Universidade do Minho
Escola de Engenharia

Alberto José Cibrão Monteiro

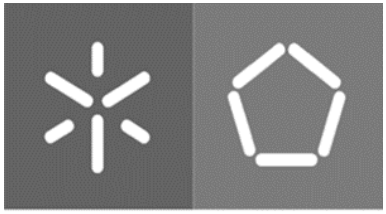
Modeling and Simulation of a *Pick&Place* System with *OpenModelica*

Junho de 2023

Modeling and Simulation of a *Pick&Place System with OpenModelica*

Alberto Monteiro

UMINHO|2023



Universidade do Minho
Escola de Engenharia

Alberto José Cibrão Monteiro

Modeling and Simulation of a *Pick&Place* System with *OpenModelica*

Master's Dissertation
Integrated Master in Mechanical Engineering

Project developed under the orientation of
Professor PhD José Mendes Machado

Coorientation of
Professor MSc Filipe Alexandre de Sousa Pereira

June 2023

ACKNOWLEDGMENTS

A deep sense of gratitude to my family, for the solid support in the different areas that the academic path implies and that culminated in this project, for all the effort and unconditional support they provided throughout my journey. A deep thank you to my parents and my sister, for whom none of this was possible, through the opportunities they provided me, the values they transmitted to me, and the sacrifices they made so that I could reach my goals.

A word of thanks to my friends and colleagues who have been part of this journey, for their patience, mutual help, and personal development, which proved to be indispensable in the resilience needed to conclude this project.

Special thanks to Professor José Machado and Professor Filipe Pereira for their guidance, with all the dedication, patience, effort, advice, and knowledge they gave me. I will be eternally grateful for the understanding and motivation that was transmitted to me.

Integrity Declaration

I declare that I have acted with integrity in the preparation of this academic work and confirm that I have not resorted to the practice of plagiarism or any form of misuse or falsification of information or results in any of the steps leading to its preparation.

I further declare that I am aware of and have respected the Code of Ethical Conduct of the University of Minho.

ABSTRACT

Modeling of multi-physical systems is a task that may be particularly complex, especially when the systems under analysis present several technologies such as pneumatics, hydraulics, mechanical systems, electrical systems, among others.

This MSc dissertation intends to present an approach for modelling a multi-physics system, using *Modelica* modeling language, focusing the connection of models, by using programming logic as ever as needed, as well as the obtaining of some guidelines for modelling multi-physical systems in a systematic way.

For the mentioned purpose, it is chosen, as basis for starting studies, a *Pick&Place* system using the modeling language *Modelica* and the simulation software *OpenModelica* whose conception was based on a manipulator's model existing in the main library of *OpenModelica* software.

Based in the initial case study, some approaches are presented for illustration about how to achieve the above-mentioned purpose, by changing and adding information, from initial considered models, to the considered final ones using simulation techniques and interpreting respective obtained results.

Throughout this work, the potentialities presented by the language are explored, bringing together the concept of a robotic manipulator inserted in *Modelica*'s main library with the outline of a detailed control, in which the concepts of inverse kinematics and *pick&place* are inserted. The incorporation with a global system is also scrutinized, in which the manipulator model is associated with a gripper and conveyor belts, designed in a system with logical sequencing.

KEYWORDS

Inverse Kinematics, *Modelica*, Modeling Multi-physics Systems, *OpenModelica*, *Pick&Place*

RESUMO

A modelação de sistemas multi-físicos é uma tarefa que pode ser particularmente complexa, especialmente quando os sistemas em análise apresentam diversas tecnologias, tais como pneumática, hidráulica, sistemas mecânicos, sistemas elétricos, entre outros.

Esta dissertação de mestrado pretende apresentar uma abordagem para a modelação de um sistema multi-físico, utilizando a linguagem de modelação *Modelica*, focando na ligação de modelos e recorrendo à lógica de programação sempre que necessário, bem como a obtenção de algumas orientações para a modelação de sistemas multi-físicos de uma forma sistemática.

Para o efeito, é escolhido, como base de estudo inicial, um sistema *Pick&Place* utilizando a linguagem de modelação *Modelica* e o software de simulação *OpenModelica*, cuja conceção foi baseada num modelo de manipulador existente na biblioteca principal do software *OpenModelica*.

Com base no caso de estudo inicial, são apresentadas algumas abordagens para ilustrar a forma de atingir o objetivo acima referido, com alteração e adição de informação, desde os modelos iniciais considerados, até aos modelos finais desenvolvidos, utilizando técnicas de simulação e interpretando os respetivos resultados obtidos.

Ao longo deste trabalho, as potencialidades apresentadas pela linguagem são exploradas, reunindo o conceito de um manipulador robótico inserido na biblioteca principal de *Modelica* com o delineamento de um controlo detalhado, em que os conceitos de cinemática inversa e *pick&place* se inserem. A incorporação com um sistema global também é escrutinada, no qual o modelo do manipulador se encontra associado com um *gripper* e tapetes transportadores, projetados num sistema com sequenciamento lógico.

PALAVRAS-CHAVE

Cinemática Inversa, *Modelica*, Modelação de Sistemas Multifísicos, *OpenModelica*, *Pick&Place*

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In this chapter, it is intended to present the area of interest that drove the development of the present work: the simulation of industrial processes using object-oriented modeling. In this sense, a small introduction, with the exposition of fundamental concepts, will be carried out, followed by the presentation and framework of the developed system.

1.1.Simulation in Industry

The importance of simulation in an industrial environment, which came later to gain relevance in cyber-physical engineering systems, was already highlighted by the first pioneers in simulation research studies since the 1980s as a top methodology. To calculate the amount of work done, some figures are provided in [1]: from 2002 to 2013, the search process with the keywords "simulation" plus "manufacturing" shows about 3000 published articles. These results are expressive and are confirmed by another survey conducted in the same year in 2014 [1].

The success of simulation is strictly related to the advent of computers [2], but the concept of simulation exists independently. A simple but meaningful definition is provided by [3]: "Simulation is a way to evaluate a proposed system for various parameters within a specific time period." Another full definition, and perhaps the most notable, is: "Simulation is the imitation of the operation of a real-world process or system over time" [4].

With particular focus on object-oriented modeling in the application of simulation in industry, this occurs at the beginning of the Software Development Life Cycle [5], allowing an improvement of the design phase of a system when an object-oriented approach is taken in software development.

Simulation applications in industry such as designing a facility or designing a material handling system [1], can be developed using object-oriented modeling.

A model can be defined as follows [6]: "A model of a system is anything to which an experiment can be applied in order to give answers to questions about that same system." This definition implies that a model is able to give answers to questions without requiring experiments with the actual, real system itself. In this way, experiments conducted on the model, which turns out to be a simplified version of the real system, are able to reflect properties that may be useful and decisive for a distinct understanding of that same system. The type of models that are used in the *OpenModelica* software are of the mathematical type.

Mathematical models built on a computer are often called virtual models or virtual prototypes. For the construction of the models, the use of the *Modelica* language allows the simplification of their construction, as well as the reuse of pre-existing models. Its application implies the following attributes by the user.

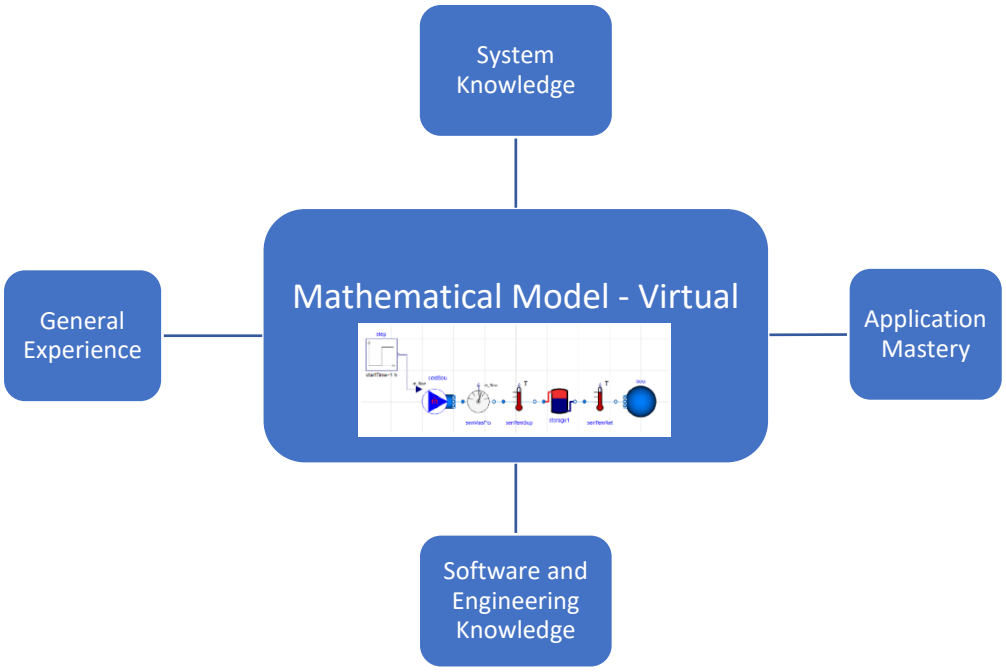


Figure 1 - Usage requirements for virtual modeling and simulation [6]

- **General Experience -** collected knowledge in the science and technology domains found in the literature and made available by different experts in the associated fields.
- **System Knowledge -** observations and familiarity with experiments already conducted on the system.
- **Application Mastery -** mastery of the application area and techniques, enabling one to best exercise the capabilities provided by the application in system modeling.
- **Software and Engineering Knowledge -** generic knowledge about the definition, treatment and representation of models and software.

Thus, the appropriate design to be taken into consideration when establishing the model building process, in order to meet the channeling and synthesizing of the information acquired, in the different forms mentioned above, is represented in the following schematization:

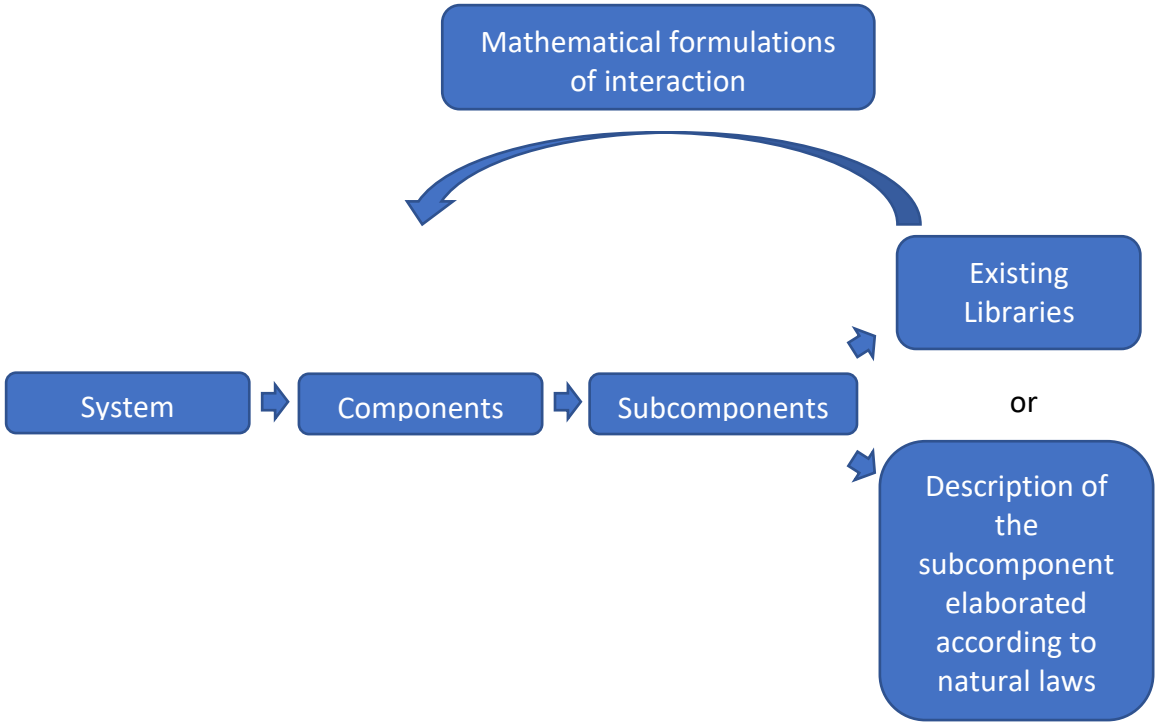


Figure 2 - Schematic of the model building process

1.2.*Modelica* modeling language and *OpenModelica* software

The use of the open-source tool *OpenModelica* using the *Modelica* language offers an excellent solution for the simulation and optimization of complex engineering systems. *OpenModelica* is a modeling tool. With this computational tool its users can create individual components or subsystem models. These models can then be used to predict the behavior of the associated physical system. In addition to being a tool that enables system modeling, it is also a platform capable of performing simulations.

Included in the *OpenModelica* simulation platform, the program features, after processing the different models, the ability to plot, analyze and "animate" the physical process adjacent to the model. This modeling and simulation capability allows users of the computational tool to explore the design environment and confirm its expected performance without the need to build expensive physical prototypes. *OpenModelica* also supports collaboration with other tools through interoperability adjacent to a set of standards [7].

The language that serves as a reference for any type of work to be developed with *OpenModelica*, is the *Modelica* modeling language. *Modelica* is a high-level declarative language that allows to describe mathematical behaviors [8]. It is usually applied in engineering systems and can be used to easily describe the behavior of different types of components in specific areas. These components in turn can be combined into systems,

subsystems and different architectures that can result. Regarding the aspects that set this language apart from others, the ability to describe, in a joint manner, continuous and discrete behaviors in a system, the fact that it supports casual and accidental approximations within the same model, and finally the fact that it is an "open" language whose specification is easily acquired, are highlighted.

1.3. Background and Motivation

Pick&Place systems are widely used in industrial manufacturing environments [9]. In fact, such systems are connected with other systems, are based on different technologies and serve as very important sub-systems in common production lines.

In the present context of digitalization of equipment, processes and factories, it is important to find approaches and tools for the correct modeling, data collecting and processes' performance improvement by using, for instance, digital twins [10] and/or integration of cyber-physical systems [11] in a digital simulation environment.

One of the approaches that can be used for this purpose is, definitively, *Modelica* [12] language well adapted for modeling multy-physics systems and processes.

Modelica modeling language can be used in different contexts and can be implemented in several softwares specially developed for implementing it. For instance, as professional approach, DYMOLA [13]; or, as free source approach, *OpenModelica* [14].

For the above-mentioned context, this work is focused in the study of a *Pick&Place* operating system, based on a detailed study of an already existing model in the *Modelica Standard Library*, using *OpenModelica* software. This model [15] is the starting point of the project, since part of its structure serves as a basis for the elaboration of the manipulator model and from which a restructuring and integration with a global system is carried out. In Figure 3 it is possible to see the underlying idea, in which can be observed a remodeling of the system for the robot/manipulator (diagram 2 in Figure 3), with the introduction of new principles in the control domain as well as in the modeling of the physical process of the manipulator itself. A model of the manipulator is thus prepared from scratch, with the purpose of not only integrating some pre-developed modeling principles but also articulating them with new models through programmed logic, and thus creating a more dynamic and intuitive system, with obtaining of greater simplicity in interaction with the user, and still serving the complex

purpose of studying a *Pick&Place* process, which finds its common applicability in industrial processes.

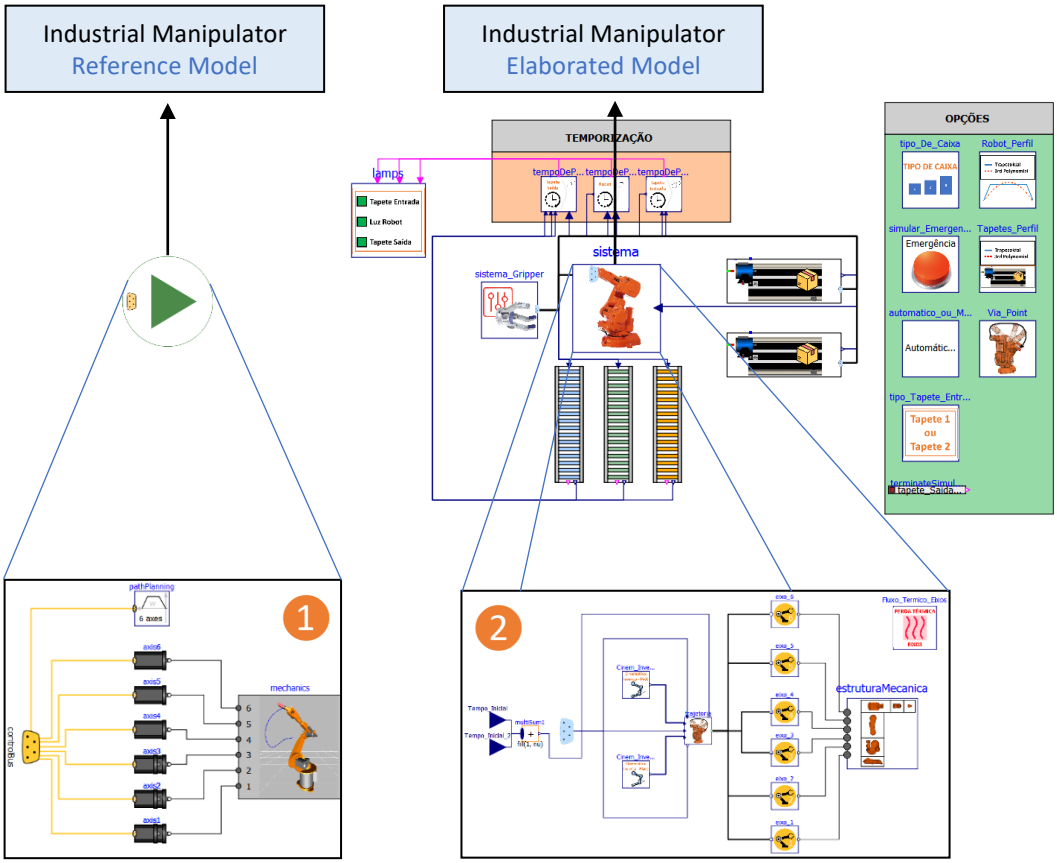


Figure 3 - Reference model of the manipulator on the left. Elaborated model of the manipulator on the right, embedded in a global *Pick&Place* system

1.4.Objectives

The main and global objective of this work is to provide a simple approach for modelling a multi-physics system, using *Modelica* modeling language, focusing the connection of models by using programming logic as ever as needed.

For this purpose, the basis for this study starts with studying a simple manipulator model (robotic application) and, step by step, this model is enlarged and being more complex, integrating other simple models and connecting itself with other models' too.

As specific goals, the study and model development of the industrial manipulator, besides a detailed study involved in the reference model already developed in *Modelica Standard Library*, allows:

- Introduction of the concept of Inverse Kinematics, improving the logistics of operation and simulation of the manipulator, for specific tasks (among which, that of *Pick&Place*).

- Studying and understanding of complementarity presented in the study of the motion profile, designing not only a trapezoidal velocity profile, but also a polynomial (3rd and 5th order) velocity profile.

- Designing a fully designed *Path Planning* for a *Pick&Place* task, with all the restructurings and improvements that it implies.

- Developing a mode with *Via Points*, trying to portray as much as possible a pick-up and delivery system observed in the industrial world.

- Studying of the thermal losses in the axes verified along the manipulator's operation.

- Hierarchical restructuring of the controller, making the cascade control system more perceptible and thus facilitating possible improvements.

On the other hand, the development of a Global System in which the robot model is inserted, allows:

- Conceptualization and development of a Gripper to be coupled to the manipulator, allowing a more realistic projection of the operation of an industrial robot.

- Conceptualization and development of conveyor belts, with an integrated cooling system.

- Modeling with sequence of operation Conveyor (input) → Robot/Gripper → Conveyor (output) using programmed logic. Operation modes are further developed with distinct definition of functional parameters.

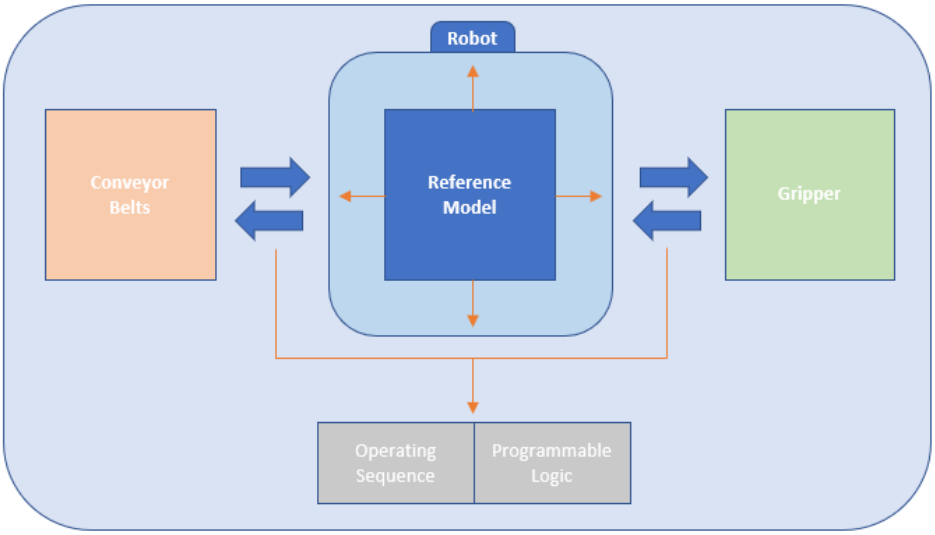


Figure 4 - Operating sequence established through the exchange of information between the models of the conveyor belts, the manipulator, and also the gripper

1.5.Dissertation Structure

With a brief state of the art elaboration, in which a short presentation of the most generic concepts was presented, chapter 2 will discuss the modeling development for each one of the subsystems of the global system. The sub-chapters for each subsystem admit a theoretical component, an implementation component in object-oriented modeling, and finally a graphic study of the simulations is conducted.

The first subchapter is dedicated to the industrial manipulator, focusing initially on the control command models, followed by the mechanical structure. In a second subchapter, we have the analysis of the development of each of the sub-models present in the different conveyor belts attached to the system, in which besides the structural and control aspects, the main differences in the elaboration of each of the sub-models are distinguished. Finally, in the last subchapter, the model of the gripper to be coupled to the manipulator is presented, showing not only the mechanical structure of the process, but also its own control circuit.

Chapter 3 concerning the visual results of the simulation is written, where the interaction between the different subsystems is observable, succeeded by a final chapter where conclusions and possible improvements to be implemented are drawn.

2. SYSTEM DEVELOPMENT IN *OPENMODELICA*

After simulating the physical environment of the system in *CoppeliaSim* (described, in detail, in a separate user's manual – appendix 4), allowing for a greater assimilation of the general concepts, inherent to the system from which it's highlighted an appropriate understanding of the kinematics involved in the movement of the arm as well as studying a simple way to control the arm that can be implemented (appendix 2), the object-oriented modeling of the overall system is then performed, and in this, certain functions are explored more clearly, of which it's highlighted:

- The modeling of the robotic arm, with a built-in inverse kinematics model, thus allowing the movement of the arm along the *Pick* and *Place* task, through the cartesian coordinates inserted at the end of the arm.
- The modeling of conveyors, responsible for transporting the package along the system. The system is composed of an input conveyor and an output conveyor, and the respective input conveyor has a modeling established according to the user's options. For the input conveyor, there are two possible conveyors with different modeling. One presents a more simplified modeling profile, in contrast to the adjacent conveyor belt, whose modeling of the physical system is slightly more complex, involving a larger number of variables under study. This distinction makes it possible to compare and evaluate different performances for different modeling of the same subsystem.

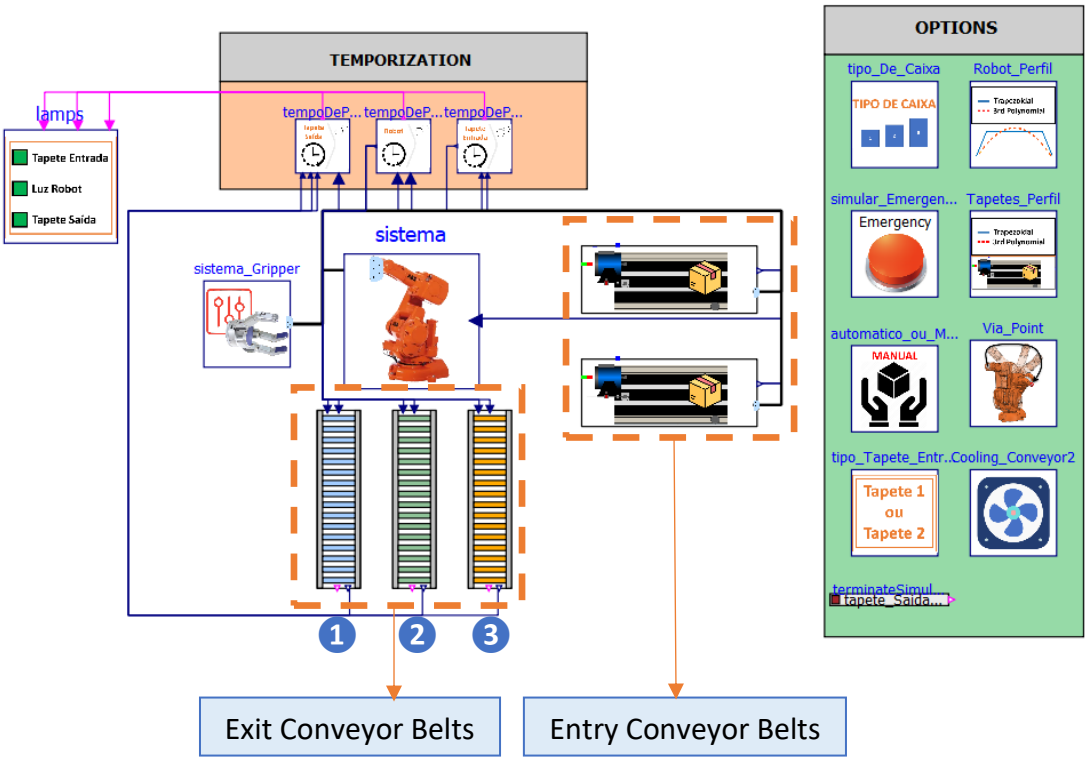


Figure 5 - Global System Model

As for the exit conveyors, these are differentiated by the type of package/box they transport, and in the case of the last conveyor (exit conveyor 3) a differentiated box transport route is presented with 2 successive stops, for different operation cells, respectively.

2.1.Robot

The design of the manipulator control, which includes the calculation of the inverse kinematics as well as the trajectory profiles, are studied in an initial phase, accompanied by its implementation in the system. The control of the different axes, by feedback cycles, is studied and implemented, but due to the similarity presented with the reference model (differing mainly from the structural point of view), it can be consulted in the appendix 3, as well as the simplifications implied in the object-oriented modeling of the control technology. Finally, the modeling of the mechanical structure is presented.

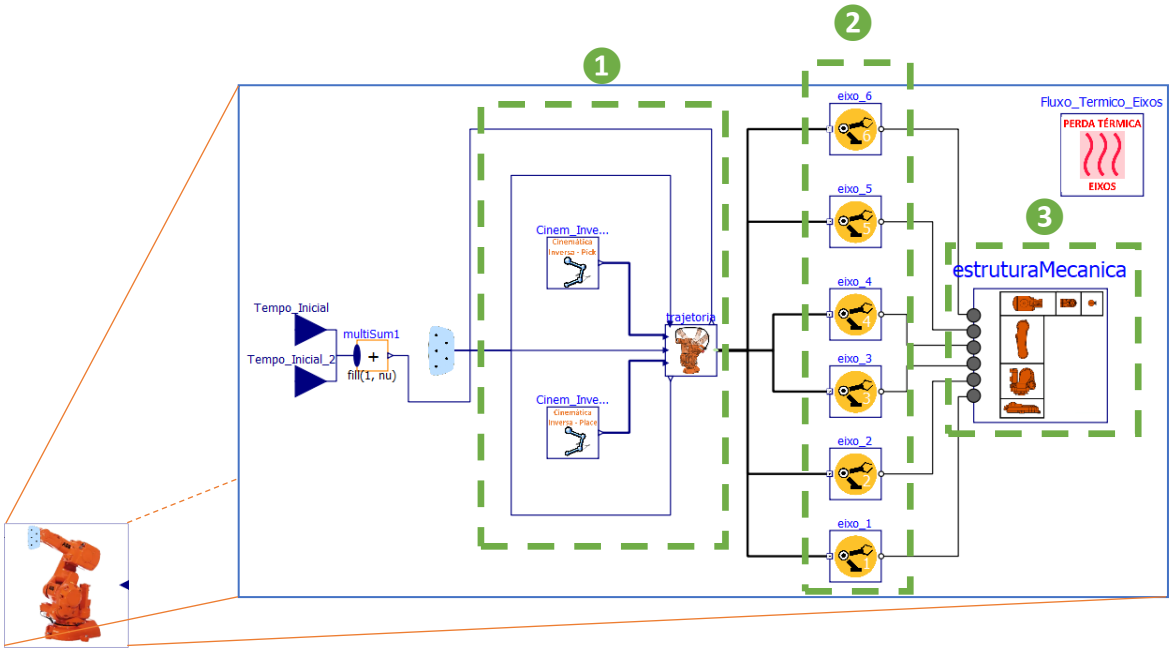


Figure 6 - Model of the manipulator consisting of different sections

The elaborated top hierarchy model, in relation to the robotic arm, can be divided into different steps:

- **1 -** A step destined to delineate the path to be taken by the arm, which includes the calculation of the inverse kinematics resulting in the angular values of the joints to be reached along time and, furthermore, through these angular values develop the position and velocity profiles along the different axes.
- **2 -** A motion control and transmission step. Where the previously calculated profiles will serve as a set of reference values in controllers. Servomotors are the respective actuators associated with the controllers, acting as the plant of the control system, where a transmission system is further coupled.
- **3 -** A stage intended exclusively for modeling the mechanical structure of the arm, where the dimensional reference structure will be that of *ABB IRB 140* industrial manipulator.

2.1.1. Inverse Kinematics

The first step for the movement of the manipulator along the operation to occur as desired is a correct calculation of the involved values that propitiate its kinematic movement, namely an accuracy in the presentation of the joint values of each axis (initial joint values and final joint values). For this purpose, an inverse kinematics model is developed, aiming to calculate

the variation of the angular values of the joints, so that the path implied in the operation occurs.

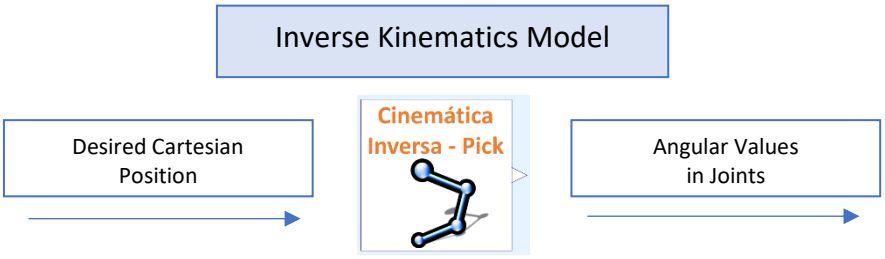


Figure 7 - Goal of the Inverse Kinematics Model

The elaboration of this model is done, in its entirety, using the *text view editor*, that is, the calculation of the angular variation of the joints is entirely programmed in code. This is done using the inverse kinematics equations already calculated in *Matlab* (user's manual – appendix 4).

Comparing scripts made in *Matlab* and with *Modelica* concerning inverse kinematics, can be concluded that in *Modelica* a "fragmentation" of the code is necessary, making its realization sometimes not very fluid. The division between parameters/inputs and the algorithm itself is necessary in *Modelica*, unlike *Matlab*, where everything can be done sequentially, making the algorithm follow-up more linear and intuitive.

This model allows simplifying the system right away, since it is easier for the user to know the final cartesian position of the arm tip, in order to perform the *Pick*, than to know the variation of the angular value of all joints so that the movement occurs as desired (as suggested in the reference model) and the final configuration is reached.

Since for one operation cycle, the implied transport must ensure picking up the package on an input conveyor and dropping the package on another output conveyor, at least two movements must be ensured. In this sense, two inverse kinematics models are used, one for *Pick* and another for *Place*, where the only differentiating elements between these models are the cartesian coordinates defined (initial and final).

Throughout this section it will be presented the theoretical component [16] aimed at calculating the inverse kinematics, which will be accompanied simultaneously by the implementation of the inverse kinematics model in *OpenModelica*.

2.1.1.1. General Theory

Assigning Frames of Reference

This assignment of frames of reference will serve as a starting point for the determination of the *Denavit - Hartenberg* parameters [17], which is based on the dimensions presented by *the IRB - 140* industrial manipulator [18]. The different references along the structure are defined by assigning a set of coordinate axes at each link of the structure.

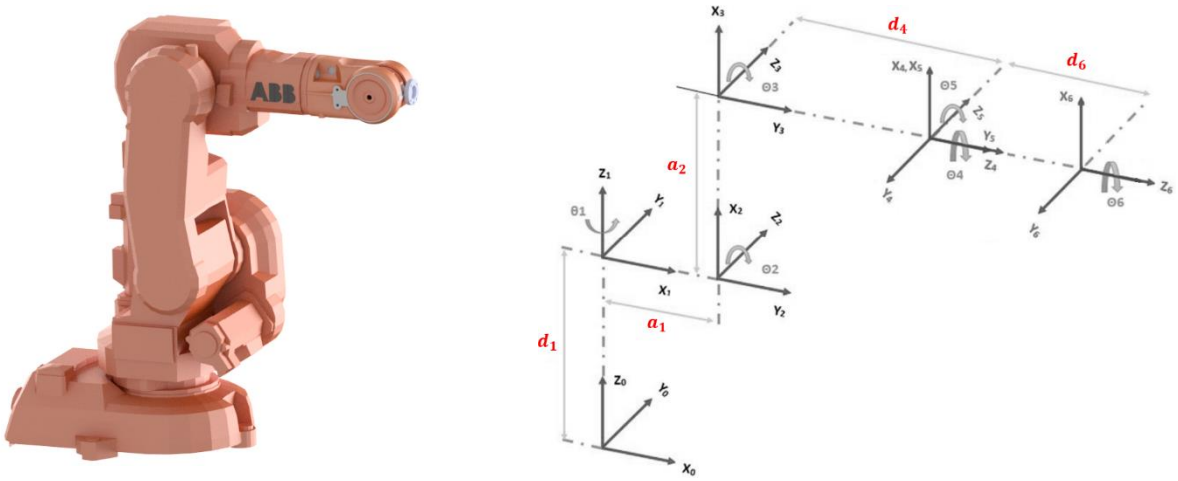


Figure 8 - IRB 140 manipulator and the frames of reference distributed between the links

Each of the dimensions shown in Figure 8 dictates the following values (Table 1).

