable

Ethercat P (P = power) is an extension of the Ethercat protocol described above. It allows you to transmit not only data, but also the power supply voltage through a single standard four-wire Ethernet cable.

From the point of view of the Ethercat protocol and Ethercat P are identical, as the extension concerns only the physical level. Specific Ethercat Slave Controllers are not required to implement Ethercat P. It could be said that Ethercat P has the same communication advantages as Ethercat, but it also provides power through the communication cable, which results in benefits and benefits for many applications. The two independent and electrically isolated 24V voltage power the Ethercat P devices, equipped with US dedicated to electronics and sensors and UP destined to periphery and actuators. Both US and UP voltages are transmitted directly to the 100mbit/s Ethercat communication line. Thanks to this, users can cascade several Ethercat P devices with a single cable. This allows for a simplification of cabling, a reduction in costs and a reduction in the size of devices, equipment and machines. Ethercat P is particularly advantageous for those independent and often isolated machine parts, as they can now be accessed with data and power through a single cable. Sensors of all types lend themselves perfectly to Ethercat P: a single M8 connector allows efficient integration of these devices into the high-speed network and connects them to the power supply. Possible errors in the connection of the devices are avoided thanks to the mechanical coding of the connectors.

Ethercat P can be used together with traditional Ethercat in the same network. Dedicated components transform the physical layer of traditional Ethercat into Ethercat P while maintaining data encoding. Similarly, a device can be powered with Ethercat P but transmit standard Ethercat output.

3.3.4 Distributed Clock for Precise Synchronization

In applications with space-distributed processes and requiring simultaneous actions, exact synchronization is particularly important, fig. refether. This is the case, for example, with multiple drives performing coordinated movements. Unlike a fully synchronous communication, the quality of which immediately suffers from communication errors, the synchronized distributed clocks have a high degree of tolerance towards the communication jitter. Therefore, the Ethercat solution for node synchronization is based on the distributed clock (DC) approach. Fully hardware syn-

chronization with propagation delay compensation. The calibration of the clocks in the individual nodes takes place at a completely hardware level. The time reference of the first DC slave is cyclically distributed to all other devices in the system. With this mechanism, the slave clocks can be synchronized precisely to that of the reference clock. The resulting sync jitter is much less than $1\mu s$. Since the time reference sent by the reference clock arrives at the other slaves with a certain propagation delay, the latter must be measured and compensated for each slave in order to ensure synchronicity and simultaneity. The delay is measured during the network startup phase and, if necessary, even at full speed, which ensures a maximum deviation between the clocks of much less than $1\mu s$.

If all nodes share the same time reference, they can activate their own outputs simultaneously or acquire their own inputs with a high-precision timestamp. In axis control applications, cycle time stability is as important as synchronism and simultaneity. In such applications, the velocity is normally derived from the measured position, so it is essential that position measurements are temporally equidistant: even minimal fluctuations in the instant of position measurement can result in a major error in the calculated speed, especially in case of fast cycle times. With Ethercat, position measurements are triggered by the precise local clock and not by the bus, which ensures much greater accuracy. In addition, the use of distributed clocks also softens the specifications on the master device; since actions such as position measurement are triggered by the local clock instead of receiving the frame, the master is not subject to special requirements regarding the sending of frames. This allows you to implement the master completely software level on a standard Ethernet hardware. Even a microsecond size jitter does not decrease the accuracy of distributed clocks! Since this accuracy does not depend on when the clock is adjusted, the exact moment of transmission of the frame becomes irrelevant. The Ethercat master only needs to ensure that the Ethercat telegram is sent sufficiently in advance of the moment when the DC signal inside the slave devices triggers the implementation of the outputs.

3.3.5 Diagnostic and Localization Errors

Experience with conventional field buses has shown that diagnostic properties play a key role in determining machine availability and commissioning times. In addition to identifying errors, it is also important to locate them when searching for faults. Ethercat allows you to scan and compare the actual network topology with that configured during the startup phase. Ethercat also inherently supports many other diagnostic functions. The Ethercat Slave Controller in each node performs a cyclic redundancy check on each frame. The data contained in it are forwarded to the slave application only if the received frame is valid. If an error is detected, a corresponding counter is incremented and subsequent nodes are informed that the frame is corrupted. Even the master device will detect that the frame has been corrupted and will discard the data. The master can locate where the frame was originally corrupted by analysing individual slave error counters. This is a huge advantage compared to conventional field buses, where errors spread over the entire physical medium making it impossible to locate its source. Ethercat allows you to locate and locate occasional disturbances before the problem impacts on machine operation. Thanks to the efficient use of the Ethercat bandwidth, which is bet-

ter orders of magnitude than industrial Ethernet technologies based on individual frames, the probability of errors caused by disturbances is much lower at the same cycle time. And, in the case of very fast cycle times - typical scenario for Ethercat - the time required for rebooting the network following an error is much less. Inside the frames, the Working Counter allows you to monitor the consistency of the data in each Datagram. Each node addressed by a Datagram and whose memory is accessible automatically increases the corresponding Working Counter. The master is therefore able to check cyclically if all nodes are working with consistent data. If the Working Counter has a different value from the expected one, the master does not forward Datagram data to the control application. The master can then determine the cause of unexpected behaviour with the help of status and error information from the nodes, as well as the state of the physical link.

Since Ethercat uses standard Ethernet frames, network traffic can be recorded via free software such as Wire-shark, which comes with an Ethercat-specific protocol interpreter.

1. Ethercat Diagnostics for Users
2. Ethercat Diagnostics for Developers

3.3.6 High Availability Requirements

For machinery and devices with high availability requirements, a cable outage or a node malfunction shall in no way determine the nn attainability of a segment of the network or the termination of the network. Ethercat allows cable redundancy with simple devices. By connecting a cable from the last node to an additional network port in the master, a line topology is extended into a ring topology. A redundancy event, such as a cable interruption or a node malfunction, is detected by a software supplement in the master device. That will be all! Slaves do not need to be modified, and they are not even aware that they are working in redundant network conditions.

Cable redundancy with standard Ethercat slave devices Physical link monitoring in slaves automatically detects and resolves redundancy cases with reaction times of less than 15 μ s, so that at most a cyclic frame is lost. This means that even axis control applications with very short cycle times can continue to operate when a cable is interrupted. Ethercat also provides master redundancy with Hot Standby functionality. Vulnerable network components, such as those connected via an cable drag chain, can be connected as a branch of the network, so that even when the cable is interrupted the rest of the machine continues to function.

3.3.7 Communication Profiles

For the purpose of configuring slave devices and extracting diagnostic information, it is possible to access the variables through acyclic communication, fig. refether. The latter is based on a reliable mailbox protocol with auto-recover functionality in case of loss or corruption of messages. To support a wide variety of devices and applications, the following Ethercat communication profiles have been defined:

1. CAN application protocol over EtherCAT (CoE)

2. Servo drive profile, IEC 61800-7-204 (SoE)
3. Ethernet over EtherCAT (EoE)
4. File access over EtherCAT (FoE)
5. Automation Device Protocol over EtherCAT (ADS over EtherCAT, AoE)

Different communication profiles can coexist in the same device.

A slave does not necessarily support all communication profiles; on the contrary, it is possible to decide which profile is best suited to specific needs. The master device is informed about which communication profiles have been implemented in the slave through the descriptive file of the slave.

3.3.8 Transparent transmission of standard IT protocols

Using the Ethernet over Ethercat (Eoe) protocol any Ethernet data traffic can be transported within Ethercat datagrams. Ethernet devices are connected to the Ethercat network through the so-called Switchports. Ethernet frames, as well as internet protocols (e.g. TCP/IP, VPN, Pppoe (DSL), etc.) are transparently conveyed over Ethercat via tunnelling. The device with Switchports functionality takes care of inserting TCP/IP fragments into Ethercat traffic and therefore prevents the determinism of communication from being compromised. In addition, Ethercat devices can support Ethernet protocols (such as HTTP) locally, and thus behave as traditional Ethernet nodes external to an Ethercat network. The master behaves like a level 2 switch, capable of sending the frames to the recipient nodes via Eoe based on their MAC addresses. In this way, you can support all internet technologies such as web servers, email, FTP transfer in an Ethercat context.

3.3.9 Services

The protocol defines interfaces and services for:

1. Data exchange between Ethercat Master devices (master-master communication),
2. Communication to man-machine interfaces (HMI),
3. A supervision controller to access devices belonging to the underlying Ethercat segments (Routing),
4. Integration of tools for plant configuration, as well as for device configuration.

3.4 Sysmac

Sysmac Studio is a fully integrated development environment for machine automation that combines configuration, programming, simulation and monitoring in a simple interface. Automation professionals across the globe use Sysmac Studio to design next-generation control systems incorporating PLCs, motion, safety, robotics and vision products. The Sysmac Studio Automation Software provides an integrated development environment to set up, program, debug, and maintain SYSMAC

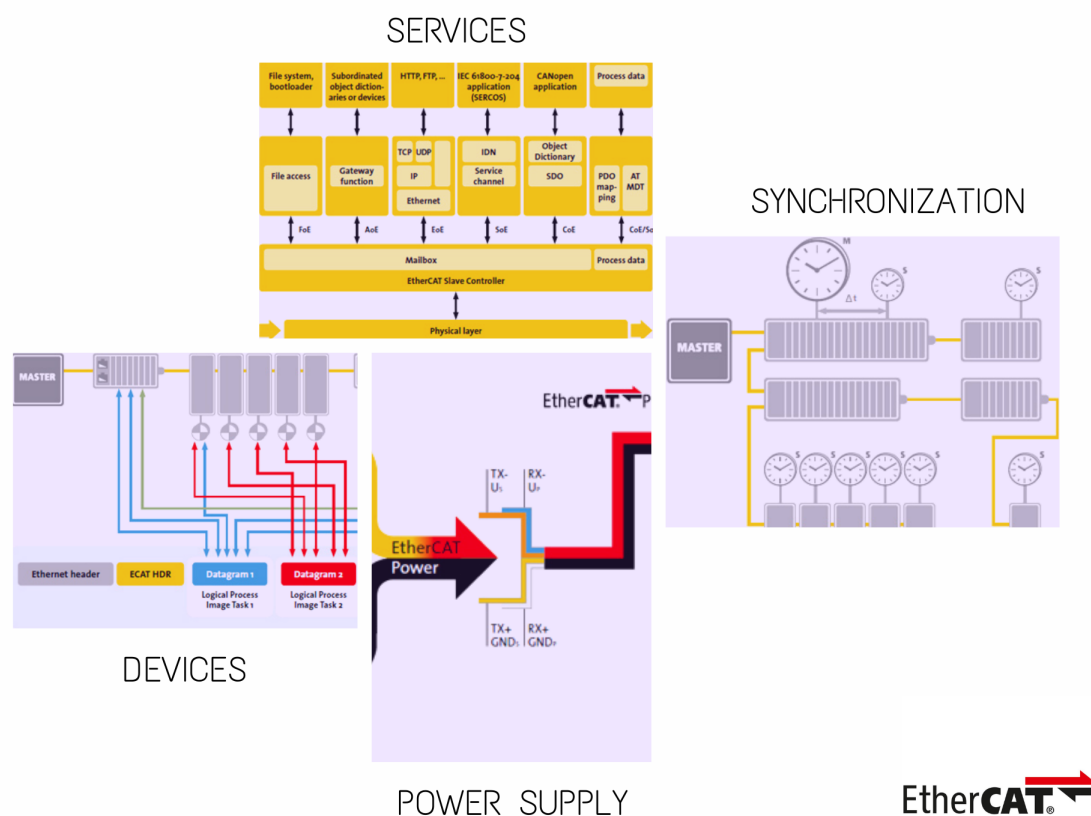


Figure 3.1: Ethercat Protocol - Communication, Power supply, Services and Synchronization

NJ/NX/NY-series Controllers and other Machine Automation Controllers, as well as EtherCAT slaves. The Sysmac Studio provides an environment for programming with variables and POU's. Programming is designed with POU's (programs, functions, and function blocks). The programs are then assigned to tasks and the program execution order is defined. This reduces the interdependence of the programs and therefore allows more than one programmer to easily work at the same time. The assignments of variables to hardware and the definitions of the relations between information that is shared between different programs can be set at any time.

As any other type of PLC software programming, Sysmac decompose a project in three main areas, that are:

1. Topological View: where it is defined how the multiple devices are connected to each other
2. Input/Output Data: where it is defined all the I/O data that the system utilizes
3. Program Files: where it is written the PLC program
4. Program Variables: where all the program variables are defined and stored

In our case, the topological view is shown in fig. 3.2, where every component can communicate with the PLC thanks to a unique IP address, who's obviously belonging to the same subnet. In our case the local web ip address and subnet are:

192.168.100.x
255.255.255.0

Once every component has a unique IP address, we have to give a name to each I/O data that every devices send to the PLC. This operation is also called *I/O cards population*, see fig.3.3.

Data can be of four types:

1. Digital Input (DI)
2. Digital Output (DO)
3. Analog Input (AI)
4. Analog Output (AO)

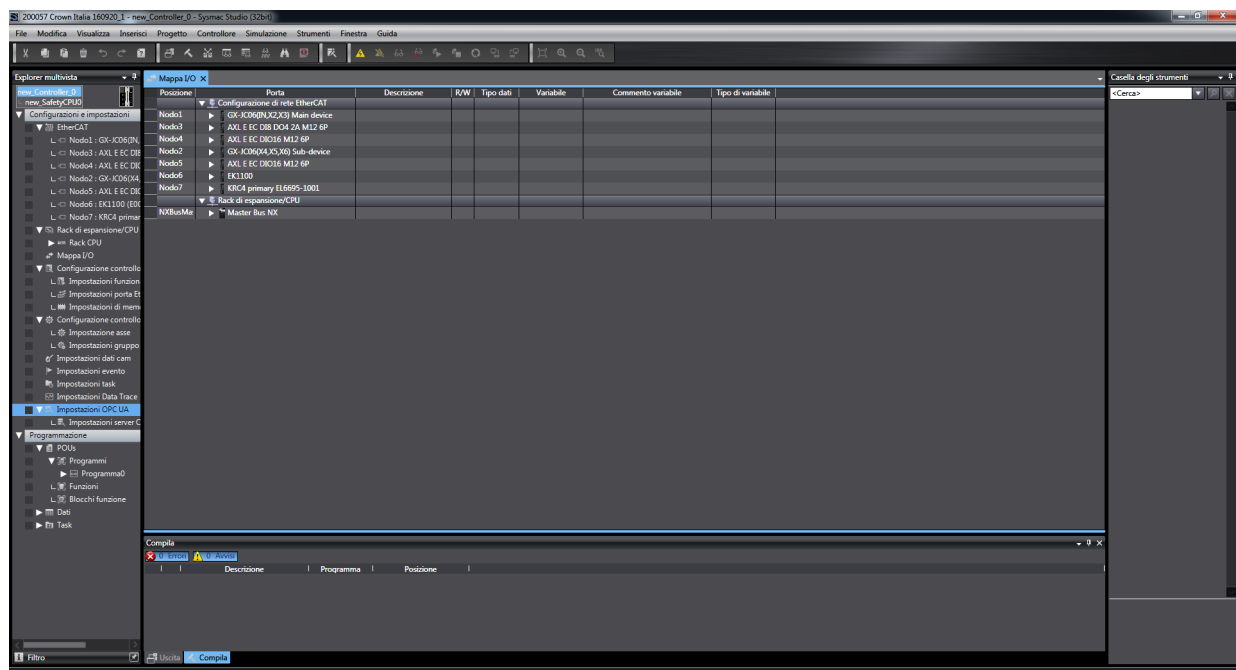


Figure 3.2: Sysmac - Topological View

Each of these data can only be read by specific PLC cards that agree with the data type.

Once population is done, it is yet possible to start programming the PLC. The latter is mostly programmed in *Ladder* language, that is a very simple and low level programming language that associate the state of a coil to the series or/and parallels of contacts.