Table 1 - Dimensions involved between the different frames of reference

	(m)
d1	0.352
a1	0.07
a2	0.36
d4	0.38
d6	0.065

These dimensions have the relevance of being used to define the *Denavit-Hartenberg* parameters, determining the position of the different reference points. However, if the focus is on all the dimensions along the structure, which will be useful in the development of the inverse kinematics, they can be described according to the vector {0.352; 0.07; 0.177; 0.36; 0.177; 0.38; 0.065}.

Denavit – Hartenberg parameters

Based on Figure 8 it is possible to determine the *Denavit-Hartenberg* parameters, which are present in table 2. For each frame of reference there are 4 parameters, which originate a transformation matrix [19] [20]. This establishes the relationship between the previous reference (i-1) and the following one (i). The parameters a_i and α_i refer to the displacement and rotation on the x axis, respectively, considering the following reference. The parameters d_i and θ_i refer to the displacement and rotation in the z axis, respectively, also taking the next reference into account.

Table 2 - *Denavit-Hartenberg* parameters relating to the schematization of the distributed coordinate axes in the IRB 140 structure

${}^{i-1}T_i$	a_i	α_i	d_i	θ_i
${}^{0}T_1$	0	0	-90	d1
${}^{1}T_2$	90	a1	90	0
${}^{2}T_3$	0	a2	0	0
${}^{3}T_4$	90	0	0	d4
${}^{4}T_5$	-90	0	0	0
${}^{5}T_6$	90	0	0	d6

Individual transformation matrices

Once the *Denavit - Hartenberg* parameters are obtained, one can resort to the matrix shown (Figure 9) in order to obtain each of the individual transformation matrices.

$$ {}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \cos(\alpha_{i-1}).\sin(\theta_i) & \cos(\alpha_{i-1}).\cos(\theta_i) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1}).d_i \\ \sin(\alpha_{i-1}).\sin(\theta_i) & \sin(\alpha_{i-1}).\cos(\theta_i) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1}).d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Figure 9 - Individual Transformation Matrix

On the other hand, the global transformation matrix of the manipulator is obtained by the product between the individual transformation matrices shown above, according to equation (1).

$$ {}^{0}T_{6} = {}^{0}T_{1} \times {}^{1}T_{2} \times {}^{2}T_{3} \times {}^{3}T_{4} \times {}^{4}T_{5} \times {}^{5}T_{6} \quad (1) $$

The global transformation matrix displays in its fourth column the position of the end effector relative to the base (P_{06}).

$$ {}^{0}T_{6} = \begin{bmatrix} & & & P_{06} \\ & & & {}^{0}x_{6} \\ & & & {}^{0}y_{6} \\ & & & {}^{0}z_{6} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Figure 10 - Position of the end effector relative to the base, taken from the global matrix

From the global matrix it is possible to determine not only the position of the end-effector but also its orientation relative to the base. This orientation is defined by the first three rows and first three columns present in the global matrix, thus defining the matrix RPY which is equivalent to the rotation matrix R_{06}, that defines the end-effector orientation relative to the base.

$$ RPY = (r_{ij})_{i,j=1,2,3} = \mathrm{Rot}(z,\phi_t)\,\mathrm{Rot}(y,\psi_t)\,\mathrm{Rot}(x,\theta_t) $$

$$ = \begin{bmatrix} c\phi_t\, c\psi_t & -s\phi_t\, c\theta_t + c\phi_t\, s\psi_t\, s\theta_t & s\phi_t\, s\theta_t + c\phi_t\, s\psi_t\, c\theta_t \\ s\phi_t\, c\psi_t & c\phi_t\, c\theta_t + s\phi_t\, s\psi_t\, s\theta_t & -c\phi_t\, s\theta_t + s\phi_t\, s\psi_t\, c\theta_t \\ -s\psi_t & c\psi_t\, s\theta_t & c\psi_t\, c\theta_t \end{bmatrix} $$

$\hat{x}_6$ $\hat{y}_6$ $\hat{z}_6$

Figure 11 - R_{06} or *RPY* rotation matrix, based on *Roll*, *Pitch*, *Yaw*, angles, dictates the final orientation of the wrist

The global matrix present in Figure 10 also has the orientation of the end effector in its first three rows and columns, where the columns correspond, respectively, to the orientation in x ($\hat{x}_6$), y ($\hat{y}_6$) and z ($\hat{z}_6$).

It is also possible to determine, based on the rotation matrix and through equations 3, 4 and 5 the value of the angles that define the orientation, *Roll* (Φ), *Pitch* (Ψ) and *Yaw* (θ) according to the nomenclature of equation 2.

$$ RPY = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (2) $$

$$\Phi = atan_2(r_{21}, r_{11}) \tag{3}$$

$$\Psi = atan_2\left(-r_{31}, \sqrt{{r_{32}}^2 + {r_{33}}^2}\right) \tag{4}$$

$$\theta = atan_2(r_{32}, r_{33}) \tag{5}$$

2.1.1.2. Angular Determination and Implementation in the Model

The calculation of each of the angles will now be analyzed from a theoretical point of view with its practical implementation using *Modelica*, specifically the "*Text View*" section for model development.

Before focusing in detail on each of the angles, the matrices present in (1) and Figure 11 described in the previous section will be implemented. The set of steps performed throughout the implementation will be duly described.

- Definition of the length of the links, based on the vector L and where the values of the dimensions $d1, a1, a2, d4$ and $d6$ implied by the *Denavit - Hartenberg* parameters are included.
- Definition of the *Pose_tip* vector, where the coordinates of the final position of the end-effector (P_{06}) are incorporated in the 3 initial values, and the *Roll*, *Pitch* and *Yaw* (RPY) angles are defined in the 3 final values, which are arbitrated with a value of 0.
- Declaration of the rotation matrix R_{06}, based on the declared values for the 3 angles involved in the end-effector orientation (RPY) and also the matrix associated with the end-effector RPY_{tip} that defines the transformed matrix between the base frame of reference and the end-effector frame of reference.
- Definition of the global matrix T_{06}, based on the rotation matrix R_{06} and the cartesian position vector of the end-effector (P_{06}).

Each of the steps is observable in the following excerpt.

```
  parameter Real L[:] = {0.352, 0.07, 0.177, 0.36, 0.177, 0.38, 0.065};
  parameter Real d1 = L[1];
  parameter Real a1 = L[2];
  parameter Real a2 = L[4];
  parameter Real d4 = L[6];
  parameter Real d6 = L[7];
  output Real joints[6];
  parameter Real Pose_tip[6, 1] = [0; 0.4965; 0.35; 0; 0; 0];
  Real P06[3, 1] = [Pose_tip[1]; Pose_tip[2]; Pose_tip[3]];
  Real RPY[1, 3] = pi / 180 * [Pose_tip[4], Pose_tip[5], Pose_tip[6]];
  Real yaw_x = RPY[1, 1];
  Real pitch_y = RPY[1, 2];
  Real roll_z = RPY[1, 3];
  Real R06[3, 3] = [cos(roll_z) * cos(pitch_y), (-sin(roll_z) * cos(yaw_x)) + cos(roll_z) * sin(pitch_y) * sin(yaw_x),
sin(roll_z) * sin(yaw_x) + cos(roll_z) * sin(pitch_y) * cos(yaw_x); sin(roll_z) * cos(pitch_y), cos(roll_z) *
cos(yaw_x) + sin(roll_z) * sin(pitch_y) * sin(yaw_x), (-cos(roll_z) * sin(yaw_x)) + sin(roll_z) * sin(pitch_y) *
cos(yaw_x); -sin(pitch_y), cos(pitch_y) * sin(yaw_x), cos(pitch_y) * cos(yaw_x)];
  Real RPY_tip[3, 3] = [0, 1, 0; 1, 0, 0; 0, 0, -1];
  Real RPY_tip_z[3, 1] = [RPY_tip[1, 3]; RPY_tip[2, 3]; RPY_tip[3, 3]];
  Real T06[4, 4] = [R06[1, 1], R06[1, 2], R06[1, 3], P06[1]; R06[2, 1], R06[2, 2], R06[2, 3], P06[2]; R06[3, 1],
R06[3, 2], R06[3, 3], P06[3]; 0, 0, 0, 1];
  Real P05_x = P05[1, 1];
  Real P05_y = P05[2, 1];
  Real P05_z = P05[3, 1];
```

Figure 12 - First excerpt of the code involved in the inverse kinematics model

- Calculation of the wrist position (P_{05}), equivalent to the subtraction between the position specified for the end-effector (P_{06}) and the distance between the end-effector and the wrist (joint 5) multiplied by the z orientation of the rotation matrix RPY (RPY_{tip_z}).

```
Real P05_x = P05[1, 1];
Real P05_y = P05[2, 1];
Real P05_z = P05[3, 1];
Real theta1_L;
Real theta1_R;
Real theta1;
Real theta1_deg = theta1 * 180 / 3.141592653589793238;
Real P05[3, 1] = P06 - (d6 + 0.0238) * RPY_tip_z;
```

Figure 13 - Second excerpt of the code involved in the inverse kinematics model

Determination of geometric θ_1

The previous determination of the wrist coordinates (P_{05}), allows to obtain geometrically the value of θ_1, based on the projection of the vector p_w in the xy plane (Figure 14), through the expression (6). In this expression, the importance of using the trigonometric function $arctan_2$ is highlighted, so that the spatial location of the robot is taken into account when calculating θ_1.

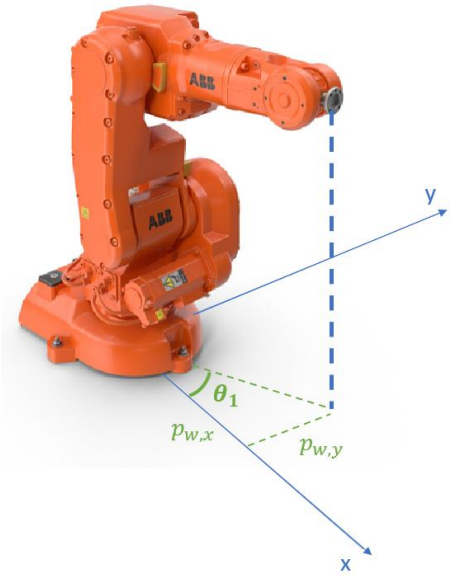


Figure 14 - Projection of vector p_w into the xy plane

$$\theta_1 = \begin{cases} arctan_2(p_{w,y}, p_{w,x}) & se \quad p_{w,y} \neq 0 \\ 0 & se \quad p_{w,y} = 0 \end{cases} \tag{6}$$

This is not the only solution for θ_1, since this angle is of rotation about z. Thus, the second solution presents a very simple calculation:

$$\theta_1{}' = \theta_1 + 180^\circ \tag{7}$$

Calculation of θ_1 - Implementation

The angles responsible for the positioning of the wrist are θ_1, θ_2 and θ_3. It can be taken into consideration that angle 1 defines the positioning of the wrist along the xy plane, while angle 2 and angle 3 define the positioning along the xz plane. Angle 1 is defined as the arc-tangent of the position of the wrist along the xy plane. If the variation seen along the y axis is close to 0, the angle value can be disregarded and assigned an approximation of 0.

- Defining the values of the 2 solutions presented for θ_1, $theta1\_L$ (when the left "elbow" is used) and $theta1\_R$ (when the right "elbow" is used).

```
equation
  if abs(P05_y) <= 0.01 then
    theta1_L = 0;
  else
    theta1_L = atan2(P05_y, P05_x);
  end if;
  theta1_R = theta1_L + pi;
  if theta1_L < (-pi) or theta1_L > pi then
    theta1 = theta1_R;
  elseif theta1_R < (-pi) or theta1_R > pi then
    theta1 = theta1_L;
  else
    theta1 = theta1_L;
  end if;
```

Figure 15 - Third excerpt of the code involved in the inverse kinematics model

Calculation of θ_2 and θ_3

For the calculation of θ_2 and θ_3 the problem can be simplified to a 2-link planar robot, where a vector between the shoulder and wrist are defined. Thus, the determination of θ_2, similarly to θ_1 depends on the previously determined shoulder (ombro) and wrist (pulso) coordinates, and the method of obtaining them is purely geometric.

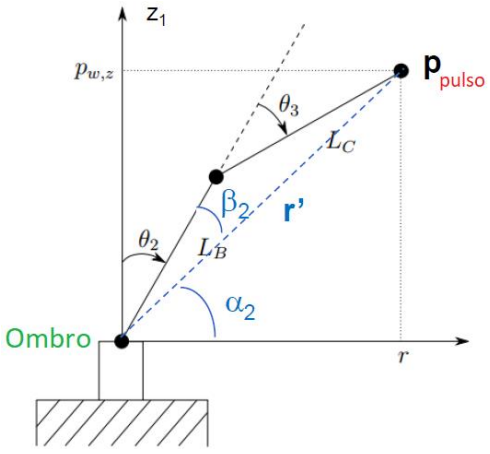


Figure 16 - Simplified problem to a 2-link planar robot

Using the schematic shown and the cosine theorem it is possible to extract the angles β_2 and, consequently, θ_2.

$$\begin{cases} \theta_2 = 90^o - \alpha_2 - \beta_2 \\ \alpha_2 = \arctan({}^{B}p_{ws,z}, r) \\ L_c^2 = L_B^2 + r'^2 - 2 \times L_B \times r' \times \cos(\beta_2) \end{cases} \quad (8)$$

Being that,

$$\beta_2 = \text{acos}\left(\frac{L_B^2 + r'^2 - L_c^2}{2 \times L_B \times r'}\right) \quad (9)$$

Only exists if,

$$-1 \leq \frac{L_B^2 + r'^2 - L_c^2}{2 \times L_B \times L_C} < 1 \quad (10)$$

By the cosine theorem it is also possible to determine β_3. This is assumed to be a considered angle that helps in obtaining θ_3. It should also be noted that $r = \sqrt{{p_{pulso,x}}^2 + {p_{pulso,y}}^2}$.

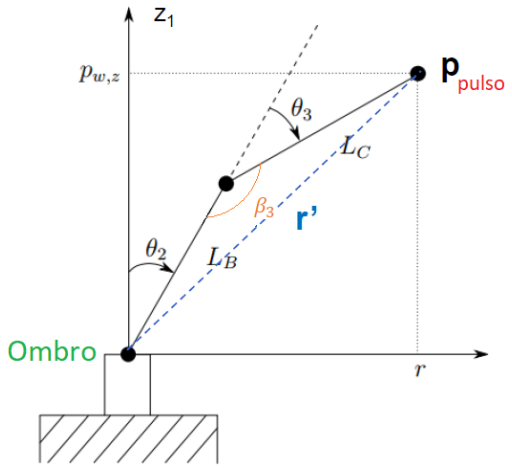


Figure 17 - Simplified problem to a 2-link planar robot (β_3 present instead of β_2)

$$\beta_3 = \text{acos}\left(\frac{L_B^2 + L_c^2 - r'^2}{2 \times L_B \times L_C}\right) \quad (11)$$

With β_3 defined, and using the geometric properties of the sketch, it is possible to obtain the equation for angle θ_3:

$$\theta_3 = \beta_3 - 90 \quad (12)$$

Calculation of θ_2 and θ_3 – Implementation

The set of steps performed throughout the implementation and aimed at calculating the angles θ_2 and θ_3 are presented:

- Calculation of the length of the vector between the shoulder and the wrist r' ($r3$ in the *script*), according to the following formulation $r3 = \sqrt{{P_{25_x}}^2 + {P_{25_y}}^2 + {P_{25_z}}^2}$. Preceded by this calculation is presented the calculation of P_{25} by subtraction between the vector between the base and the pulse P_{05} and the vector P_{02} defined between the base and the second axis of the manipulator, being $P_{25} = \{P_{25_x},\ P_{25_y},\ P_{25_z}\}$.
- Calculation of $\beta_3(beta3)$ according to equation 11, preceded by calculation of the argument $(arg3)$ $\frac{L_B^2+L_C^2- r'^2}{2\times L_B\times L_C}$where L_B and L_C equal the dimensions $a2$ and $d4$, respectively.
- Calculation of the two solutions of θ_3 ($theta3\_L$ e $theta3\_R$). The solution profiling the left elbow is given by equation 12, while the solution established by the right elbow is calculated using the following formula.

$$\theta_3 = 270 - \beta_3 \tag{13}$$

- Calculation of r' projected along the xy plane (r)
- Calculation of α_2 ($alfa2$) using the coordinate of vector P_{25} ($P25\_Z$) and r
- Calculation of $\beta_2(beta2)$ according to equation 9, preceded by calculation of the argument $(arg2)$ $\frac{L_B^2+ r'^2- L_C^2}{2\times L_B\times r'}$.
- Calculation of the two solutions of θ_2 ($theta2\_L$ e $theta2\_R$). The solution profiling the left elbow is given by equation 8, while the solution established by the right elbow is calculated using the following formula.

$$\theta_2 = 90^o - \alpha_2 + \beta_2 \tag{14}$$

```
Real P02[3, 1] = [a1 * cos(theta1); a1 * sin(theta1); d1];
Real P25[3, 1] = P05 - P02;
Real P25_x = P25[1, 1];
Real P25_y = P25[2, 1];
Real P25_z = P25[3, 1];
Real r3 = sqrt(P25_x ^ 2 + P25_y ^ 2 + P25_z ^ 2);
Real LB = a2;
Real LC = d4;
Real arg3 = (LB ^ 2 + LC ^ 2 - r3 ^ 2) / (2 * LB * LC);
Real beta3 = acos(arg3);
Real theta3_L = beta3 - 3.141592653589793238 / 2;
Real theta3_R = 3 * (3.141592653589793238 / 2) - beta3;
Real theta3 = theta3_L;
Real theta3_deg = theta3 * 180 / 3.141592653589793238;
Real r = sqrt(P25_x ^ 2 + P25_y ^ 2);
Real alfa2 = atan2(P25_z, r);
Real arg2 = (LB ^ 2 + r3 ^ 2 - LC ^ 2) / (2 * LB * r3);
Real beta2 = acos(arg2);
Real theta2_L = -(3.141592653589793238 / 2 - alfa2 - beta2);
Real theta2_R = -(3.141592653589793238 / 2 - alfa2 + beta2);
Real theta2 = theta2_L;
Real theta2_deg = theta2 * 180 / 3.141592653589793238;
```

Figure 18 - Fourth excerpt of the code involved in the inverse kinematics model

Calculation of θ_4 , θ_5 and θ_6

Using the *Denavit - Hartenberg* parameters, it is possible to calculate the first 3 individual transformation matrices and from there extract the corresponding rotation matrix ${}^0R_3$. Considering ${}^0R_6$, the rotation matrix RPY (which is known), it is possible to solve the equation for ${}^3R_6$.

$$ {}^3R_6 = {}^3R_0 \times {}^0R_6 = inv({}^0R_3) \times {}^0R_6 \tag{15} $$

Having obtained the final values of the matrix ${}^3R_6$ and knowing that its elements may be defined in the way illustrated in Figure 19, it's possible to calculate θ_4, θ_5 e θ_6.

$$ {}^3R_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \cos(\theta_4)\cos(\theta_5)\cos(\theta_6) - \sin(\theta_4)\sin(\theta_6) & -\cos(\theta_6)\sin(\theta_4) - \cos(\theta_4)\cos(\theta_5)\sin(\theta_6) & -\cos(\theta_4)\sin(\theta_5) \\ \sin(\theta_5)\cos(\theta_6) & -\sin(\theta_5)\sin(\theta_6) & \cos(\theta_5) \\ -\cos(\theta_4)\sin(\theta_6) - \cos(\theta_5)\cos(\theta_6)\sin(\theta_4) & \cos(\theta_5)\sin(\theta_4)\sin(\theta_6) - \cos(\theta_4)\cos(\theta_6) & \sin(\theta_4)\sin(\theta_5) \end{bmatrix} $$

Figure 19 - Rotation matrix ${}^3R_6$

Resorting to the element r_{23} and knowing that $\sin(\theta) = \pm\sqrt{1 - \left(cos(\theta)\right)^2}$, the determination of θ_5 becomes possible, whose dual solution can be taken from the following equation.

$$\theta_5 = arctan_2\ (\pm\sqrt{1-(r_{23})^2}, r_{23}) \tag{16}$$

Similarly, using the formulations taken from matrix present in Figure 20 allows, based on elements r_{33} and r_{13}, to extract the value of θ_4 as well as, using r_{22} and r_{21}, the value of θ_6 .

$$\begin{cases} \theta_4 = arctan2\ (r_{33}, -r_{13}\ ) \\ \theta_6 = arctan2\ (-r_{22}, r_{21}\ ) \end{cases} \tag{17}$$

The deductions of these values present, however, the condition that $\sin(\theta_5) \neq 0$ aimed at avoiding singularities. It is in this sense that the following conditional definition for both angles θ_4 and θ_6 is further considered.

$$\begin{cases} \theta_4 = 0\ \wedge \theta_6 = arctan_2\ (r_{12}, r_{32}) & if\ \theta_5 = 0 \\ \theta_4 = 0\ \wedge \theta_6 = -\ arctan_2(-r_{12}, -r_{32}) & if\ \theta_5 = \pi \end{cases} \tag{18}$$

Calculation of θ_4 , θ_5 and θ_6 – Implementation

The set of steps performed throughout the implementation and aimed at calculating the angles θ_4 , θ_5 and θ_6 are presented:

- Statement of the *Denavit - Hartenberg* parameters a_i , α_i , d_i and θ_i for the different frames of reference ($DH\_alpha,\ DH\_a, DH\_theta, DH\_d$).
- Calculation of the first 3 individual transformation matrices ${}^0T_1$, ${}^1T_2$ e ${}^2T_3$ ($T01$, $T12$ e $T23$).
- Calculation of the transformation matrix ${}^0T_3$ ($T03$) and consequent extraction of the rotation matrix ${}^0R_3$ ($R03$) involved in the first 3 frames of reference, proceeded by its inverse ${}^3R_0$ ($R30$).
- Calculation of the matrix ${}^3R_6$ ($R36$), necessary for the calculation of the 3 angles. Based on the definition of this matrix, each of the constituent elements is subsequently determined (r_{11}, r_{12}, r_{13}, etc).

```
  Real DH_alpha[6, 1] = 3.141592653589793238 / 180 * [0; 90; 0; 90; -90; 90];
  Real DH_a[6, 1] = [0; a1; a2; 0; 0; 0];
  Real DH_theta[6, 1] = 3.141592653589793238 / 180 * [-90; 90; 0; 0; 0; 0];
  Real DH_d[6, 1] = [d1; 0; 0; d4; 0; d6];
  Real T01[4, 4] = [cos(DH_theta[1, 1] + theta1), -sin(DH_theta[1, 1] + theta1), 0, DH_a[1, 1]; cos(DH_alpha[1, 1]) *
sin(DH_theta[1, 1] + theta1), cos(DH_theta[1, 1] + theta1) * cos(DH_alpha[1, 1]), -sin(DH_alpha[1, 1]), -
sin(DH_alpha[1, 1]) * DH_d[1, 1]; sin(DH_alpha[1, 1]) * sin(DH_theta[1, 1] + theta1), sin(DH_alpha[1, 1]) *
cos(DH_theta[1, 1] + theta1), cos(DH_alpha[1, 1]), cos(DH_alpha[1, 1]) * DH_d[1, 1]; 0, 0, 0, 1];
  Real T12[4, 4] = [cos(DH_theta[2, 1] + theta2), -sin(DH_theta[2, 1] + theta2), 0, DH_a[2, 1]; cos(DH_alpha[2, 1]) *
sin(DH_theta[2, 1] + theta2), cos(DH_theta[2, 1] + theta2) * cos(DH_alpha[2, 1]), -sin(DH_alpha[2, 1]), -
sin(DH_alpha[2, 1]) * DH_d[2, 1]; sin(DH_alpha[2, 1]) * sin(DH_theta[2, 1] + theta2), sin(DH_alpha[2, 1]) *
cos(DH_theta[2, 1] + theta2), cos(DH_alpha[2, 1]), cos(DH_alpha[2, 1]) * DH_d[2, 1]; 0, 0, 0, 1];
  Real T23[4, 4] = [cos(DH_theta[3, 1] + theta3), -sin(DH_theta[3, 1] + theta3), 0, DH_a[3, 1]; cos(DH_alpha[3, 1]) *
sin(DH_theta[3, 1] + theta3), cos(DH_theta[3, 1] + theta3) * cos(DH_alpha[3, 1]), -sin(DH_alpha[3, 1]), -
sin(DH_alpha[3, 1]) * DH_d[3, 1]; sin(DH_alpha[3, 1]) * sin(DH_theta[3, 1] + theta3), sin(DH_alpha[3, 1]) *
cos(DH_theta[3, 1] + theta3), cos(DH_alpha[3, 1]), cos(DH_alpha[3, 1]) * DH_d[3, 1]; 0, 0, 0, 1];
  Real T03[4, 4] = T01 * T12 * T23;
  Real R03[3, 3] = T03[1:3, 1:3];
  Real R30[3, 3] = inv(R03);
  Real R36[3, 3] = R30 * R06;
  Real r11 = R36[1, 1];
  Real r12 = R36[1, 2];
  Real r13 = R36[1, 3];
  Real r21 = R36[2, 1];
  Real r22 = R36[2, 2];
  Real r23 = R36[2, 3];
  Real r31 = R36[3, 1];
  Real r32 = R36[3, 2];
  Real r33 = R36[3, 3];
```

Figure 20 - Fifth excerpt of the code involved in the inverse kinematics model

- Compute the two solutions for θ_5, using the elements present in equation 16.

```
Real theta5_R = atan2(sqrt(1 - r23 ^ 2), r23);
Real theta5_L = atan2(-sqrt(1 - r23 ^ 2), r23);
Real theta5 = theta5_L;
Real theta5_deg = theta5 * 180 / 3.141592653589793238;
Real theta4;
Real theta6;
Real theta4_deg;
Real theta6_deg;
```

Figure 21 - Sixth excerpt of the code involved in the inverse kinematics model

- Calculation of the generic solutions for θ_4 and θ_6, as well as the conditional solutions dependent on the value of θ_5.
- Final definition of the *joints* vector consisting of the 6 angles, being this vector equivalent to the only output defined for the model.

```
if abs(theta5) < 0.01 then
  theta4 = 0;
  theta6 = atan2(r12, r32);
elseif abs(theta5 - pi) < 0.01 then
  theta4 = 0;
  theta6 = -atan2(-r12, -r32);
else
  theta4 = atan2(r33 / sin(theta5), -r13 / sin(theta5));
  theta6 = atan2(-r22 / sin(theta5), r21 / sin(theta5));
end if;
theta4_deg = theta4 * 180 / 3.141592653589793238;
theta6_deg = theta6 * 180 / 3.141592653589793238;
joints = {theta1, theta2, theta3, theta4, theta5, theta6};
```

Figure 22 - Seventh excerpt of the code involved in the inverse kinematics model

2.1.2. Trajectory Definition

After developing the code that dictates obtaining the desired angles, there is the input of these same angles in the model for trajectory calculation or also called "motion planning". In robotic control systems, motion planning is the computational problem responsible for outlining the path to be executed, free of obstacles, through the sequence of a set of valid configurations that move the object from the origin to the profiled destination. For a better explanation of this section, it starts with the first approach taken in the development of this model, in which a *Pick&Place* with no *via point* mode is considered. At a later stage, however, the improvement with the *via point* mode is implemented, thus bringing complementary aspects to the initially conceived model.

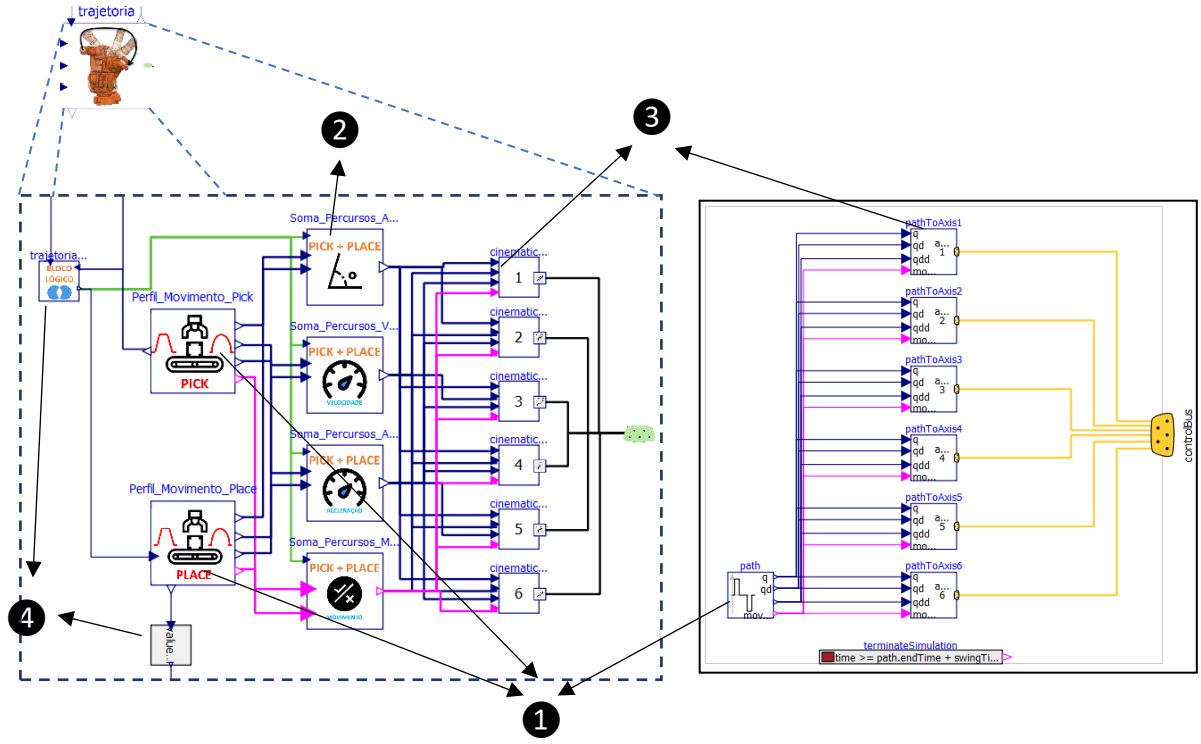


Figure 23 - On the left, the elaborated model involved in the trajectory definition. On the right, the reference model used.

In Figure 23 presented, it is possible to observe on the left the initially designed model of the path planning and, analogously, the model [21] with the same purpose inserted in the reference model present in the *Modelica Standard Library*. Some aspects introduced can be highlighted, such as the existence (1) of 2 trajectory definition models, or movement profile, contrasting with 1 in the reference model, the existence of a section (2) that collects and sums both movement profiles over time, coming from the 2 previous models, thus defining a single movement profile, and also presenting the specificity of dividing its state in position, velocity,

acceleration and also movement state, through the elaboration of 4 distinct models. A third section (3) present in both models, responsible for distributing each of the information from the motion profile along the 6 axes and, finally, the elaboration of small logic models (4), which allow the association of the model with a precedent and subsequent process, in which the updating of states over time allows the execution of the remaining blocks in the model, only for a specific time interval.

Due to the amount of computations required for this process, the global trajectory definition model is assumed to be dense in terms of internal models. In order to clarify the function of each one more clearly, the aforementioned division will be accompanied by a clearer explanation of the sections.

2.1.2.1. Models for the Kinematic Profile

Through the inputs, which characterize the initial and final joint angle, as well as kinematic constraints, namely velocity and maximum angular acceleration, which can be reached by the axis, the trajectory algorithm processes the respective information and defines the output variables. These correspond to the instantaneous values of position, velocity and acceleration, and there is also a Boolean output variable that takes the positive value when there is angular movement (evolution of the algorithm from the inputs).

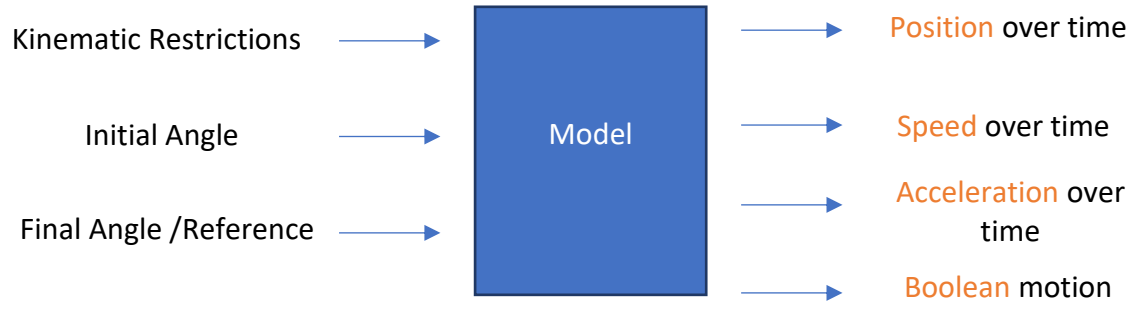


Figure 24 - Inputs and Outputs of the intended model for the motion profile

The goal is the fastest possible movement from the start position to the end position under certain kinematic constraints. This block generates position, velocity and acceleration as output values. The signals are constructed in such a way that no faster motion is possible due to the maximum allowed velocity and acceleration assigned to it.