Those contacts can be seen as switches and their state can be set to be

- NO Normally Open contact, when the contact is not excited its state is said to be low or 0, otherwise it is high or 1

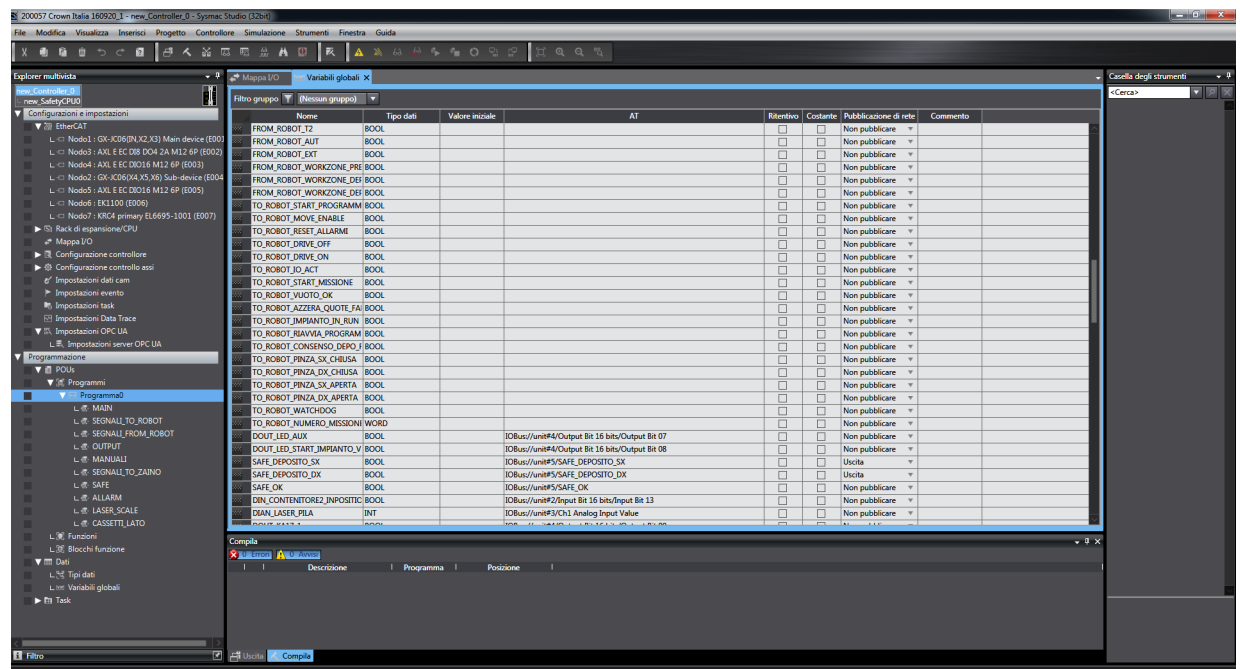


Figure 3.3: Sysmac - Populated I/O Card

- NC Normally Closed contact, when the contact is not excited its state is said to be high or 1, otherwise it is low or 0

More complex bit operations are possible, among them it is worth recalling:

1. MOVE: move the data in input to the output bit,byte,word, double word, etc.
2. SHIFTL : shift the first bit of a byte towards left
3. SHIFTR : shift the first bit of a byte towards right
4. ADD : consist of two inputs, one is the data to be manipulated nd the other is the quantity who has to be summed up to the latter
5. INC : increments of 1 the input data
6. SUB : : consist of two inputs, one is the data to be manipulated nd the other is the quantity who has to be subtracted up to the latter

The ladder language is not the only possible choice to program an OMRON PLC, in fact it is possible to use standard structured programming language, that is a higher level programming language which provides the usage of for, while loops, as well as if, end if statements.

The advantages of ladder language stand in its intrinsic simplicity and clarity, while its main drawback is that loops are not possible and so the same logic for n-different tag must be re written n-times. The advantages of the structured language stands in the coding speed and it has theoretically no disadvantages.

3.5 Structure of the Code

We have chosen to implement the PLC logic, both in ladder and in structured language. The ladder part will deal with all the signals that the PLC receives is

input and send in output from/to the field as well as all the safety devices and HMI communication. Instead the structured language part will only deal with the PLC to ROBOT communication. In particular, our company its used to work with the following framework:

1. the robot PC stores the robot program with all the movement and functions the robot has to accomplish, but can not make it run
2. the external PLC stores all the I/O data from the field and sends enabling signals in order to make the robot program start.

The advantage of this implementation is the separation between the robot and the field, in this way it is possible to make the robot work just aa muscle, while the external PLC works as the brain of the plant.

3.5.1 Mission Concept

As explained above, the external PLC sends a signal in order to enable the robot program. In practice, the external PLC can enable both the entire robot program(all the functions and movements that the robot was programmed to execute) or just a subprogram(just the withdraw function or the palletizing one). Everyone of this functions has an ID that specify itself univocally with respect to all other sub-functions. We use to refer to this subprograms with the name of missions, and we refer to the ID numbers as mission numbers. In practice, when the external PLC wants to enable a mission, it sends the mission number to the robot PC through an analog output variable and waits for the robot PC to answer back that same mission number before enabling the robot to accomplish that mission. This mechanism introduce a redundancy that is particularly important for the safe of safety. In fact, without this mechanism if there was some kind of disturb in the communication between the two PLC, it could be possible to activate another mission with respect to the called one, while instead, with this method that we call ECO, this probability is very low. We can sketch the program flow as in fig.3.4

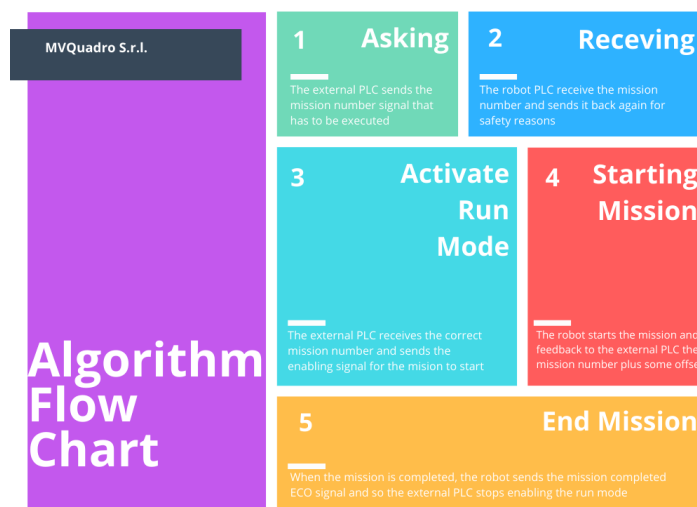


Figure 3.4: Algorithm Flow Chart

1. First the External PLC sends the enabling signal for the robot program to start
2. Then the robot PC sends back the same mission code received
3. When the external PLC receives correctly the ECO from the robot PC it enables the robot mission
4. The robot start the mission it was asked to and sends back in ECO the mission number plus an increasing offset (i.e 300,305,310)
5. When the robot has finished its mission, it will send back to the PLC the following ECO signal $ECO_{end} = MissionStart + 99$
6. The External PLC stops enabling the robot movements and th robot stand still waiting for the next mission number to be executed.

Hazard events can not make the robot move because of enabling communication redundancy (ECO). Safety inputs can stop the robot program everytime under any condition. The safety modules are programmed in order to send the robot to the Home Position, with open gripper, in case of safety issues.

3.5.2 External PLC - Robot PC Communication

The external PLC can communicate with the Robot PC thanks to EtherCAT communication protocol. In particular there are areas of memory in both PLC dedicated to the intercommunication. This data can be accessed by their address, (i.e. Variable : *Distance*, Address: DI 3.0). So the two devices have to point to the respective exchange memory area and therefore it is needed to track this variable's name and addresses. This is done with an excel sheet. In fig.3.5 there's a fragment of this list.

A	B	C	D	E	F	G	H
	WORD	Byte	Variable PLC	Type	Indirizzo PLC	Indirizzo Robot	Utilizzata
OUT_ROBOT_02	7	13	START_MISSION	BOOL	Q316.0	3031 SI	
			RIAVVIA_PROGRAMMA	BOOL	Q316.1	3033 SI	
			PRIMA_HOME_OK	BOOL	Q316.2	3034 SI	
					Q316.3	3035	
					Q316.4	3036	
					Q316.5	3037	
				BOOL	Q316.6	3038	
				BOOL	Q316.7	3039	
			PINZA_CHIUSA	BOOL	Q317.0	3040 SI	
			PINZA_APERTA	BOOL	Q317.1	3041 SI	
		14		BOOL	Q317.2	3042	
				BOOL	Q317.3	3043	
				BOOL	Q317.4	3044	
				BOOL	Q317.5	3045	
				BOOL	Q317.6	3046	
					Q317.7	3047	
	8	15			Q318.0	3048	
					Q318.1	3049	
					Q318.2	3050	
					Q318.3	3051	
					Q318.4	3052	
					Q318.5	3053	
					Q318.6	3054	
					Q318.7	3055	
					Q319.0	3056	
					Q319.1	3057	
					Q319.2	3058	
					Q319.3	3059	
					Q319.4	3060	
					Q319.5	3061	
					Q319.6	3062	
		16					
			WATCHDOG	BOOL	Q319.7	3063 SI	

Figure 3.5: Fragment of I/O External PLC to Robot PC Mapping - Excel Sheet

Chapter 4

Coding

4.1 Sysmac

In this project we have used both the ladder and structured language for programming the communications between the robot pc and the external plc. In particular in the ladder part, we set some important bits and words in order to make the robot start/stop a cycle, as well as to send data to the hmi, in order to give to the user some insight about the state of the current working conditions. Here, we are going to discuss about both the ladder and the structured text code separately.

4.1.1 Ladder Main Program

The Main program is divide into 22 ranks, see fig.4.1 to 4.5.
In rank:

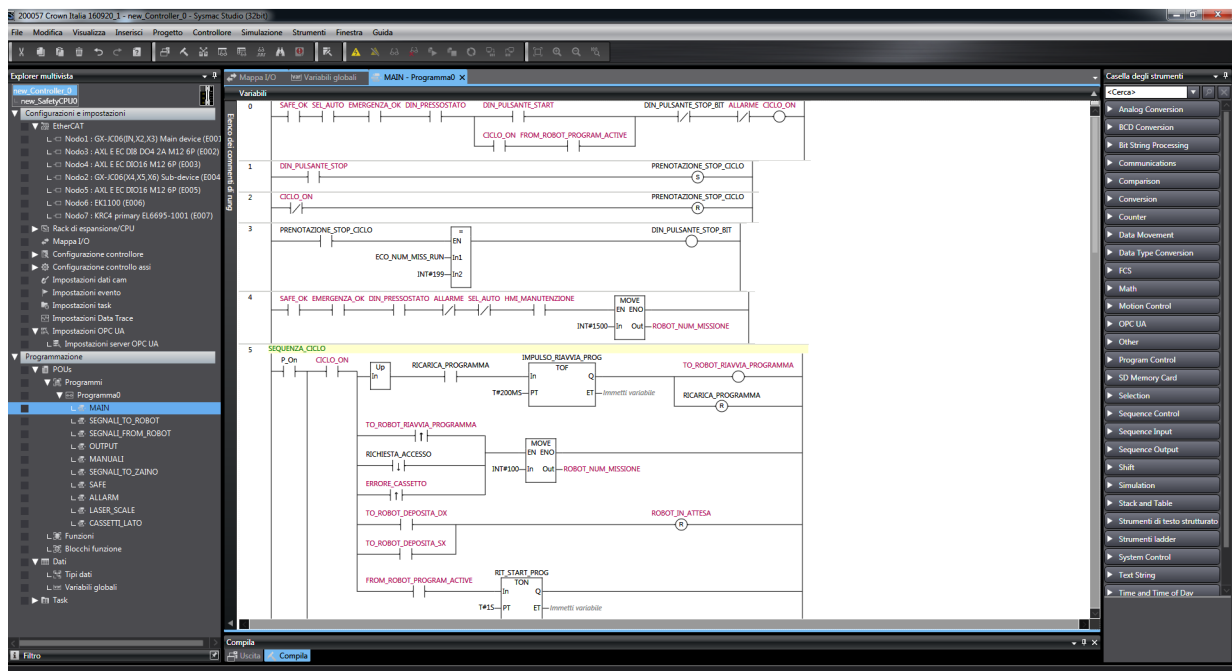


Figure 4.1: Plc Code - Ladder 1/5

- [illegible]

Figure 4.2: Plc Code - Ladder 2/5

9. we can see that after 200ms that the robot sends the signal gripper closed, the plc resets the command bit for closing the gripper and sets to true the variable gripper close.
10. we can see that if there's no emergency we will not set the bit `dout_ev_generale_auto`
11. we can see that if an operator asks to withdraw some cans in order to perform a quality test, we set the corresponding bit to 1
12. we can see the same as in rank 11, but this time the input comes from the push-buttons
13. we can see that if an operator asks to deposit some cans in order to perform a quality test, we set the corresponding bit to 1
14. we can see the same as in rank 13, but this time the input comes from the push-buttons.

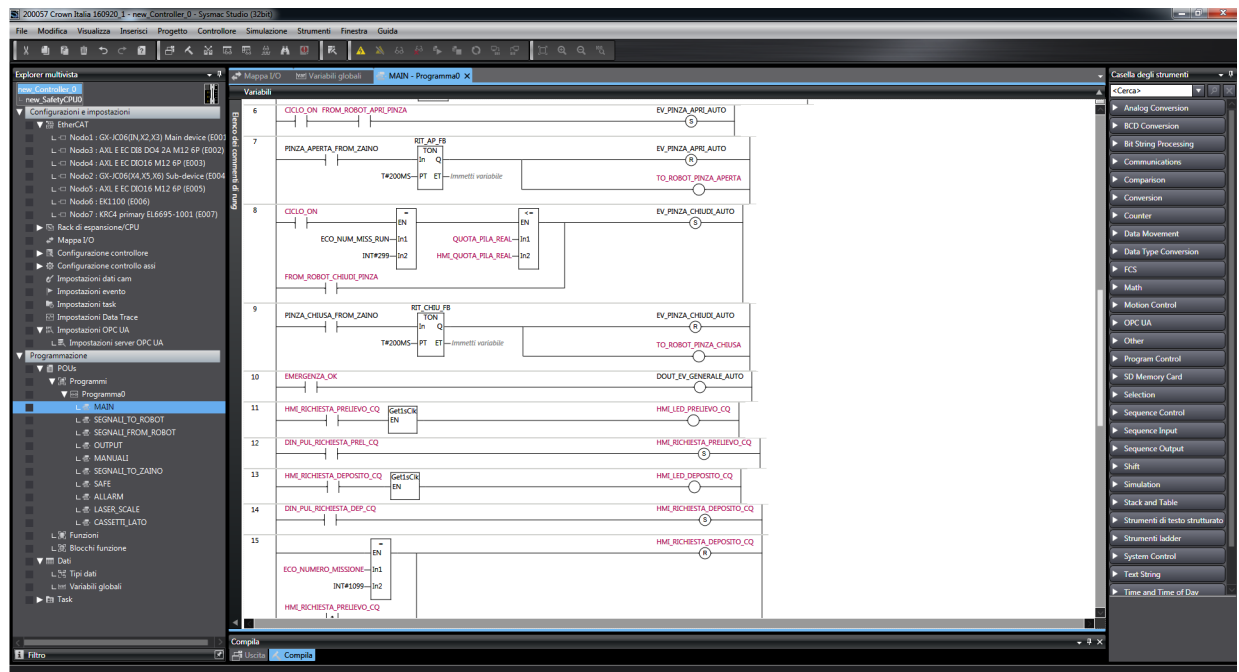


Figure 4.3: Plc Code - Ladder 3/5

15. we can see that if the deposit mission for the Quality Test is over or we get a request for withdrawing cans for quality test purposes, we set the bit `HMI_richiesta_Deposito_CQ` to keep track of this request
16. we can see that if the withdrawing mission for the Quality Test is over or we get a request for depositing cans for quality test purposes, we set the bit `HMI_richiesta_Prelievo_CQ` to keep track of this request
17. we can see that if `ciclo_on` is true and there are no access request nor alarm nor robot waiting signals, then we move 0 in the variable `hmi_indicatore_stato`
18. we can see that if `ciclo_on` is true and there is an access request but not alarm nor robot waiting signals, then we move 0 in the variable `hmi_indicatore_stato`.