The initally defined kinematic motion (trapezoidal), presents an initial movement of acceleration, proceeded by constant speed and in a final phase is characterized by a deceleration, established when the end point is close to being reached. In this way, the reference values to be reached by the manipulator will be continuously supplied to the axes, in order to be further processed and thus trigger the desired movement.

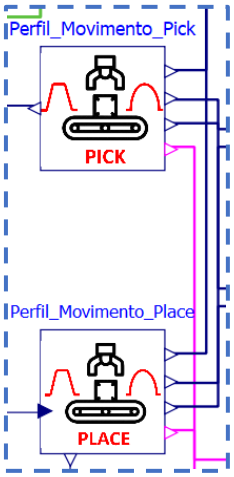


Figure 25 - Profile Generation model for Pick (on top) and Place (on bottom) movements

Before a clearer understanding of the algorithm, the inputs and outputs along the kinematic profile generation section will be explained. For each generation model, the kinematic characteristics are defined as well as the time instant at which the algorithm and the calculation of the respective profile start. For the case of the *Pick* profile generation model, the initial joint angular values are the values considered for the initial position of the robot which is independent of the type of transport path being executed, while the final angular values will be calculated by the inverse kinematics model. The kinematic constrictions, relating to speed and acceleration, are user-set parameters, that is they are predefined fixed values. The time instant at which the profile calculation starts is a variable dependent on the package transport time executed by the conveyor at the entrance of the global system.

In the *Place* profile generation model, the initial angular values correspond to the final angular values of the *Pick* profile, and the final angular values admit the values calculated by the corresponding inverse kinematics model. The time instant at which the profile calculation starts depends on the gripper's operation time in gripping the package present on the conveyor. Thus, the instant of time corresponds to the instant when the *Pick* movement is finished plus the gripper operation time.

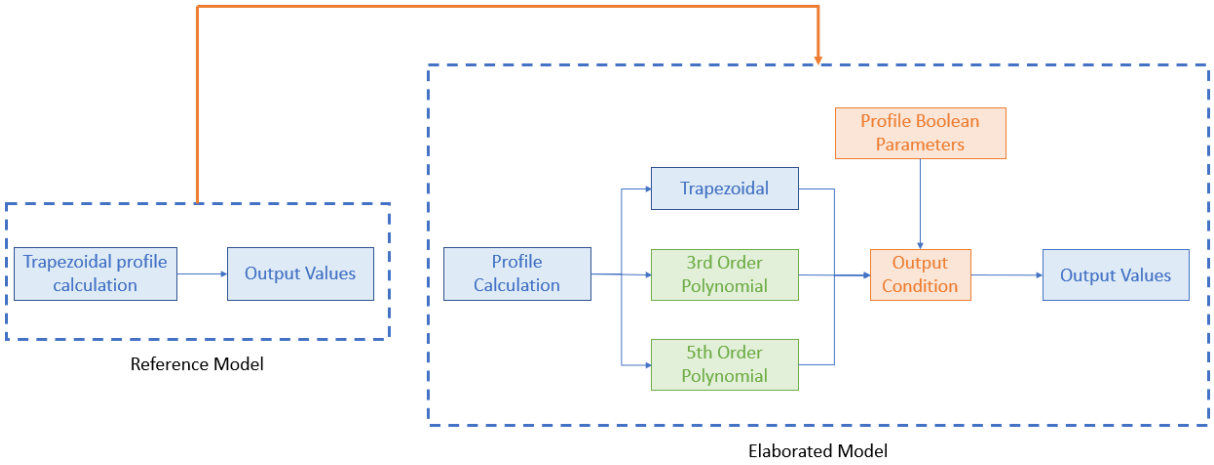


Figure 26 - Structure of the code involved in generating the movement profile

In Figure 26 the structure of the code for generating the motion profile present in the models is shown. This structure has been extended in order to admit additional motion profiles, if they are selected. In addition to the trapezoidal velocity profile, already designed in the reference model present in the library [22], the calculation of 3rd and 5th order polynomial profiles have been introduced, seeking to make the configuration of the trajectory more complete, according to the user's wishes. The different types of profiles are calculated simultaneously in the model, with only one subsequently admitting the final values to be transmitted.

2.1.2.2. Kinematic Trapezoidal Profile

Trapezoidal Profile - Theory

To generate the algorithm involved in the kinematic trajectory of the arm, it is necessary to first define the type of kinematic profile to be adopted. Using the example of a robotic arm provided by *OpenModelica* as an aid, the kinematic profile initially defined is the trapezoidal motion profile, and its study will be carried out [23] before the subsequent implementation of other profiles. The trapezoidal velocity profile gives from the start, in comparison with a 3rd order polynomial path, a shorter duration of motion for a maximum velocity and equal angular displacement.

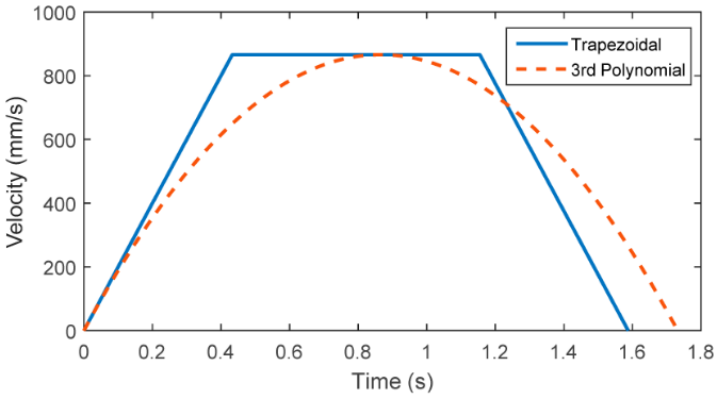


Figure 27 - Trapezoidal and 3rd order polynomial velocity profile, for the same maximum velocity

A simple example of this speed profile for a point-to-point displacement can be seen in Figure 28, where a smooth acceleration and deceleration between the starting point and the reference point to be reached can be observed, and a constant speed is admitted between the acceleration and deceleration.

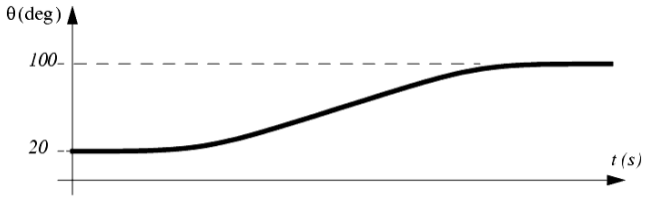


Figure 28 - Position variation along a trapezoidal velocity profile

The angular displacement, between two points, can be specified by a number of parameters, which together define the motion profile. For the simple trapezoidal profile, these parameters are distance, maximum velocity, maximum acceleration and deceleration. The trapezoidal profile is divided into 3 phases: acceleration, constant velocity, and deceleration. If the displacement is positive, the acceleration is positive and constant in the first phase. The velocity is, still in this phase, a linear function in time and the position is a parabolic curve in time. In section 2 of the profile, the acceleration is zero and the velocity is a constant value. Thus, position is a linear function in time in this section. Finally, a negative acceleration constant is imposed in the 3rd section and the velocity is linearly reduced. Figure 29 shows the position, velocity and acceleration of a trapezoidal motion profile, where the 3 graphs are illustrated.

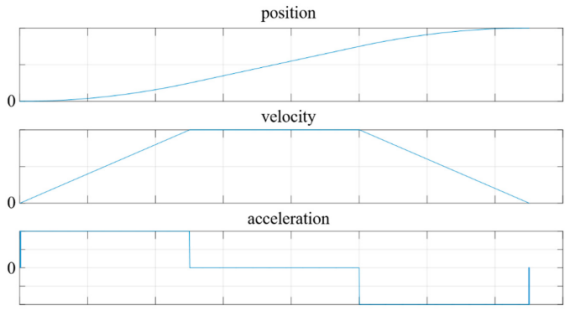


Figure 29 - Position, velocity, and acceleration of a trapezoidal motion profile

To design a trapezoidal displacement profile, the kinematic parameters must be predefined and the displacement given. If the maximum velocity and acceleration are given, the duration of the acceleration and displacement can be derived. Typically, the operator can assign the maximum values of velocity and acceleration to define the motion parameters of an industrial robot. It is assumed that the velocities at the start and end points are zero, as can be seen in Figure 30, and that the acceleration and deceleration times are identical.

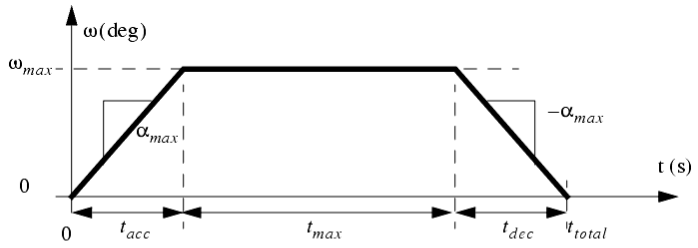


Figure 30 - Trapezoidal motion profile, with description of acceleration, constant speed, and deceleration times

If the maximum velocity is v_{max} and the maximum acceleration a_{max}, the acceleration time t_a can be derived as follows.

$$t_a = \frac{v_{max}}{a_{max}} \tag{19}$$

The displacement is defined as Δx and is assumed to be positive. The time required to reach the target, through displacement Δx, is t_f and can be computed with the following formula.

$$t_f = \frac{\Delta x}{v_{max}} + t_a \tag{20}$$

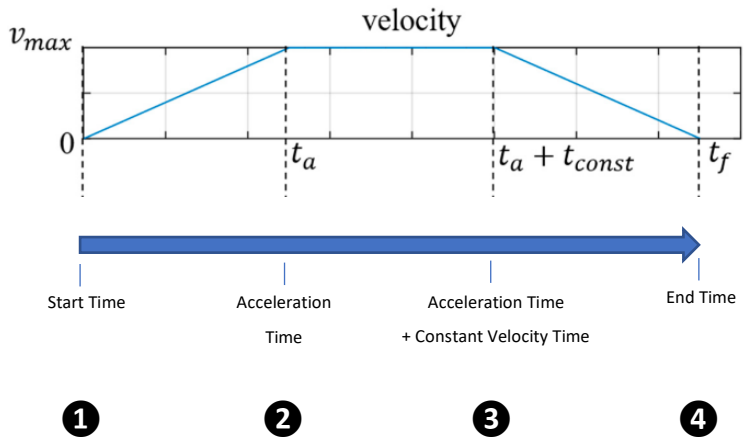
For an established symmetry between acceleration and deceleration, the time duration in which the shaft adopts a constant velocity is calculated as follows.

.

$$t_{const} = t_f - 2t_a \tag{21}$$

In order to define the different positions verified over time and that will have special importance in the code to be developed later, the trapezoidal velocity profile presents its relevance, since the positions subject to calculation can be defined directly through equations that define the profile area at different instants in time. Based on this principle the following Table 3 has been realized. In it, the calculation of the absolute positions is presented, for the different reference times shown in the table image as well as over the instants present between the reference times, and whose calculation is stated through functions.

Table 3 - Calculating the positions along the trapezoidal velocity profile



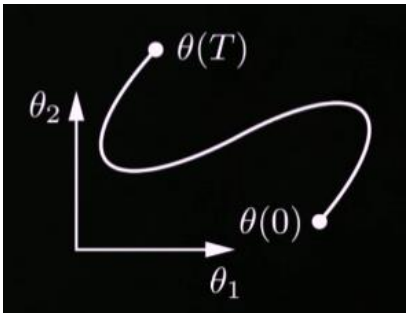
Position	Value/Formula
Position 1 (Start)	$\boldsymbol{p_1 = 0}$
Position Function (1 → 2)	$\boldsymbol{p(t) = \frac{1}{2} \times t \times v_{max}}$ **or** $\boldsymbol{p(t) = \frac{1}{2} \times t^2 \times a_{max}}$
Position 2 (position at t_a)	

	$$p_2 = \frac{1}{2} \times t_a \times v_{max}$$ or $$p_2 = \frac{1}{2} \times {t_a}^2 \times a_{max}$$
Position Function (2 → 3)	$$p(t) = p_2 + v_{max} \times (t - t_a)$$
Position 3 (position at $t_a + t_{const}$)	$$p_3 = p_2 + v_{max} \times t_{const}$$
Position Function (3 → 4)	$$p(t) = p_3 + \frac{1}{2} \times (t - t_a - t_{const}) \times v_{max}$$ or $$p(t) = p_3 + \frac{1}{2} \times (t - t_a - t_{const})^2 \times a_{max}$$
Position 4 (Final Position t_f)	$$p_4 = p_3 + \frac{1}{2} \times t_d \times v_{max}$$ $$p_4 = p_3 + \frac{1}{2} \times {t_d}^2 \times a_{max}$$ **Being that $t_d = t_a$**

The calculation formulas presented in table 3 show their significance throughout the development of the code, however the final values obtainable through the calculations can be disregarded, since the calculation formulas are to be aided by a time scale, and this will be explained below.

The configuration of the 6 axes of a manipulator as a function of time is called trajectory.

Trajectory: $\theta(t), t \in [0, T]$



Path: $\theta(s), s \in [0,1]$

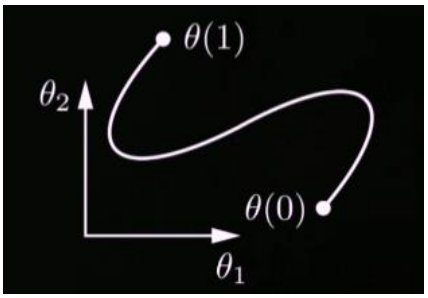


Figure 31 - Trajectory and path along a 2-axis system

Where T is assumed to be the value of the total travel time. However, the interest of the configuration development is not always initially dependent on time, but rather on the configured space. That is, the curve observed in Figure 31 in which can be observed the evolution of the 2-axis angles, instead of evolving according to time t, evolves according to a function whose parameter s varies between 0 (initial configuration) and 1 (final configuration), thus defining the path.

A path can be transformed into a trajectory, in turn, by defining s as a function over time. In this way, the trajectory can be defined:

$$\theta\big(s(t)\big), s : [0, T] \rightarrow [0,1]$$

The definition of a path, through the implementation of a trajectory is thus called time scaling, where the speed with which the path is traveled is established. Thus, the final implementation required will involve the use of the formulas presented in the table 3 in conjunction with the angular displacement delineated between the points of the path, causing the final values of position to vary between 0 and 1, resulting from a relative and not absolute positioning.

In equation 21, t_{const} is positive when t_f is greater than $2t_a$. This means that t_{const} can be zero if t_f is equivalent to $2t_a$. This happens when the displacement Δx is relatively small. The constant velocity zone in the profile may not exist, thus giving rise to a triangular velocity profile where only the acceleration and deceleration zones appear, as profiled in Figure 32.

For a kinematics algorithm that offers greater consistency, this velocity profile is also defined in the event that the angular displacement of a particular axis is reduced.

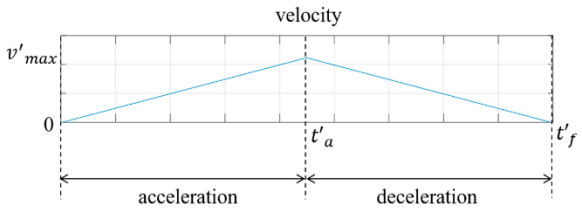


Figure 32 - Triangular speed profile

For this profile, the maximum speed is not reached when the maximum acceleration a_{max}, is imposed. It is in this sense that the variables t_a, t_f and v_{max} need to be recalculated, giving rise to t'_a, t'_f e v'_{max}.

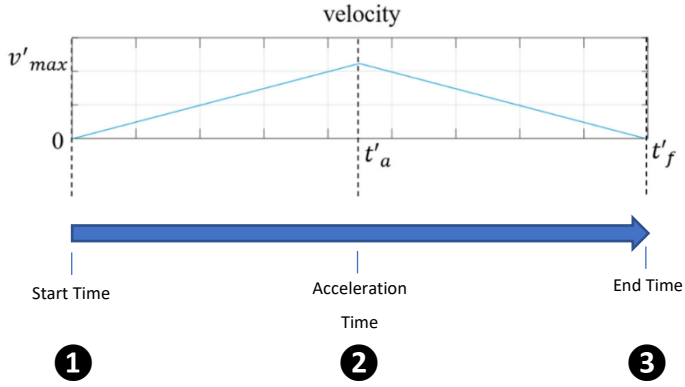
$$t'_a = \sqrt{\frac{\Delta x}{a_{max}}} \tag{22}$$

$$v'_{max} = a_{max} \times t'_a \tag{23}$$

$$t'_f = 2t'_a \tag{24}$$

The positions and position functions are again calculated, this time for the triangular profile.

Table 4 - Calculating the positions along the triangular velocity profile



Position	Value/Formula
Position 1 (Start)	$\boldsymbol{p_1 = 0}$
	$\boldsymbol{p(t) = \frac{1}{2} \times t \times v'_{max}}$

Position Function (1 → 2)	$p(t) = \frac{1}{2} \times t^2 \times a_{max}$
Position 2 (position at t'_a)	$p_2 = \frac{1}{2} \times t'_a \times v'_{max}$ $p_2 = \frac{1}{2} \times t'^2_a \times a_{max}$
Position Function (2 → 3)	$p(t) = p_2 + \frac{1}{2} \times (t - t'_a) \times v'_{max}$ $p(t) = p_2 + \frac{1}{2} \times (t - t'_a)^2 \times a_{max}$
Position 3 (position at $t'_a + t'_d$)	$p_3 = p_1 + v_{max} \times t_{const}$

The detailed implementation of the trapezoidal profile, already present in the reference model, can be seen in appendix 1.

2.1.2.3. 3rd Order Polynomial Kinematic Profile

3rd Order Polynomial Profile - Theory

In order to be able to explore different kinematic profiles, a polynomial profile of motion is elaborated along the *Pick* and *Place* kinematic models. The equations for calculating the polynomial are inserted into the kinematic models and thus allow the trapezoidal profile to be compared to a 3rd order polynomial. The user is also given, as input parameters in the global system, the option to choose which of the two kinematic profiles he wants for the delineation of the angular trajectories along the robot's axes.

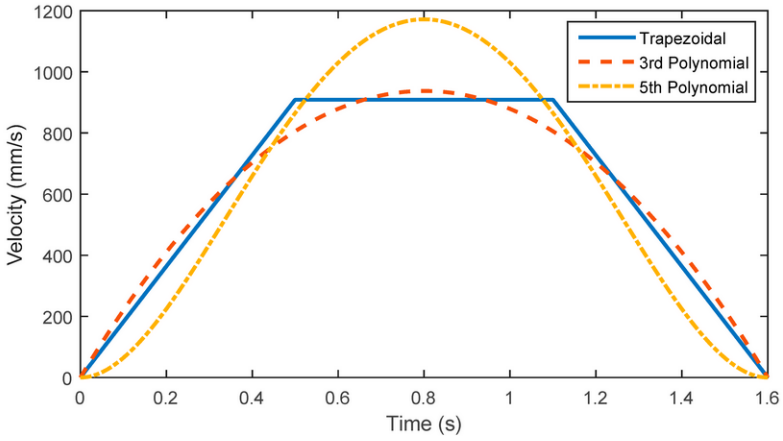


Figure 33 - Trapezoidal, 3rd order polynomial and 5th order polynomial velocity profile for the same time interval

In Figure 33, comparing the trapezoidal profile with the polynomial one, it can easily be seen that the use of a polynomial profile projects a smoother movement, without radical changes in slope, however this smoothness of movement is contrasted with higher values of speed and maximum acceleration.

The study of the polynomial trajectory that allows the manipulator to evolve from one position to another will be, throughout this section, explored, first in mathematical terms and then with its implementation in *Modelica*.

The mathematical deduction of the coefficients of a cubic polynomial equation will be briefly discussed. The choice of a cubic polynomial function is admitted for the following motivations:

- A cubic polynomial function is sufficient for the four constraints to be used;
- A polynomial function ensures smooth motion, because it is a continuous function and has a first derivative that is also continuous;
- If the first and last point are within the joint space, then the points in between will be as well.

The deductions expressed are designed for the case in which the initial and final velocity are zero, however they are also valid if one chose to use *via points* [24], a situation in which specific points of trajectory are usually designed in which the manipulator will pass with non-zero velocity.

Equation 25 contains the expression that describes the evolution of the angle of each joint over time. In this expression it is intended to determine a_0, a_1, a_2 e a_3.

$$\theta_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (25)$$

Deriving equation 25 gives equation 26, which allows obtaining the angular velocity at each joint over time.

$$\dot{\theta}_i(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{26}$$

Restrictions

The restrictions [25] for the intended trajectory are found in equations 27-30. These are the initial ($\theta_{i,0}$) and final ($\theta_{i,f}$) angle of the joint and the initial ($\dot{\theta}_{i,0}$) and final ($\dot{\theta}_{i,f}$) velocity of the joint. In equations 28 and 30, T is the duration time of the motion, which must be the same for all joints.

$$\theta_i(0) = \theta_{i,0} \tag{27}$$

$$\theta_i(T) = \theta_{i,f} \tag{28}$$

$$\dot{\theta}_i(0) = \dot{\theta}_{i,0} \tag{29}$$

$$\dot{\theta}_i(T) = \dot{\theta}_{i,f} \tag{30}$$

Determination of a_0, a_1, a_2 and a_3

From the constraints in equations 25-26 and equations 27-30 the system of equations [25] below can be written. This system of equations shows the constraint that gave rise to each equation.

$$\begin{cases} a_0 = \theta_{i,0} \\ a_1 = \dot{\theta}_{i,0} \\ \theta_{i,f} = \theta_{i,0} + \dot{\theta}_{i,0} T + T^2 a_2 + T^3 a_3 \\ \dot{\theta}_{i,f} = \dot{\theta}_{i,0} + 2a_2 T + 3a_3 T^2 \end{cases} \tag{31}$$

By solving this system it is possible to obtain the expressions for calculating the coefficients of the cubic polynomial equation that gives the evolution of a joint angle over time.

$$\begin{cases} a_0 = \theta_{i,0} \\ a_1 = \dot{\theta}_{i,0} \\ a_2 = -\frac{1}{T}\dot{\theta}_{i,f} - \frac{2}{T}\dot{\theta}_{i,0} + \frac{3}{T^2}(\theta_{i,f} - \theta_{i,0}) \\ a_3 = -\frac{2}{T}(\theta_{i,f} - \theta_{i,0}) + \frac{1}{T^2}(\dot{\theta}_{i,f} + \dot{\theta}_{i,0}) \end{cases} \tag{32}$$

3rd Order Polynomial Profile - Implementation

Having presented the theoretical principles of the 3rd order polynomial calculus, its implementation in *Modelica* will now be shown. In a first instance, the polynomial path coefficients for each of the axes are calculated. The coefficient implied in the constriction of the initial position adopts the value present in the *q_begin* array, while that of the initial velocity adopts the value 0. An initial velocity of 0, thus implies that $a_2 = \frac{3}{T^2}(\theta_{i,f} - \theta_{i,0})$ and also that $a_3 = -\frac{2}{T}(\theta_{i,f} - \theta_{i,0})$, whose angular variation, $(\theta_{i,f} - \theta_{i,0})$, is determined by the difference between the two vectors of initial and final position defined in the model.

Coefficients

(Axes)

1

```
Real a0_1 = q_begin[1];
Real a1_1 = 0;
Real a2_1 = if startTime <> 0 then 3 / T ^ 2 * (q_end[1] - q_begin[1]) else 0;
Real a3_1 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[1] - q_begin[1]) else 0;
```

2

```
Real a0_2 = q_begin[2];
Real a1_2 = 0;
Real a2_2 = if startTime <> 0 then 3 / T ^ 2 * (q_end[2] - q_begin[2]) else 0;
Real a3_2 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[2] - q_begin[2]) else 0;
```

3

```
Real a0_3 = q_begin[3];
Real a1_3 = 0;
Real a2_3 = if startTime <> 0 then 3 / T ^ 2 * (q_end[3] - q_begin[3]) else 0;
Real a3_3 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[3] - q_begin[3]) else 0;
```

4

```
Real a0_4 = q_begin[4];
Real a1_4 = 0;
Real a2_4 = if startTime <> 0 then 3 / T ^ 2 * (q_end[4] - q_begin[4]) else 0;
Real a3_4 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[4] - q_begin[4]) else 0;
```

5

```
Real a0_5 = q_begin[5];
Real a1_5 = 0;
Real a2_5 = if startTime <> 0 then 3 / T ^ 2 * (q_end[5] - q_begin[5]) else 0;
Real a3_5 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[5] - q_begin[5]) else 0;
```

6

```
Real a0_6 = q_begin[6];
Real a1_6 = 0;
Real a2_6 = if startTime <> 0 then 3 / T ^ 2 * (q_end[6] - q_begin[6]) else 0;
Real a3_6 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[6] - q_begin[6]) else 0;
```

Figure 34 – Excerpt from the implementation of a 3rd order polynomial kinematic profile, contained in the code involved in the kinematic profile generation model

The fact, that this calculation ends up describing the calculation of coefficients for motion profile models where a direct trajectory is verified, should be emphasized. That is, without *Via Point* mode activated. In situations where *Via Points* are used, these define points through which the trajectory of the manipulator's tip must pass, but not necessarily stopping at them in an absolute way. The objective is to observe a slowdown, in order to make sure that the passage through the reference point occurs, without sudden changes in trajectory direction and also without the occurrence of an absolute stop. Thus, the final speed value up to the *via point* and the initial speed value from the *via point* should have non-zero values. This change in speed restrictions changes the way the other coefficients (a_2 *e* a_3) are calculated.

```
  Real junta1 = if startTime <> 0 and time < endTime then a0_1 + a1_1 * (time - startTime) + a2_1 * (time -
startTime) ^ 2 + a3_1 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[1] else q_end[1];
  Real junta2 = if startTime <> 0 and time < endTime then a0_2 + a1_2 * (time - startTime) + a2_2 * (time -
startTime) ^ 2 + a3_2 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[2] else q_end[2];
  Real junta3 = if startTime <> 0 and time < endTime then a0_3 + a1_3 * (time - startTime) + a2_3 * (time -
startTime) ^ 2 + a3_3 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[3] else q_end[3];
  Real junta4 = if startTime <> 0 and time < endTime then a0_4 + a1_4 * (time - startTime) + a2_4 * (time -
startTime) ^ 2 + a3_4 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[4] else q_end[4];
  Real junta5 = if startTime <> 0 and time < endTime then a0_5 + a1_5 * (time - startTime) + a2_5 * (time -
startTime) ^ 2 + a3_5 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[5] else q_end[5];
  Real junta6 = if startTime <> 0 and time < endTime then a0_6 + a1_6 * (time - startTime) + a2_6 * (time -
startTime) ^ 2 + a3_6 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[6] else q_end[6];
  Real juntas[6] = {junta1, junta2, junta3, junta4, junta5, junta6};
  Real vel_juntas[6] = der(juntas);
```

Figure 35 - Excerpt from the implementation of a 3rd order polynomial kinematic profile, contained in the code involved in the kinematic profile generation model - Continuation

Once the coefficients are calculated, the next step is to write the equation of motion (equation 25), that is, the evolution of the angle over time. The code passage took into account that the evolution of each of the angles, described by the equation, must occur at the proper instant, so its execution only occurs within the established condition. This condition defines the variation of the angular value only when the simulation time interval of the global *Pick&Place* system is between the instant when the model is activated, being this instant defined by the parallel modeling to this model, and the time instant defined by the initial time when the model is "activated" + the final time of the movement profile. The function is defined for t with $0 < t < T$, being t defined as the total simulation time minus the instant of time when this model initializes the motion profile calculation and T as the total displacement time of the 6 axes.

2.1.2.4. 5th Order Polynomial Kinematic Profile

5th Order Polynomial Profile – Theory

In the event that an even smoother movement is preferred compared to the 3rd order polynomial function, a 5th order polynomial function is developed, in which acceleration constraints can be set, at the beginning and end of the movement, with a value of 0. This way, it is confronted with 2 more coefficients, along the function, which are used to satisfy the acceleration constraints mentioned.

Equation 33 contains the expression that allows the description of the evolution of the angle of each joint over time. In this expression it's intended to determine a_0, a_1, a_2, a_3, a_4 and a_5.

$$\theta_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (33)$$

Deriving equation 33 gives equation 34, which gives the angular velocity at each joint over time.

$$\dot{\theta}_i(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \quad (34)$$

Restrictions

The constraints [25] for the intended trajectory are found in equations 35-40. These are the initial ($\theta_{i,0}$) and final ($\theta_{i,f}$) angle of the joint, the initial ($\dot{\theta}_{i,0}$) and final ($\dot{\theta}_{i,f}$) velocity, and finally the initial ($\ddot{\theta}_{i,0}$) and final ($\ddot{\theta}_{i,f}$) acceleration of the joint. In the equations, T is the time duration of the motion, which must be the same for all joints.

$$\theta_i(0) = \theta_{i,0} \quad (35)$$

$$\theta_i(T) = \theta_{i,f} \quad (36)$$

$$\dot{\theta}_i(0) = \dot{\theta}_{i,0} \quad (37)$$

$$\dot{\theta}_i(T) = \dot{\theta}_{i,f} \quad (38)$$

$$\ddot{\theta}_i(0) = \ddot{\theta}_{i,0} \quad (39)$$

$$\ddot{\theta}_i(T) = \ddot{\theta}_{i,f} \quad (40)$$

Determination of a_0, a_1, a_2, a_3, a_4 and a_5

From the constraints present in equations 33-34 and equations 35-40 it is possible to write the system of equations [25]. In this system of equations, the constraint that gave rise to each equation is marked.

$$\begin{cases} a_0 = \theta_{i,0} \\ a_1 = \dot{\theta}_{i,0} \\ a_2 = \dfrac{\ddot{\theta}_i}{2} \\ a_3 = \dfrac{20\theta_f - 20\theta_0 - (8\dot{\theta}_f + 12\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f){t^2}_f}{2{t^3}_f} \\ a_4 = \dfrac{30\theta_0 - 30\theta_f + (14\dot{\theta}_f + 16\dot{\theta}_0)t_f + (3\ddot{\theta}_0 - 2\ddot{\theta}_f){t^2}_f}{2{t^4}_f} \\ a_5 = \dfrac{12\theta_f - 12\theta_0 - (6\dot{\theta}_f + 6\dot{\theta}_0)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f){t^2}_f}{2{t^5}_f} \end{cases} \quad (41)$$

5th Order Polynomial Profile - Implementation

Similarly to the coefficients calculated for the 3rd order polynomial calculation, the coefficient implied in the constriction of the initial position adopts the value present in the *q_begin* array, while that of the initial velocity adopts the value of 0 and, similarly, the coefficient a_2 of the initial acceleration, as well. Considering the null values given for the coefficients for the initial constraints, one gets $a_3 = \frac{20(\theta_{i,f} - \theta_{i,0})}{2t^3{}_f}$, $a_4 = \frac{30(\theta_{i,0} - \theta_{i,f})}{2t^4{}_f}$ and $a_5 = \frac{12(\theta_{i,f} - \theta_{i,0})}{2t^5{}_f}$.

```
Real a0_1_5th = q_begin[1];
Real a1_1_5th = 0;
Real a2_1_5th = 0;
Real a3_1_5th = if Tempo_Inicial then (20 * (q_end[1] - q_begin[1])/(2*T^3)) else 0;
Real a4_1_5th = if Tempo_Inicial then (30 * (q_begin[1] - q_end[1])/(2*T^4)) else 0;
Real a5_1_5th = if Tempo_Inicial then (12 * (q_end[1] - q_begin[1])/(2*T^5)) else 0;

Real a0_2_5th = q_begin[2];
Real a1_2_5th = 0;
Real a2_2_5th = 0;
Real a3_2_5th = if Tempo_Inicial then (20 * (q_end[2] - q_begin[2])/(2*T^3)) else 0;
Real a4_2_5th = if Tempo_Inicial then (30 * (q_begin[2] - q_end[2])/(2*T^4)) else 0;
Real a5_2_5th = if Tempo_Inicial then (12 * (q_end[2] - q_begin[2])/(2*T^5)) else 0;

Real a0_3_5th = q_begin[3];
Real a1_3_5th = 0;
Real a2_3_5th = 0;
Real a3_3_5th = if Tempo_Inicial then (20 * (q_end[3] - q_begin[3])/(2*T^3)) else 0;
Real a4_3_5th = if Tempo_Inicial then (30 * (q_begin[3] - q_end[3])/(2*T^4)) else 0;
Real a5_3_5th = if Tempo_Inicial then (12 * (q_end[3] - q_begin[3])/(2*T^5)) else 0;

Real a0_4_5th = q_begin[4];
Real a1_4_5th = 0;
Real a2_4_5th = 0;
Real a3_4_5th = if Tempo_Inicial then (20 * (q_end[4] - q_begin[4])/(2*T^3)) else 0;
Real a4_4_5th = if Tempo_Inicial then (30 * (q_begin[4] - q_end[4])/(2*T^4)) else 0;
Real a5_4_5th = if Tempo_Inicial then (12 * (q_end[4] - q_begin[4])/(2*T^5)) else 0;

Real a0_5_5th = q_begin[5];
Real a1_5_5th = 0;
Real a2_5_5th = 0;
Real a3_5_5th = if Tempo_Inicial then (20 * (q_end[5] - q_begin[5])/(2*T^3)) else 0;
Real a4_5_5th = if Tempo_Inicial then (30 * (q_begin[5] - q_end[5])/(2*T^4)) else 0;
Real a5_5_5th = if Tempo_Inicial then (12 * (q_end[5] - q_begin[5])/(2*T^5)) else 0;

Real a0_6_5th = q_begin[6];
Real a1_6_5th = 0;
Real a2_6_5th = 0;
Real a3_6_5th = if Tempo_Inicial then (20 * (q_end[6] - q_begin[6])/(2*T^3)) else 0;
Real a4_6_5th = if Tempo_Inicial then (30 * (q_begin[6] - q_end[6])/(2*T^4)) else 0;
Real a5_6_5th = if Tempo_Inicial then (12 * (q_end[6] - q_begin[6])/(2*T^5)) else 0;
```

Figure 36 - Excerpt from the implementation of a 5th order polynomial kinematic profile, contained in the code involved in the kinematic profile generation model

Once the coefficients are calculated, the next step is to write the equation of motion (equation 33), similar to what is done for the 3rd order polynomial profile.

```
  Real juntal_5th = if Tempo_Inicial and time < endTime then a0_1_5th + a1_1_5th * (time - startTime) + a2_1_5th * (time - startTime) ^ 2 + a3_1_5th *
(time - startTime) ^ 3 + a4_1_5th * (time - startTime) ^ 4 + a5_1_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[1] else
q_end[1];
  Real junta2_5th = if Tempo_Inicial and time < endTime then a0_2_5th + a1_2_5th * (time - startTime) + a2_2_5th * (time - startTime) ^ 2 + a3_2_5th *
(time - startTime) ^ 3 + a4_2_5th * (time - startTime) ^ 4 + a5_2_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[2] else
q_end[2];
  Real junta3_5th = if Tempo_Inicial and time < endTime then a0_3_5th + a1_3_5th * (time - startTime) + a2_3_5th * (time - startTime) ^ 2 + a3_3_5th *
(time - startTime) ^ 3 + a4_3_5th * (time - startTime) ^ 4 + a5_3_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[3] else
q_end[3];
  Real junta4_5th = if Tempo_Inicial and time < endTime then a0_4_5th + a1_4_5th * (time - startTime) + a2_4_5th * (time - startTime) ^ 2 + a3_4_5th *
(time - startTime) ^ 3 + a4_4_5th * (time - startTime) ^ 4 + a5_4_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[4] else
q_end[4];
  Real junta5_5th = if Tempo_Inicial and time < endTime then a0_5_5th + a1_5_5th * (time - startTime) + a2_5_5th * (time - startTime) ^ 2 + a3_5_5th *
(time - startTime) ^ 3 + a4_5_5th * (time - startTime) ^ 4 + a5_5_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[5] else
q_end[5];
  Real junta6_5th = if Tempo_Inicial and time < endTime then a0_6_5th + a1_6_5th * (time - startTime) + a2_6_5th * (time - startTime) ^ 2 + a3_6_5th *
(time - startTime) ^ 3 + a4_6_5th * (time - startTime) ^ 4 + a5_6_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[6] else
q_end[6];
```

Figure 37 - Excerpt from the implementation of a 5th order polynomial kinematic profile, contained in the code involved in the kinematic profile generation model - Continuation

2.1.2.5. Direct Trajectory and Trajectory with Via Points

The *Pick* and *Place* development initially aimed to execute each of the two paths using a direct trajectory, i.e., the movement profile is made based on the initial point where the robot is and the final point where the gripper movement is initialized, and thus make the interaction with the package to be grabbed/laid down.