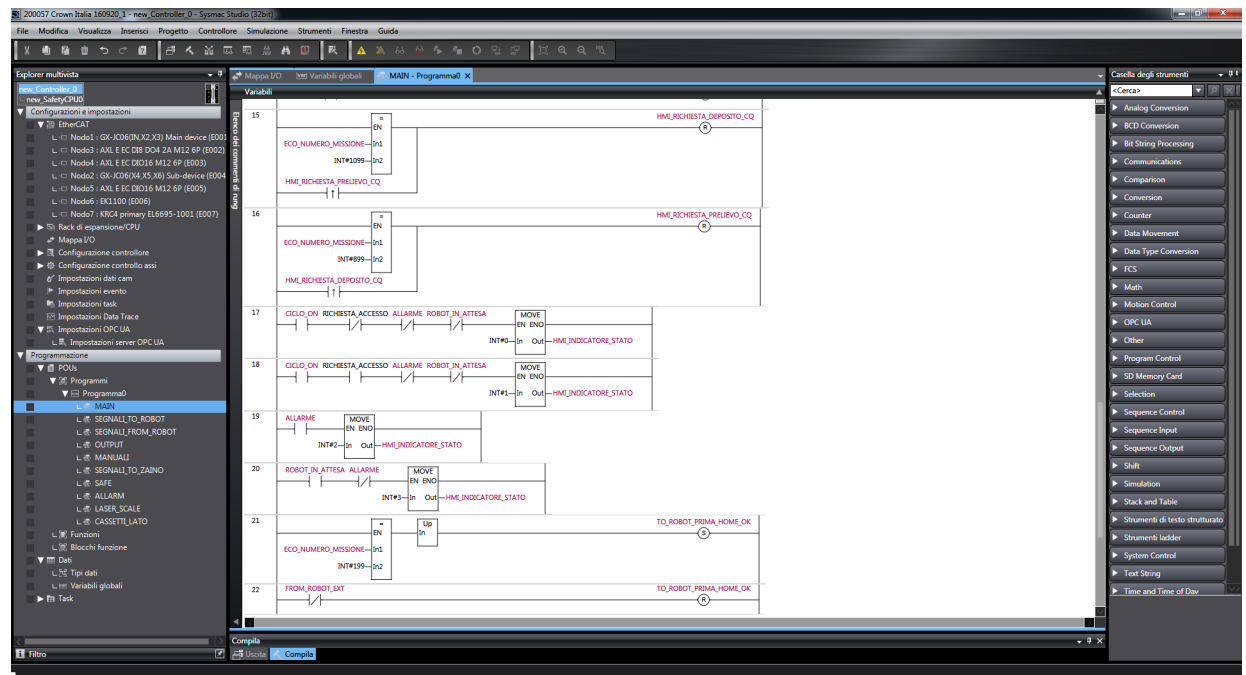


Figure 4.4: Plc Code - Ladder 4/5

19. we can see that if there's the alarm signal we move 2 inside the variable `hmi_indicatore_stato`
20. we can see that if the robot is waiting and there are no alarms then we move 3 inside the variable `hmi_indicatore_stato`.
21. we can see that as soon as the home position mission is over we set the bit `to_robot_prima_home_ok` to make it remember that the robot were there before.
22. we can see that if the robot is not in ext operation mode then we reset the previous `to_robot_prima_home_ok` bit.

In fig.4.6, we have depicted the Main function flow chart.

4.1.2 Structured Code

The goal of this part of the code is just to exchange data from the external plc and the robot one, see fig.4.7. In this part of the code we have implemented the exchange of data like:

1. mission numbers
2. data needed for the hmi visualization
3. gripper state
4. alarms state
5. status feedback

The full program code can be seen in the last pages, pag.[A.1](#)

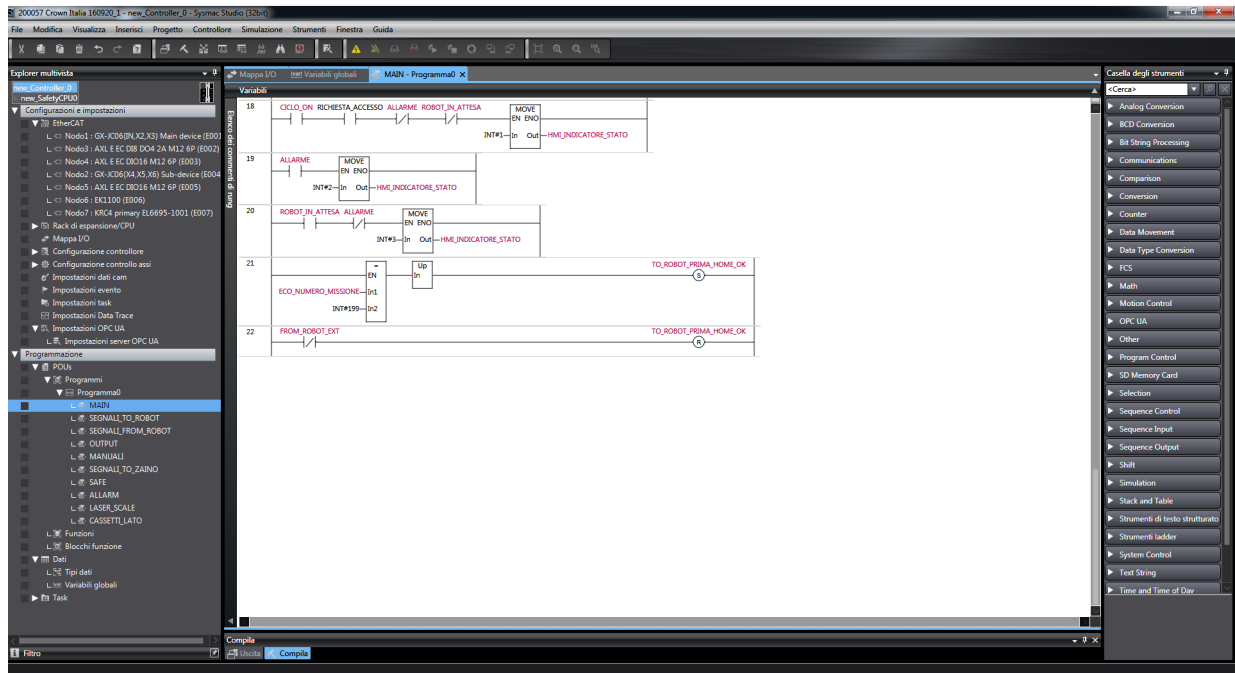


Figure 4.5: Plc Code - Ladder 5/5

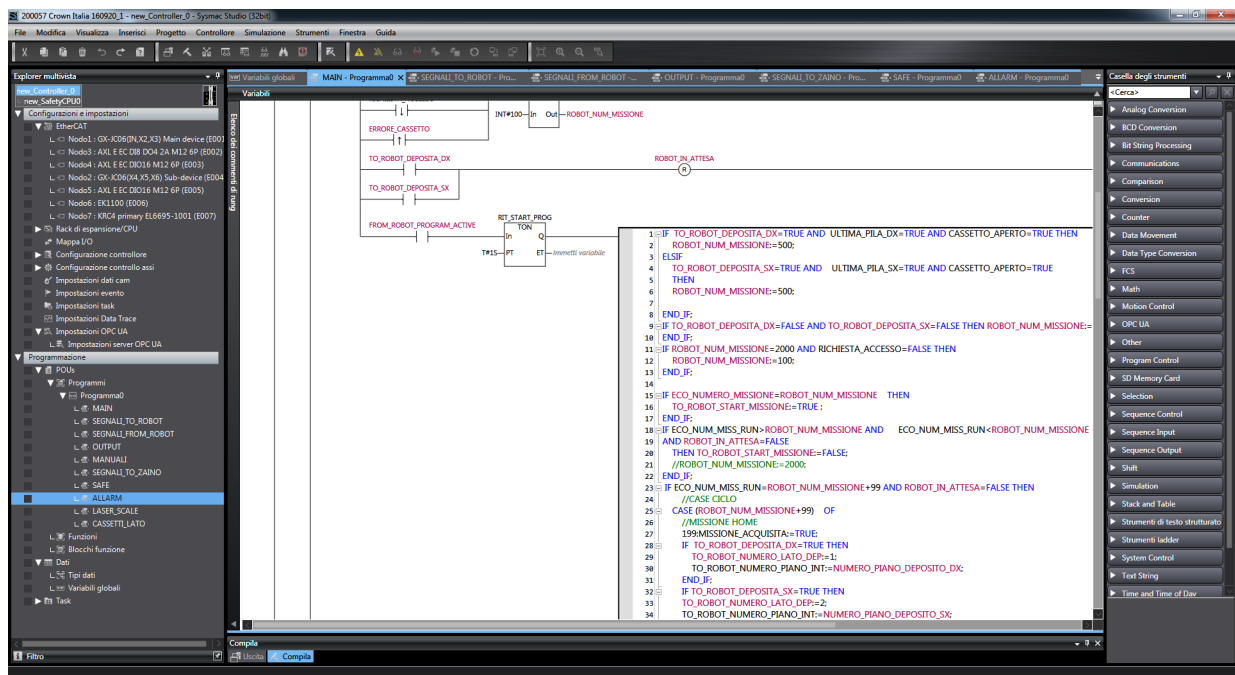


Figure 4.7: Plc Code - Structured Language

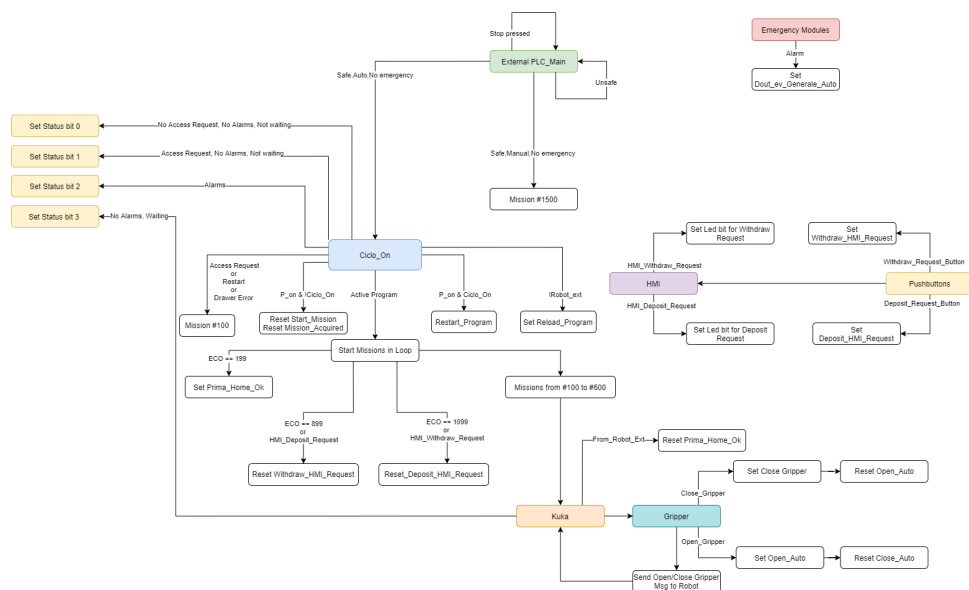


Figure 4.6: Plc Main - Flow Chart

4.2 WorkVisual

The WorkVisual(WV) software package is intended exclusively for the configuration, programming and diagnosis of a KUKA industrial robot or customer-specific kinematic system.

4.2.1 Files Extensions

The WV Editor can be used to edit the following file formats:

1. ADD
2. BAT
3. CONFIG
4. CMD
5. DEL
6. INI
7. KFD
8. KXR
9. LOG
10. REG
11. TXT
12. XML

But WV editor is used primarily to deal with the following files:

1. `.dat` where WV stores all the positions of the robot program (i.e. HomePosition, Withdraw Position, etc.) as an array of n elements who can represent axis angles or digital bit who indicate some particular configurations.
2. `.src` where it is written the source code that is the heart of the root program. All the instructions and movement are defined here
3. `.config` where configuration parameters are defined

For the sake of this thesis work, it is important to deal with the robot code and the logic structure behind it, so we are going to deal with the robot program, taking it for granted that the reader has programming experience.

4.3 Robot Program

The complete robot program is subdivided into 10 missions, that are:

1. Home Position 100 : The robot is sent to a safe position where it can wait for next orders.
2. Approaching Cans 200 : The robot is sent to a point very close to the incoming pile of cans, before the actual grabbing.
3. Withdraw Cans 300 : By closing the gripper, the robot is able to grab the cans pile.
4. Open Drawer 400 : By using the end effector's nail, the robot can open the drawer while grabbing the cans.
5. Close Drawer 500 : By using the end effector's nail, the robot can close an opened drawer.
6. Cans Deposit 600 : The robot actuate all the needed movements to store the cans on the right position inside the drawer.
7. Maintenance 700 : For safety reasons, when maintenance is needed, the robot is send to a special waiting position.
8. Quality Check Withdraw 800 : In order to make an operator check the quality of the produced cans, the robot will withdraw some of them.
9. Quality Check Deposit 1000 : In order to make an operator check the quality of the produced cans, the robot will deposit them in a special area for supervisions.
10. Manual Control 1500 : When the robot is controlled manually, that means through the teach pendant, the mission number that the plc sends to the robot's PC is 1500.

We can further distinguish between missions who are carried out automatically and in loop(missions from 100 to 600, see fig.4.8) as the normal operations missions and the ones who instead are called just in some specific situations(700-800-1000-1500), called special operations, see fig.4.15. We are going now to see the code of each one of these missions.

4.3.1 Normal Operations

Normal Operations are the missions who are carried out automatically and in loop (missions from 100 to 600, see fig.4.8)

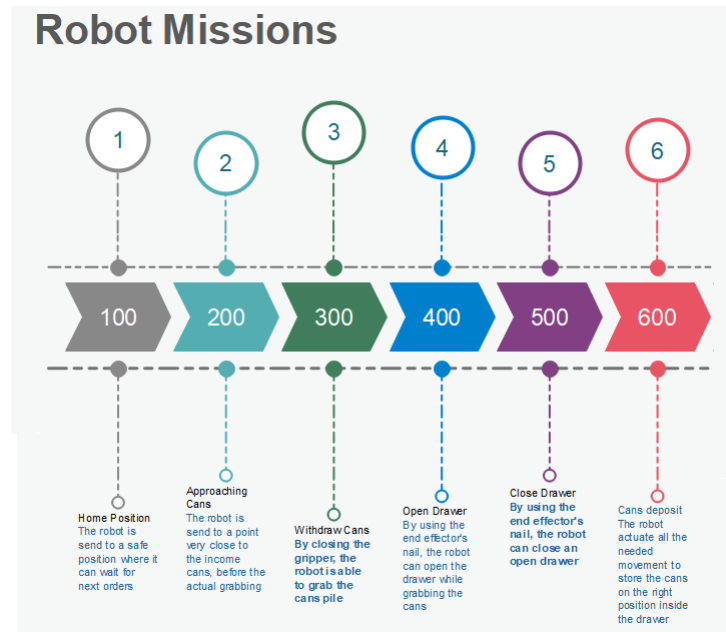


Figure 4.8: Robot missions set for normal Operations

Mission 100 : Home Position

The Home Position is the robot configuration where it can stand still waiting for inputs or the intermediate position between two different missions. This procedure is needed in order to avoid tangling of robot's cables, i.e. pneumatic air transmission. Although this mission should be very simple as it should consist of only one movement, there are different scenarios to take into account. The full code is depicted in fig.4.9. Before programming the robot movements, it is needed to specify the following configuration parameters:

1. The frames with respect to whom all the positions are computed, that are the base and tool frames.
2. The speed of the movements, expressed in percentage.

Having specified the frames and speed, the cases that we needed to take care are depicted in fig.4.9 and in particular:

1. Is the robot yet in Home Position?
2. If not it could be in : Left/Right Rack zone

Depending on this two cases, we can further distinguish the movements and directives needed to safely go into the Home Position. Moreover, during the computations, we may need to apply some corrections in order to not make the robot move beyond certain limits of the z axis, so that it will not collide with the roof. The same has to be done for the x axis, in order to not make the tool collide with the chest

of drawers. By looking at fig.4.9, we can see that if the robot actual position tell us that the robot is in the left rack zone, we can distinguish three different situations:

1. The robot is in the Deposit Cans area, so we need to increase the z axis current value of $80mm$ and then perform two linear movements (LIN instruction) on for x axis, $x = 0$ and the other for the y one, $y = 1100$, finally we can use a point to point motion to bring the robot in the Home Position
2. The robot is in the Withdraw Drawer area, so we need to increase the z axis current value $20mm$ and then perform two linear movements (LIN instruction) on for x axis, $x = 0$ and the other for the y one, $y = 1100$, finally we can use a point to point motion to bring the robot in the Home Position
3. The robot is in the Withdraw Cans area, so we need to check if the z axis current value is greater than $1350mm$ and then perform two linear movements (LIN instruction) on for y axis, $y = 0$ and the other for decreasing the x one, but making sure that its value it's not lower than $400mm$, in order to not collide with the Withdraw Cans structure, finally we can use a point to point motion to bring the robot in the Home Position.

The exact same considerations can be applied to the right hand side. Instead if the robot was already in Home Position or very close to it, we just perform a point to point motion in order to remain in the Home Position.