This first approach is undoubtedly the most simplistic and the one that reduces as much as possible both the size of the trajectory calculations involved and the robot running time to perform the operation properly. However, it is important to emphasize how inadequate this way of execution can be, both for safety reasons, considering the less sequenced trajectory and without slowdown points, and from the logistics point of view, with the possibility of the gripper hitting the packages before positioning itself to tie them from above. Based on these principles, and in the fact that most industrial processes implement a movement trajectory using *Via Points*, there is an attempt to replicate this procedure through programmed logic in object-oriented modeling and thus try to analyze a more realistic performance of the process in question.

The *Via Points* end up being reference points in space, over which the robotic arm passes, before reaching the desired Cartesian point. The *Via Points* adopted end up being points above each of the boxes to be transported, at a point in space slightly above the position where the gripper movement is initialized. A first approximation to the package thus occurs whose Cartesian xy position already obeys the desired final position, varying only on the z axis. The use of *Via Points* could be extended to a larger number of cartesian points, or even to repeatedly passing through the approach point after the package is gripped, however for the purposes of testing the concept, the following sequence of movements is understood to be sufficient.

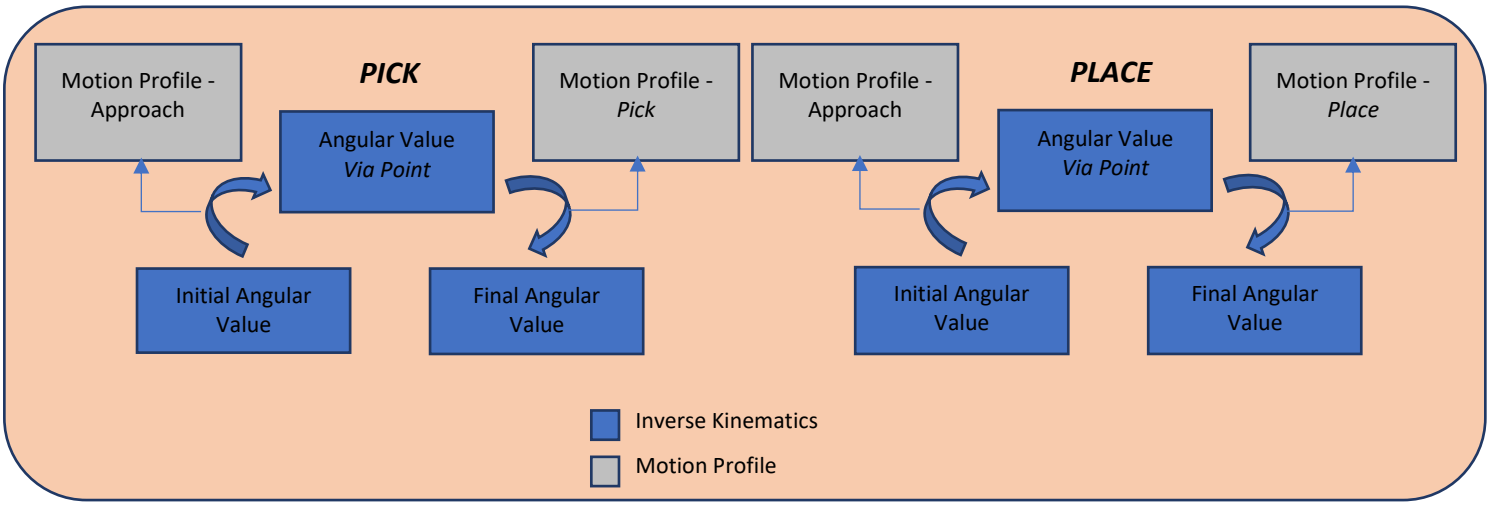


Figure 38 - Motion sequence with *Via Point* mode

The implementation of the *via points* mode of operation involved restructuring the modeling of the inverse kinematics as well as the motion profile. Instead of two inverse kinematic and two motion profile calculations, now there are four for each of the different phases. The changes that this new concept introduces in *Modelica* modeling will be readily explained.

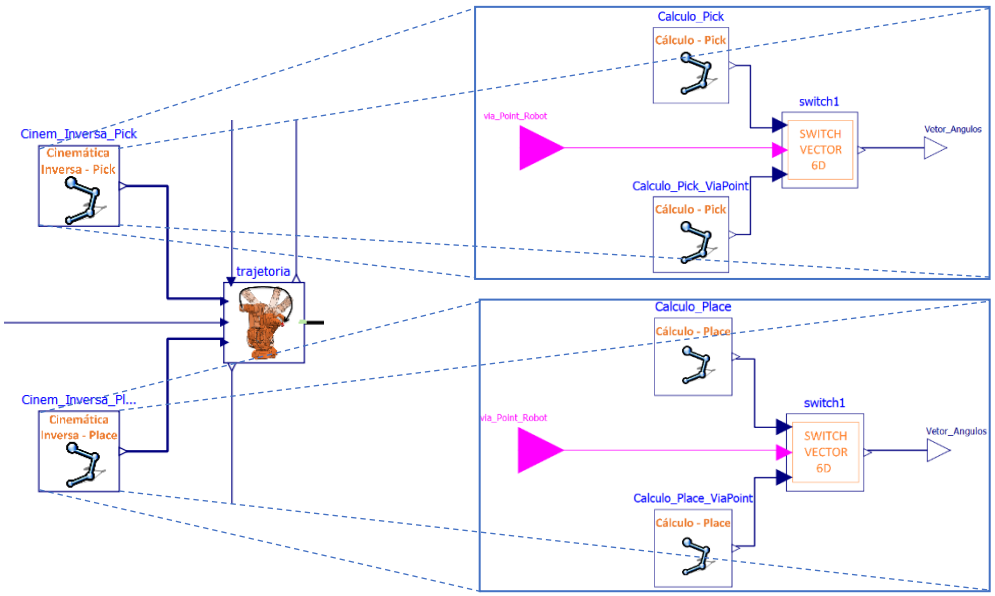


Figure 39 - Restructuring in the modeling of inverse kinematics. Pick and Place model on the left, each in turn consisting of 2 inverse kinematics models (one for the approach and one for the completion of the movement)

Instead of two inverse kinematics models, in which the final angular values of *Pick* and *Place* are calculated directly and then associated with the model in which the motion profile of the trajectory is calculated, a model is conceived in which only inside a calculation model is

presented. Thus, inside the inverse kinematics model for the *Pick*, a diagram consisting of two calculation models is presented, one for the inverse kinematics of the *Pick*'s final Cartesian point if the *Via Point* mode is not activated, and the other for the inverse kinematics of the *Pick*'s approach Cartesian point if the *Via Point* mode is activated. The selection of one of the two comes after a Boolean, whose parameter is directly equaled to the *Via Point* parameter defined in the global system by the user. If the Boolean is true, the model that calculates the inverse kinematics of the *Via Point* transmits the vector of the angles of the six axes via a *switch* model (Figure 39). The *switch* model is developed from scratch with the particularity that, compared to the *switch* of the *Modelica Standard Library*, it is intended for a 6-dimensional array.

Once this rearrangement has been established, in order to put together a model according to the trajectory requirements declared by the user, all that remains is a readjustment in the movement profile, this time a little more complex, and an additional calculation for the inverse kinematics relative to the *Pick* and *Place*, if the *Via Point* mode is activated. The model of the movement profile prepared will be explained next with numbered assignment, in order of operation, to each of the sections of the diagram present in the model.

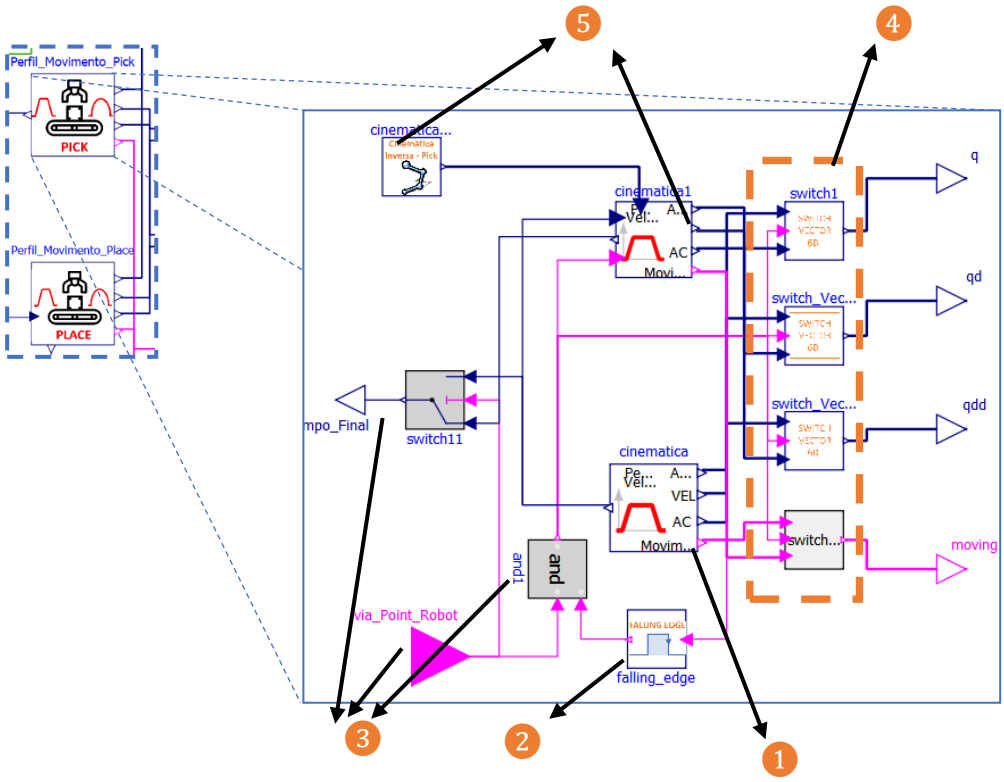


Figure 40 - Restructuring of the motion profile model

1 - First model concerning the calculation of the movement profile Initial Position - *Pick* Position. If the *Via Point* mode is active, the programmed values that enter this model are the angular values of the joints of the initial position as well as of the *Pick*'s approach position. If, on the other hand, this mode is not active, the values of the end joints correspond to the angular variations for the *Pick*'s final Cartesian position.

2 - A model elaborated from scratch, which admits as output a positive boolean, of indeterminate time, after a *falling edge* is checked. In the diagram, this *falling edge* ultimately refers to the cessation of motion by the axis control. In the *via point* mode, the motion cessation concerns the approach to the package, while in the absence of this mode the *Pick* (of the arm) motion completed in its fullness.

3 - A set consisting of a real input that determines the existence or not of the mode *via point*, an *and* condition that sets the output of a boolean depending on the joint existence of the mode and the *falling edge* (i.e., calculation of the first completed motion profile), and a *switch* whose output state is determined by the *and* condition. Thus, in the absence of the mode *via point*, when the motion profile model finishes calculating the variation of position, velocity, and acceleration of the joints, the finishing time leaves the profile model and enters the *switch* in the false state, since the boolean *and* is not true. If on the other hand the *via point* mode is active, the boolean *and* adopts the true state, transmitting this state to the 2nd model of the motion profile, so that the calculation in it is initialized. Under this mode, the final time output will only take place after the calculation in the 2nd motion profile.

4 - In the same way that there is a *switch* responsible for transmitting the calculation end time of the entire *Pick* movement, established either for a direct movement or for a movement with approach through the middle of the path, it is also presented in the rightmost section of the model, a set of *switches*, this time for vectors in 6 dimensions, responsible for transmitting to the model outputs each of the positions, velocities, accelerations and movement states, defined throughout the profile calculation models over time. The boolean *and* is, again, the condition that defines which of the commands will be subsequently sent through the *switches*.

5 - In the event that the *via point* mode is active, a second profile will have to be calculated, representing the movement between the approach reference point and the final cartesian point. For this purpose, the second profile model is given the time instant value at which it

starts, corresponding to the end instant of the 1st profile model, and also the initialization boolean, allowing not only the activation of the *switches* but also triggering the necessary calculation within the profile model. For the 2nd model of the movement profile, there are also the initial angular values that correspond to the final angular values of the 1st model, this equivalence being programmed in the *text view* of the program, and the final angular values are calculated with an inverse kinematics calculation model, also duly programmed, in which the angular values are calculated for the Cartesian point in question, this varying according to the input conveyor chosen by the user.

2.1.2.1. Sum of Trajectories

Once the *Pick* kinematic profile and the *Place* kinematic profile have been calculated individually, the two profiles are added up over time, making a single kinematic profile that can be sent to the control of the different axes.

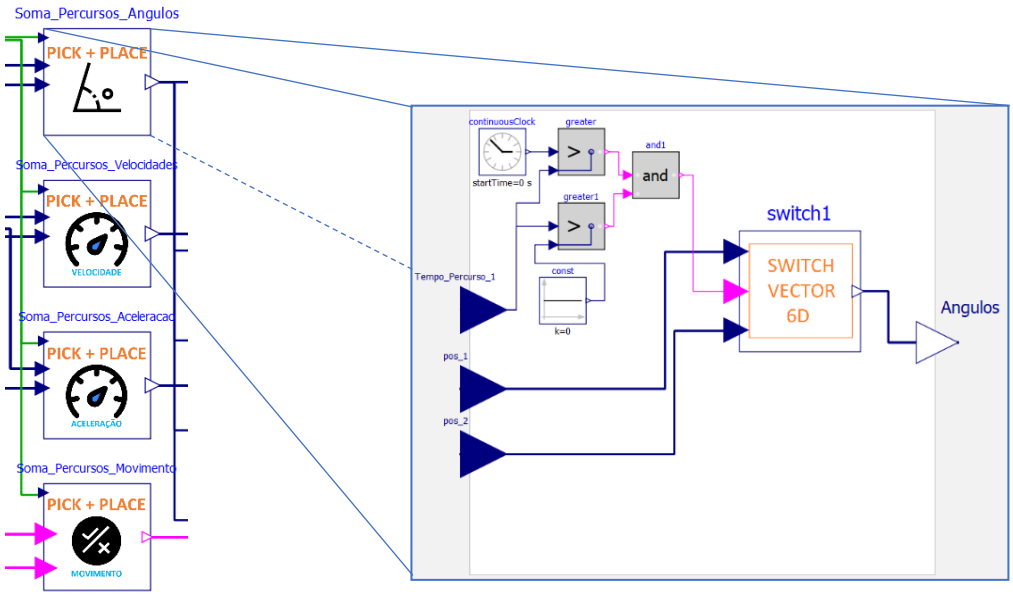


Figure 41 - Models intended to sum the motion profiles over time (*Pick* + *Place*)

The design of this unique profile is done with the help of a final vector calculation model, whose output is the profile vector. This model can be seen in Figure 42.

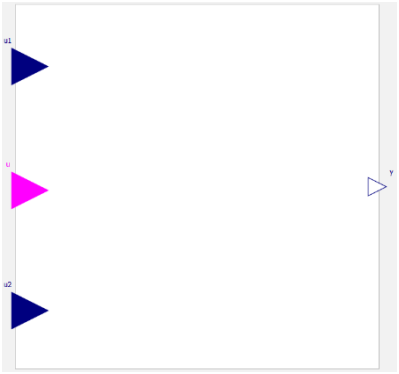


Figure 42 - Model that defines the final vector

The output vector is equivalent to the input vector coming from the *Pick* profile, until a Boolean condition is verified, this Boolean variable being activated when the time instant in the stopwatch (whose value is started at the beginning of the simulation) is greater than the time instant entering the model. The value of the time instant entering the model is equivalent to the end time of the kinematic profile *Pick*. After the Boolean condition is checked the output vector becomes equivalent to the vector coming from the kinematic profile *Place*. There is thus an "exchange" of profiles over time, so that at the block's output the vector is the sum of both.

The models for calculating the final vectors of position, velocity, acceleration, and motion all have the same structure. There is the exception, however, for the model that depicts whether the robot is moving in each of the axes or not, where a small change is made by using Boolean vectors rather than real vectors at the input of the auxiliary calculation model.

2.1.2.2. Distribution of the final profiles by axis

The output variables of the summation models of the 2 profiles, at the level of position, velocity, acceleration and motion present the following form:

$$q = \begin{bmatrix} q_1 \\ \vdots \\ q_7 \end{bmatrix};\ qd = \begin{bmatrix} qd_1 \\ \vdots \\ qd_7 \end{bmatrix};\ qdd = \begin{bmatrix} qdd_1 \\ \vdots \\ qdd_7 \end{bmatrix};\ moving = \begin{bmatrix} True/False \\ \vdots \\ True/False \end{bmatrix} \quad (42)$$

Each of these vectors are finally distributed according to the axis in question, that is, each of the variables $q[i]$, $qd[i]$, $qdd[i]$ and $moving[i]$ are connected to the respective i axis block.

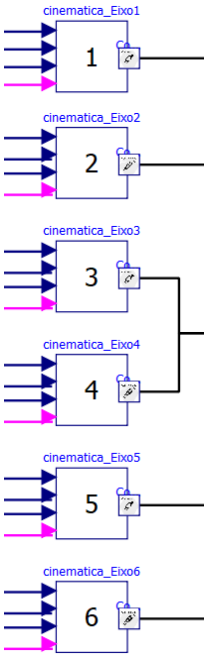


Figure 43 - Models responsible for collecting the different variables intended for each axis command

2.1.3. Mechanical Structure

In order to properly simulate the system, a mechanical structure of the robotic arm performing the Pick and Place task is developed. At the structural level, the initial goal would be to model based on the physical profile of the *ABB IRB 140* robot [26], and in a simulation environment, each of the links modeled in *Modelica* would adopt the actual shape of the different components of the *IRB 140*, in compatibility with the respective components modeled in CAD software. However, the compatibility process that would allow the transcription of the different components failed, largely due to the limitations offered by the open-source program *OpenModelica*. Therefore, an effort is made to model each of the robot's links, from scratch, in *OpenModelica*, based exclusively on the geometric shapes provided by the program (cylinders and parallelepipeds). The result is shown in Figure 44.

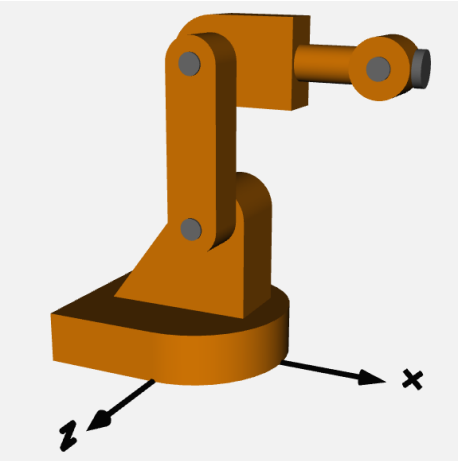
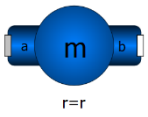
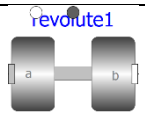
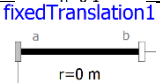
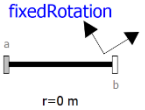
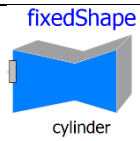


Figure 44 - Simulation representation of the modeled mechanical structure

The total structure is obtained through a model consisting essentially of "*bodyshape's*" that depict each of the linkages or structural components. In these, the mass, center of mass and shape (height, width and length) are defined, using, fundamentally, vectors. Between each of the links, rotating joints or "*revolute's*" are modeled, constituting the joints whose axes will be respectively driven by the servomotors. The fixing of the joints at a specific location in the link is done using the "*fixedTranslation*" or "*fixedRotation*" components, which host the vector between the end of the link and the beginning of the joint, and vice versa. To complement the physical shape of the model, physical shapes are also added through "*fixedShape's*".

Table 5 - Components used for the design of the mechanical structure

Component	Illustration
BodyShape	
Revolute	
FixedTranslation	
FixedRotation	

FixedShape	

Each of the axes, coming from the modeling section intended for axial control, enters the structure model and connects to the *revolutes*, thereby actuating the respective joints, whenever required.

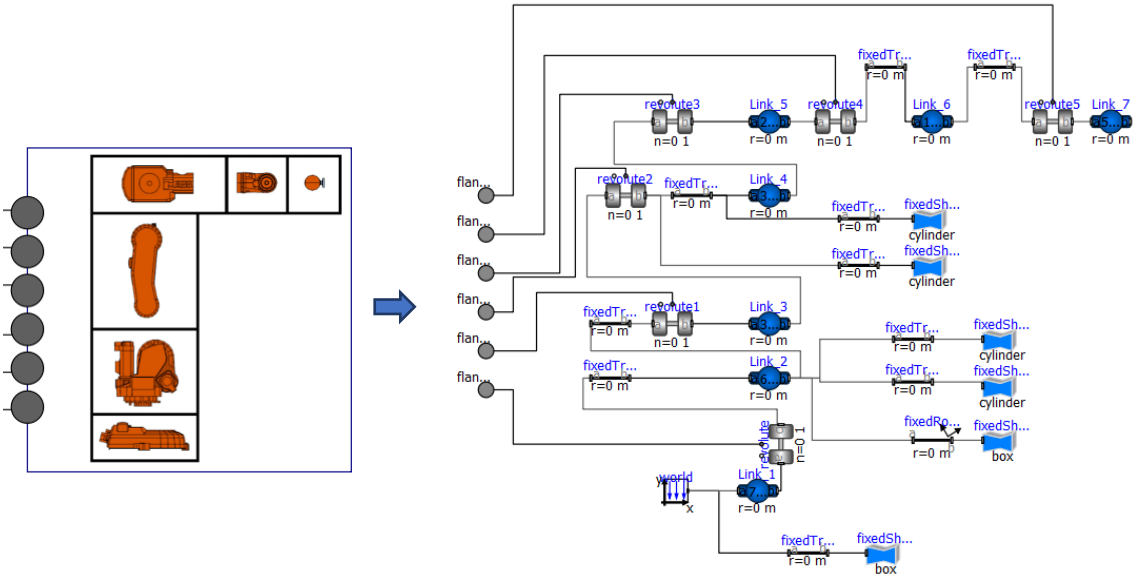


Figure 45 - Model of the mechanical structure of the manipulator

2.1.3.1. Robot – Gripper Connection

An attempted robot - Gripper connection was explored with some exhaustion but successively failed, in which a *frame* connection between the mechanical structure of the robot and the mechanical structure of the 3 constituent fingers of the gripper was modeled. The modeling presented successive errors due to an overloading of the system. Thus, the gripper is simulated independently, from the structural point of view, of the manipulator. However, in order not to compromise the study of the manipulator's performance, a point of mass is added at the end of the manipulator's structure, thus trying to replicate the mass of the gripper's presence along the structure.

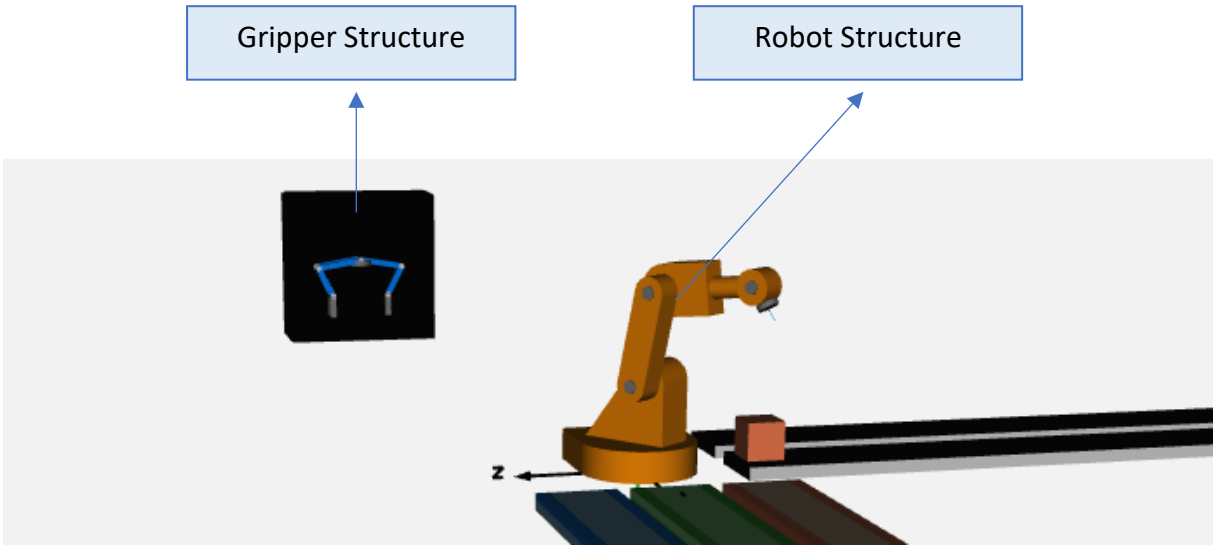


Figure 46 - Gripper structure modeled independent of the manipulator, from a structural point of view

In Figure 47 it can be seen the block corresponding to the additional mass that aims to study, as already mentioned, the replication of the gripper presence at the end of the robot structure.

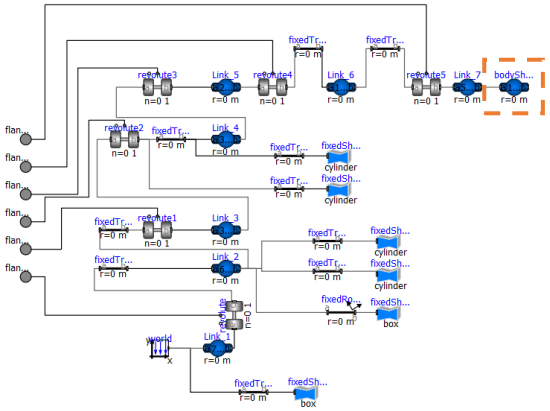


Figure 47 - Attaching the gripper mass to the manipulator frame

It can be concluded that, and as becomes clear from a graphical observation, an increase in mass applied at the end of the structure implies an increase in torque applied along each of the robot's axes.

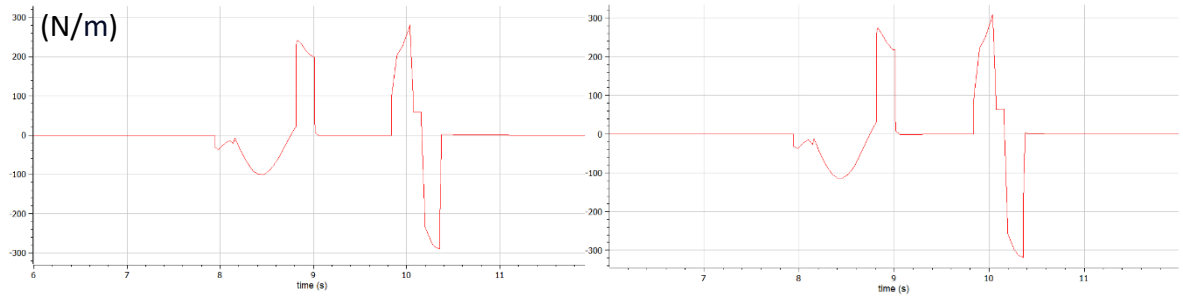


Figure 48 - Torque amplitude over time is increased on each axis. Graph on the left plots torque at 5kg gripper mass. Graph on the right plots torque at 10kg gripper mass

2.1.4. Thermal Flow – Motor Shafts

In order to try to replicate possible heat losses throughout the operation of the robot, a heat transfer model between an input *port* and the exterior is developed. The heat transfer to the outside occurs under convective effect, and can be represented by the following formula.

$$\dot{Q} = h \times A \times dT \tag{43}$$

Where $\dot{Q}$ corresponds to the heat transfer $[W]$, h and A correspond respectively to the heat transfer coefficient $[W/m^2°C]$ and the total area $[m^2]$ on which the transfer occurs, and finally the temperature difference between the 2 environments [°C].

The model developed aims to study the heat transfer by convection, from the surface under study, with its own heat capacity, to the outside that admits a constant temperature over time. In the project under study, this model is used to study the thermal losses in the motor(s), as well as the friction losses along the component designed for this purpose, which is located in the transmission section.

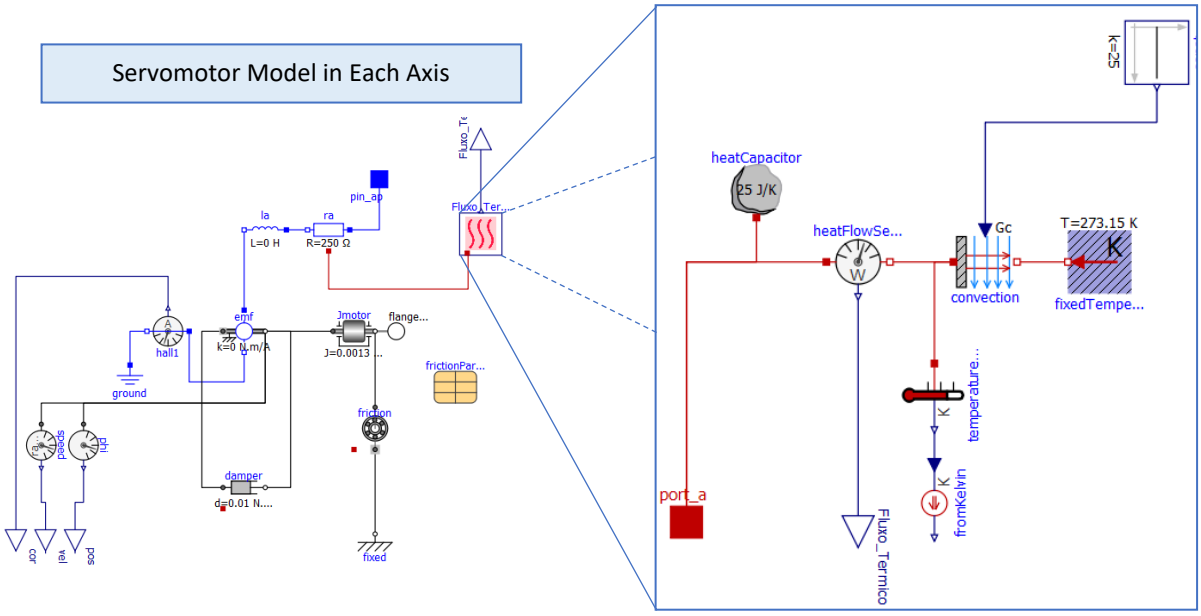
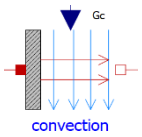
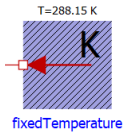
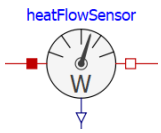


Figure 49 - Thermal flow model associated with each axis

The possibility of using a representative transfer component in the form of conduction, before the convection modeling, would possibly replicate a more realistic situation of the process to be modeled here, however, the fact that the conduction transfer component has a standard temperature (20 degrees) different from the initial ambient temperature to be considered for the bearing and motor (15 degrees), implies an initial heat transfer to bridge this temperature difference along the transfer section. The heat transfer as a function of the system operation would be barely noticeable, as there would be another factor for it to occur.

In the following table 6 it will be possible to observe how the concepts discussed, convective transfer, as well as heat capacity, are incorporated into the modeling presented in Figure 49. For this purpose a set of components are used and whose respective description will be readily presented.