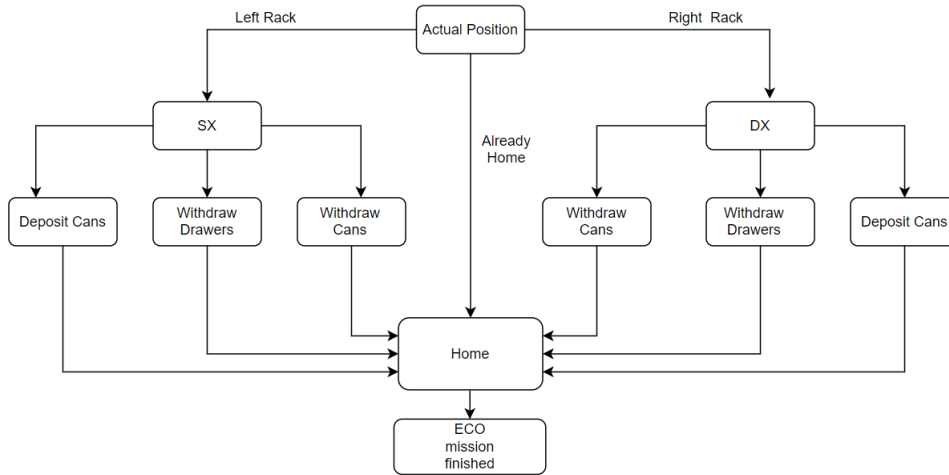


Figure 4.9: Flow Chart Mission 100 - Home Position

Mission 200 - Withdraw Cans Approach

The goal of this mission is to prepare the robot to grab the cans pile. So in order to do so, we make sure that the robot will reach a certain configuration, where it is very near to the cans to be taken, so that in the next mission it can close the gripper and withdraw the cans. In the first place, we open the gripper for safety reasons, then we make the robot move to a safe position for withdrawing cans, where with safety we mean a position who will not make the cables tangling up. After this, we make the robot go into the actual approach position. For the sake of testing and programming simplicity, it is possible to insert some corrections to this position quotes, thank to the HMI panel. So during testing procedure, it is possible to modify in real time

this approaching position in order to tune it to the best. Finally, we increase the z axis, written in the tool's frame, of 150mm , in order to reach the withdrawing cans structure.

In fig.4.10, we find the complete flow chart of this simple mission.

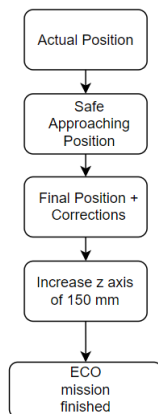


Figure 4.10: Flow Chart Mission 200 - Withdraw Cans Approach

Mission 300 - Withdraw Cans

The goal of this mission is to grab the cans pile. Since the previous mission have sent the robot to the right position to withdraw the cans, we make the robot close the gripper and once it has grabbed the pile, we program it to first move outside the withdrawing cans zone along the x axis, with a linear motion and then it is send away from this point thanks to a linear motion along x and z axis. At the end of this motion, the robot can safely return to the safe position defined in the variable *PositionSafePrelievo*.

In fig.4.11, we find the complete flow chart of this mission.

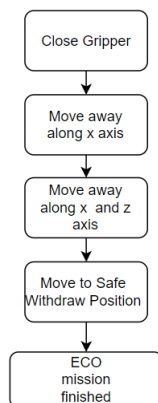


Figure 4.11: Flow Chart Mission 300 - Withdraw Cans

Mission 400 - Open Drawer

The goal of this mission is to first decide in which box should the cans of pile be deposited and then deposit it. Since the previous called mission has opened the right drawer, this mission starts looking for the Plane number, that is the number of drawer to be used, starting from the lowest one up t the highest one, 1-15. At first,

we set the speed in percentage depending on plane number, then depending on the left or right rack, we have to know in which drawer the robot has to operate and basing on this, it will be at first sent to a safe position (low, middle or high). After this the robot PC will load the final deposit position as well as the approaching position. Now the robot can get to the approaching position and then to the final one opening the gripper in order to allow the cans to be put in the right place. Once this operation is done, it follows three linear motions that are intended to safely get away from this position, each of these motions will involve movement along Y and Z axis, with different slopes. Once these movements are accomplished, the robot is sent to the Safe Left Rack position or the right one depending on the previous operations. In fig. 4.12, we find the complete flow chart of this mission.

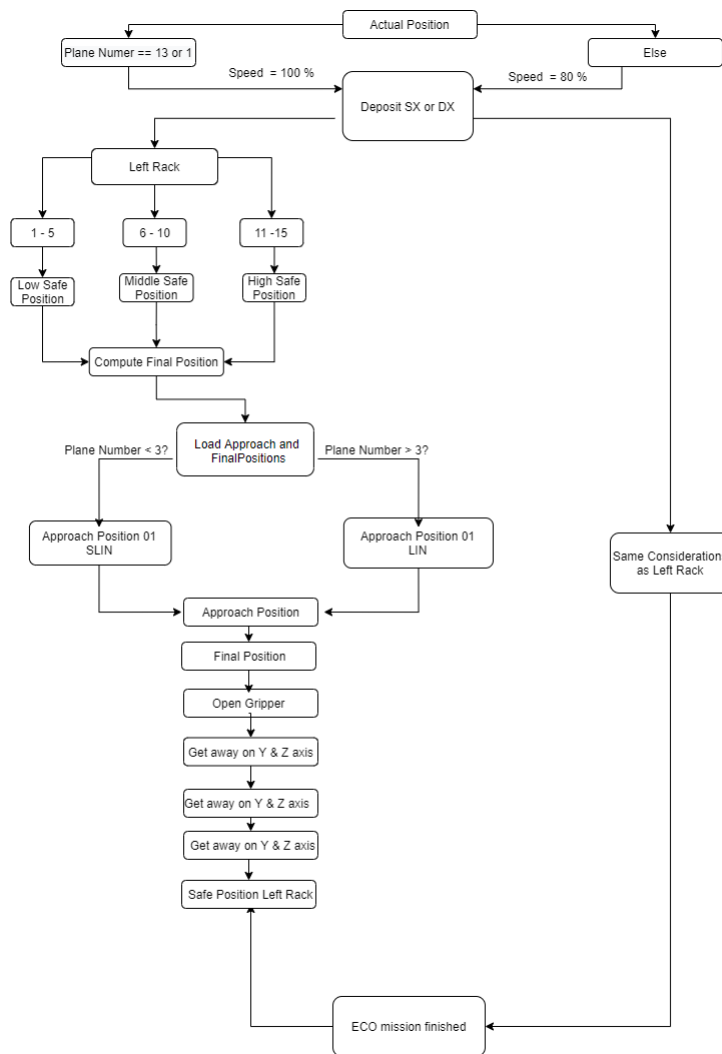


Figure 4.12: Flow Chart Mission 400 - Deposit Cans

Mission 500 - Close Drawer

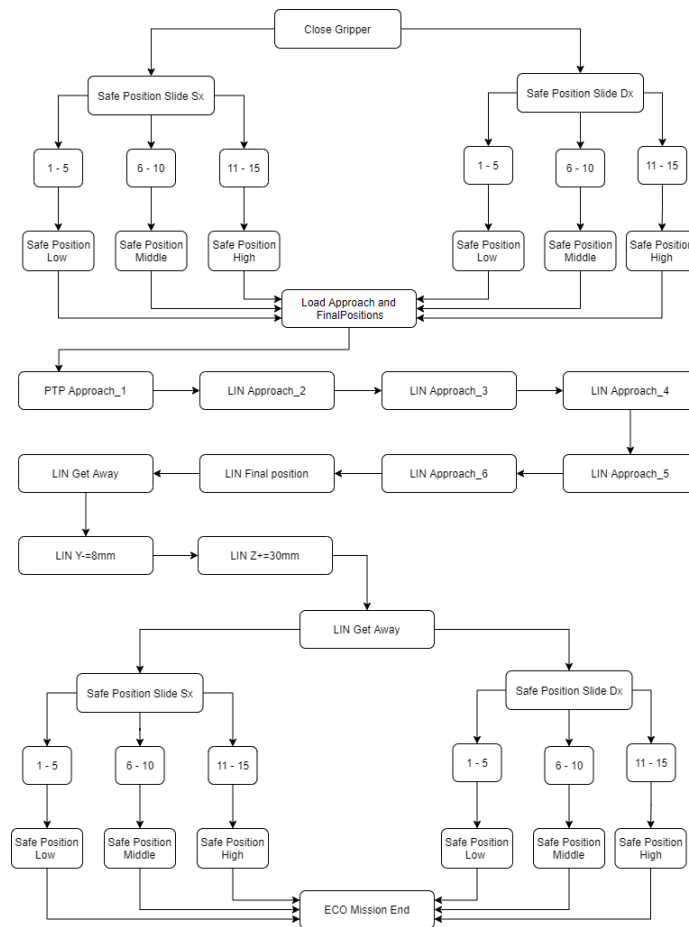


Figure 4.13: Flow Chart Mission 500 - Close Drawer

The goal of this mission is to close an opened drawer. The robot has to know which of the two rack is involved, the left or right one. After it has recognized where it has to operate, it will be sent in a safe position called "Safe Position Slide" and then, depending on the height of the drawer to be closed, it will move into the next safe position, called "Safe Position Low/Middle/High". At this stage, the robot PC will load the approaching and final points. In particular, we have programmed the robot to follow 6 approaches points before getting to the final one, in order to be sure that the mission is correctly accomplished without collisions assuring repeatability. Once the robot reaches the final position it starts to get away at first parallel to the ground and then along a 3D line, instructions LIN Y=-8mm and LINZ+=300mm. At this point the drawer is closed and the robot its far away from it, so depending on the previous knowledge of the involved rack, left or right and on the height of the drawer, the robot will be sent to the Safe Position Low/Middle/High and the mission is completed.

In fig.4.13, we find the complete flow chart of this mission.

Mission 600 - Cans Deposit

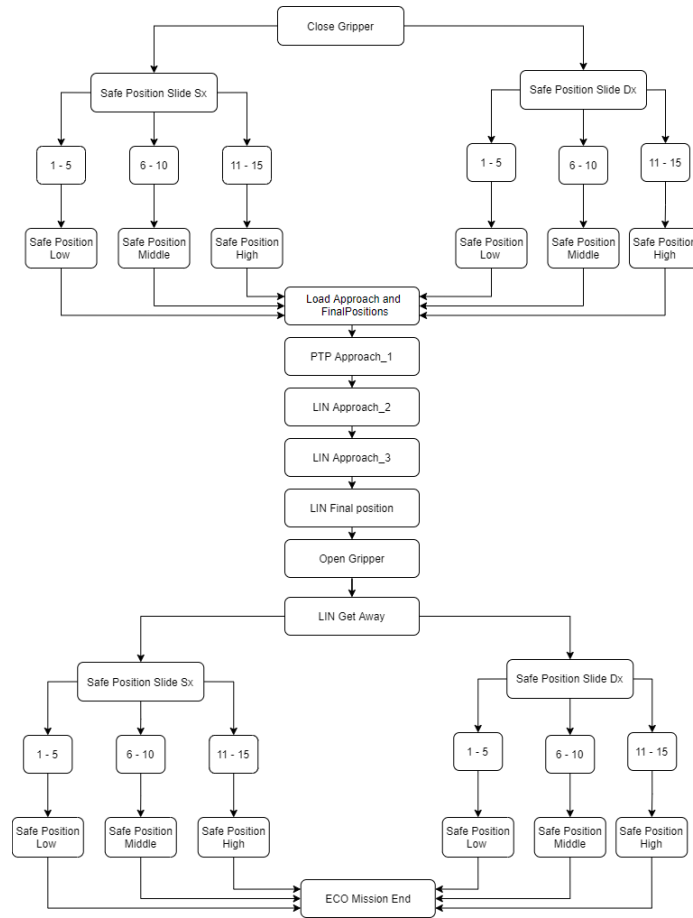


Figure 4.14: Flow Chart Mission 600 - Cans Deposit

The goal of this mission is to deposit cans in the right place inside a drawer. At first we have programmed to close the gripper, in order to be sure the robot won't leave the pile of cans, moreover, the robot has to know which of the two rack is involved, the left or right one. After it has recognized where it has to operate, it will be sent in a safe position called "Safe Position Slide" and then, depending on the height of the drawer to be closed, it will move into the next safe position, called "Safe Position Low/Middle/High". At this stage, the robot PC will load the approaching and final points. In particular, we have programmed the robot to follow 3 approaches points before getting to the final one, in order to be sure that the mission is correctly accomplished without collisions assuring repeatability. Once the robot reaches the final position it opens the gripper so that the pile of cans can be correctly deposited. Then it starts to get away from them. At this point, depending on the previous knowledge of the involved rack, left or right and on the height of the drawer, the robot will be sent to the Safe Position Low/Middle/High and the mission is completed.

In fig.4.14, we find the complete flow chart of this mission.

4.3.2 Special Operations

Special Operations missions are called just in some specific situations (missions 700, 800, 1000 see fig. 4.15)



Figure 4.15: Robot missions set for special Operations

Mission 700 - Maintenance

The maintenance mission is never called by the external plc, as this is a prediction for future updates. It can happen that the operators have the necessity to operate on the robot or on its tool and therefore that they need that the robot goes in a particular configuration different from that of HomePosition. It is for this reason that the maintenance function is important, however in our application it was not requested, but it was still designed in case of future customer needs.

Anyway, in fig. 4.16, we find the flow chart of this mission.

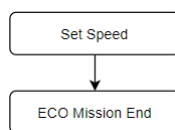


Figure 4.16: Flow Chart Mission 700 - Maintenance

Mission 800 - Quality Check Withdraw

The goal of this mission is to withdraw cans in a special place, so that an operator can check the quality of the produced cans. At first, we have programmed to open the gripper, for preparing the robot to take a new pile of cans, the robot has to go to the Quality Check position for withdrawing the cans. At this stage, the robot will be sent first to an approach position and then to the final one, so that it can easily and safely grab the cans. Once the robot reaches the final position it closes the gripper so that the pile of cans can be correctly withdrawn. Then it starts to get away from them and finally it returns to the Quality Check Safe Position.

In fig. 4.17, we find the complete flow chart of this mission.

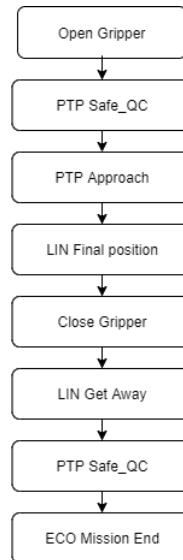


Figure 4.17: Flow Chart Mission 800 - Quality Check Withdraw

Mission 1000 - Quality Check Deposit

The goal of this mission is to deposit cans in a special place, so that an operator can check the quality of the produced cans. At first, the robot has to go to the Quality Check position for withdrawing the cans. Then, it will be sent first to an approach position and then to the final one, so that it can easily and safely bring the cans with it. Once the robot reaches the final position it opens the gripper so that the pile of cans can be correctly deposited. Then it starts to get away from this position and finally it returns to the Quality Check Safe Position.

In fig.4.18, we find the complete flow chart of this mission.



Figure 4.18: Flow Chart Mission 1000 - Quality Check Deposit

Mission 1500 - Quality Check Deposit

This is not properly a mission, because the robot does not operate automatically. But anyway, when the robot is controlled manually, through the teach pendant, the mission number that the plc sends to the robot's PC is 1500.

4.4 Full Missions Code

The full missions code can be seen in the last pages [pag.106](#)

Chapter 5

HMI Operator Panel

A Human-Machine Interface (HMI), see fig.5.1 is a user interface or dashboard that connects a person to a machine, system, or device. While the term can technically be applied to any screen that allows a user to interact with a device, HMI is most commonly used in the context of an industrial process. Although HMI is the most common term for this technology, it is sometimes referred to as Man-Machine Interface (MMI), Operator Interface Terminal (OIT), Local Operator Interface (LOI), or Operator Terminal (OT). HMI and Graphical User Interface (GUI) are similar but not synonymous: GUIs are often leveraged within HMIs for visualization capabilities. In industrial settings, HMIs can be used to:

- Visually display data
- Track production time, trends, and tags
- Oversee KPIs
- Monitor machine inputs and outputs
- And More

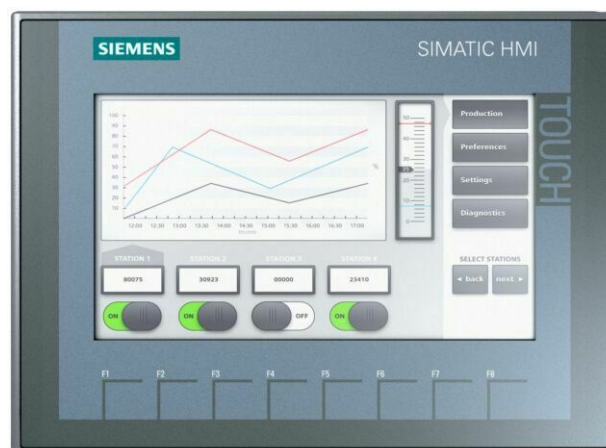


Figure 5.1: HMI - Human-Machine Interface

Similar to how you would interact with your air-conditioning system to check and control the temperature in your house, a plant-floor operator might use an HMI to check and control the temperature of an industrial water tank, or to see if a certain

pump in the facility is currently running. HMIs come in a variety of forms, from built-in screens on machines, to computer monitors, to tablets, but regardless of their format or which term you use to refer to them, their purpose is to provide insight into mechanical performance and progress. HMI technology is used by almost all industrial organizations, as well as a wide range of other companies, to interact with their machines and optimize their industrial processes.