Table 6 - Components present in the heat flow model

Component	Description
	Component that shows the heat capacity C, that is, the amount of heat supplied so that the temperature rises one unit, considering a constant temperature of the body throughout its volume. The heat capacity is obtained by multiplying the specific heat c_p by the mass of the body m. In modeling using this component, the establishment of C is a directly defined parameter, so there must be inherent knowledge of the body to be studied, such as the material of which the body is made as well as its mass and/or volume. $$C = m \times c_p$$
	Component that defines the convective state of the transfer, specifically the heat transfer coefficient and the total transfer area, as shown in the equation. For this purpose the parameter Gc is defined in which it is determined based on the multiplication of the area by the coefficient. $$Gc = h \times A$$
	Component that defines the outside temperature and whose difference to the body generates the thermal potential necessary for the transfer to take place.
	Component responsible for measuring the heat transferred between the object of study and the outside. It is connected to the model's output.

For the situation of the bearing, it can be seen below the temperature difference dT between it and the ambient temperature (which adopts the value of 15 degrees). When throughout the process this temperature difference increases, it is possible to observe that this increase is also accompanied by an increase in energy transfer in the form of heat, whose values are obtained by multiplying the heat capacity of the bearing.

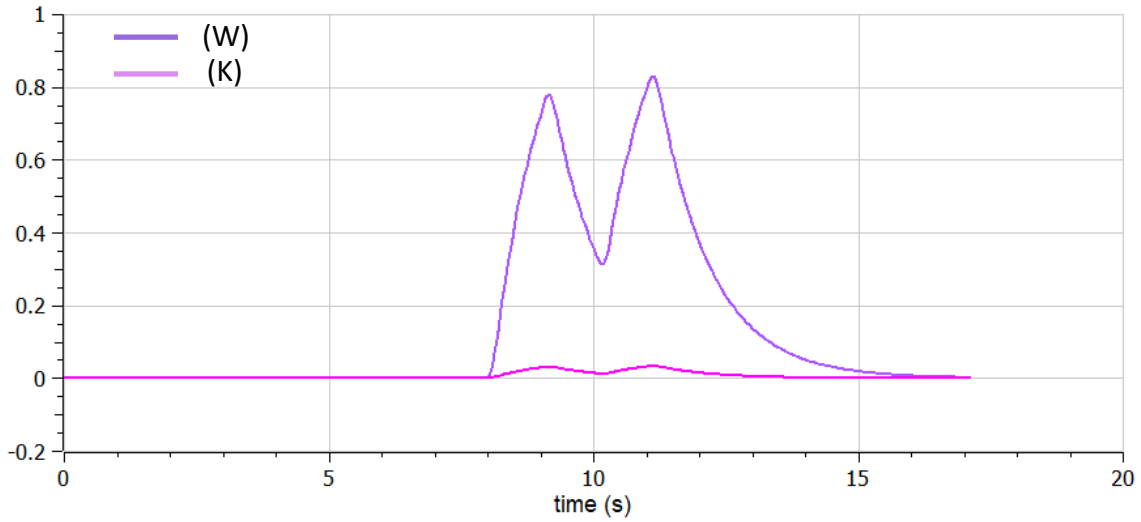


Figure 50 - Temperature variation and consequent increase in energy transfer

In an initial phase, when the bearing is at rest, heat transfer is zero, since the bearing, before being driven, has its temperature equalized to the ambient temperature. When the *Pick* movement is initiated and thus the movement of the bearing, the heat increase, due to friction, raises the heat transfer until the arm movement ceases and the gripper operating stage is initiated, resulting in a drop in transfer given the stabilization in temperature differences. When the *Place* motion is initiated, there is an increase in transfer again and the process repeats. Note the fact, that for this particular situation, although the *Place* movement is of shorter duration than the *Pick*, this is where the peak of heat transfer is reached. This is explained by the fact, that the increase in transference at the time of the *Place* is preceded by a state of the bearing in which, although stopped, it still presents a temperature deviation due to the performance in the *Pick*. There is thus a need for a considerable pause in the operation of the manipulator if the temperature values are to return to ambient values.

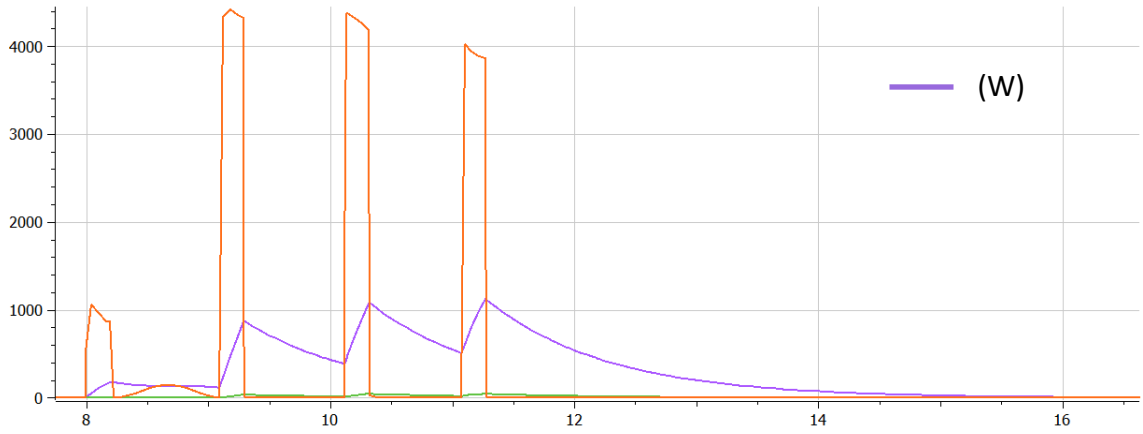


Figure 51 - Heat transfer (purple) from the motor (axis 1)

Regarding the thermal transfer that comes from the motor, it can be observed that it occurs in the moments of acceleration and deceleration of the motor (in orange). During moments of acceleration, the inflow of current and voltage increases. Similarly, during deceleration moments, the current value increases again, this time in the opposite direction. At current peaks, variations in temperature values are naturally observed, which in turn trigger a convective heat transfer.

The heat transfer losses in each axis can be seen in the following graph.

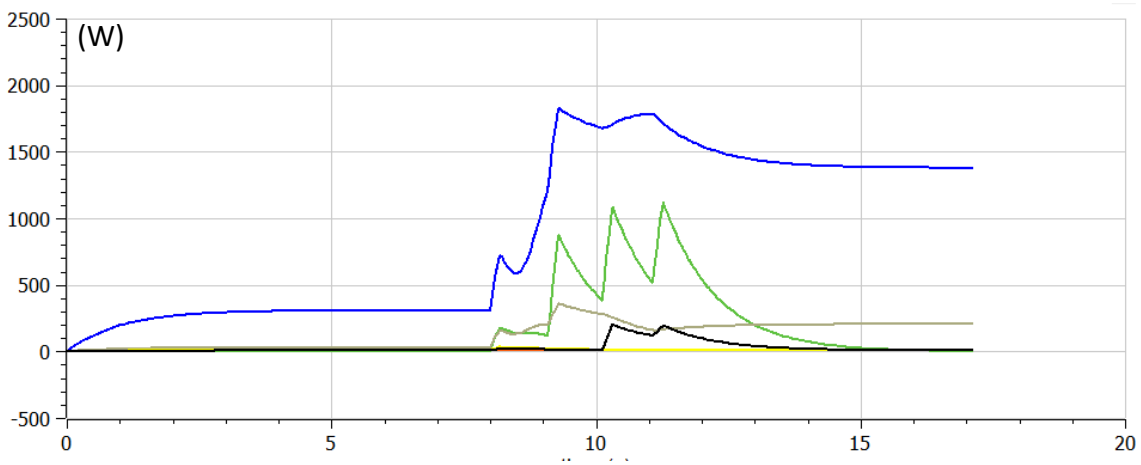


Figure 52 - Heat transfers (W) involved in the 6 axes

The thermal losses are calculated individually in each of the axes, however, in order to make it easier to consult them, a model is designed with the purpose of channeling the different values and presenting them in a single model with dynamic indicators.

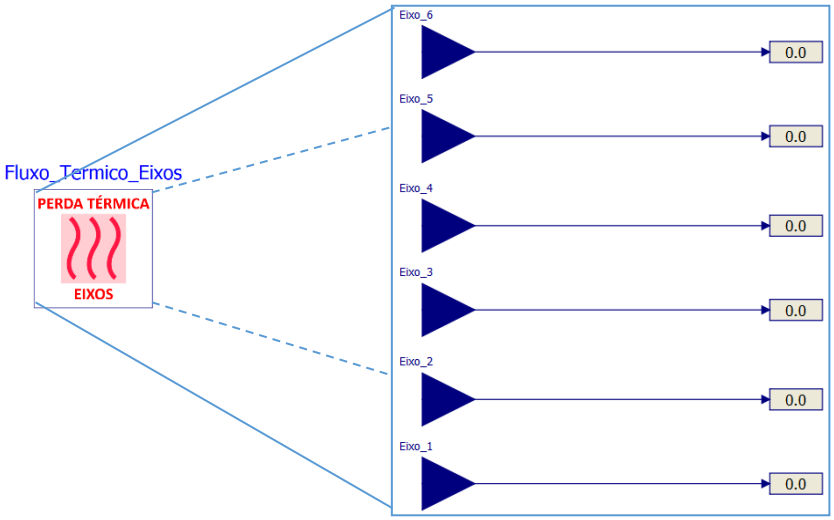


Figure 53 - Model presenting the thermal losses in all axes

2.1.5. Continuation of the Simulations Study – Robot

Some results derived from the simulations and arising from the manipulator are presented here.

Regarding the variation of the angles associated with each of the manipulator's axes, it is observed in the graph that axes 1, 3 and 5 have an angular variation in *Pick* and a variation in the opposite direction along the *Place*. Axis 4 is not driven along *Pick&Place* since the configurations required for the total path do not require it. Axis 6 presents an exclusive variation in the *Place* movement and axis 2 has the particularity of being driven twice in the same direction.

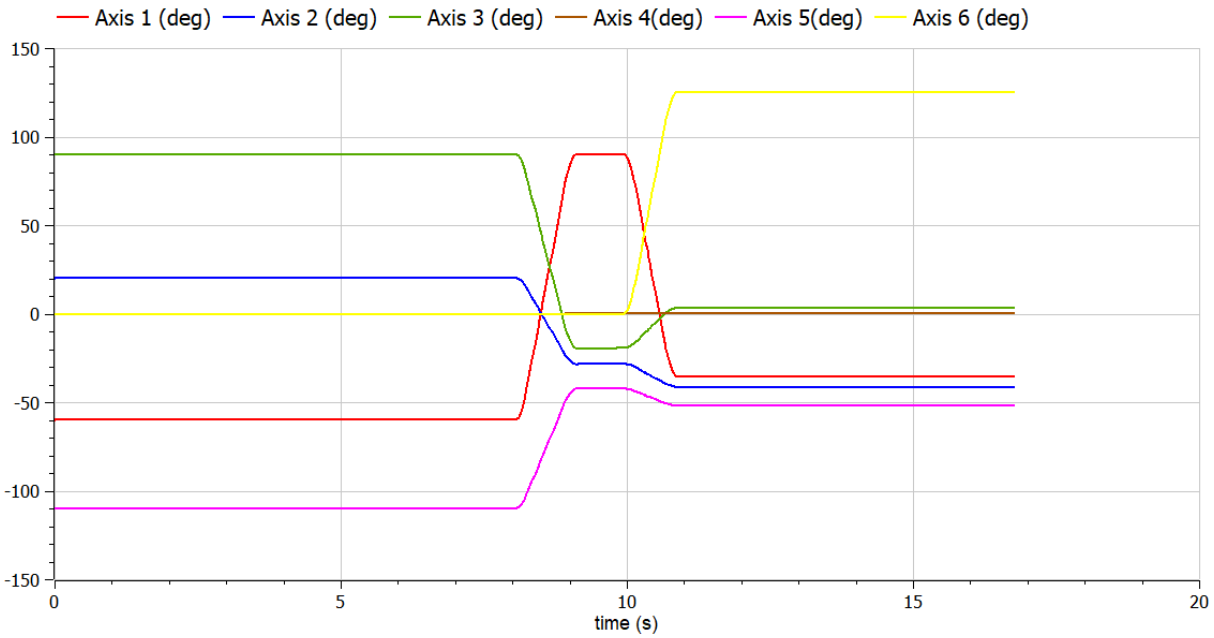


Figure 54 - Angular variation in all 6 axes, with trapezoidal velocity profile and no Via Point mode - Entry Conveyor 2 and Box 1 transport

The trapezoidal velocity profile can be observed (Figure 55). The time required for each trapezoidal profile performed is equal, as noted above, allowing each of the profiles to start and end at the same times. This required trajectory time equals the time from the axis that takes the longest to complete the trapezoidal profile taking into account the angular displacement for that axis and also the axial kinematic constraints involved. Only in this way is it guaranteed that all axes are able to complete the trapezoidal profile in the same time interval. The direct consequence of this imposition is that only one axis reaches the maximum speed (the axis whose ratio between angular displacement and maximum speed defined is greater), and the remaining axes have a peak speed lower than the maximum speed defined

individually. For the situation presented in Figure 55, only axis 1 adopts its maximum velocity defined whose value is 3 rad/s.

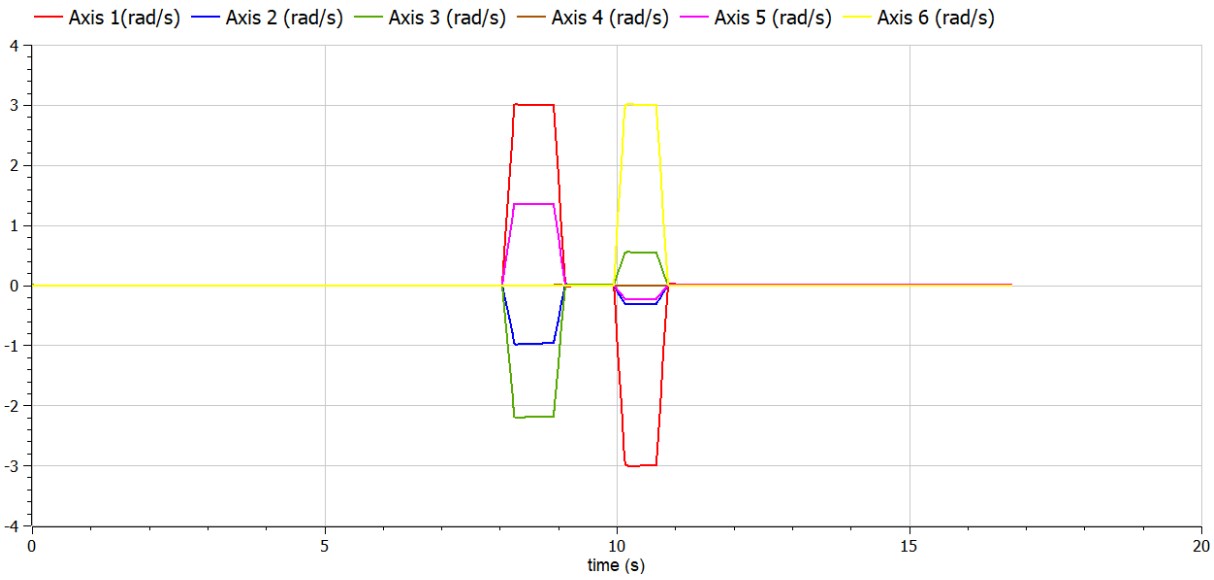


Figure 55 – Trapezoidal gear profile on all 6 axles, without *Via Point* mode - Entry Conveyor 2 and Box 1 Transport

In Figure 56 it's possible to see the 3 velocity profiles calculated for axis 1 of the manipulator. The trapezoidal velocity profile shown in the graph is extracted from the position sensors present on the axis (i.e. it is the implemented profile) while the other 2 profiles are calculated without implementation through the trajectory generation model.

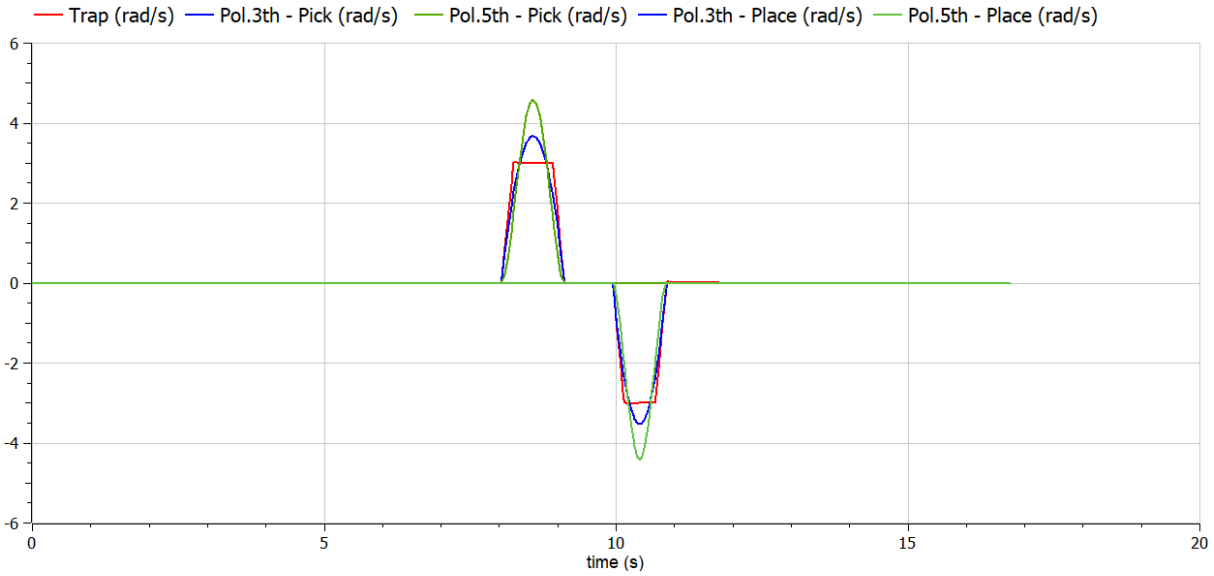


Figure 56 - Trapezoidal, polynomial 3rd order and polynomial 5th order velocity profile on axis 1

Regarding the *Via Point* mode, can be concluded in the graph of Figure 57, that only 3 axes (axis 2, 3 and 5) are involved in an angular motion of approach and then an angular motion of collection/deposition, ensuring a "double" trapezoidal profile both in *Pick* and *Place*. On the

other hand, axis 1 and 6 have their angular motion exclusively involved in approach in both *Pick* and *Place*, without the presentation of a "double" trapezoidal profile.

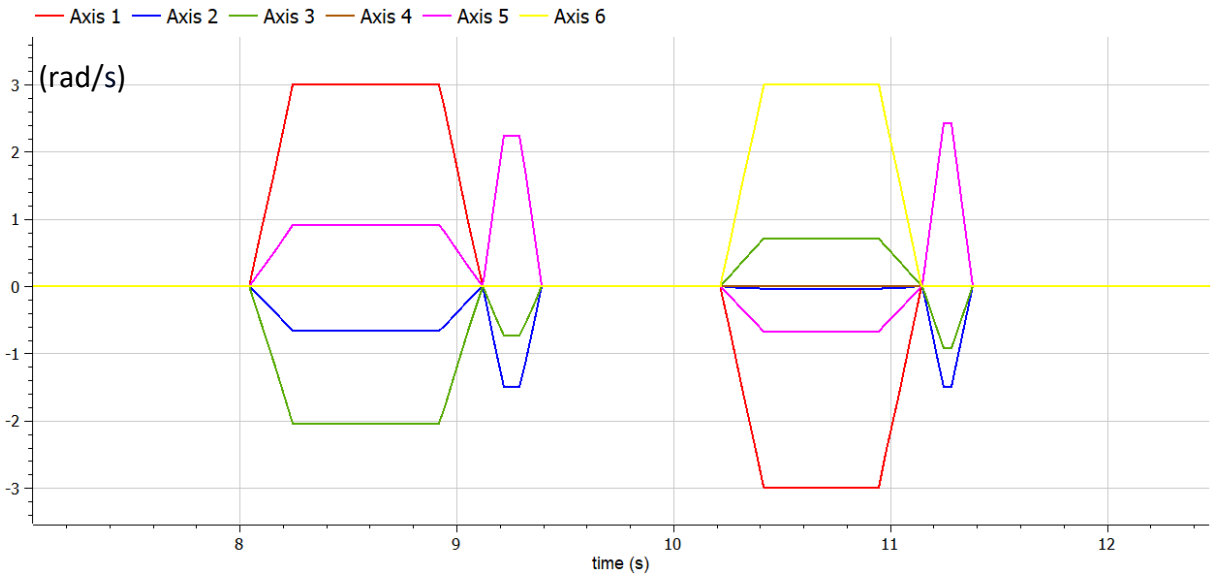


Figure 57 - Trapezoidal speed profile in all 6 axes, with *Via Point* mode

Since in the phase between the approach and collection/deposition point only 3 axes start angular motion, the axis that reaches its maximum velocity set changes (axis 2 instead of axis 1, reaching a maximum velocity of 1.5 rad/s).

As for the impact of the motion profile on performance, the order obtained from lowest to highest thermal expenditure throughout the operation is Polynomial 3rd Order → Trapezoidal → Polynomial 5th Order. The fact that the 3rd order polynomial has less thermal expenditure than the trapezoidal is explained by the smoother movement of the motor. On the other hand, a greater expense than the trapezoidal, by the 5th order polynomial, comes from the fact that it has higher accelerations, which necessarily translates into higher electrical and thermal expense. Thus, when choosing a motion profile from the point of view of thermal expenditure, the choice should fall on a polynomial profile in which a balance between motion smoothing and accelerations is verified.

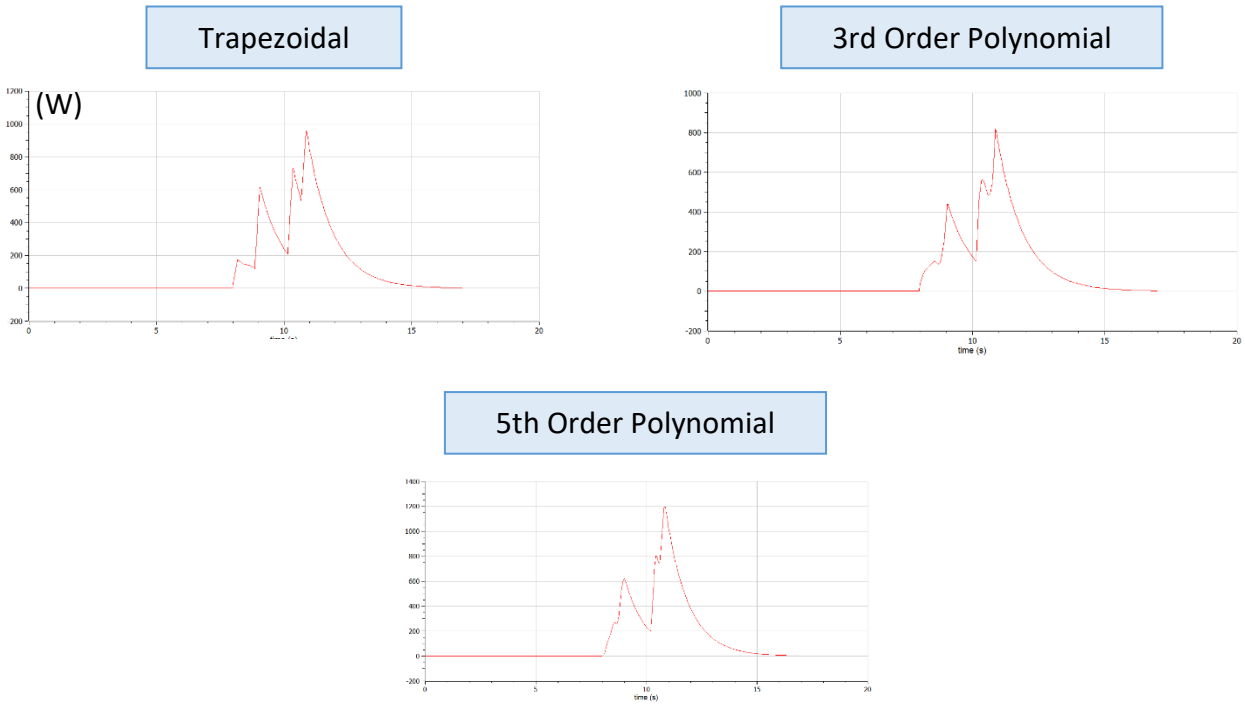


Figure 58 - Thermal losses (watts) along the 1st axis, as a function of the chosen movement profile

2.2.Conveyor Belts

For the different conveyor belts used throughout the process, the type of conveyor chosen is the conveyor belt. The different belts present in the system have, as a whole, a modeling structure very close to each other.

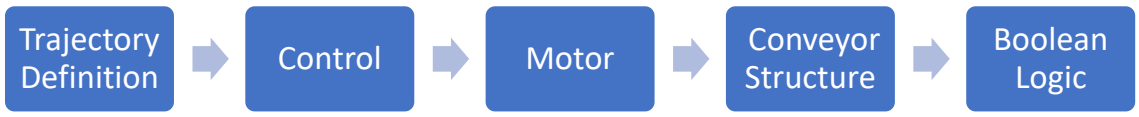


Figure 59 - Base modeling structure, for each of the conveyor belts

The relevance of developing models for conveyors with divergent tasks allowed, without losing focus on the basic modeling structure, to refine each of the phases individually whenever necessary. The different phases integrated into the conveyor belt modeling framework will be described.

2.2.1. Trajectory Definition

It is important to note that the motion path of the conveyor belts, over time, is designed in such a way as to carry the box through only one operating cycle. Otherwise, the interpretation

and modeling of the conveyor operation would most likely be different. It is throughout this phase that the reference position/speed, referring to the trajectory of the box along the conveyor, are defined, and these differ depending on the specific task performed by the transport. Thus, two ways are considered for trajectory generation: generation of a "trapezoidal" trajectory, where the start, uniform linear motion and stop times are incorporated, conditioned by the desired kinematic constraints; and generation of a trajectory with start and uniform linear motion designed by pulses or *step* signals. The latter being a manifestation of the attempt to replicate deposition belts at the exit of the system, where the stop in the transport movement is not verified.

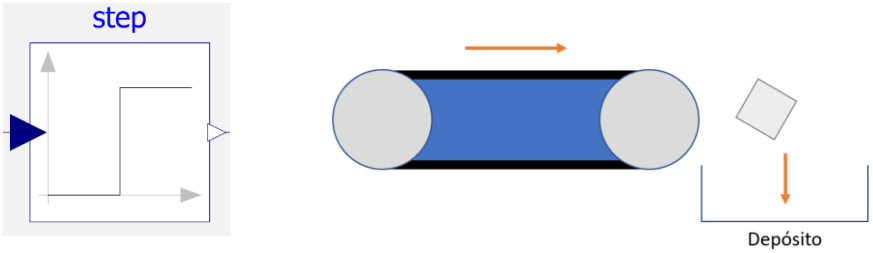


Figure 60 – Exit conveyor belt 1/2 that aims to deposit the packaging

The fact that we are only focusing on one operation cycle, and that the need to place the package in a specific location of the conveyor does not exist, means that a deceleration or stop stage is disregarded in the definition of the trajectory, thus triggering a uniform linear movement of the conveyor which is exclusively extinguished when the simulation is interrupted.

On the other hand, in the exploration of trapezoidal trajectories, two types of trajectories are considered: trajectory with single velocity trapezoidal profile; and also trajectory with cyclic velocity trapezoidal profile.

The conveyor belts at the entrance collect a package and transport it to the collection point made by the robotic arm, thus existing a start and end point where the absolute velocity is zero and a trapezoidal velocity profile defined between those 2 points, thus constituting a single trapezoidal profile. On the other hand, in one of the exit belts, it can be seen the existence of different operation cells along the package transport, thus resulting in a cyclic trapezoidal speed path, in which successive stops of the package are verified.

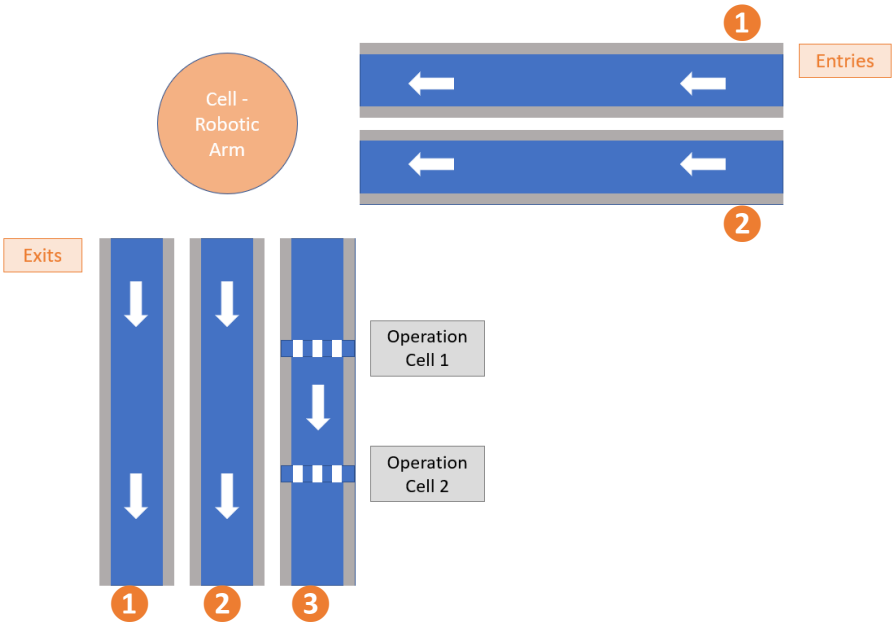
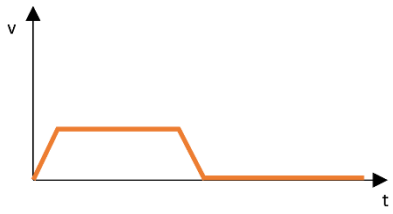
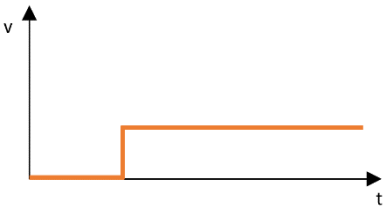
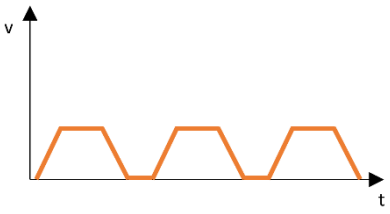


Figure 61 - Arrangement of the conveyor belts, with a last exit belt that features stops throughout an operation cycle

For the different conveyors in the system visualized in Figure 61, the speed profiles are shown in the table.

Table 7 - Velocity profiles on the different conveyor belts

Conveyor Belt	Velocity Profile
Entry Conveyor 1 and 2	
Exit conveyor 1 e 2	
Exit conveyor 3	

In the case of the single trapezoidal profile, the model elaborated resorts to the algorithm used in trajectory elaboration on the robotic arm. Once the kinematic constraints inherent to the trapezoidal profile have been defined, such as initial position, final position, maximum speed, and maximum acceleration, the position and speed profile is generated.

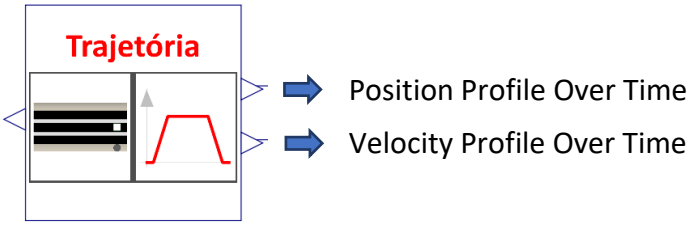


Figure 62 - Model used to generate the single trapezoidal profile

For the cyclic trapezoidal profile, the trajectory model used was developed using the previous model.

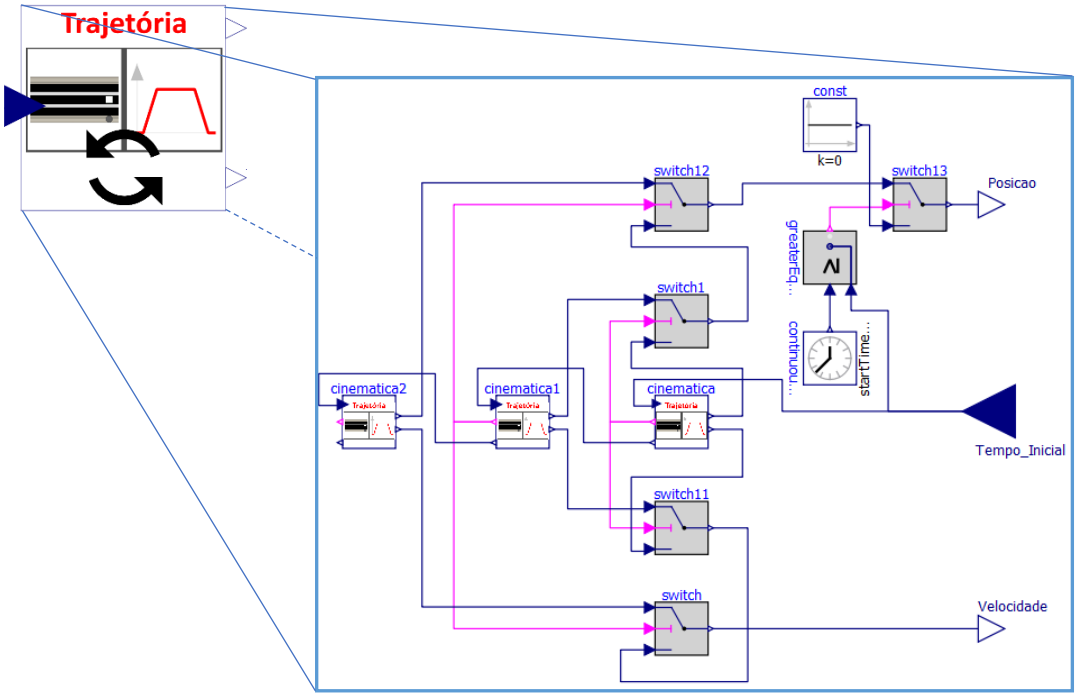


Figure 63 - Model used to generate the cyclic trapezoidal profile

Before going on to explain the diagram inside the model, it is important to clarify that the final speed profile consists of 3 trapezoidal pulses, one between the start point, where the box is dropped, and the point where the first operation is performed on the box. A second pulse between the two operation points, and a third between the second operation point and the

end point of the conveyor, where the box must admit the 0 speed, in order to be processed in a further process.

In the diagram presented, the modeling of the velocity and position profile, which can be divided into three as mentioned, allows the use of the trapezoidal profile model already developed, subject only to minor changes, specifically the input and output variables that take place in each of the trajectory models. Thus, a sequencing is developed through the transmission of initiation signals through Booleans, and the sending of timing values, i.e., the time instants in which the respective profiles should be calculated.

Figure 64 shows the control cycle of the conveyor's movement, thus constituting the final architecture of the model of the different conveyors. Regarding the trajectory generation, and as previously mentioned, the input conveyors and output conveyor 3 generate position and velocity profiles (the presence of a trapezoidal velocity profile in the transport is more easily obtained according to a cascaded position-speed control) and in the output conveyors 1 and 2 velocity profiles are generated exclusively, due to a less judicious control need, as a consequence of the absence of a slowdown at the end of the movement.

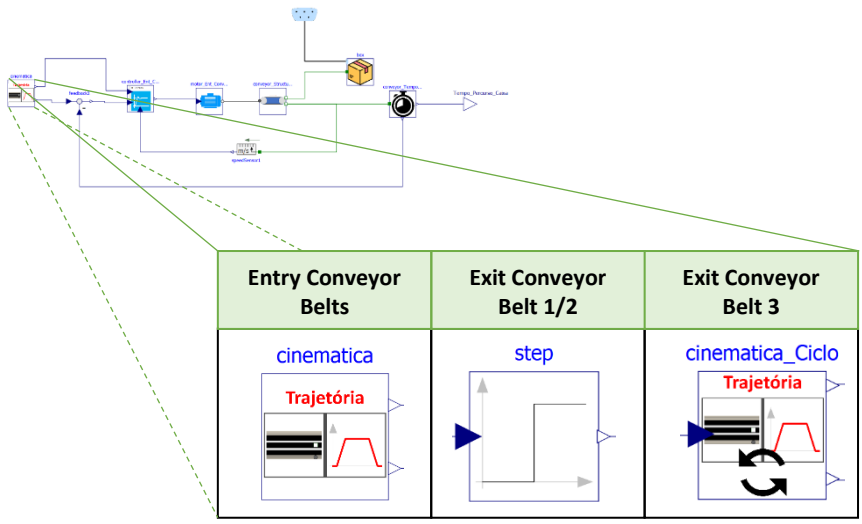


Figure 64 - Control cycle present in the conveyor models and the options present for generating the movement profile

2.2.2. Control and Motor

Similar to the servomotor developed in the manipulator model, the modeling of the conveyor motor has the same format, with the only distinction being how the current is transmitted to the electrical circuit. In this, a *SignalVoltage* block is used giving input directly into the circuit. The voltage along the developed closed circuit is thus mediated by this block, which transmits

a continuous function of potential difference over time. This voltage, in turn, ends up being the command provided by the P-PI control.

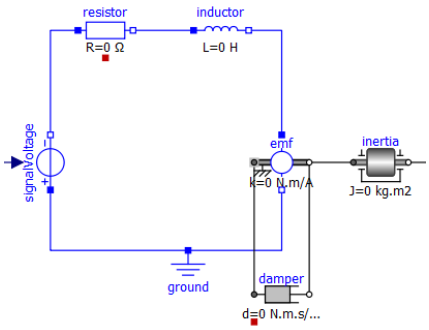


Figure 65 - Modeling of the motor used in the conveyor belts

P-PI control is directly associated with trajectory generation, with proportional position control and PI speed control (input conveyors).

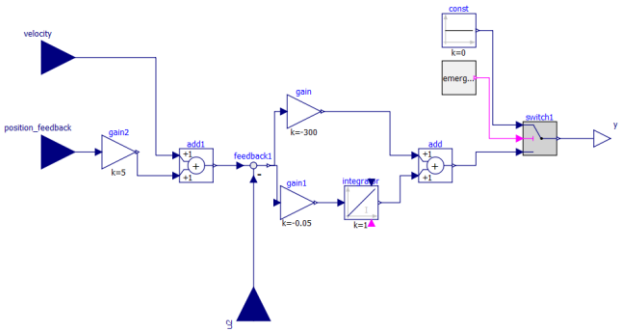


Figure 66 - Motor associated P-PI control on the entry conveyors

In the output conveyors, the control modeling is modified. In the first 2 output conveyors, the absence of a proportional control term for position is noteworthy. The absence of a position control on these 2 conveyors determines a unique PI speed control. Finally, for the control of output conveyor 3, a cascade control that attends to the position, speed and current commands (similar to the control profile inserted in the manipulator's motors) is implemented.

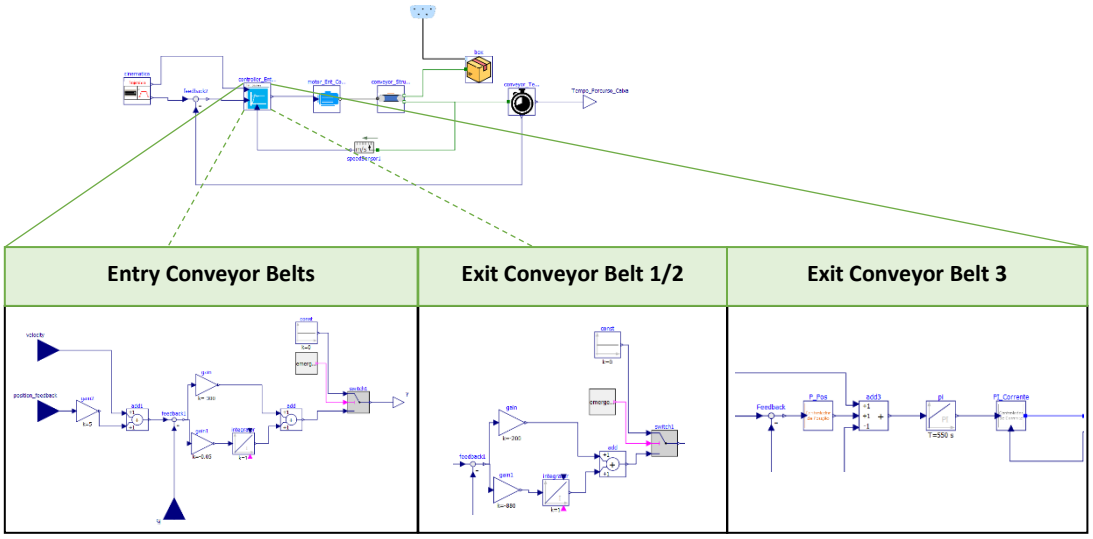


Figure 67 - Control cycle present in the models of the conveyor belts and the options present for the controller

There is, therefore, an increase in the impositions inherent to the control cycles, meeting the needs required by each task performed by each conveyor belt.

2.2.3. Conveyor Belt Structure

Starting by analyzing the conveyor model itself, that is, the entire conjuncture inherent to the operability of the mechanical load. The mechanical load represents the force required to move successive packages along the conveyor in an industrial environment. Modeling the transport of the mechanical load involves dividing the conveyor system into different competencies, with the initial approach consisting of the gears, the inertia of the system, damping, the transmission that translates the linearity of the movement, and finally the mass to be transported, being this initial approach based on the schematization present in [27].

For the modeling of the structure and mechanical load, two distinct models are also developed, one more simplified and the other with greater complexity, i.e., with a larger number of parameters and variables involved, thus determining a more rigorous study of the system. The set of different components used will be discussed in more detail below.

It is important to realize that for the motor to accelerate or decelerate the conveyor, the motor has to overcome not only the inertia of the mass being transported, but also the conjectural inertia of the entire system.

Simplification of the system could lead us to a system consisting of a conveyor geared to two cylinders/pulleys. These pulleys in turn can be classified according to their action in the

system. Motor or drive pulleys are those responsible for transmitting force. On the other hand, the pulley that receives the movement is called the driven or conducting pulley.

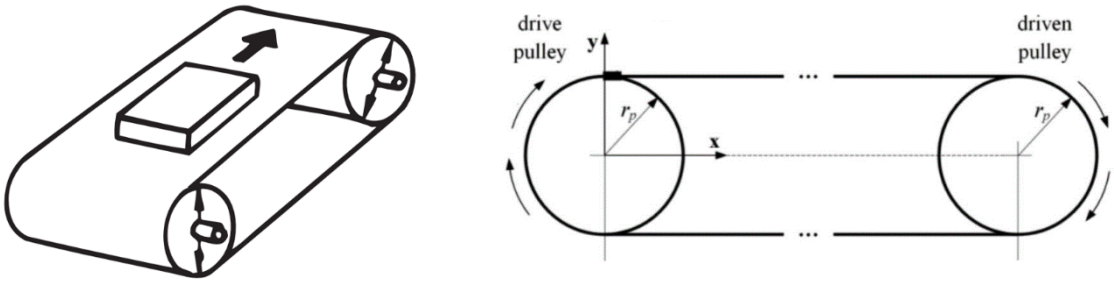


Figure 68 - Conveyor belt with associated rollers/pulleys

The drive pulley, driven by the motor shaft, triggers the movement of the conveyor and, attached between it and the motor shaft, a set of highly efficient gears is introduced, increasing the output torque and decreasing the speed, of the motor output, to be transmitted.

Regarding inertia/moment of inertia, this is a measure of the distribution of mass of a body around an axis of rotation, which for the modeled system is the drive shaft. The moment of inertia evaluates the difficulty of the drum/cylinder inherent to the transmission to rotate about its axis, and this same difficulty is directly related to the moments of inertia of the load (belt and packages) as well as to the moments of inertia of the transport drums. The total inertia thus depends on several parameters implicit to the type of conveyor present in the system.

The main points of interest already mentioned to be taken into consideration throughout the modeling of a conveyor: gears, system inertia, damping, transmission that translates the linearity of the movement, and the mass to be transported, will serve as a basis for the more simplistic development of the system structure model. Of these, the point that requires the most careful analysis is the total system inertia, involved in the torque required for the movement to occur as desired. Thus, the appropriate formulations will be presented.

2.2.3.1. Calculation of Torque and Total System Inertia

Assuming for example a trapezoidal speed profile (verified on exit conveyor 3), the different torques required throughout the process are torque for acceleration, constant speed and

deceleration torque, and in most applications, the maximum torque occurs during acceleration. Because the maximum value occurs mostly at the beginning, in processes involved with conveyors, a constant torque, in many applications, is provided to the system by the motor, and its calculation is performed by the square root of all the different torques required throughout the process (the use of constant torque will be disregarded in the modeling). The different torques involved in the different phases will be presented.

Torque – Continuous Speed

For a system such as a conveyor, the motor torque required during constant speed is simply the total axial force (F_a) on the conveyor multiplied by the radius of the pulley (r_p).

$$T_c = \frac{F_a \times r_p}{\eta} \tag{44}$$

Where the efficiency η of the system is included in the torque equation. This efficiency relates to the friction losses between the belt and the pulleys/cylinders. It is also important to note that it is assumed that both the drive and driven pulleys have the same diameter, which turns out to be the case for most conveyor belts used in linear motion systems.

Conveyor belts are not normally designed to withstand external axial forces, so the total axial force consists exclusively of the force required to move the load, which is the weight ($m.g$) of the package to be transported as well as the belt, multiplied by the coefficient of friction of the load bearing guide.

$$F_a = m \times g \times \mu \tag{45}$$

Torque – Acceleration

The acceleration phase of the motion profile is typically the period when the maximum torque required by the motor is reached, and this torque (T_a) is often also considered the intermittent torque. The torque required during belt acceleration includes the torque required at constant speed (T_c) plus the torque required for load acceleration (T_{ac}).

$$T_a = T_c + T_{ac} \tag{46}$$

The acceleration torque of the load is obtained by multiplying the total inertia of the system (J_t) by the angular acceleration (α).

$$T_{ac} = J_t \times \alpha \tag{47}$$

The total inertia of the system includes the inertia of the motor, coupling, pulleys/cylinders, and also the load (belt and packing).

$$J_S = J_{P1} + J_{P2} + J_T + J_C + J_M + J_{Acop} \tag{48}$$

Table 8 - Calculation of the different constituent inertias in the inertia of the overall system

Component	Inertia Calculation
Cylinders Inertia	$J_{P1} = \frac{1}{8} \times m_{P1} \times D^2$; $J_{P2} = \frac{1}{8} \times m_{P2} \times D^2$ or $J_{P1} = \frac{1}{2} \times m_{P1} \times r^2$; $J_{P2} = \frac{1}{2} \times m_{P2} \times r^2$
Conveyor Belt Inertia	$J_T = \frac{1}{4} \times m_T \times D^2$ or $J_T = m_T \times r^2$
Box Inertia	$J_C = \frac{1}{4} \times m_c \times D^2$ or $J_C = m_c \times r^2$
Motor Inertia	J_M
Coupling Inertia	J_{Acop}

Torque – Deceleration

$$T_d = T_c - T_{acc} \tag{49}$$

The torque required during deceleration of the conveyor includes the torque required at constant speed minus the torque required for acceleration of the load.

Between the inertia, which represents the resistance of the total load being carried, and the transmission that converts the rotary motion to linear, a damping present in the bearings is modeled. The damping force or torque (T_a) is equal to the multiplication of the damping constant (c_a) and the relative angular velocity (w_r). The energy loss (P) in the bearings is equal to the damping torque multiplied by the relative speed.

$$T_a = c_a \times w_r \tag{50}$$

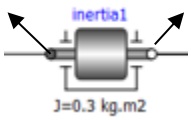
$$P = T_a \times w_r \tag{51}$$

2.2.3.2. Implementation

The motor torque is responsible for the movement of all the mechanical load involved in the belt. In the event of wanting to maintain a constant speed, and disregarding the friction losses between the belt and the pulleys/cylinders, an increase in load drives a "counter-torque" (torque in the opposite direction) on the motor, which triggers the belt speed to decrease, so an increase in motor torque is required so that a counter-balance, between the torque that triggers the movement and the torque that offers resistance to the movement, exists, and thus prevents the speed from decreasing. Thus, when the torque and counter-torque are equal, a stabilization of speed is generated, making the speed constant. When the torque is greater than the counter-torque there is an increase in speed and a consequent acceleration. On the other hand, for a higher counter-torque, a deceleration is verified.

Table 9 - Transmission and inertia component used in the modeling

Component	Description
	Component involved in the calculation of torque at constant speed, using the axial force required to carry the load. The transmission ratio subject to the transformation from rotational to linear motion is taken into account. $T_c = F_a \times r_p$

Flange A  Flange B 	Component that defines the total inertia of the system, also calculating the acceleration torque of the load based on the acceleration allowed on the flanges. $J_S = J_{P1} + J_{P2} + J_T + J_C + J_M + J_{Acop}$ $T_{ac} = J_t \times \alpha$

Acceleration torque T_{ac}, being solely responsible for the speed increase. It corresponds to the motor torque over acceleration T_a minus the constant speed torque T_c (corresponding to the counter-torque T_{ct}) exerted by the mechanical load.

$$\begin{aligned} T_a &= T_{ac} - T_c \\ \leftrightarrow\ T_a &= T_{ac} - T_{ct} \\ \leftrightarrow\ T_{ac} &= T_a + T_{ct}\text{ , with } \|T_a\| > \|T_{ct}\| \end{aligned} \tag{52}$$

The acceleration torque is 0 in the constant speed phase, and in this same phase the motor torque throughout acceleration is set by a value equal but symmetrical to the counter torque.

$$T_{ct} = -T_a \text{ , for } T_{ac} = 0 \tag{53}$$

On the other hand, for the deceleration torque T_{des}, it is exclusively responsible for the decrease in speed. The calculation is formulated in the same way as the acceleration torque, with the particularity that:

$$T_{des} = T_a - T_{ct} \text{ , with } \|T_a\| < \|T_{ct}\| \tag{54}$$

One can consider flange b as the flange where the counter torque is defined, flange a as the flange where the constant velocity torque is defined, and finally the total inertia of the system, introduced in the block, which defines the acceleration torque, together with the acceleration.

```
J*a = flange_a.tau + flange_b.tau;
```

Figure 69 - Code excerpt in the inertia component, in which the equation establishing the torque is defined

Simplified Model

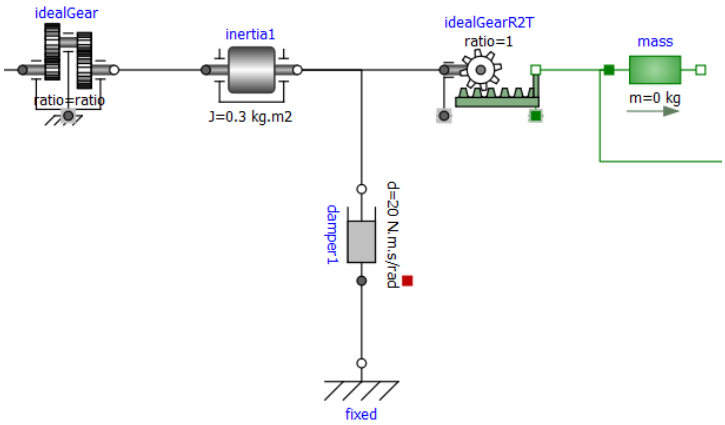


Figure 70 - Simplified modeling of the conveyor structure

The 1-dimensional modeling of the transformation from rotation to translation is achieved through a component/object characterized as an ideal gearbox (without mass and inertia considered), in which the transmission ratio is properly defined, along with the blocks defined in table 9, a block intended for rotary damping, allowing to ensure a smooth and controlled rotary motion, and finally the component intended for the mass of the package to be transported.

Model with considered losses

For the 2nd model, which aimed at a more in-depth study of the development of the mechanical load, it is based on the set of objects already developed, adding some more, thus expanding the total model.

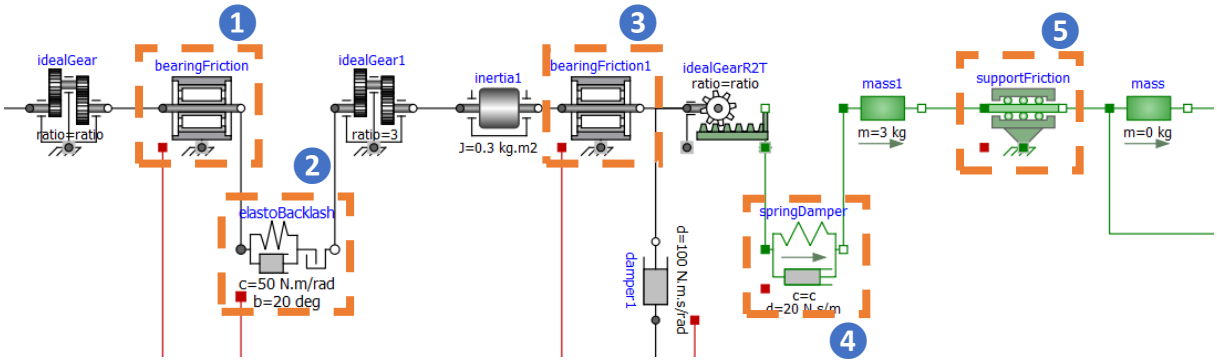
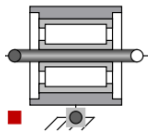
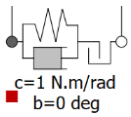
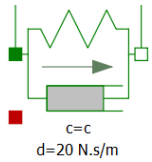
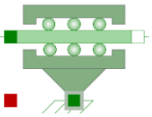


Figure 71 - Modeling with associated losses of the conveyor belt structure

This model admits not only one, but two gear reductions. The additional components focus on verified energy losses in the form of friction, damping, elasticity, and also backlash.

Table 10 - Description of the components associated with energy losses in the conveyor system

Component	Description
	Component that characterizes the Columbus friction between the flange along a present support (e.g. a bearing). Here, the existing friction results from contact between the flange and a bearing surface.
	Component that establishes a series connection between a backlash element and a parallel connection, consisting of a spring and a damper, respectively. In this way, the elasticity, damping, and backlash effects can be extracted for the second gear reduction in the model, thereby being able to model its gearbox more accurately.
	The block for the rotary-linear transmission initially present in the first model of the belt is ideal, and therefore disregards possible losses right away. This block, however, models elasticity and damping present in that transmission, thus ensuring a transmission definition closer to the real one.

	Similar to the rolling model/component of friction, this component also focuses on the Columbus friction, but this time not dependent on the frictional torque and the absolute angular velocity, since it is a translational motion, but rather between the frictional force, acting between the surface and the load, and the absolute translational velocity.

Thus there are in the 1st model, disregarding losses, 2 blocks that define the torque required by the motor. These, however, may be complemented with the blocks reproduced in table 10 and that properly portray the losses involved. These complementary blocks/models will thus define the equations already discussed.

1 - Representative blocks of the final efficiency related to losses between the gears and the belt and pulleys/cylinders.

2 – Block representing the friction coefficient of the load bearing guide.

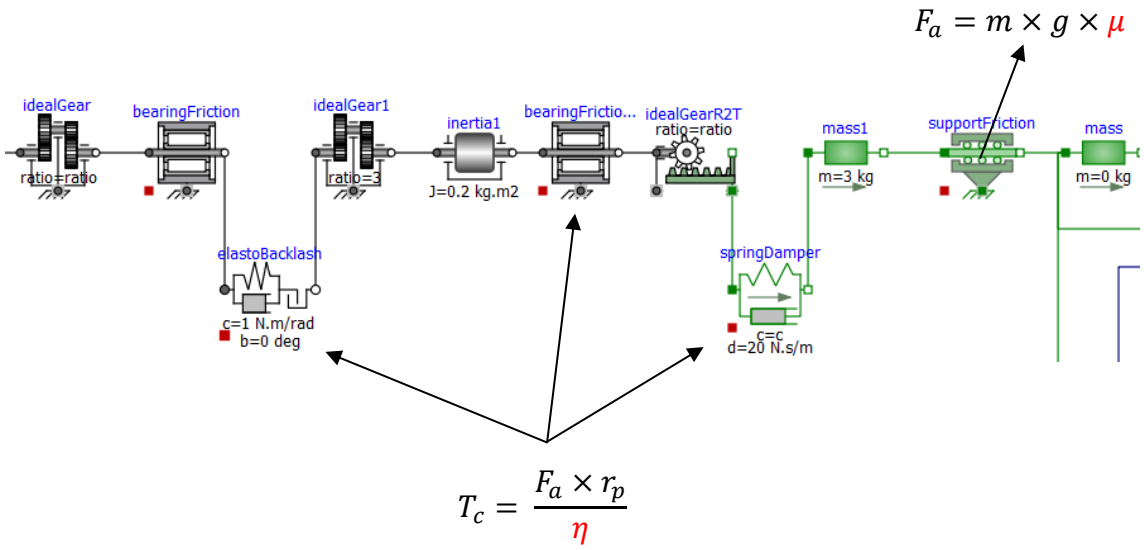


Figure 72 – System efficiency variables (red) present in the equations involved in the required torque and axial force

Given the different modeling of the structure of the input conveyors, they are inserted in the final cycle of the conveyor model, preceded by the motor model and the associated control.

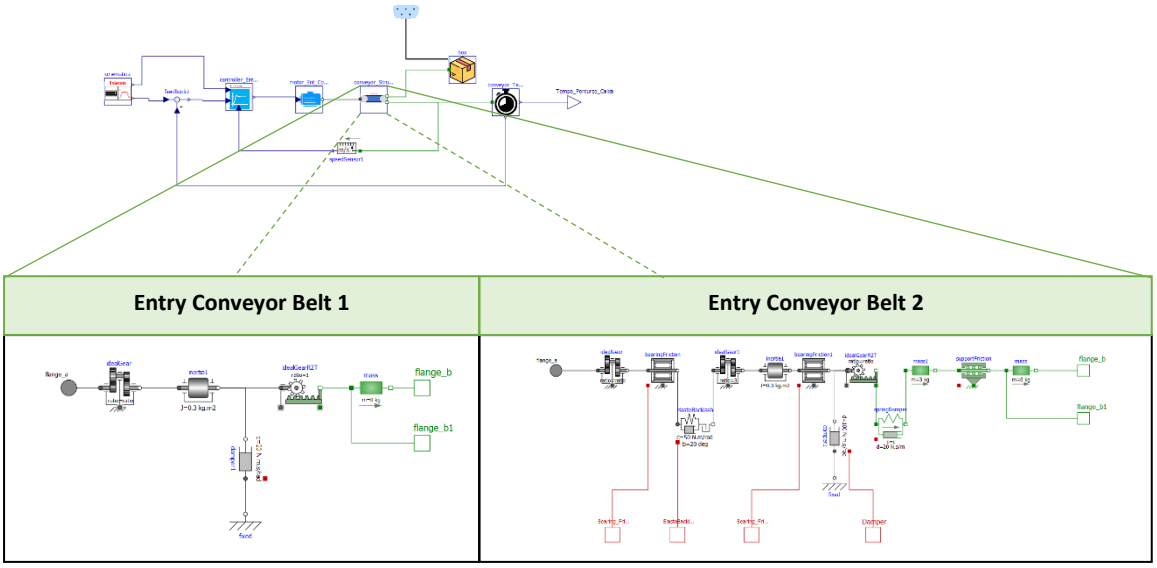


Figure 73 - Control cycle present in the conveyor belt models and the options for the conveyor belt structure

2.2.4. Timing

In order to establish the travel time of the box along the conveyor, a section consisting of logic blocks is elaborated.

A position sensor is connected to the box transported by the conveyor and a feedback is responsible for subtracting the value of the length of the path to be performed, by the box, and its position along the path, measured by the sensor. When this difference equals 0, with a tolerance of 0.01 meters, a *timer* is activated.

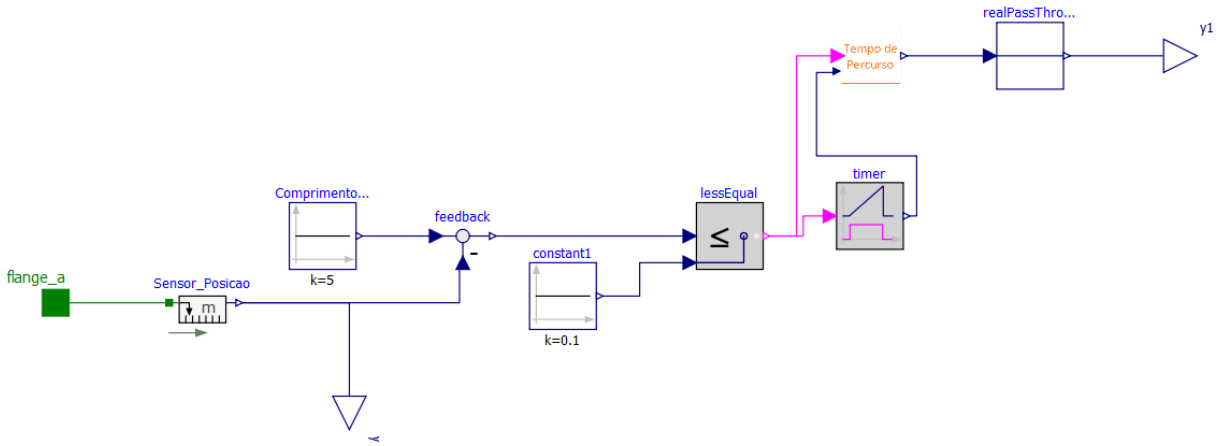


Figure 74 - Modeling the path timing of the conveyor belt

The travel time block collects the Boolean signal, which determines whether the travel has been completed or not, and the signal coming from the *timer*, which works as a stopwatch that is initialized when the Boolean adopts the value of 1. Once the signals are collected, they

are treated by the equation present in the block, where if the Boolean is 1, the output value corresponds to the difference between the simulation time minus the time marked on the *timer* output. The simulation time and the time on the *timer* are values that grow continuously, marking the passage of time, so the subtraction between both will be a real value, a value that corresponds to the time instant in which the box reaches the position where it will be later collected.

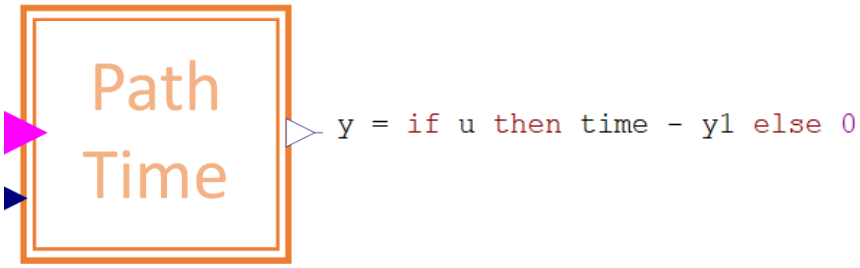


Figure 75 - Calculation model of the travel time

2.2.5. Comparison – Conveyor Belt Modeling

A relevant aspect to observe is the differences in the simulation results obtained for each of the input conveyors. Here can be concluded that the complexity of the modeling involved in the different factors that can be observed in real life makes a difference. The importance of approximating the real process as accurately as possible should thus be emphasized.

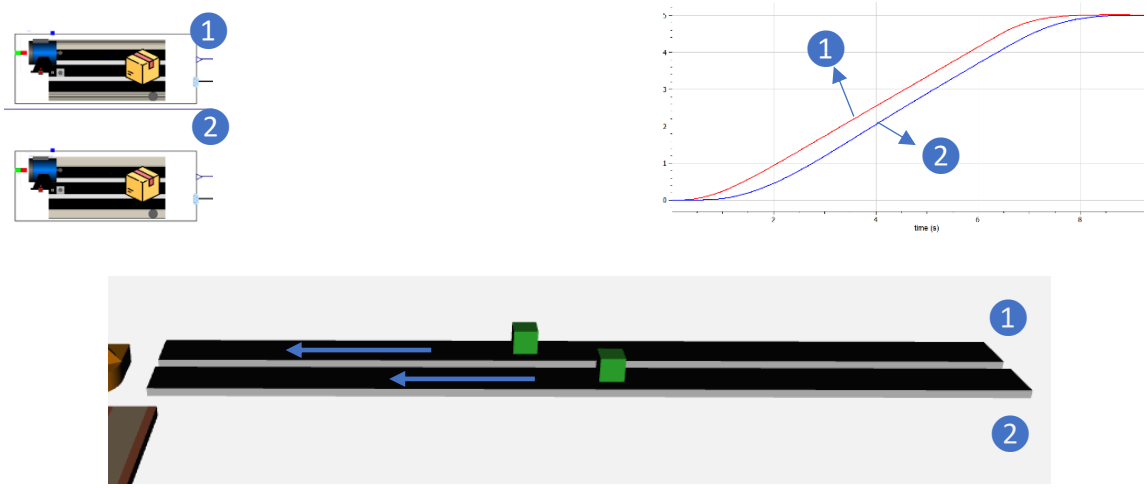


Figure 76 - Simulation of the 2 types of modeling performed for the entry conveyor and the respective graphical results

In addition to distinguishing this importance from how close the modeling may be in attributing factors involved in the actual process, the development of more simplified models

in parallel with more complex models may have its importance, as it allows one to understand the significance of these same factors and how they are reflected in the final performance of the system.

2.2.6. Cooling System

2.2.6.1. Thermal Flux to the Outside

Similarly to the set of heat flux models developed for the motor, a thermal dissipation exploration is also performed for one of the conveyor models, more specifically the 2nd entry conveyor modeled. The choice of this particular conveyor is due to the particularity that, when modeling the conveyor structure, higher mechanical and thermal losses are considered. Until now, thermal losses have been analyzed from a passive point of view, i.e. the heat fluxes involved in the axes of the robot joints have been calculated, but never from the perspective of changing these fluxes and thereby avoiding overheating. Either the operation of an industrial manipulator, or the operation of a *pick&place* system supplemented with several conveyor belts, does not normally require a cooling system, unless the production flow is very large. This need becomes even less relevant when only one operating cycle is addressed, as analyzed throughout the model simulations of this project. However, in order to examine this possible need (for example if it is necessary that the temperature variation of the packages to be transported maintain a fairly strict tolerance) as well as investigate and introduce a new area of interest for modeling in the system, a cooling system is developed, associated with the conveyor.

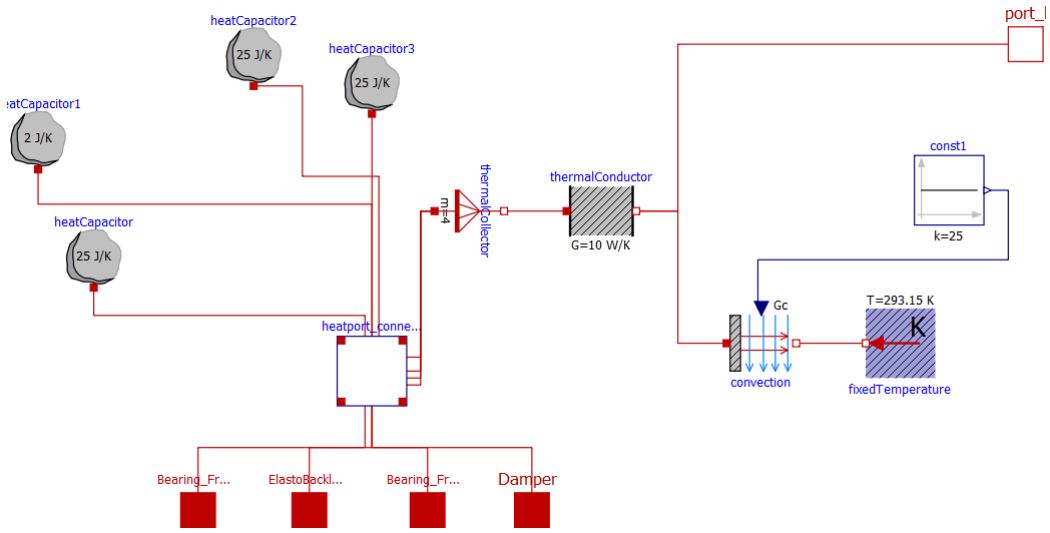
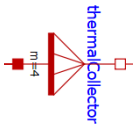
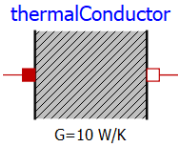
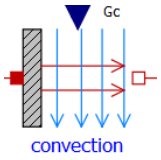
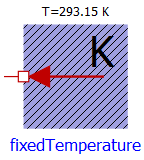


Figure 77 - Modeling the heat flow to the outside

Associated with each of the components of the conveyor that one seeks to study thermal losses (friction, damping and backlash components), it can be seen that several components are introduced destined for different heat capacities, one for each component, thus representing the resistance to temperature variation of each component. Once the heat capacities are established, a component is used that allows the heat fluxes of each of the components to be summed up and thus a single heat flux to be associated with a conductive element. This conductive element can be defined as the physical border between the conveyor and the outside, in which besides a direct contact with the packaging, there is a sharing of a convective thermal flux with the ambient temperature, on its outside.