Industries using HMI include:

- Energy
- Food and beverage
- Manufacturing
- Oil and gas
- Power
- Recycling
- Transportation
- Water and waste water
- And many more

The most common roles that interact with HMIs are operators, system integrators, and engineers, particularly control system engineers. HMIs are essential resources for these professionals, who use them to review and monitor processes, diagnose problems, and visualize data.

5.1 Common Uses of HMI

HMIs communicate with Programmable Logic Controllers (PLCs) and input/output sensors to get and display information for users to view. HMI screens can be used for a single function, like monitoring and tracking, or for performing more sophisticated operations, like switching machines off or increasing production speed, depending on how they are implemented. HMIs are used to optimize an industrial process by digitizing and centralizing data for a viewer. By leveraging HMI, operators can see important information displayed in graphs, charts, or digital dashboards, view and manage alarms, and connect with SCADA and MES systems, all through one console. Previously, operators would need to walk the floor constantly to review mechanical progress and record it on a piece of paper or a whiteboard. By allowing PLCs to communicate real-time information straight to an HMI display, HMI technology eliminates the need for this outdated practice and thereby reduces many costly problems caused by lack of information or human error.

5.2 Premium HMI

In order to be able to design an HMI, it is needed some programming interface. There are various IDE for such a purpose. In our case, we have used Premium HMI, see fig.5.2. Premium HMI, [6], offers the most advanced graphic technologies based

on XAML standards and it is the only visualization solution supporting XAML vector graphics also on Windows CE operating system.

- Premium HMI introduces a new 16 million colours graphic rendering engine supporting XAML advanced graphic technologies
- Sophisticated management of transparency and shading effects
- Automatic re-Dimensioning of screens for devices with different graphic resolutions; this feature of Premium HMI allows existing projects to be easily reused on different systems regardless of the graphic resolution of the display
- Rich gallery of vector graphic objects (buttons, switches, analogue displays, sliders, etc.) to realise unprecedented user interface projects
- Complete set of graphic animations (including movement of objects along definable routes)
- SVG import functionality
- Alias support and inheritance of symbols with definition of public symbols and automatic propagation of modifications from parent object to child object
- Integrated support for multi-monitor systems

Premium HMI has a complete communication drivers library for the most used PLCs on the market. Premium HMI also provides:

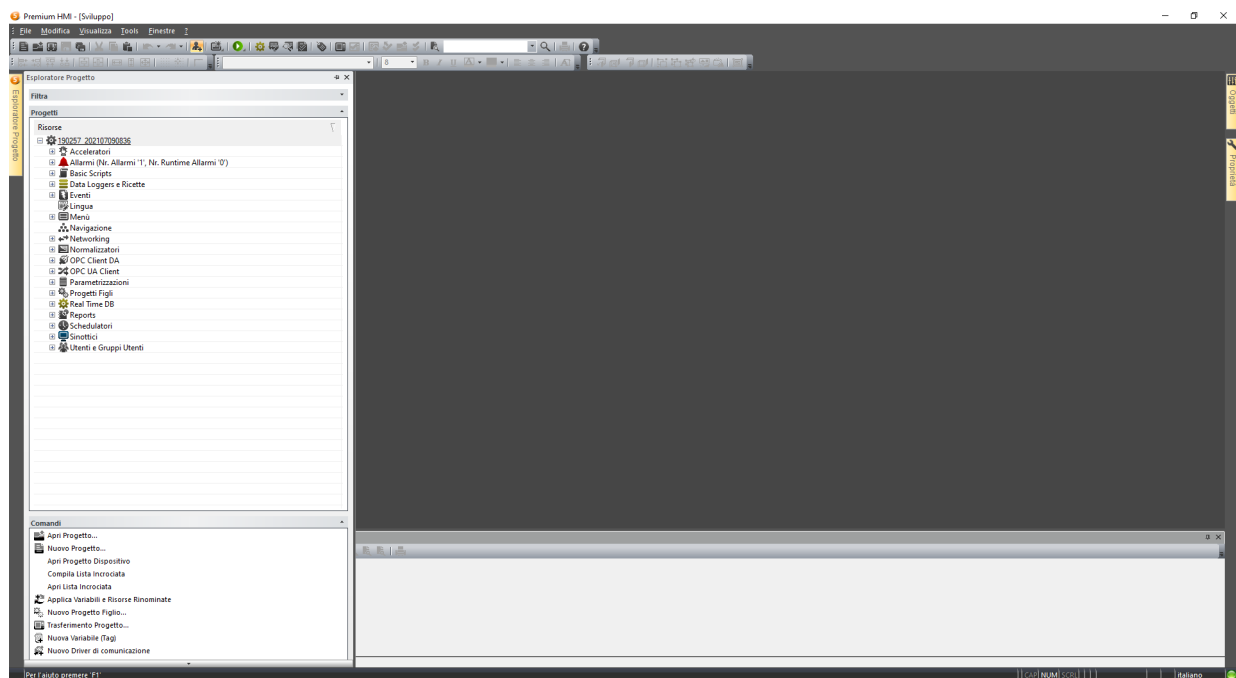


Figure 5.2: Premium HMI IDE

- High performance and reactivity of controls to meet the most demanding requirements of machine manufacturers that need fast data updating and a prompt dispatch of commands to actuators

- Support for multi-protocol interfacing with data transfer function (gateway) between communication channels
- Real-Time I/O ODBC Link provides connectivity towards company's information systems. Each variable (Tag) has the reading-writing connectivity to an external relational DB. Therefore the Real-Time DB of the project can be shared automatically (partially or entirely) on a DB table, allowing sharing of plant's real-time data with the company's ERP
- Availability of normalisers for the application of non-linear transformations to the variables

Premium HMI provides maximum reliability in events management, guaranteeing continuous and immediate system/ machine monitoring, improving its efficiency and minimising production downtime

- Alarms are managed according to ISA S-18 standards, but they are entirely customizable with high-configurable objects and templates-oriented programming (threshold alarms, digital alarms, warning messages without recognition cycle, etc.)
- Simple definition and configuration of repetitive alarms using templates
- Fixed or variable triggering thresholds determine activation of the alarm, managing the four standard operating statuses (ON, OFF, ACK and RST) and the consequent representation of active alarms in visualization objects, managed by Windows or Banners with several filters (by time, area, priority, period, etc.) and the possibility to dynamically combine help and wizards on external files (CHM, HTML, PDF)
- Library tools for the organic visualization of active alarms, alarms awaiting acknowledgement and the alarm log with the possibility to apply visualization filters for a simple search and analysis
- The Alarms Window and the Historic Log Window are the tools to visualize active or stored alarms and can be inserted and configured as objects in any screen
- Premium HMI introduces the possibility to select an active alarm and directly view its history in the alarm window
- The Alarm Log automatically records all the events (Alarms, Driver Events or System Events) on the relational database (even on Windows CE) or on text files
- Alarm Dispatcher to promptly send alarms or messages via SMS or E-mail; the notification is sent to the specific User or Group of Users and can be customised depending on timetables, calendars, work shifts, etc. SMS notification dispatcher based on SMPP protocol (dispatches SMSs via Internet without modem)

5.2.1 Designed HMI

The machine has an operator interface called HMI. With it you can monitor the correct operation of the system. Manage the system in manual mode, monitor the

status of alarms, check the history of events, set parameters adjustment. The operator panel (HMI) consists of two structures that are common to all pages realized. The upper part of the interface operator, is composed from the bar of the title.

- BOTTONE DI DISCONNESSIONE
- SMARTpad
- KEYSWITCH
- PULSANTE DI SICUREZZA
- SPACE MOUSE
- PULSANTI DI AVANZAMENTO
- PROGRAM
- OVERRIDE
- AVANZAMENTO
- OVERRIDE
- MENU PRINCIPALE
- DISPLAY TOUCH
- SCREEN
- PULSANTI DI STATO
- PULSANTE START
- PULSANTE DI AVVIO IN BACKWARDS
- PULSANTE STOP
- MOSTRA TASTIERA
- TOUCH

The button bar or navigation bar is located on the left side of the interface and serves to navigate between the pages of the HMI, to request the removal of lids from the axis of supply and subsequent storage within the CQ station, to request the picking of from the CQ station and palletizing inside the box, for display the status of the machine and to select the mode of operation (Manual or Automatic). The operations of Request withdrawal for CQ and Request deposit from CQ can be also carried out by quality control control control panel placed next to its workstation.

5.3 Home Page

The main page of the panel is the home page from which you can have a overview of the operating status of the plant. In the background is represented the layout of the machine with two lights that light red if they should be disconnected RFID sensors, cause access door opening and CQ drawer. At the bottom of left there is a button with which you can vary the operating status and switch from Manual

(Yellow Color) to Automatic (Green Color). There are two buttons called "Reset Left/Right Box" that allow zeroing of the cycle; the reset buttons shall be pressed downstream of the following procedure:

- Remove the complete box from its position (after having inserted the safety, placed in the upper part of the box, and have checked that all the drawers are locked and safely);
- Place the empty box on the location of interest and remove the safety;
- Press the button "Reset Left/Right Box" to reset the cycle parameters;
- Press "Start" to start the cycle.

There are also two indicators relating to the working cycle: number of the Battery being processed and number of the Worktop. In low there is a window where the machine's operating state is reported and process speed: the "Plant Status" indicates the operating status, in real time, of the machine; the "Command Miss" indicates the operation performed by the PLC in real time; the "Miss Robot" means the operation performed by the Robot in real time. Ensure that the safety has been removed or is likely to damage the drawer extraction component and the Gripper Failure to press the Reset button when processing a new box, may cause incorrect arrangement of lids, as well as damage to machine components. Always press the key Reset Box when processing a new box. In addition, there is an Alarms button with which you can interact to open its and check which alarms have been triggered.

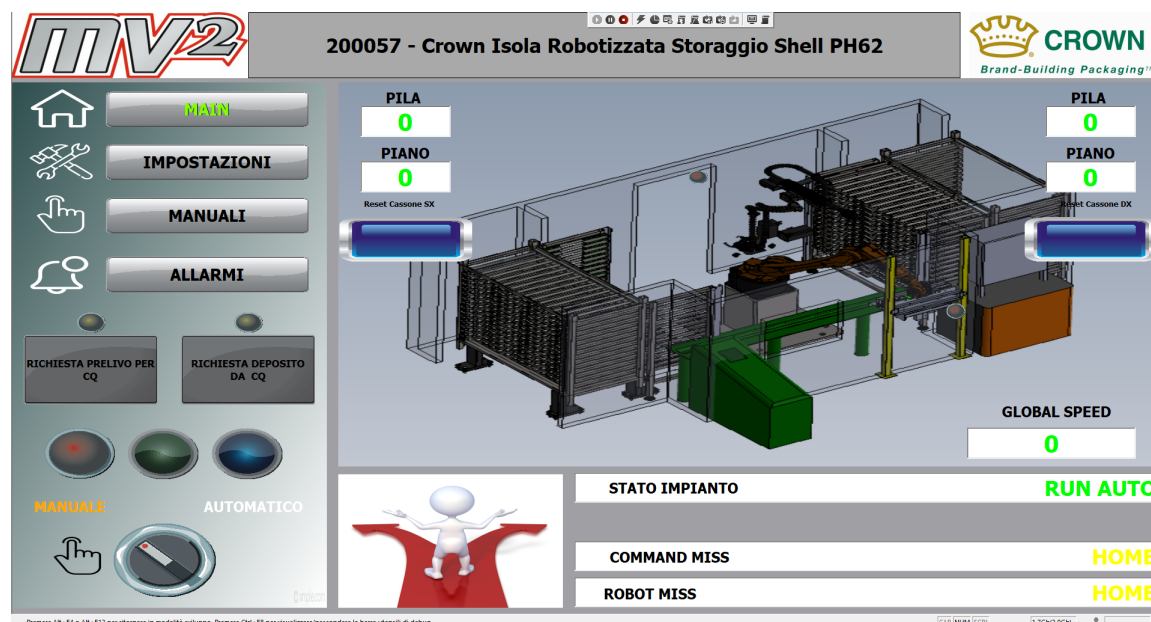


Figure 5.3: HMI - Home Page

5.3.1 Manual Operation Page

If necessary, you can control the Robot Gripper, in the only manual mode, and through this page you can activate the actions of the pliers. As well as monitoring

the status of the associated facilities. It is also important to monitor the status of the plant. **WARNING:** After a collision, always check that the deposit number is consistent with that which is actually the situation of machine downtime. If it is not consistent, set the correct deposit number to prevent the Robot go to collide. This is an operation to do with the plant in **MANUAL** mode and the robot in **HOME** position. The number on the panel corresponds to the deposit that the **ROBOT** must carry out.

1. STATO
2. CASSONE
3. SELETTORE
4. AUT/MAN
5. RESET
6. CASSONE
7. CONTROLLO
8. QUALITA'
9. INDICATORI STATO
10. MACCHINA
11. PULSANTE
12. ALLARMI
13. INDICATORI DI STATO
14. DELLA MACCHINA
15. INDICATORI
16. SENSORI RFID

5.3.2 Setup Page

The parameters can only be changed when the system is stationary, in manual mode. In addition the modification of some parameters shall be carried out by checking that there is consistency with the standstill situation.

On the **MAIN** page you can change the following parameters:

1. Stack - Number of the stack to be deposited, from 1 to 13, in the plan indicated (modifiable only when the system is stationary, in manual mode, checking that there is consistency with the stationary situation to avoid collisions).
2. Plane- Number of the plan to work from 1 to 15 (modifiable only to plant stationary, in manual mode, checking that there is consistency with the situation of machine downtime to avoid collisions).
3. Global Speed - Variable robot handling speed from 0 to 100.

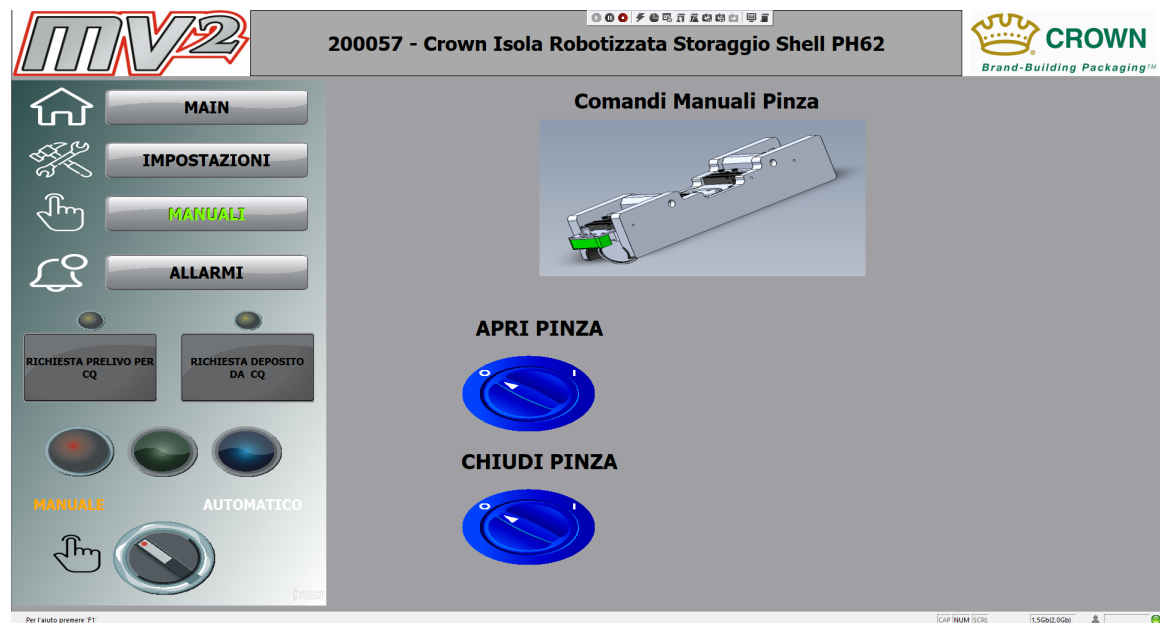


Figure 5.4: HMI - Manual Page

4. Palletization change side - Allows you to change the palletizing side from left to right and vice versa. At the pressure a Pop-Up window will appear that will ask to confirm the side change; pressing YES will confirm the choice, press NO to cancel it.

On the Setup page you can change the following parameters:

1. Laser height set socket - Distance between the light source of the laser sensor and battery head covers. Determines the length that must have the stack of lids to palletize. The lower the quota and the higher the length of the cover stack. Once reached the set altitude the Robot is authorized at the removal of the lids on the feeding axis.
2. Max number of planes on the left/right side - Maximum number of planes the robot must fill inside the box during the production cycle. The values can be set between 1 and 15.

Read Laser Quota: Indicates the instant quota read by the sensor (can not be modified). Once the parameters are changed, they become immediately operational.

5.3.3 Alarm Pages

This page monitors the alarms of the machine. They are referred to as Leds that are coloured red in case they are activated. All alarms are reported from the luminous column with the lighting of the red light, which disappears at the moment of Reset. The alarms managed are the following:

1. ALLARME CIRCUITO EMERGENZE
2. ALLARME CIRCUITO CARTER
3. ALLARME MICRO PORTA ACCESSO

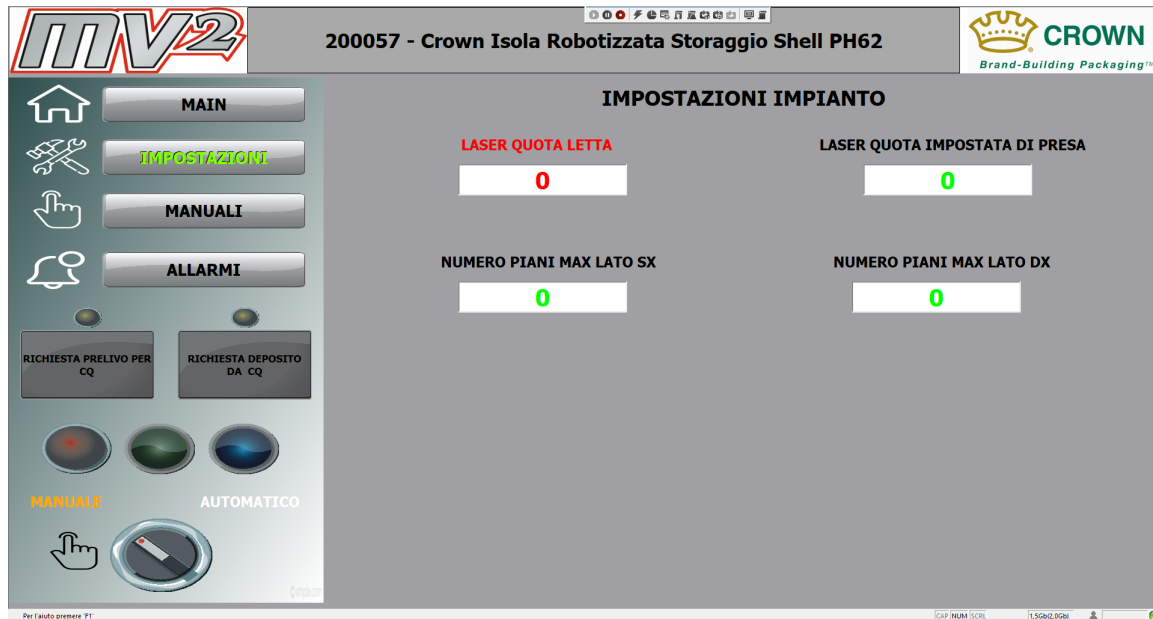


Figure 5.5: HMI - Setup Page

4. ALLARME MICRO PORTA QUALITA
5. ALLARME PRESENZA CASSETTIERA DX
6. ALLARME PRESENZA CASSETTIERA SX
7. ALLARME QUOTA SICUREZZA LASER
8. ALLARME DISCREPANZA CASSETTI DX
9. ALLARME DISCREPANZA CASSETTI SX
10. CASSETTIERA PIENA DX
11. CASSETTIERA PIENA SX

Moreover we have the following alarms:

- **ALLARME CIRCUITO EMERGENZE:** it is connected to emergency mushrooms and is activated at pressure of one of them. After restoration, it is important to reset the alarm. The occurred Execution of the reset is indicated by the color of the button ALARMS in green.
- **ALLARME CIRCUITO CARTER:** It is activated when the circuit that connects the sensors of the perimeter access is interrupted, that is when you try to access the work area from two openings available: for opening the operator access door and/or for opening the CQ drawer. The alarms are visible on the HMI by means of two indicators, positioned on the layout in the Home screen at the above accesses, which emit light red when active. To restore the state of operation of the machine must close the accesses, ensure that no operator is present within the perimeter of the machine and press the Start button.
- **ALLARME MICRO PORTA ACCESSO:** Active when operator access door comes open. The sensor is restored by closing the door and pressing the Start button on the operator panel. The activation of this alarm involves the plant

shutdown, of Consequently, the machinery must be restored to its state of operation.

- **ALLARME MICRO PORTA QUALITA'**: Active when quality desk drawer is opened. The sensor is restored by closing the door and pressing the button Start on operator panel. Activation of this alarm does not stop the process Unless the robot is heading to the quality control station.
- **ALLARME PRESENZA CASSETTIERA DESTRA/SINISTRA**: It is activated when the box is not present in its location or has not been correctly positioned. The presence of this alarm does not allow to start the production cycle.
- **ALLARME QUOTA SICUREZZA LASER**: It is an alarm that is activated when the battery of lids present on the feed axis, at the outlet from the furnace for the application of mastic, has not reached the level of work set, that is, when the number of lids is not enough to obtain a complete stack for palletizing.
- **ALLARME DISCREPANZA CASSETTI DX/SX**: Indicates that the number of the Top inserted in the Home screen Plan box does not match with what was processed before manual insertion by HMI. To restore machine operation enter the correct Plan number and press Start.
- **CASSETTIERA PIENA DX/SX**: At the end of the production cycle, the robot takes in a rest position and signal to the operator that you have to remove the box. After the removal of the full box, you will have to place a new empty box, remove the safe drawers and reset the alarm from the operator panel.

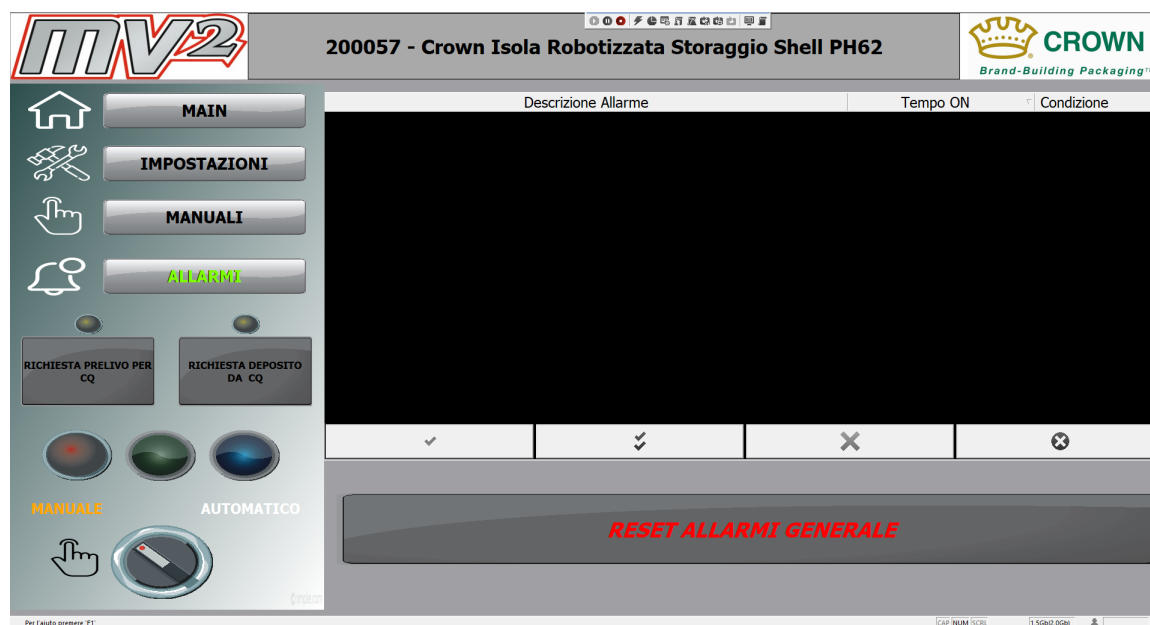


Figure 5.6: HMI - Alarm Page

Chapter 6

Conclusions

This project has tried with success to meet the customer's needs: "Obtain an automatic palletizing system for aluminium lids thanks to the use of a robotic manipulator" To this end, two racks containing boxes, an ad-hoc end effector for the application and a safety barrier for operators have been constructed from a mechanical point of view, From the electrical point of view, an electric panel has been created able to house an Omron PLC and Sick safety modules, from the point of view of programming, both the robot's PC and the external PLC have been programmed, sick modules and HMI operator interface panel.

The proposed solution, with the external PLC architecture, proved to be valid and robust, with a fairly fast working time and good general stability. This result is consistent with the expectation initially expressed in the elaboration, according to which the use of an external PLC would have created a separation between the robot and the rest of the plant, making it act a simple muscle, which performs the orders given by the external PLC. This use of the manipulator is optimal because the computation power of the robot PC is engaged only in the calculation of the trajectories to follow, making the system less overloaded with respect to the simulated robot PLC case.

However, it is important to keep in mind that this solution is more expensive in general and involves the use of many different devices, which must be programmed in different environments and also present incompatibilities with regard to the communication protocol, that have to be solved. Therefore, although this architecture is valid and meets all customer expectations, it would be possible in the future to integrate more devices with the same communication protocol to speed up programming operations. In addition, the use of an external PLC also increases the size of the final electrical panels, so it should be considered the possibility of using just the simulated robot PLC, but being very careful to prevent loop and stability issues.

Bibliography

- [1] John J Craig. *Introduction to Robotics: Mechanics and Control*. 2020.
- [2] Géza Husi. *Position Singularities and Ambiguities of the KUKA KR5 Robot*. 2015. URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.979.2398&rep=rep1&type=pdf>.
- [3] *IEC 61508*. 2011. URL: https://en.wikipedia.org/wiki/IEC_61508.
- [4] Bruno Siciliano Lorenzo Sciavicco Luigi Villani Giuseppe Oriolo. *Robotics, Modelling, Planning and Control*. 2009.
- [5] Majid Pakdel. *Advanced PLC Programming*. 2020.
- [6] *Premium HMI*. 2021. URL: <https://asem.it/it/prodotti/72/premium-hmi.html>.
- [7] Alessandro Rizzo. *Robotics course slides*. 2020.
- [8] Raffaella Sesana. *Applied Mechanics and Machine Design course slides*. 2020.
- [9] Angela Summers. *Evaluation of Uncertainty in Safety Integrity Level (SIL) Calculations*. 2016. URL: <https://sis-tech.com/wp-content/uploads/2016/11/Evaluation-of-Uncertainty-in-Safety-Integrity-Level.pdf>.
- [10] Yuhan Chen Xiao Luo Baoling Han Yan Jia Guanhao Liang Xinda Wang. “A General Approach Based on Newton’s Method and Cyclic Coordinate Descent Method for Solving the Inverse Kinematics”. In: *Applied Sciences* (2019).

Appendix A

Appendix Title



Figure A.1: Property of MVQuadro S.r.L

A.1 Sysmac - Structured Language Code

```
IF TO_ROBOT_DEPOSITA_DX=TRUE AND ULTIMA_PILA_DX=TRUE AND CASSETTO_APERTO=TRUE THEN
ROBOT_NUM_MISSIONE:=500;
ELSIF
TO_ROBOT_DEPOSITA_SX=TRUE AND ULTIMA_PILA_SX=TRUE AND CASSETTO_APERTO=TRUE
THEN
ROBOT_NUM_MISSIONE:=500;

END_IF;
IF TO_ROBOT_DEPOSITA_DX=FALSE AND TO_ROBOT_DEPOSITA_SX=FALSE...
THEN ROBOT_NUM_MISSIONE:=2000;
END_IF;
IF ROBOT_NUM_MISSIONE=2000 AND RICHIESTA_ACCESO=FALSE THEN
ROBOT_NUM_MISSIONE:=100;
END_IF;

IF ECO_NUMERO_MISSIONE=ROBOT_NUM_MISSIONE THEN
TO_ROBOT_START_MISSIONE:=TRUE ;
END_IF;
IF ECO_NUM_MISS_RUN>ROBOT_NUM_MISSIONE AND ...
ECO_NUM_MISS_RUN<ROBOT_NUM_MISSIONE +100
AND ROBOT_IN_ATTESA=FALSE
THEN TO_ROBOT_START_MISSIONE:=FALSE;
//ROBOT_NUM_MISSIONE:=2000;
END_IF;
IF ECO_NUM_MISS_RUN=ROBOT_NUM_MISSIONE+99 AND ROBOT_IN_ATTESA=FALSE THEN
//CASE CICLO
CASE (ROBOT_NUM_MISSIONE+99) OF
//MISSIONE HOME
199:MISSIONE_ACQUISITA:=TRUE;
IF TO_ROBOT_DEPOSITA_DX=TRUE THEN
TO_ROBOT_NUMERO_LATI_DEP:=1;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_DX;
END_IF;
```

```
IF TO_ROBOT_DEPOSITA_SX=TRUE THEN
TO_ROBOT_NUMERO_LATO_DEP:=2;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_SX;
END_IF;

IF RICHIESTA_ACCESSO=FALSE THEN
IF CASSETTO_APERTO=TRUE THEN
ROBOT_NUM_MISSIONE:=200;
MISSIONE_ACQUISITA:=FALSE;
ELSE
IF TO_ROBOT_DEPOSITA_DX=TRUE THEN ...
TO_ROBOT_NUMERO_LATO_DEP:=1; ROBOT_NUM_MISSIONE:=400;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_DX;
MISSIONE_ACQUISITA:=FALSE;
ELSIF TO_ROBOT_DEPOSITA_SX=TRUE THEN...
TO_ROBOT_NUMERO_LATO_DEP:=2;ROBOT_NUM_MISSIONE:=400;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_SX;
MISSIONE_ACQUISITA:=FALSE;
END_IF;
END_IF;
ELSE
ROBOT_NUM_MISSIONE:=2000;
MISSIONE_ACQUISITA:=FALSE;
END_IF;

//MISSIONE PREPARAZIONE PRELIEVO
299: MISSIONE_ACQUISITA:=TRUE;
IF RICHIESTA_ACCESSO=FALSE THEN
IF ((QUOTA_PILA_REAL<=(HMI_QUOTA_PILA_REAL+20)) AND...
(EV_PINZA_CHIUDI_AUTO=TRUE)) THEN
ROBOT_NUM_MISSIONE:=300;
MISSIONE_ACQUISITA:=FALSE;

END_IF;
IF PRENOTAZIONE_STOP_CICLO=TRUE OR RICHIESTA_ACCESSO=TRUE THEN
ROBOT_NUM_MISSIONE:=100;
END_IF;
ELSE
ROBOT_NUM_MISSIONE:=100;
MISSIONE_ACQUISITA:=FALSE;
END_IF;

//MISSIONE DEPOSITO
399:MISSIONE_ACQUISITA:=TRUE;
IF HMI_RICHIESTA_DEPOSITO_CQ=TRUE THEN
ROBOT_NUM_MISSIONE:=1000;
ELSE

IF TO_ROBOT_DEPOSITA_DX=1 THEN TO_ROBOT_NUMERO_LATO_DEP:=1;
ELSIF TO_ROBOT_DEPOSITA_SX=1 THEN TO_ROBOT_NUMERO_LATO_DEP:=2;
MISSIONE_ACQUISITA:=FALSE;
ELSE
ROBOT_NUM_MISSIONE:=100;
ROBOT_IN_ATTESA:=TRUE;
MISSIONE_ACQUISITA:=FALSE;
END_IF;
```

```
IF ROBOT_IN_ATTESA=FALSE AND TO_ROBOT_NUMERO_LATO_DEP=1 THEN
TO_ROBOT_NUMERO_FILA_PIANO_INT:=NUMERO_DEPOSITO_PILE_DX;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_DX;
ROBOT_NUM_MISSIONE:=600;
MISSIONE_ACQUISITA:=FALSE;
ELSIF
ROBOT_IN_ATTESA=FALSE AND TO_ROBOT_NUMERO_LATO_DEP=2 THEN
TO_ROBOT_NUMERO_FILA_PIANO_INT:=NUMERO_DEPOSITO_PILE_SX;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_SX;
ROBOT_NUM_MISSIONE:=600;
MISSIONE_ACQUISITA:=FALSE;
END_IF;

END_IF;

699: MISSIONE_ACQUISITA:=TRUE;
IF TO_ROBOT_DEPOSITA_DX AND NUMERO_DEPOSITO_PILE_DX>1 AND ...
NUMERO_DEPOSITO_PILE_DX<13 AND
... HMI_RICHIESTA_PRELIEVO_CQ=TRUE THEN
ROBOT_NUM_MISSIONE:=800;
ELSIF TO_ROBOT_DEPOSITA_SX AND NUMERO_DEPOSITO_PILE_SX>1 AND...
NUMERO_DEPOSITO_PILE_SX<13 AND...
HMI_RICHIESTA_PRELIEVO_CQ=TRUE THEN
ROBOT_NUM_MISSIONE:=800;
ELSE

IF ULTIMA_PILA_DX=TRUE OR ULTIMA_PILA_SX=TRUE THEN
ROBOT_NUM_MISSIONE:=500;
MISSIONE_ACQUISITA:=FALSE;
ELSE
ROBOT_NUM_MISSIONE:=200;
MISSIONE_ACQUISITA:=FALSE;
END_IF;
END_IF;

599: MISSIONE_ACQUISITA:=TRUE;
IF TO_ROBOT_DEPOSITA_DX=1 THEN TO_ROBOT_NUMERO_LATO_DEP:=1;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_DX;
MISSIONE_ACQUISITA:=FALSE;
ELSIF TO_ROBOT_DEPOSITA_SX=1 THEN TO_ROBOT_NUMERO_LATO_DEP:=2;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_SX;
MISSIONE_ACQUISITA:=FALSE;
END_IF;
ULTIMA_PILA_DX:=FALSE;
ULTIMA_PILA_SX:=FALSE;
ROBOT_NUM_MISSIONE:=400;

IF CASSETTIERA_PIENA_SX=TRUE AND CASSETTIERA_PIENA_DX=TRUE THEN
ROBOT_NUM_MISSIONE:=2000;
END_IF;

499:MISSIONE_ACQUISITA:=TRUE;

ULTIMA_PILA_DX:=FALSE;
ULTIMA_PILA_SX:=FALSE;

IF CASSETTO_APERTO=FALSE THEN
```

```
ERRORE_CASSETTO:=TRUE;
MISSIONE_ACQUISITA:=FALSE;
ELSE
ROBOT_NUM_MISSIONE:=200;
MISSIONE_ACQUISITA:=FALSE;
APERTURA_FALLITA:=FALSE;
END_IF;

899: HMI_RICHIESTA_PRELIEVO_CQ:=FALSE;
IF TO_ROBOT_DEPOSITA_DX=1 THEN TO_ROBOT_NUMERO_LATO_DEP:=1;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_DX;
MISSIONE_ACQUISITA:=FALSE;
ELSIF TO_ROBOT_DEPOSITA_SX=1 THEN TO_ROBOT_NUMERO_LATO_DEP:=2;
TO_ROBOT_NUMERO_PIANO_INT:=NUMERO_PIANO_DEPOSITO_SX;
MISSIONE_ACQUISITA:=FALSE;
END_IF;
ROBOT_NUM_MISSIONE:=600;

1099:HMI_RICHIESTA_DEPOSITO_CQ:=FALSE;
ROBOT_NUM_MISSIONE:=200;
END_CASE;
END_IF;
```