Table 11 - Modeling Components used in the heat flow to the outside on the conveyor

Component	Description
	Heat capacity, that is, the amount of heat supplied for the temperature to rise by one unit, considering a constant temperature of the body throughout its volume. The heat capacity is ultimately obtained by multiplying the heat transfer coefficient k by the surface area of the body A.

	Component that gathers at its inlet different heat ports (m). The sum of the heat fluxes presented at each of the ports at the component's entrance are added and the result of this sum is presented in the form of heat flux at the thermal port at the exit. The temperature established both at the inlet and outlet of the component is equivalent to the result of the thermal equilibrium of the system in which the component is located.
	Component that portrays the thermal conductivity of a material under study. From a mathematical point of view it can be defined as the amount of heat transmitted per unit time, through an object with unit thickness L, in a direction normal to the surface area of its straight section A, also unit due to a unit temperature variation ΔT between the longitudinal ends.
	Component that defines the convective state of the transfer, specifically the heat transfer coefficient and the total transfer area, as shown in the equation. For this purpose the parameter Gc is defined in which it is determined based on the multiplication of the area by the coefficient. $$Gc = h \times A$$
	Boundary condition where a fixed temperature is stipulated.

The convection process introduced in this model follows the transfer equation already addressed in the robot axes, so the modeling introduced here is identical to that previously modeled in the thermal flow along the axes, with the convection component directly connected to a component that characterizes the outside temperature. There is however, a

thermal conduction component introduced, whose transfer is defined using the following formula.

$$\dot{Q} = \frac{Q}{\Delta t} = \frac{k \times A \times \Delta T}{L} \tag{55}$$

Where the thermal conductance $\frac{k \times A}{L}$ is parameterized in the component for the body's thermal conductivity.

As can be seen from the heat dissipation process analyzed, a continuous heat transfer occurs, between the conveyor and the external environment, whenever it starts operating and its overall temperature increases. Thus, it is easy to see that a cooling is always inevitable, when the conveyor operation is interrupted and the convective heat transfer rate, implied by a heat transfer to the surrounding environment, is higher than the heat generation by the conveyor components.

The development of a cooling system arises from the need to increase this heat transfer rate whenever desired and thus more consistently control the average conductive surface temperature of the conveyor.

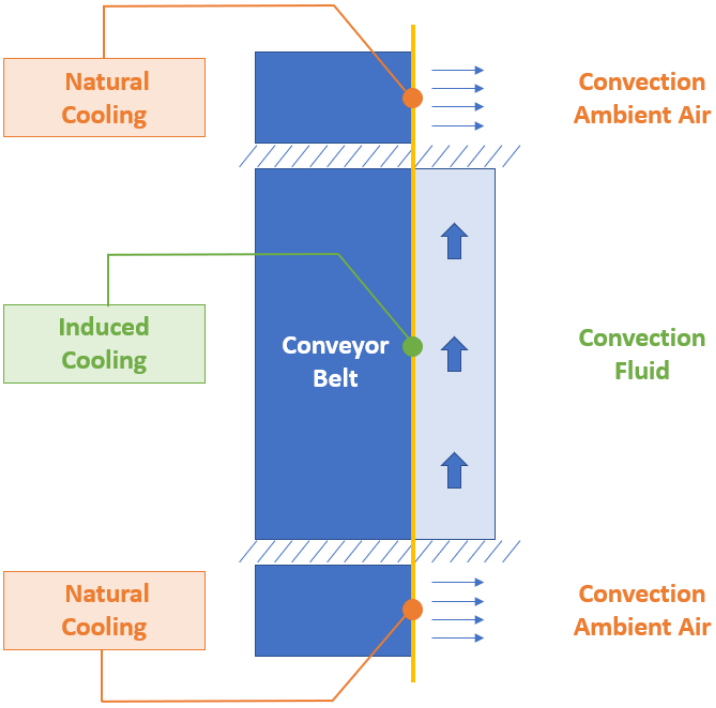


Figure 78 - Natural cooling and induced cooling

In order to attach the fluid section, and cause additional cooling when needed, a new portion will have to be modeled and associated with the component that portrays the thermal conductivity of the conveyor surface, thus serving as an intermediate component between the total flow coming from inside the conveyor structure and also the flow coming from the pipe through which the fluid circulates.

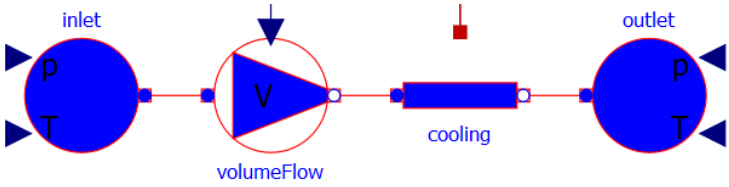
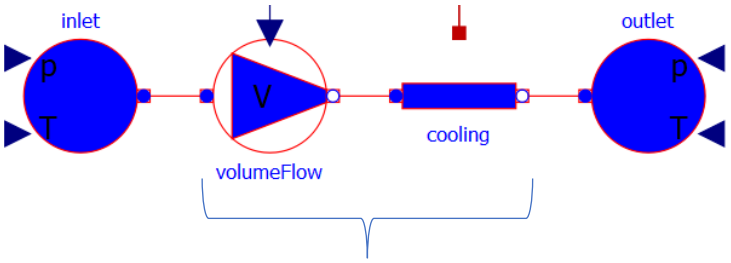


Figure 79 - Modeling of the attached cooling system

2.2.6.2. Thermodynamic Aspects

The cooling system model has a number of thermodynamic aspects that will be readily referred to. The different parameters and variables that are relevant in the two main models of the cooling system can be seen in Figure 80. The coolant circulation system is inspired by an example in the *Modelica* library [28].



TwoPort	• *Dp* - Pressure drop between the two ports • *V_flow* (Volume flow) = Mass Flow (port a) (kg/s) / Density (kg/m3) • *T_a* = H - Specific Enthalpy (port a) / Cp – Specific Heat Capacity (J/(kg.K)) – at constant pressure • *T_b* = H - Specific Enthalpy (porte b) / Cp - Specific Heat Capacity (J/(kg.K)) – at constant pressure
flowPort	• *P* - Pressure (Pa) • *M_flow* – Mass Flow Rate (kg/s) • *H* - Specific Enthalpy • *H_flow* – Enthalpy Flow Rate (watts)
medium	• *Rho* - Density (kg/m3) • *Cp* - Specific Heat Capacity (J/(kg.K)) – at constant pressure • *Cv* - Specific Heat Capacity (J/(kg.K)) – at constant volume • *Lambda* - Thermal Conductivity (W/(m.K)) • *Nu* – Kinematic Viscosity (m2/s)

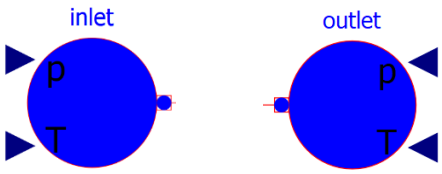
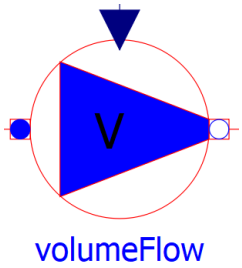
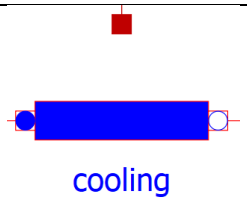
Figure 80 - Different parameters and variables that have relevance in the 2 main cooling system models

Partial *TwoPort* model in which the fluid properties are defined. These properties serve as a basis for the *volumeFlow* and *cooling* models. This partial model consists of 2 *flowports* in which pressure *p* (Pa), mass flow *m_flow* (kg/s), specific enthalpy *h* and enthalpic flow *h_flow* (watts) are defined. In these 2 *flowports* a *medium* parameter, with properties relative to the fluid, is further established, in which density *Rho* (kg/m3), specific heat capacity *cp* (J/(Kg.K)) at constant pressure, specific heat capacity *cv* (J/(Kg.K)) at constant volume, thermal conductivity *lambda* (W/(m.K)) and kinematic viscosity *nu* (m2/s) are stated.

The *TwoPort* partial model calculates the variation between the pressures in the 2 *flowports*, the volumetric flow rate by dividing the mass flow rate in the first *flowport* by the density, the temperatures in the 2 *flowports* by the ratio between the specific enthalpy and the specific heat capacity at constant pressure, and finally the temperature relevant for heat exchange with the environment. Throughout this partial model a mass balance is also defined between the two ports and an energy balance where the enthalpic flow in the two *flowports* plus the heat exchanged with the outside/environment equals $m \,.\, c_v \,.\, dT$ where m is the mass of the

fluid, c_v is the specific heat capacity at constant volume and finally dT is the derivative of the temperature in the second *flowport* over time.

Table 12 - Description of the components present in the cooling system

Component	Description
	*Inlet* and *outlet* are defined as two components that determine the boundary conditions of the system, where an environment with the properties of pressure and temperature are established.
	Component that has the ability to directly define the volumetric flow rate (m3/s). In the absence of such definition by the user, the volumetric flow rate taken into account comes from the thermodynamic equations evident in the partial model of the two ports.
	Allows an ideal heat exchange with the outside.

2.2.6.3. Cooling System Control

In modeling the cooling system, there is a concern to optimize the cooling by controlling the volumetric flow rate. A higher value of volumetric flow results in a greater heat exchange arising between the piping and the thermal conduction surface on the conveyor, thus decreasing the temperature.

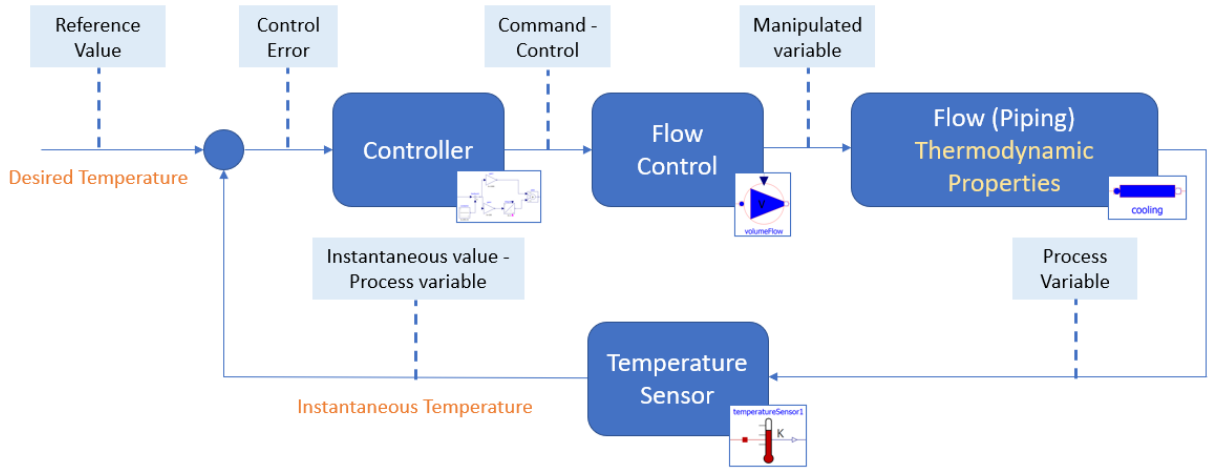


Figure 81 - Schematic of the control system

In order to ensure this flow control, the temperature at the outlet of the thermal port (process variable) in the *cooling* block is continuously measured, and its value is input to a PI controller. This controller in turn manipulates over time the volumetric flow rate in the system, so as to ensure that the difference between the desired temperature at the thermal conduction surface of the conveyor and the instantaneous temperature obtained is close to absolute 0.

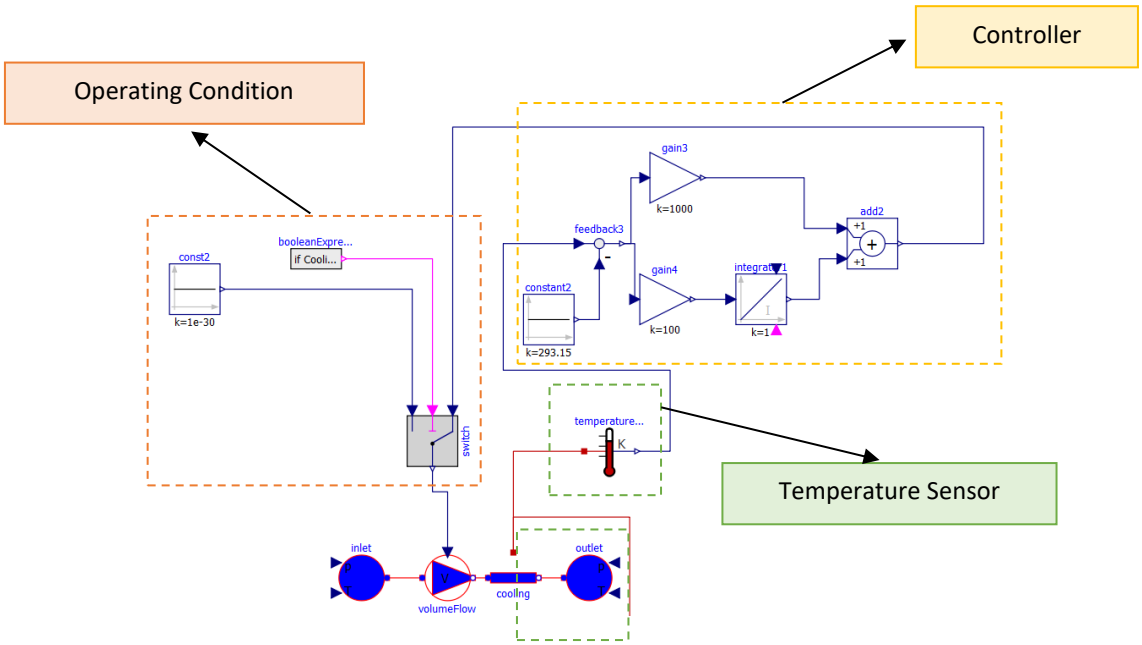


Figure 82 - Control system modeling

An additional condition is inserted throughout the system, which is related to activation of the cooling system, in the event that the respective activation boolean is true the control system

operates as mentioned above, otherwise the volumetric flow rate admitted into the system is 0 and the occurrence of additional cooling is not verified.

The final model of the conveyor 2, with the thermal dissipation and cooling properties of the system, can be seen in the illustration below.

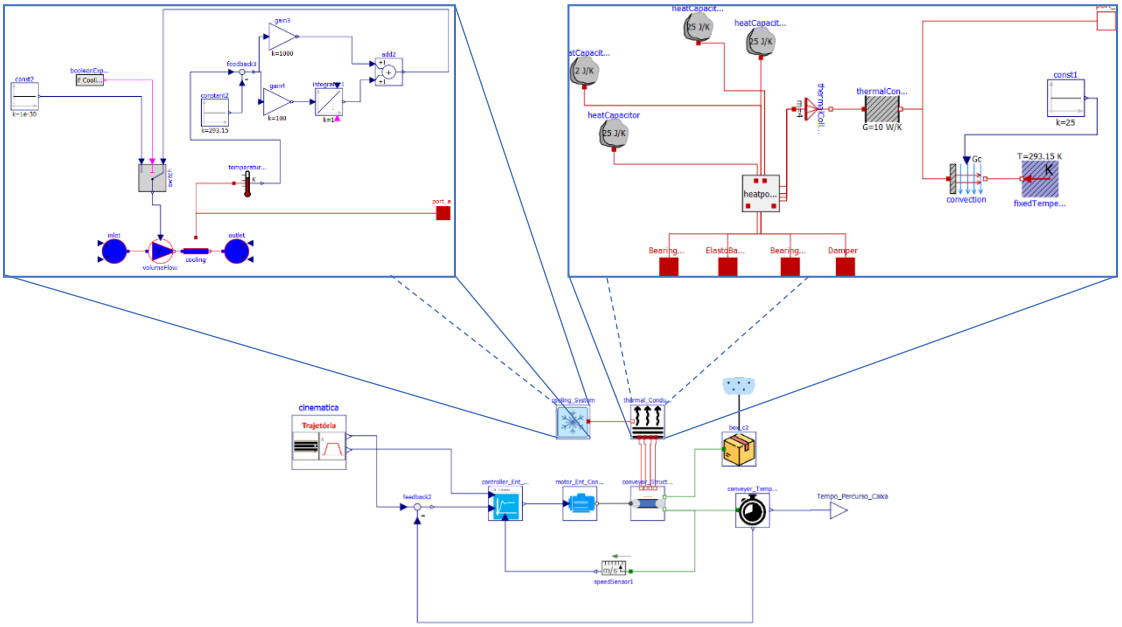


Figure 83 - Final model of conveyor 2 with conduction, convection and thermal cooling system

2.2.6.4. Cooling - Results

- Without induced cooling

It can be seen that an increase in temperature along the conveyor takes place, simultaneously with its movement. When the movement of the conveyor is finished, the peak temperature is reached, and the temperature subsequently goes into decline as a result of heat transfer to the outside.

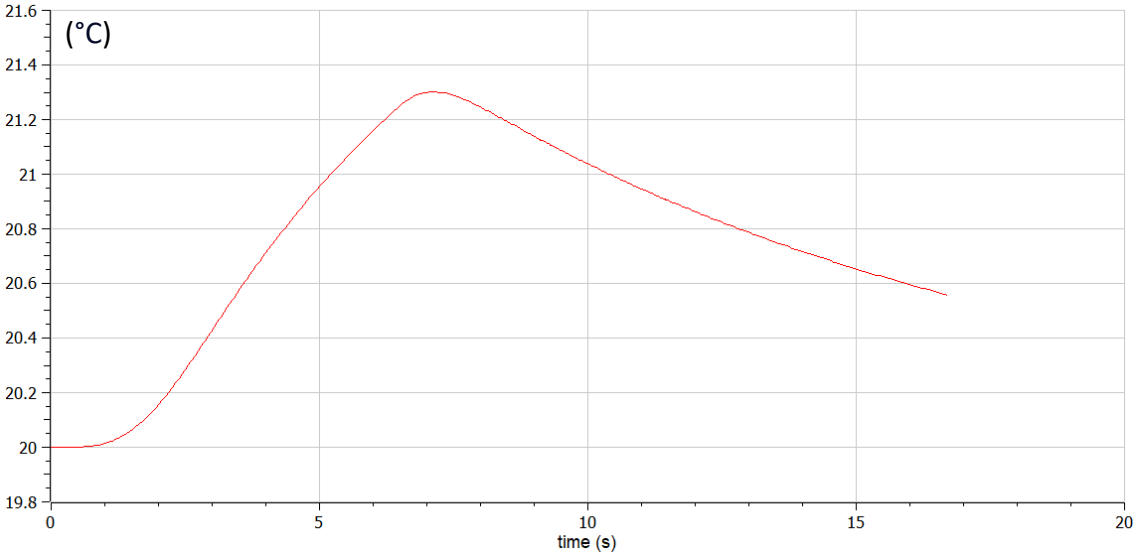


Figure 84 - Temperature variation along the conveyor without cooling system activated

- With induced cooling

With the active cooling system, the difference in temperature variation is notorious. There is a peak increase in temperature variation of 0.116 as opposed to 1.299 when not subject to supplemental cooling.

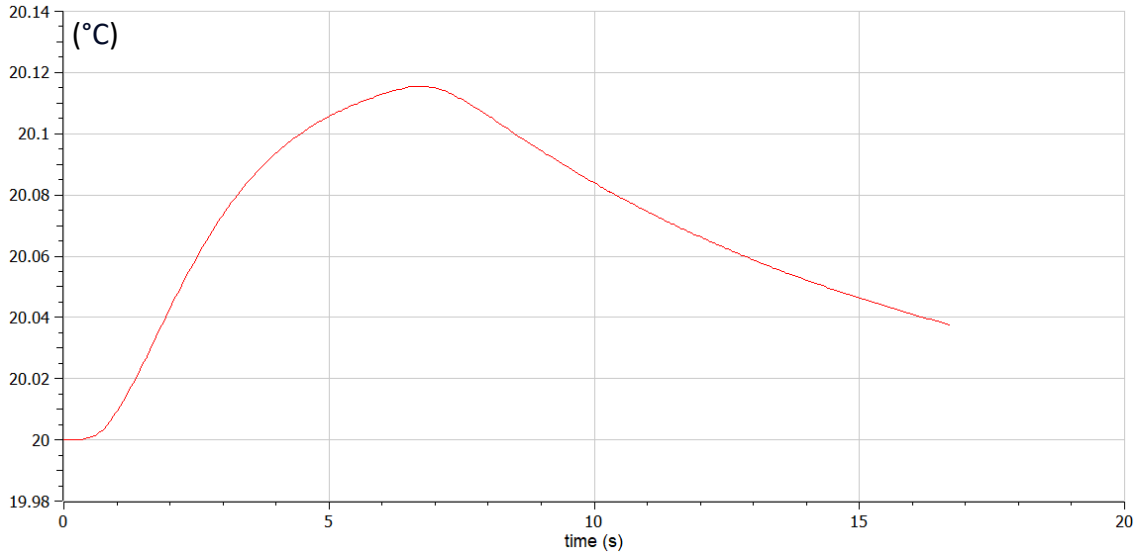


Figure 85 - Temperature variation along the conveyor without cooling system activated

2.2.7. Continuation of the Simulations Study – Conveyor Belts

Some results derived from the simulations and arising from the conveyors are presented here.

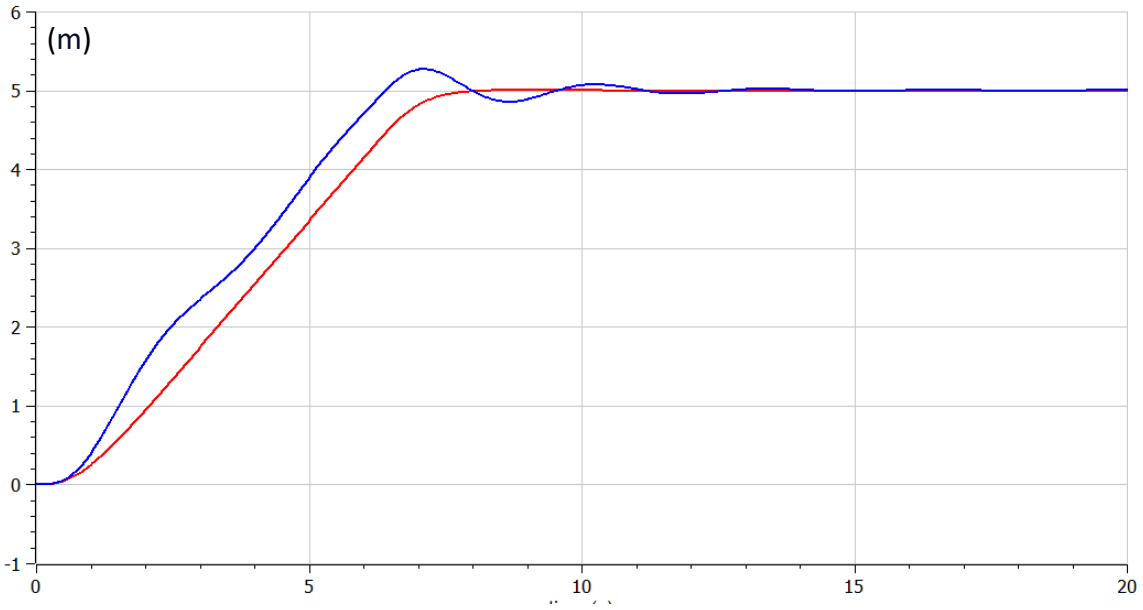


Figure 86 - Entry conveyor belt 1 with (red) and without (blue) damper

In this graph (Figure 86) it is possible to observe the input conveyor 1 that is simulated apart from the global system with 2 different configurations, in order to allow observing the effect that the insertion of a damper has on the performance of the different conveyors. It is noticeable that its insertion (function in red) allows a much smoother movement, without disturbances and/or overshooting.

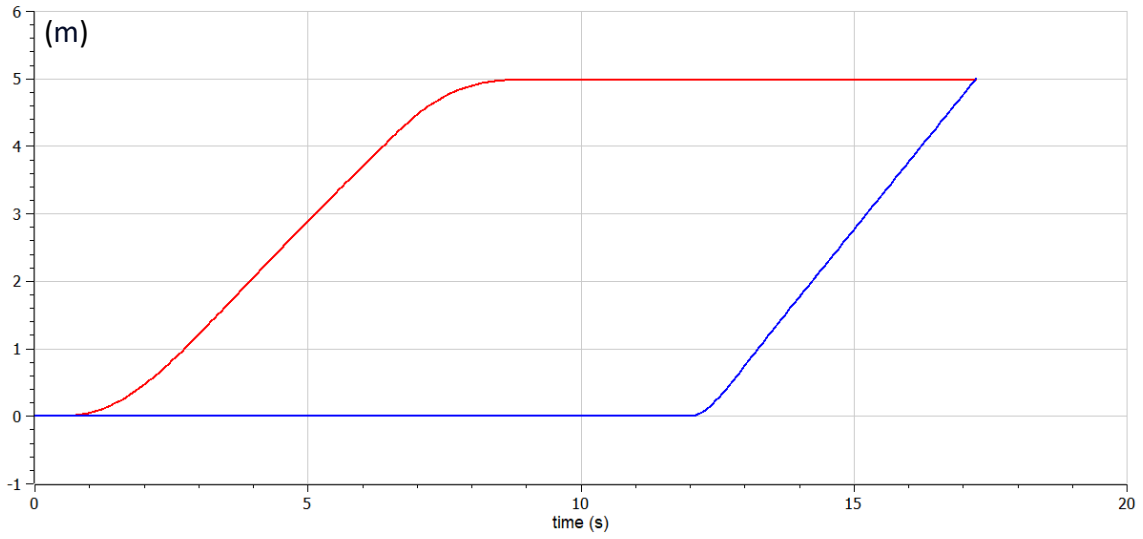


Figure 87 – Linear movement (meters) of entry conveyor 2 (red) and exit conveyor 1 (blue)

Here it is possible to observe the movement of the package on the entry conveyor 2 and exit conveyor (package 1). With the use of the input conveyor considering losses, accelerations

and decelerations are smoother in contrast to the output motion, where losses are not considered.

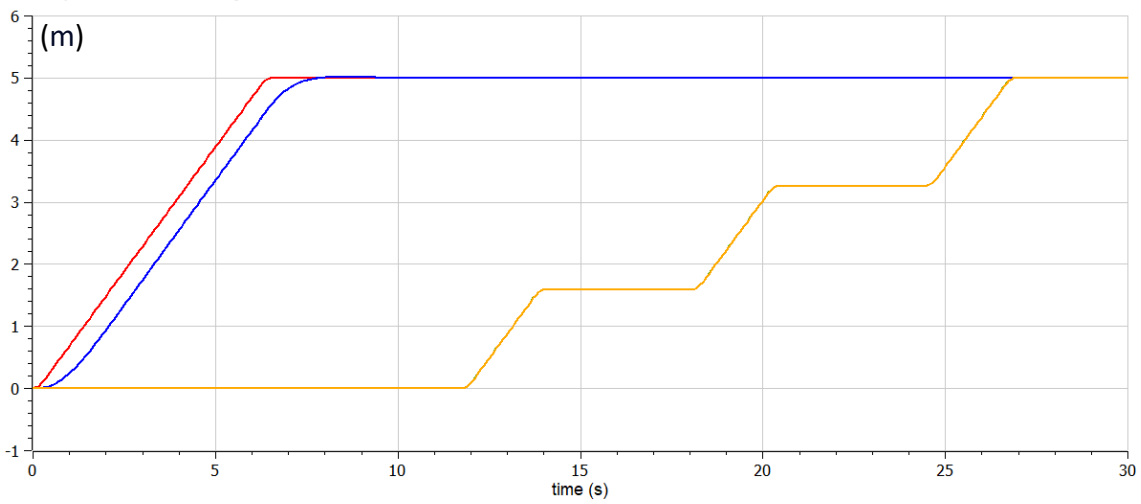


Figure 88 – Command (red) and Position/Movement (blue) of the entry conveyor 1, and command (green, not visible – overlaid by the instant position) and Position/Movement (yellow) of the exit conveyor 3

The differences established between the position commands and the positions actually adopted can be observed in Figure 88, for entry conveyor 1 and exit conveyor 3. While the input conveyor presents a mismatch, the output conveyor presents an almost perfect behavior, with the overlapping of the acquired position with the desired one. The mismatch verified in the input conveyor can be explained by a not very accurate tuning and by the absence of the current control, which is present in the output conveyor control. The fact that the output conveyor requires successive stops at specific operation posts and whose 4-second rest time must be obeyed with very little tolerance, the development of the control on this conveyor is more rigorous.

Regarding the torques, it is important to note that acceleration will require an acceleration torque ($T_{ac} = J_t \times \alpha$). When acceleration ceases and a constant speed is reached, the torque is equal to counter-torque (difference between the two becomes zero as verified graphically), generating an equilibrium of motion.

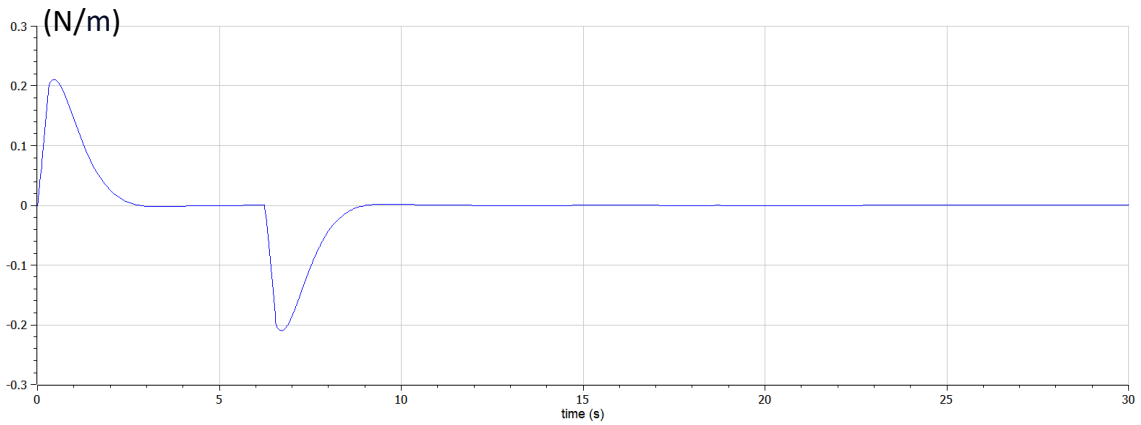


Figure 89 - Difference between torque and counter-torque

When acceleration occurs, the resulting motor torque is greater than the counter-torque (resulting in a positive difference where acceleration is verified), and in a final phase there is a lower motor torque than the counter-torque resulting from the inertia of the system (resulting in a negative difference where deceleration is verified).

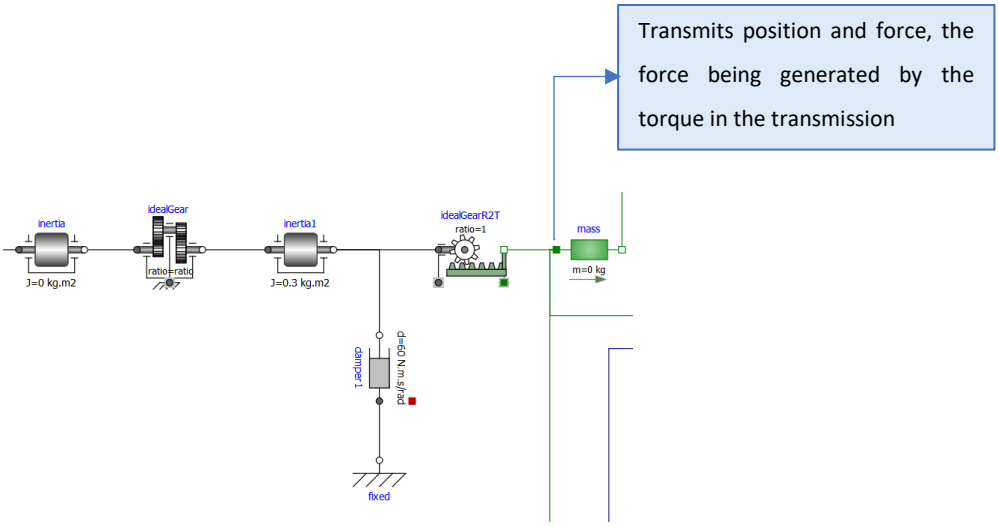


Figure 90 - The torque involved in the transmission is directly proportional to the axial force required for the transport

The axial force, and consequently the torque on the rotation-translation transmission block, will always adopt a value close to 0 between acceleration and deceleration, since, disregarding losses, the load will maintain a uniform rectilinear motion without any axial force acting. Therefore the torque will always tend towards a zero value, in this phase, regardless of the inertia of the total system or the damping constant considered.