Figure A.2: Property of MVQuadro S.r.L

A.2 Kuka - Missions Code

A.2.1 Mission 100

```
DEF GEST_HOME()

;FOLD RIDUZIONE DELLA VELOCITA' SE IL ROBOT E' STATO MOSSO MANUALMENTE
PTP $POS_ACT
IF NOT(PRIMA_HOME_OK) THEN
$RED_VEL=20

;ENDFOLD

;FOLD SELEZIONE TOOL E BASE PER MOVIMENTI SAFE
BAS(#TOOL, 0)
BAS(#BASE, 0)
;ENDFOLD

;FOLD ESEGUO I MOVIMENTI SOLO SE NON SONO IN HOME

$ADVANCE = 3

;FOLD VERIFICO LA POSIZIONE DEL ROBOT IN ASSI
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP(VEL_MISS, ACC_MED, APO_PTP_NORMALE)

;VERIFICO LA POSIZIONE ATTUALE DEL ROBOT IN ASSI ED IN CARTESIANO
AX_ATTUALE=$AXIS_ACT
```

```

POS_ATTUALE = $POS_ACT

IF (OUT_WORKZONE_RASTR_SX) THEN
;FOLD IN WORKZONE RASTRELLIERA SX
IF ((POS_ATTUALE.Y > 1225) AND (POS_ATTUALE.Y < 1235)) THEN
;IL ROBOT SI TROVA IN UNA POSIZIONE DI DEPOSITO DISCHI
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Z = POS_ATTUALE.Z + 80
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS
ELSE
;IL ROBOT SI TROVA IN UNA POSIZIONE DI PRELIEVO CASSETTO
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Z = POS_ATTUALE.Z + 20
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS
ENDIF

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.X = 0
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Y = 1100
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS

; NUM_MISS_RUN = NUM_MISS_RUN + 1
; AX_APPOGGIO=$AXIS_ACT
; AX_APPOGGIO.A4 = -180
; AX_APPOGGIO.A5 = 40
; AX_APPOGGIO.A6 = 90
; VelAccApoPTP(VEL_MISS, ACC_MAX, APO_PTP_NORMALE)
; PTP AX_APPOGGIO C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP(VEL_MISS, ACC_MAX, APO_PTP_NORMALE)
PTP POS_SAFE_HIGH_SLIDE_SX C_DIS
;ENDFOLD
ELSE
IF (OUT_WORKZONE_PREL) THEN
;FOLD IN WORKZONE PRELIEVO
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Z = POS_ATTUALE.Z + 500
; SE IL VALORE DELLA Z E' SUPERIORE AL VALORE LIMITE DEFINITO
;(+ UNA QUOTA DI SICUREZZA) MODIFICO IL VALORE
IF (POS_ATTUALE.Z > 1350) THEN
POS_ATTUALE.Z = 1350
ENDIF

VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Y = 0
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS

```

```

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.X = POS_ATTUALE.X - 300
; SE IL VALORE DELLA X E' INFERIORE AL RAGGIO DELLA PINZA
;(+ UNA QUOTA DI SICUREZZA)MODIFICO IL VALORE
IF (POS_ATTUALE.X < 400) THEN
POS_ATTUALE.X = 400
ENDIF

VelAccApoLIN (1, 100, 0)
LIN POS_ATTUALE C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP(VEL_MISS, ACC_MAX, APO_PTP_NORMALE)
PTP POS_SAFE_PREL C_DIS
;ENDFOLD
ELSE
IF (OUT_WORKZONE_RASTR_DX) THEN
;FOLD IN WORKZONE RASTRELLIERA DX
IF ((POS_ATTUALE.Y > -1255) AND (POS_ATTUALE.Y < -1245)) THEN
;IL ROBOT SI TROVA IN UNA POSIZIONE DI DEPOSITO DISCHI
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Z = POS_ATTUALE.Z + 80
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS
ELSE
;IL ROBOT SI TROVA IN UNA POSIZIONE DI PRELIEVO CASSETTO
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Z = POS_ATTUALE.Z + 20
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS
ENDIF

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.X = 0
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ATTUALE.Y = -1100
VelAccApoLIN (0.5, 100, 0)
LIN POS_ATTUALE C_DIS

;NUM_MISS_RUN = NUM_MISS_RUN + 1
;AX_APPOGGIO=$AXIS_ACT
;AX_APPOGGIO.A4 = -180
;AX_APPOGGIO.A5 = 40
;AX_APPOGGIO.A6 = 90

;VelAccApoPTP(VEL_MISS, ACC_MAX, APO_PTP_NORMALE)
;PTP AX_APPOGGIO C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP(VEL_MISS, ACC_MAX, APO_PTP_NORMALE)
PTP POS_SAFE_HIGH_SLIDE_DX C_DIS

;ENDFOLD
ENDIF

```



```
ENDIF
ENDIF
;ENDFOLD

;FOLD MOVIMENTO IN HOME
$ADVANCE = 3
NUM_MISS_RUN = NUM_MISS_HOME_FINE
$RED_VEL=100
VelAccApoPTP(VEL_MISS, 100, APO_PTP_AMPPIO)

$H_POS=XHOME
PDAT_ACT=PDEFAULT
BAS (#PTP_DAT )
FDAT_ACT=FHOME
BAS (#FRAMES )
BAS (#VEL_PTP,100 )
PTP XHOME
;ENDFOLD

IF NOT(OUT_IN_HOME) THEN

$H_POS=XHOME
PDAT_ACT=PDEFAULT
BAS (#PTP_DAT )
FDAT_ACT=FHOME
BAS (#FRAMES )
BAS (#VEL_PTP,100 )
PTP XHOME
;ENDFOLD
ENDIF
ENDIF
;ENDFOLD
$ADVANCE = 3
NUM_MISS_RUN = NUM_MISS_HOME_FINE
$RED_VEL=100
WAIT SEC 0.5

END
```

A.2.2 Mission 200

```
DEF GEST_APPR_PREL()
;FOLD Missione per la gestione dell'approccio al prelievo
$ADVANCE=2
NUM_MISS_RUN = NUM_MISS_RUN + 1
; Apri pinza per sicurezza
APRI_PINZA()

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_PREL C_DIS

;Setto il TOOL e la BASE corretti
```

```

NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_PREL)

;Verifico eventuali correzioni da apportare al punto di prelievo dischi
POS_FINALE_DISCHI = POS_PREL_DISCHI
POS_FINALE_DISCHI.X = POS_PREL_DISCHI.X + CORR_X
POS_FINALE_DISCHI.Y = POS_PREL_DISCHI.Y + CORR_Y
POS_FINALE_DISCHI.Z = POS_PREL_DISCHI.Z + CORR_Z

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_PREL = POS_FINALE_DISCHI
POS_APPROCCIO_PREL.Z = POS_FINALE_DISCHI.Z + 150
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_APPROCCIO_PREL C_DIS
;WAIT FOR IN_FBK_GRIPPER_OPN

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoLIN (0.5, 100, 0)
LIN POS_FINALE_DISCHI

WAIT SEC 0.0
NUM_MISS_RUN = NUM_MISS_APPR_PREL_FINE

MISS_RUN = FALSE

;ENDFOLD
END

```

A.2.3 Mission 300

```

DEF GEST_PRELIEVO_DISCHI()
;FOLD Missione per la gestione del prelievo dei dischi dalla macchina
;PTP $POS_ACT

NUM_MISS_RUN = NUM_MISS_RUN + 1
;Apri pinza
;APRI_PINZA()
;WAIT FOR IN_FBK_GRIPPER_OPN

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_PREL)

;NUM_MISS_RUN = NUM_MISS_RUN + 1
;VelAccApoLIN (0.5, 100, 0)
;LIN POS_FINALE_DISCHI C_DIS

;Istruzioni commentate per tempistiche dovute al Laser
;WAIT SEC 0.0
NUM_MISS_RUN = NUM_MISS_RUN + 1
;Chiudi pinza per prelievo dischi
;CHIUDI_PINZA()
COM_PINZA_OPN = FALSE
COM_PINZA_CLS = TRUE

```

```
WAIT FOR IN_FBK_GRIPPER_CLS
COM_PINZA_CLS = FALSE
WAIT FOR IN_FBK_GRIPPER_CLS

$ADVANCE = 3

NUM_MISS_RUN = NUM_MISS_RUN + 1
; Calcolo per la posizione di allontanamento
POS_ALLONTANAMENTO_01 = POS_FINALE_DISCHI
POS_ALLONTANAMENTO_01.X = POS_FINALE_DISCHI.X + 60
VelAccApoLIN (2.5, 100, 5)

LIN POS_ALLONTANAMENTO_01 C_DIS

;NUM_MISS_RUN = NUM_MISS_RUN + 1
;; Calcolo per la posizione di allontanamento
;POS_ALLONTANAMENTO_02 = POS_FINALE_DISCHI
;POS_ALLONTANAMENTO_02.X = POS_FINALE_DISCHI.X + 20
;VelAccApoLIN (0.5, 100, 0)

;LIN POS_ALLONTANAMENTO_02 C_DIS

;NUM_MISS_RUN = NUM_MISS_RUN + 1
;; Calcolo per la posizione di allontanamento
;POS_ALLONTANAMENTO_03 = POS_FINALE_DISCHI
;POS_ALLONTANAMENTO_03.X = POS_FINALE_DISCHI.X + 40
;VelAccApoLIN (2.0, 100, 0)

;LIN POS_ALLONTANAMENTO_03 C_DIS

NUM_MISS_RUN = NUM_MISS_PREL_FINE
MISS_RUN = FALSE
POS_ALLONTANAMENTO_04 = POS_FINALE_DISCHI
POS_ALLONTANAMENTO_04.X = POS_FINALE_DISCHI.X + 100
POS_ALLONTANAMENTO_04.Z = POS_FINALE_DISCHI.Z + 200
VelAccApoLIN (3.0, 100, 30)

LIN POS_ALLONTANAMENTO_04 C_DIS

;Setto il TOOL e la BASE corretti
;NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_PREL C_DIS

NUM_MISS_RUN = NUM_MISS_PREL_FINE
$RED_VEL=100

;ENDFOLD
END
```

A.2.4 Mission 400

```
DEF GEST_DEPOSITO_DISCHI()
;FOLD Missione per la gestione del deposito dei dischi nella rastrelliera
IF (NUM_FILA_PIANO == 13) OR (NUM_FILA_PIANO == 1) THEN
$RED_VEL = 100
ELSE
$RED_VEL = 80
ENDIF

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1

SWITCH NUM_LATO_DEP
CASE 1
; Vai nella posizione safe Rastrelliera SX
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_RASTR_SX C_DIS

SWITCH NUM_PIANO
CASE 1,2,3,4,5
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_RASTR_SX C_DIS

CASE 6,7,8,9,10
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_RASTR_SX C_DIS

CASE 11,12,13,14,15
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_RASTR_SX C_DIS
ENDSWITCH

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_RASTR_SX)

CASE 2
; Vai nella posizione safe Rastrelliera DX
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_RASTR_DX C_DIS

SWITCH NUM_PIANO
CASE 1,2,3,4,5
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_RASTR_DX C_DIS

CASE 6,7,8,9,10
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_RASTR_DX C_DIS

CASE 11,12,13,14,15
```

```

VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_RASTR_DX C_DIS
ENDSWITCH

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_RASTR_DX)

ENDSWITCH

;Calcolo della quota di deposito e verifica eventuali correzioni da apportare
;al punto di deposito dischi
POS_FINALE_DEP = POS_DEP_DISCHI
POS_FINALE_DEP.X = POS_DEP_DISCHI.X + CORR_X + ((NUM_FILA_PIANO-1) * 90)
POS_FINALE_DEP.Y = POS_DEP_DISCHI.Y
POS_FINALE_DEP.Z = POS_DEP_DISCHI.Z + CORR_Z + ((NUM_PIANO-1) * 88.0)

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_01 = POS_FINALE_DEP
POS_APPROCCIO_01.X = 270
POS_APPROCCIO_01.Z = POS_FINALE_DEP.Z + 80
;VelAccApoLIN (0.5, 100, 0)
;LIN POS_APPROCCIO_01 C_DIS

IF NUM_PIANO <= 3 THEN
SPLINE
SLIN POS_APPROCCIO_01 WITH $ORI_TYPE=#CONSTANT, $VEL.CP = VEL_MISS_LIN
ENDSPLINE
ELSE
VelAccApoLIN (VEL_MISS_LIN, 100, 20)
LIN POS_APPROCCIO_01 C_DIS
ENDIF
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_DEP = POS_FINALE_DEP
POS_APPROCCIO_DEP.X = POS_FINALE_DEP.X
POS_APPROCCIO_DEP.Z = POS_FINALE_DEP.Z + 80
VelAccApoLIN (1.0, 100, 10)
LIN POS_APPROCCIO_DEP C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoLIN (0.5, 100, 0)
LIN POS_FINALE_DEP
WAIT SEC 0.0
;Comanda l'apertura della pinza
APRI_PINZA()
; WAIT FOR IN_FBK_GRIPPER_OPN
IF (NUM_LATO_DEP == 1) THEN
OUT_PIEZZO_DEP_SX_OK = TRUE
ELSE
OUT_PIEZZO_DEP_DX_OK = TRUE
ENDIF

$ADVANCE=3
NUM_MISS_RUN = NUM_MISS_DEPO_FINE
MISS_RUN = FALSE
;Allontanamento dal punto di Deposito

```

```

;NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ALLONTANAMENTO_DEP = POS_FINALE_DEP
;POS_ALLONTANAMENTO_DEP.Y = POS_FINALE_DEP.Y + 10
POS_ALLONTANAMENTO_DEP.Z = POS_FINALE_DEP.Z + 20

VelAccApoLIN (0.5, 100, 2)
LIN POS_ALLONTANAMENTO_DEP C_DIS

;NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ALLONTANAMENTO_01 = POS_FINALE_DEP
;POS_ALLONTANAMENTO_01.Y = POS_FINALE_DEP.Y + 10
POS_ALLONTANAMENTO_01.Z = POS_FINALE_DEP.Z + 80

VelAccApoLIN (1, 100, 10)
LIN POS_ALLONTANAMENTO_01 C_DIS

;NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ALLONTANAMENTO_02 = POS_FINALE_DEP
POS_ALLONTANAMENTO_02.X = 270
POS_ALLONTANAMENTO_02.Y = POS_FINALE_DEP.Y + 10
POS_ALLONTANAMENTO_02.Z = POS_FINALE_DEP.Z + 80

VelAccApoLIN (1, 100, 10)
LIN POS_ALLONTANAMENTO_02 C_DIS

;Setto il TOOL e la BASE corretti
;NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

IF (NUM_LATO_DEP == 1) THEN

IF (NUM_PIANO <= 5) THEN
;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_RASTR_SX C_DIS
ENDIF

IF ((NUM_PIANO > 5) AND (NUM_PIANO <= 10 )) THEN
;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_RASTR_SX C_DIS
ENDIF

IF (NUM_PIANO > 10) THEN
;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_RASTR_SX C_DIS
ENDIF

;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_RASTR_SX C_DIS
ENDIF

IF (NUM_LATO_DEP == 2) THEN

IF (NUM_PIANO <= 5) THEN

```

```
;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_RASTR_DX C_DIS
ENDIF

IF ((NUM_PIANO > 5) AND (NUM_PIANO <= 10 )) THEN
;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_RASTR_DX C_DIS
ENDIF

IF (NUM_PIANO > 10)THEN
;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_RASTR_DX C_DIS
ENDIF

;NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_RASTR_DX C_DIS
ENDIF
NUM_MISS_RUN = NUM_MISS_DEPO_FINE
$RED_VEL=100
WAIT SEC 0.0
;ENDFOLD
END
```

A.2.5 Mission 500

```
DEF GEST_OPEN_SLIDE()

;DICHIARAZIONE DELL'INTERRUPT DI ESTRAZIONE CASSETTO.
; MI FERMO SE SI ALZA LA VARIABILE CASSETTO IN POSIZIONE.
INTERRUPT DECL 20 WHEN SLIDE_IN_POS == TRUE DO SLIDE_IN_POS()

;FOLD Missione per la gestione dell'apertura del cassetto della rastrelliera
NUM_MISS_RUN = NUM_MISS_RUN + 1
; Chiudi pinza per prelievo dischi
CHIUDI_PINZA()
;WAIT FOR IN_FBK_GRIPPER_CLS

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1

SWITCH NUM_LATO_DEP
CASE 1
; Vai nella posizione safe Rastrelliera SX
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_SLIDE_SX C_DIS

SWITCH NUM_PIANO
CASE 1,2,3,4,5
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_SLIDE_SX C_DIS
```

```

CASE 6,7,8,9,10
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_SLIDE_SX C_DIS

CASE 11,12,13,14,15
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_SLIDE_SX C_DIS
ENDSWITCH

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_RASTR_SX)

CASE 2
; Vai nella posizione safe Rastrelliera DX
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_SLIDE_DX C_DIS

SWITCH NUM_PIANO
CASE 1,2,3,4,5
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_SLIDE_DX C_DIS

CASE 6,7,8,9,10
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_SLIDE_DX C_DIS

CASE 11,12,13,14,15
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_SLIDE_DX C_DIS
ENDSWITCH

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_RASTR_DX)

ENDSWITCH

;Calcolo della quota di prelievo cassetto e verifica eventuali correzioni da apportare
;al punto di prelievo cassetto
POS_FINALE_SLIDE = POS_OPEN_SLIDE
POS_FINALE_SLIDE.Z = POS_OPEN_SLIDE.Z + ((NUM_PIANO-1)*87.8) + CORR_Z

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_01 = POS_FINALE_SLIDE
POS_APPROCCIO_01.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_01.Y = POS_FINALE_SLIDE.Y + 635
POS_APPROCCIO_01.Z = POS_FINALE_SLIDE.Z + 150
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_MINIMO)
PTP POS_APPROCCIO_01 C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_07 = POS_FINALE_SLIDE

```



```
POS_APPROCCIO_07.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_07.Y = POS_FINALE_SLIDE.Y + 650
POS_APPROCCIO_07.Z = POS_FINALE_SLIDE.Z
VelAccApoLIN (1.0, 100, 0)
LIN POS_APPROCCIO_07 C_DIS
```

```
$RED_VEL=80
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_02 = POS_FINALE_SLIDE
POS_APPROCCIO_02.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_02.Y = POS_FINALE_SLIDE.Y
POS_APPROCCIO_02.Z = POS_FINALE_SLIDE.Z
```

```
VelAccApoLIN (1.0, 100, 0)
LIN POS_APPROCCIO_02 C_DIS
```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_03 = POS_FINALE_SLIDE
POS_APPROCCIO_03.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_03.Y = POS_FINALE_SLIDE.Y + 20
POS_APPROCCIO_03.Z = POS_FINALE_SLIDE.Z
```

```
VelAccApoLIN (1.0, 100, 0)
LIN POS_APPROCCIO_03 C_DIS
$RED_VEL=100
```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_04 = POS_FINALE_SLIDE
POS_APPROCCIO_04.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_04.Y = POS_FINALE_SLIDE.Y + 20
POS_APPROCCIO_04.Z = POS_FINALE_SLIDE.Z + 30
```

```
VelAccApoLIN (1.0, 100, 0)
LIN POS_APPROCCIO_04 C_DIS
```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_05 = POS_FINALE_SLIDE
POS_APPROCCIO_05.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_05.Y = POS_FINALE_SLIDE.Y - 20
POS_APPROCCIO_05.Z = POS_FINALE_SLIDE.Z + 30
```

```
VelAccApoLIN (1.0, 100, 0)
LIN POS_APPROCCIO_05 C_DIS
```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_06 = POS_FINALE_SLIDE
POS_APPROCCIO_06.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_06.Y = POS_FINALE_SLIDE.Y - 12
POS_APPROCCIO_06.Z = POS_FINALE_SLIDE.Z + 4
```

```
VelAccApoLIN (1.0, 100, 0)
LIN POS_APPROCCIO_06 C_DIS
```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoLIN (0.5, 100, 0)
LIN POS_FINALE_SLIDE C_DIS
```

```

;Apertura cassetto
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ALLONTANAMENTO_02 = POS_FINALE_SLIDE
POS_ALLONTANAMENTO_02.Y = POS_FINALE_SLIDE.Y + 530
POS_ALLONTANAMENTO_02.Z = POS_FINALE_SLIDE.Z - 3
VelAccApoLIN (0.5, 100, 0)
$RED_VEL=70
LIN POS_ALLONTANAMENTO_02 C_DIS

;Apertura cassetto
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ALLONTANAMENTO_01 = POS_FINALE_SLIDE
POS_ALLONTANAMENTO_01.Y = POS_FINALE_SLIDE.Y + 630
POS_ALLONTANAMENTO_01.Z = POS_FINALE_SLIDE.Z - 2

VelAccApoLIN (0.5, 100, 0)
$RED_VEL=30
OPEN_SLIDE ( )
$ADVANCE = 3
$RED_VEL=100

NUM_MISS_RUN = NUM_MISS_RUN + 1

LIN_REL {Y - 8}
LIN_REL {Z + 30}

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ALLONTANAMENTO_03 = POS_FINALE_SLIDE
POS_ALLONTANAMENTO_03.Y = POS_FINALE_SLIDE.Y + 650
POS_ALLONTANAMENTO_03.Z = POS_FINALE_SLIDE.Z + 150

VelAccApoLIN (1.0, 100, 0)
LIN POS_ALLONTANAMENTO_03 C_DIS

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

IF (NUM_LATO_DEP == 1) THEN

IF (NUM_PIANO <= 5) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_SLIDE_SX C_DIS
ENDIF

IF ((NUM_PIANO > 5) AND (NUM_PIANO <= 10 )) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_SLIDE_SX C_DIS
ENDIF

IF (NUM_PIANO > 10) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_SLIDE_SX C_DIS

```

```
ENDIF

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_SLIDE_SX C_DIS
ENDIF

IF (NUM_LATO_DEP == 2) THEN

IF (NUM_PIANO <= 5) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_LOW_SLIDE_DX C_DIS
ENDIF

IF ((NUM_PIANO > 5) AND (NUM_PIANO <= 10 )) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_MED_SLIDE_DX C_DIS
ENDIF

IF (NUM_PIANO > 10) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_HIGH_SLIDE_DX C_DIS
ENDIF

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_SLIDE_DX C_DIS
ENDIF
NUM_MISS_RUN = NUM_MISS_OPEN_SLIDE_FINE

$RED_VEL=100
WAIT SEC 0.0
;ENDFOLD
END
Mission 600
DEF GEST_CLOSE_SLIDE()
;FOLD Missione per la gestione della chiusura del cassetto della rastrelliera
NUM_MISS_RUN = NUM_MISS_RUN + 1
; Chiudi pinza per prelievo dischi
CHIUDI_PINZA()
;WAIT FOR IN_FBK_GRIPPER_CLS

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1

SWITCH NUM_LATO_DEP
CASE 1
; Vai nella posizione safe Rastrelliera SX
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_SLIDE_SX C_DIS
```

```

SWITCH NUM_PIANO
CASE 1,2,3,4,5
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_SLIDE_SX C_DIS

CASE 6,7,8,9,10
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_SLIDE_SX C_DIS

CASE 11,12,13,14,15
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_SLIDE_SX C_DIS
ENDSWITCH

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_RASTR_SX)

CASE 2
; Vai nella posizione safe Rastrelliera DX
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_SLIDE_DX C_DIS

SWITCH NUM_PIANO
CASE 1,2,3,4,5
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_SLIDE_DX C_DIS

CASE 6,7,8,9,10
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_SLIDE_DX C_DIS

CASE 11,12,13,14,15
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_SLIDE_DX C_DIS
ENDSWITCH

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, NUM_BASE_RASTR_DX)

ENDSWITCH

;Calcolo della quota di chiusura cassetto
;e verifica eventuali correzioni da apportare
;al punto di chiusura cassetto
POS_FINALE_SLIDE = POS_CLOSE_SLIDE
POS_FINALE_SLIDE.X = POS_CLOSE_SLIDE.X
POS_FINALE_SLIDE.Z = POS_CLOSE_SLIDE.Z + ((NUM_PIANO-1) * 87.9) + CORR_Z

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_01 = POS_FINALE_SLIDE
POS_APPROCCIO_01.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_01.Y = POS_FINALE_SLIDE.Y + 635

```

```
POS_APPROCCIO_01.Z = POS_FINALE_SLIDE.Z + 150
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_MINIMO)
PTP POS_APPROCCIO_01 C_DIS
```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_02 = POS_FINALE_SLIDE
POS_APPROCCIO_02.X = POS_FINALE_SLIDE.X
POS_APPROCCIO_02.Y = POS_FINALE_SLIDE.Y + 650
POS_APPROCCIO_02.Z = POS_FINALE_SLIDE.Z
VelAccApoLIN (1.0, 100, 0)
LIN POS_APPROCCIO_02 C_DIS
```

```
;Chiusura cassetto
NUM_MISS_RUN = NUM_MISS_RUN + 1
$RED_VEL=50
VelAccApoLIN (0.5, 100, 0)
LIN POS_FINALE_SLIDE C_DIS
$RED_VEL=100
```

```
; Allontanamento
WAIT SEC 0.5
NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_ALLONTANAMENTO_01 = POS_FINALE_SLIDE
POS_ALLONTANAMENTO_01.Y = POS_FINALE_SLIDE.Y + 200
```

```
VelAccApoLIN (1.0, 100, 0)
LIN POS_ALLONTANAMENTO_01 C_DIS
```

```
;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)
```

```
IF (NUM_LATO_DEP == 1) THEN
```

```
IF (NUM_PIANO <= 5) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_LOW_SLIDE_SX C_DIS
ENDIF
```

```
IF ((NUM_PIANO > 5) AND (NUM_PIANO <= 10 )) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_MED_SLIDE_SX C_DIS
ENDIF
```

```
IF (NUM_PIANO > 10) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_HIGH_SLIDE_SX C_DIS
ENDIF
```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_SLIDE_SX C_DIS
ENDIF
```

```

IF (NUM_LATO_DEP == 2) THEN

IF (NUM_PIANO <= 5) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMP10)
PTP POS_SAFE_LOW_SLIDE_DX C_DIS
ENDIF

IF ((NUM_PIANO > 5) AND (NUM_PIANO <= 10 )) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMP10)
PTP POS_SAFE_MED_SLIDE_DX C_DIS
ENDIF

IF (NUM_PIANO > 10) THEN
NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMP10)
PTP POS_SAFE_HIGH_SLIDE_DX C_DIS
ENDIF

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMP10)
PTP POS_SAFE_SLIDE_DX C_DIS
ENDIF

IF NUM_LATO_DEP==1 THEN
OUT_SLIDE_SX_CLS = TRUE
ELSE
OUT_SLIDE_DX_CLS = TRUE
ENDIF
NUM_MISS_RUN = NUM_MISS_CLOS_SLIDE_FINE

$RED_VEL=100
WAIT SEC 0.0

;ENDFOLD
END

```

A.2.6 Mission 700

```

DEF GEST_PREL_CQ()
;FOLD Missione per la gestione del prelievo dei dischi dalla stazione CQ

NUM_MISS_RUN = NUM_MISS_RUN + 1
; Apri pinza per sicurezza
APRI_PINZA()
;WAIT FOR IN_FBK_GRIPPER_OPN

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMP10)
PTP POS_SAFE_CQ C_DIS

;Setto il TOOL e la BASE corretti

```

```
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, 0)

;Verifico eventuali correzioni da apportare al punto di prelievo dischi
POS_FINALE_CQ = POS_PREL_CQ
POS_FINALE_CQ.X = POS_PREL_CQ.X + CORR_X
POS_FINALE_CQ.Y = POS_PREL_CQ.Y + CORR_Y
POS_FINALE_CQ.Z = POS_PREL_CQ.Z + CORR_Z

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_PREL = POS_FINALE_CQ
POS_APPROCCIO_PREL.Z = POS_FINALE_CQ.Z + 200
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_APPROCCIO_PREL C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoLIN (0.5, 100, 0)
LIN POS_FINALE_CQ C_DIS

WAIT SEC 0.0
NUM_MISS_RUN = NUM_MISS_RUN + 1
; Chiudi pinza per prelievo dischi
CHIUDI_PINZA()
;WAIT FOR IN_FBK_GRIPPER_CLS

NUM_MISS_RUN = NUM_MISS_RUN + 1
; Calcolo per la posizione di allontanamento

POS_ALLONTANAMENTO_01 = POS_FINALE_CQ
POS_ALLONTANAMENTO_01.Z = POS_PREL_CQ.Z + 200
VelAccApoLIN (0.5, 100, 0)
LIN POS_ALLONTANAMENTO_01 C_DIS

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPIO)
PTP POS_SAFE_CQ C_DIS

NUM_MISS_RUN = NUM_MISS_PREL_CQ_FINE

$RED_VEL=100
WAIT SEC 0.5
;ENDFOLD
END
```

A.2.7 Mission 800

```
DEF GEST_DEP_CQ()
;FOLD Missione per la gestione del deposito nella stazione CQ
```

```
;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_CQ C_DIS

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1

BAS(#TOOL, NUM_TOOL_LAVORO)
BAS(#BASE, 0)

;Verifico eventuali correzioni da apportare al punto di prelievo dischi
POS_FINALE_CQ = POS_DEP_CQ
POS_FINALE_CQ.X = POS_DEP_CQ.X + CORR_X
POS_FINALE_CQ.Y = POS_DEP_CQ.Y + CORR_Y
POS_FINALE_CQ.Z = POS_DEP_CQ.Z + CORR_Z

NUM_MISS_RUN = NUM_MISS_RUN + 1
POS_APPROCCIO_PREL = POS_FINALE_CQ
POS_APPROCCIO_PREL.Z = POS_FINALE_CQ.Z + 200
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_APPROCCIO_PREL C_DIS

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoLIN (0.5, 100, 0)
LIN POS_FINALE_CQ C_DIS

WAIT SEC 0.0
NUM_MISS_RUN = NUM_MISS_RUN + 1
; Apri pinza per deposito dischi
APRI_PINZA()
;WAIT FOR IN_FBK_GRIPPER_OPN

NUM_MISS_RUN = NUM_MISS_RUN + 1
; Calcolo per la posizione di allontanamento

POS_ALLONTANAMENTO_01 = POS_FINALE_CQ
POS_ALLONTANAMENTO_01.Z = POS_PREL_CQ.Z + 200
VelAccApoLIN (0.5, 100, 0)
LIN POS_ALLONTANAMENTO_01 C_DIS

;Setto il TOOL e la BASE corretti
NUM_MISS_RUN = NUM_MISS_RUN + 1
BAS(#TOOL, 0)
BAS(#BASE, 0)

NUM_MISS_RUN = NUM_MISS_RUN + 1
VelAccApoPTP (VEL_MISS, ACC_MAX_MISS, APO_PTP_AMPPIO)
PTP POS_SAFE_CQ C_DIS

NUM_MISS_RUN = NUM_MISS_DEP_CQ_FINE
```



```
$RED_VEL=100  
WAIT SEC 0.5  
;ENDFOLD  
END
```

A.2.8 Mission 1000

```
DEF GEST_MANUTENZIONE()  
;FOLD Missione per la gestione del movimento del robot  
; e portarlo in posizione di manutenzione  
  
$RED_VEL=100  
WAIT SEC 0.5  
;ENDFOLD  
END
```