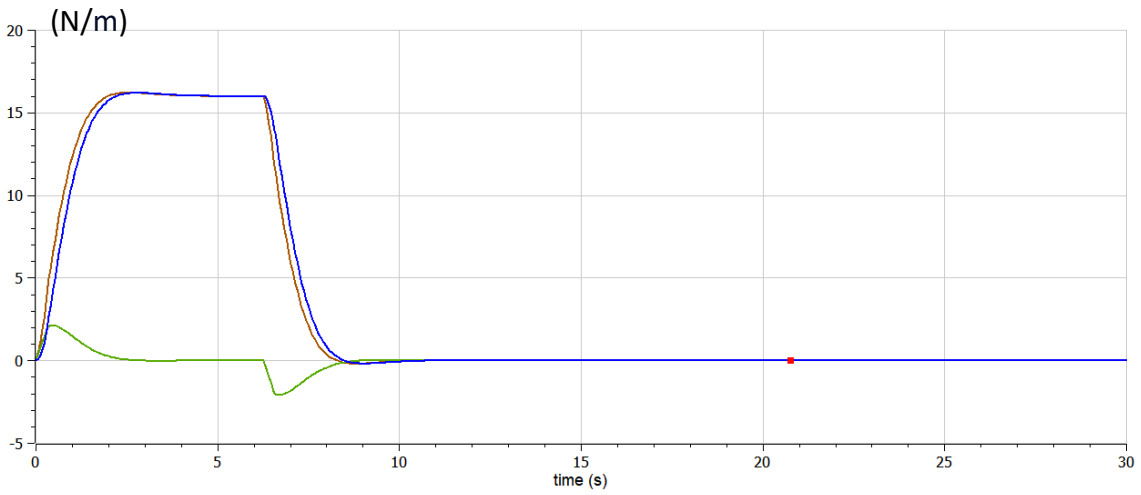


Figure 91 - In brown is the torque required by the motor. In green the acceleration and deceleration counter-torque as a function of the axial force required and finally in blue the counter-torque resulting from damping

In the graph (Figure 91), the torque from the transmission associated with the motor is shown in brown. Analyzing the formula resulting from the torque with constant speed, it would be expected that the torque required from the motor would adopt a value close to 0, since there is no acceleration in this phase, the force resulting from the speed variation does not exist either, resulting in a torque close to 0. The axial force of motion exists, however, not due to the speed variation, but due to the counter-torque, resulting from the use of a damper (to avoid overshooting). The counter-torque has its torque directly proportional to the speed, so it assumes, like the speed, a profile close to trapezoidal. The motor torque (brown line) thus adopts a value over time, resulting from the sum between the torque for accelerating and decelerating the load (green line) and the counter-torque (blue line) resulting from the frictional force on the damper. When the counter-torque of the damper exceeds the torque coming from the motor, deceleration is verified.

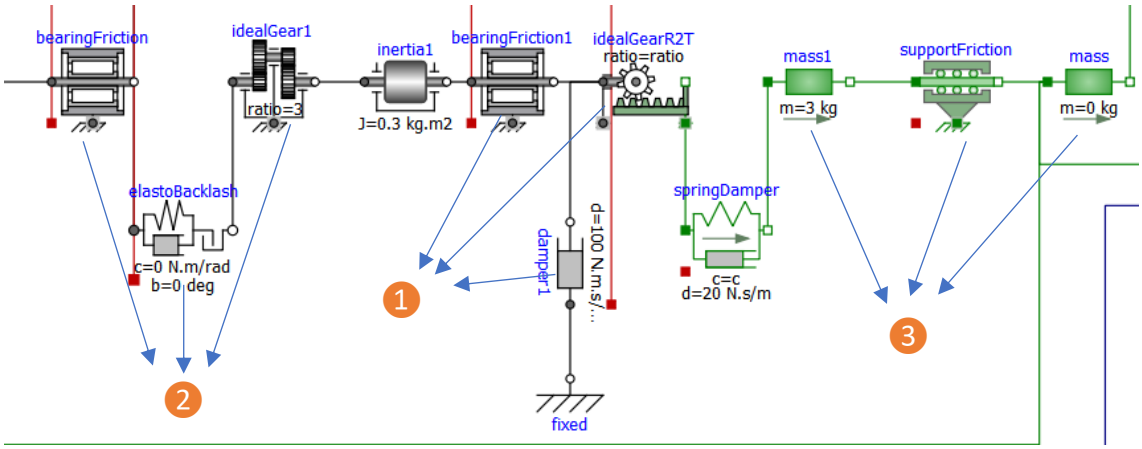


Figure 92 – Structure modeling of the entry conveyor 2

Regarding the simulation of the model with a structure in which the losses are comprised, similarly to the model of input conveyor 1, the torque present in the inertia block is equivalent (1) to the counter-torque of the friction in the damper plus acceleration and deceleration torque of the load, but in this case the counter-torque present in the friction of the bearings is also considered. The torque present in the inertial block is not, however, the torque required by the motor, since this block is preceded by a backlash gear transmission (2). The backlash effect is responsible for eliminating any kind of contact between the gears for a set angular range, triggering a higher torque when that contact is introduced again, in order to compensate for the absence of previous transmission, thus generating a more abrupt and less smooth speed variation. An additional resistance to motion is further introduced with a 2nd friction block in the bearings.

In this example, a guide frame (3) is modeled that serves as the base for the mechanical load (belt plus packaging, with the mass of the belt having been disregarded). The axial force exerted on the guide structure (red line) is thus higher than the axial force exerted on the mechanical load (blue line), since axial frictional force modeling is performed. Also noteworthy is the presence of an axial force and consequently a torque throughout acceleration that is sharper than that seen throughout deceleration. This peak of force and torque required is fundamentally explained by the backlash, which acts in a first phase in which the load moves from the resting state to the moving state.

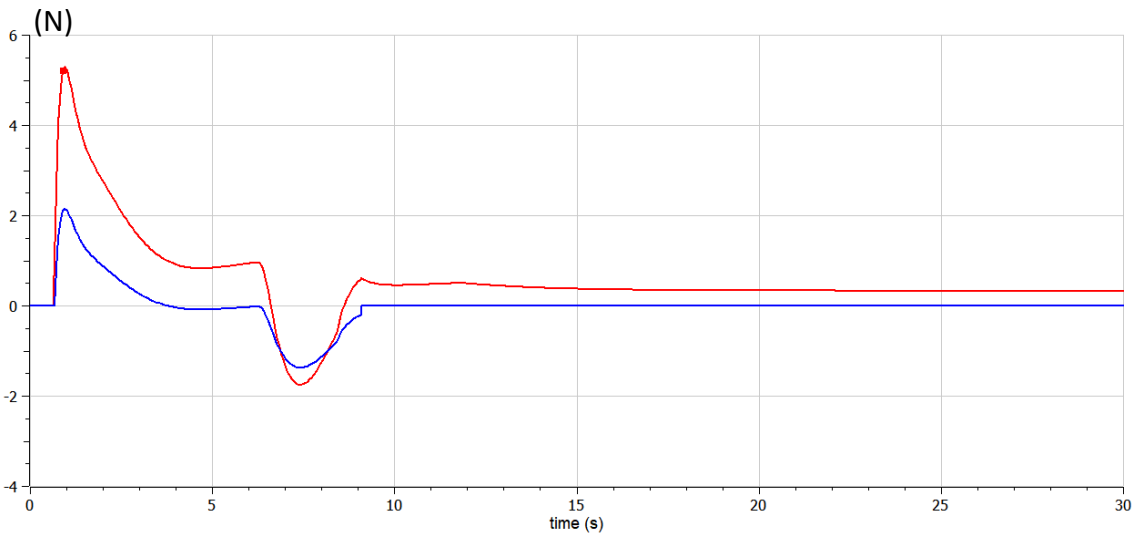


Figure 93 - Axial force acting on the guide frame (red) and axial force acting on the mechanical load (blue)

Finally, the final torque coming from the motor (green line) corresponds to the sum of the acceleration and deceleration torque (orange line, whose variation profile is equal to the axial force required in the guide structure), plus the torques required to suppress the different resistances to movement involved in damping, friction, and backlash, which together admit a profile close to the trapezoidal one.

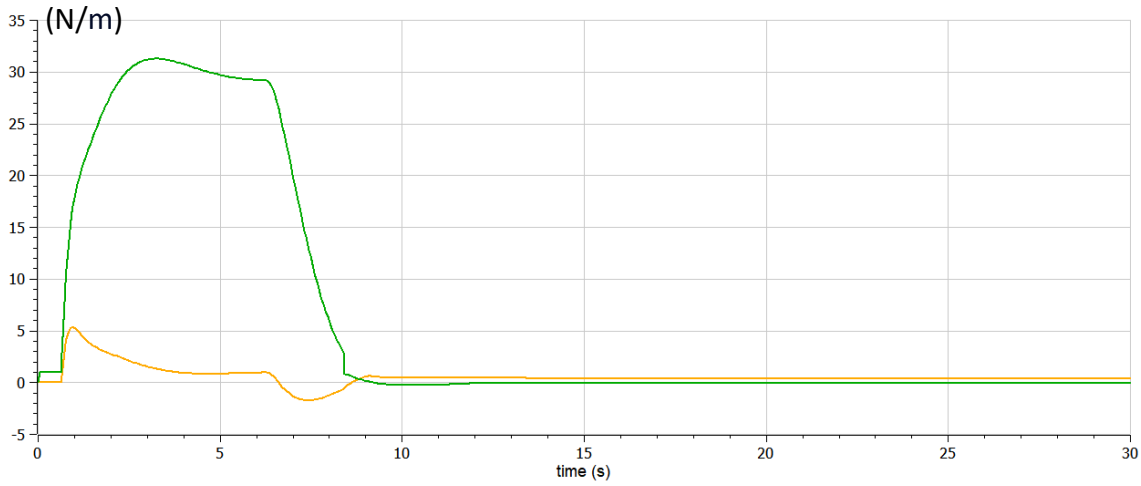


Figure 94 - Torque from the motor (green) and acceleration and deceleration torque (yellow)

Note that the torque involved in the rotation-translation transmission, in contrast to the simplified modeling of the structure, is thus no longer close to 0, since for a constant speed to be maintained a torque has to be provided in order to overcome the frictional forces involved between the guide structure and the mechanical load.

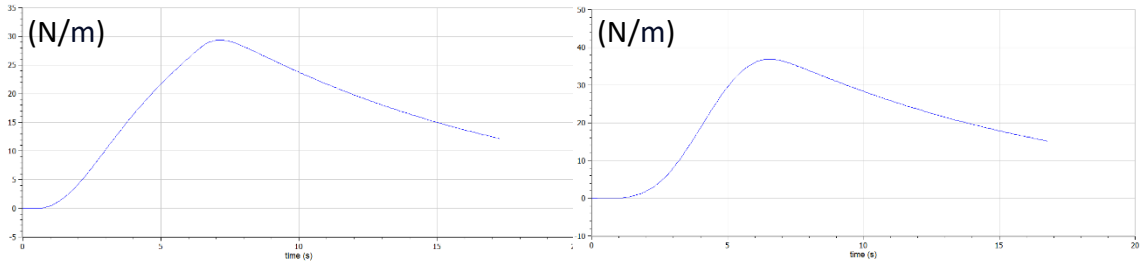


Figure 95 - Convective heat transfer on the entry conveyor 2, without cooling system, with trapezoidal (left) and 3rd order polynomial (right) velocity profile

In the robot, the trapezoidal velocity profile presents higher heat transfer values relative to the 3rd order polynomial. On the conveyor, the inverse is verified. This can be explained by the range of motion and the time involved in each of the movements. The acceleration peaks are always verified in the trapezoidal profiles, but while these peaks, in the manipulator, translate into greater heat transfer, in the conveyors this greater transfer is not guaranteed because the polynomial profile although it does not reach high values of acceleration, presents acceleration values mostly far from zero, unlike the trapezoidal which allows a constant speed between acceleration and deceleration.

Thus, although the polynomial profile is smoother without sudden increases or decreases in velocity, it may present a sufficiently constant velocity variation throughout the entire travel time, in contrast to the trapezoidal profile, thus dictating a greater heat transfer. The discrepancy between the torques and heat transfer involved in the two profiles is thus defined by the maximum acceleration delimited in the trapezoidal profile and also in the range of motion.

One of the particularities of studying different types of packaging has to do with the fact that different moments of inertia can be analyzed. A greater mass will necessarily imply a greater inertia exerted on the motors. This aspect is therefore of greater significance when modeling conveyor number 2 is considered, since the losses considered are much greater.

Table 13 - Trajectory times for the different boxes

Box	Temporization (s)	
	Entry Conveyor 1	Entry Conveyor 2
Box 1 - 2 kg	7.99	8.03
Box 2 - 2.5 kg	7.87	7.98
Box 3 - 3 kg	7.77	7.94

The greater the mass of the package, the shorter the travel time. These results may at first glance seem contradictory, since it would be expected, perhaps, a longer travel time in proportion to the weight of the package to be transported. The inverse is verified and this can be explained by the system response based on the same values assigned to the controller parameters. Thus, if the parameters are maintained, a larger mass to be transported will require a more robust and consequently faster response. In return, there will be greater overshooting and possibly a greater imbalance in the response given. Under these conditions, the ideal, in order to maintain a continuously balanced response regardless of the weight carried, would be an automatic tuning of the parameter values of the controller, and then, for greater mass, longer travel times would be verified. The use of equal parameter values for different masses ends up not compromising the integrity of the system, since the different masses used have values very close to each other, thus ensuring adequate responses regardless of the type of packaging.

2.3.Gripper

The gripper that can be connected to the robot and thus be used to pick up the package explored the mimic of a claw with three fingers, a central finger arranged laterally and two others arranged symmetrically on the opposite side. The modeling of it is done keeping in mind that the physical system incorporates position and force sensors, as well as electric actuators, specifically DC motors.

2.3.1. Gripper Design

The gripper structure, as previously mentioned, has 3 fingers. A categorization in similitude to the human hand could be translated into the middle finger and index finger on one side of the gripper and the thumb on the other. Electric actuators are installed adjacent to the first joint

at the base of the fingers, connected to the wrist. The joints of each finger have 1 degree of freedom each, providing a rotational movement. Each of the fingers thus has, in total, 3 degrees of freedom.

Thus, the following considerations are taken:

- The 3 available contacts should be applied simultaneously on the object, a synchronous application of the constrictions can help reduce uncontrolled movements.
- The contact elements should house a set of proximity sensors facilitating the control of approach procedures, as well as force sensors assisting also in controlling the contact forces involved.
- The existence of only 3 actuators, in order to limit size and weight, and limit the complexity of the system at the end of the arm. One actuator for each finger with each finger being actuated independently.

Each finger is composed of three phalanges, each phalange being connected by tendons/toothed belts and by pulleys.

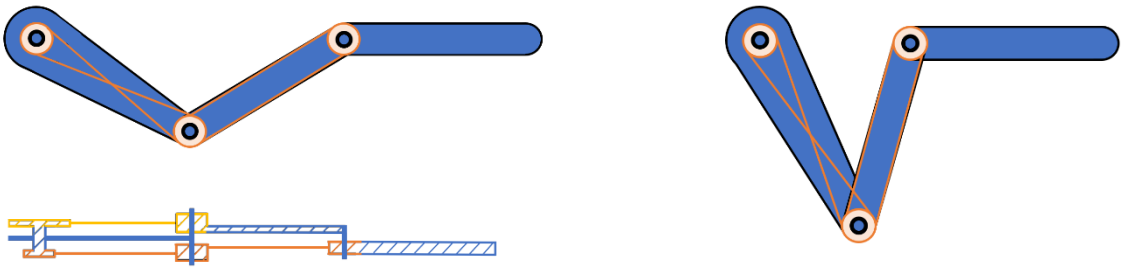


Figure 96 - Finger structure with transverse view

The matrix of the first axis is associated with the wrist, being the only axis of the structure directly driven by the servomotor. The axes of the other pulleys are driven by mechanical transmission. The mechanical transmission is designed so that the third, and last, phalanx is arranged continuously parallel to the position initially assumed, with an arrangement parallel to the surface of the package to be transported. This way, it is understood that the surface area of contact with the package to be grabbed is larger, allowing the action to be performed in a more compact and safe way.

Regarding the distribution of the pulleys along the 3 interconnected phalanges, 2 divisions in the structure can be considered. In the first one, which is in the upper area of the cross-sectional view image, there are only 2 pulleys, which dictate the respective positioning and orientation of the 2nd phalanx. While in the bottom section of the cross-sectional image, 3 pulleys are presented, being responsible for dictating the orientation of the 3rd phalanx. It can thus be considered that in a first phase, the position and orientation of the first two phalanges is achieved, and the orientation of the third phalange is later adjusted, according to the profiles of the preceding phalanges.

Table 14 - Pulleys belonging to each of the fingers and their angular variation

Section	**Angular Displacement**			
	1st Pulley	**2nd Pulley**	**3rd Pulley**	**Exit Angle**
Upper	α	2α	----	**3α**
Description	Pulleys that drive the kinematic motion of the first and second phalanx, with the total absolute angular displacement being equivalent to the sum of the motion of the first and second pulleys.			
Lower	α	-2α	-3α	**-3α**
Description	Pulleys that determine the final orientation of the third phalanx. The absolute angular displacement of the third phalanx is dictated by the last pulley.			

In order to maintain the same orientation in space, the angular displacement of the joint at the start of the third flange has to be the inverse value observed for the absolute angular displacement of the second flange. With this in mind, the proportion of angular displacements is established, in accordance with the values entered in the table 14, and from there the dimensional proportions of the different pulleys are taken.

Table 15 - Pulleys belonging to each of the fingers and their dimensional proportion

Section	Diameter		
	1st Pulley	2nd Pulley	3rd Pulley
Upper	2d	d	----
Lower	2d	d	2/3d

2.3.2. Modeling

In order to model the gripper that is attached to the robotic arm wrist, it is necessary to develop not only a control system that includes the movement of the phalange joints, aiming at opening and closing the gripper during the *Pick&Place* task, but also the force control to be exerted when closing it. All this implies a sequencing of control in different stages:

- In an initial stage a control by feedback is elaborated with an angular setpoint as reference, trying to define in this way the closing of the gripper and the approach of the fingers to the package.
- In a second stage, there is a feedback control with a force setpoint as reference. Here, contact is established as well as a compression force, allowing gripping the package. Simultaneous to this phase, there is the arm that transports the package between the entry and exit conveyor.
- Finally, an angular reference setpoint is again set, trying to replicate the opening of the gripper and the placement of the package.

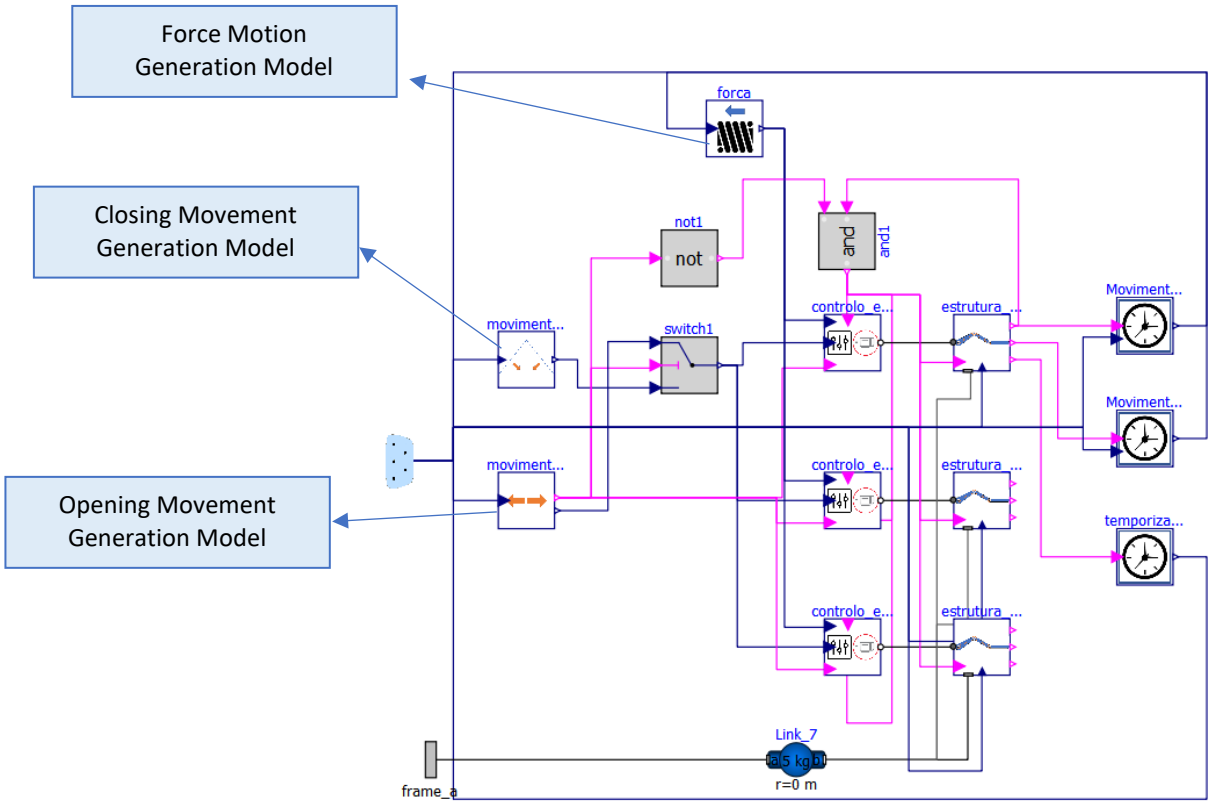


Figure 97 - Gripper model

In Figure 97 it's possible to observe the gripper model where several modeling instances can be immediately distinguished, such as the closing, force and opening movement models, which express the signals sent to the control system and that dictate the structure's movement.

Regarding the modeling of each finger, a prismatic joint is associated to the 3rd phalanx, with a spring and damper in parallel, with the purpose of simulating the contact arising from the gripper with the package to be transported.

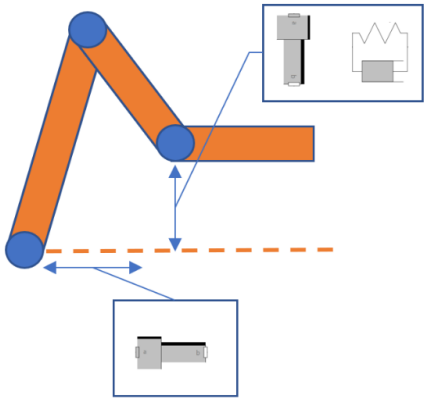


Figure 98 - Specific components used in the modeling of the finger structure

The prismatic joint, where the spring axis is attached, allows the movement of the spring to be perpendicular to the surface of the third phalanx, whose orientation remains constant over time. In this way, the force to be calculated using the spring and damper will also be perpendicular to the contact surface of the finger with the package. The spring and damper are actuated along the 3rd phalanx and are connected to the gripper shaft, whereby only motion in the direction of the shaft is permitted, as expressed through the use of a 2nd prismatic joint. By Newton's 3rd law, the spring applies a force equal and opposite to the force applied by the flange. In the prismatic joint, the displacement x along the axis is the displacement of any point along the 3rd flange and $v = dx/dt$ is its velocity. The force acting by the flange on the damped spring along the prismatic joint can be expressed as follows:

$$F = -k(x - x_0) - bv \tag{56}$$

Where the admitted parameters are the spring constant k, the natural length of the spring x_0, and also the damping constant b.

The logic expressed in the structure also presents its complexity, arising from the need to modify not only one of the three movement profiles, opening, force or closing, but also the mechanical structure itself, over time, with the assignment of the spring properties when the force profile is initiated.

When the movement of the arm to the collection point is finished, a signal of the time instant in which it occurs takes on a value other than 0 and feeds into the higher hierarchy model. This in turn enters the model for generating the motion profile for the closure, with the activation of a trapezoidal profile and sending it to the control model.

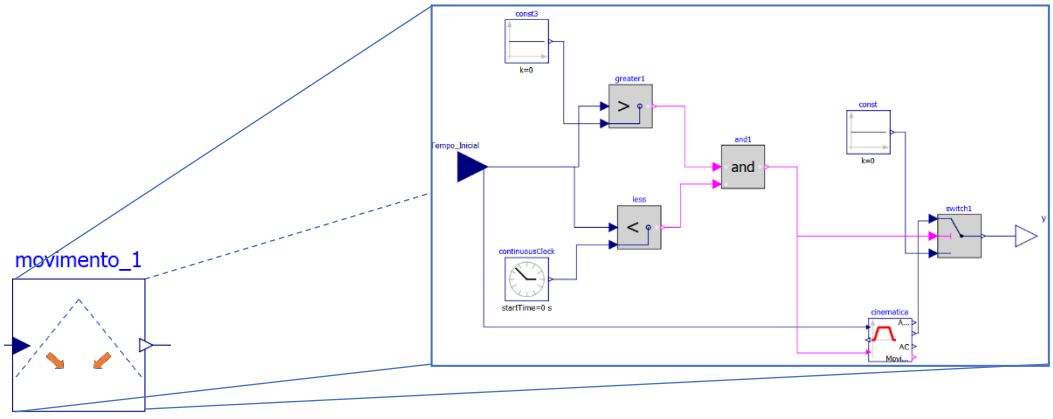


Figure 99 - Model for generating the motion profile for the closure

In the control model, 2 inputs are shown on the left, the top input defines the force profile and the bottom input defines the motion profiles (in the top hierarchy model, the closing or opening motion profiles have their input in this model determined by a *switch*).

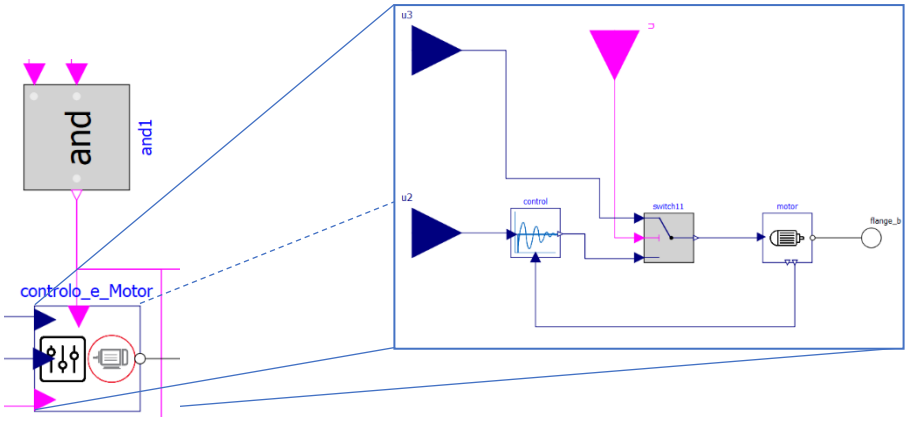
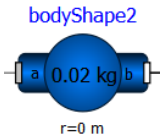
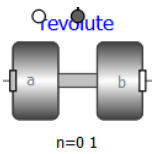


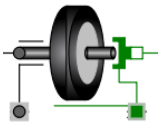
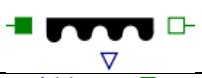
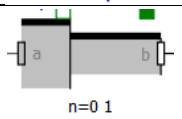
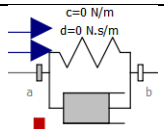
Figure 100 - Central gripper control model

The profiles are thus generated and sent as commands to the motor, driving the motor according to the operating stage the gripper is in. Finally, the motion is transmitted to the flanges via the output flange of the model illustrated above.

In modeling the final gripper structure, the gripper components are modeled as follows.

Table 16 - Components used in modeling the final gripper structure

Components	Illustration	Description
Phalanx		Use of *bodyshapes* for each of the phalanges, assuming the values of mass, center of mass, total mass, and moments of inertia. Some *bodyshapes* are also added to the model, for the purpose of its runnability, but without significant interference on its performance.
Rotary Joint		Represents the rotation axes along the structure. The first one is driven directly by the servomotor and the others by transmission through pulleys and belts. In the second joint, 2 axes are elaborated, one for the angular displacement that constitutes the kinematic movement of the second phalanx, and a second axis/rotating joint that characterizes

		the angular movement to be transmitted to the lower pulley, and thus the desired transmission rate is ensured.
Pulley		It constitutes the different pulleys distributed along the structure. The modeled pulleys are considered toothed, where the efficient diameter, the number of teeth, and also the moment of inertia are parameterized.
Belt		Belt that transmits momentum between the pulleys. Its elasticity can be considered by defining an elastic constant.
Prismatic Joint		Prismatic joints that define the physical constrictions of the spring movement along the gripper movement, with the intention that the force exerted on it be exclusively vertical.
Damping spring		Damping spring, involved in the force required for the gripper to grip the box.

The modeling of the finger structure can be seen below. For a better understanding of the modeling, it is divided into different parts. In the component highlighted in orange are the joints and phalanges, which can later be seen in the images of the simulated model. The sections highlighted in green are the upper and lower sections of the pulleys and belts responsible for driving the joints and moving the flanges. The yellow section refers to the components that subject the structure to a series of kinematic constrictions, namely prismatic joints and a damping spring that is activated, exclusively, when the force setpoint is admitted in the control, trying to replicate the contact with the packaging. The development of the modeling that targets the structure is partially inspired by a model developed in [29].

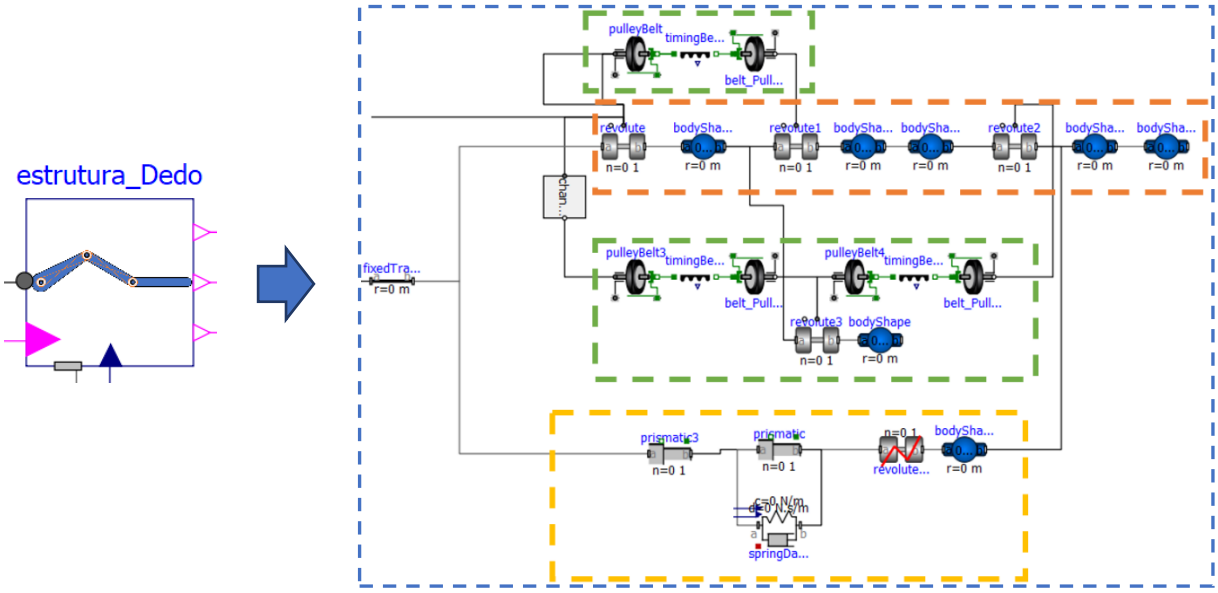


Figure 101 - Modeling the structure of a finger

The damping spring is modeled so that the parameters k (spring constant) and b (damping constant) vary over time, in accordance with the inputs received in the model of the spring.

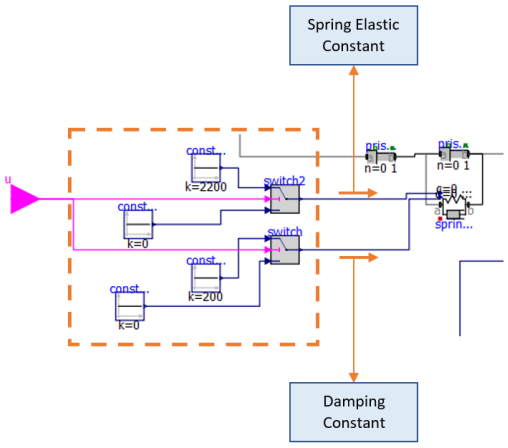


Figure 102 - Modeling inputs on the *springDamper* component

There are thus 2 inputs on the component, one for each parameter, whose values are determined by a pair of *switches*. Each of these *switches* is activated by a boolean, with a value other than 0 being admitted when the compression force is performed to grip the package, in contrast to a value of 0 when the gripper movement is to be free, either to bring the fingers together before the compression force, or to open the fingers, and allow the package to be dropped.

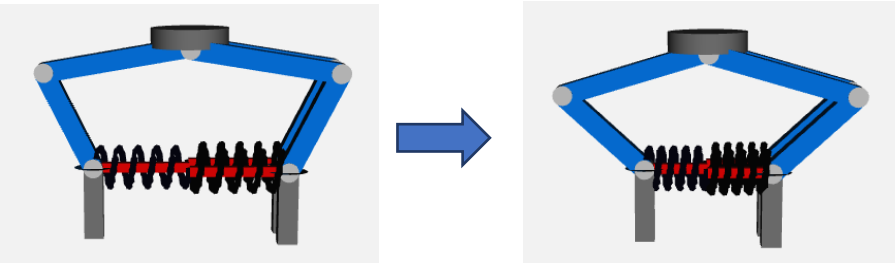


Figure 103 - Illustration of the gripper in simulation at the opening and closing moment

2.3.3. Gripper – Control

It was discussed the fact that there is a control model with 3 inputs that define the commands to the motor, depending on the operating stage in which the gripper is, and that only one of them appropriates, each time, the control signal sent to the motor. The definition of the instants in which either the closing movement command, the force command or the opening movement command are defined, is guaranteed by a set of associated feedbacks in the gripper structure model and that end up acting as angular change sensors of the first joint, allowing to perceive when the different closing, force and opening movements are completed.

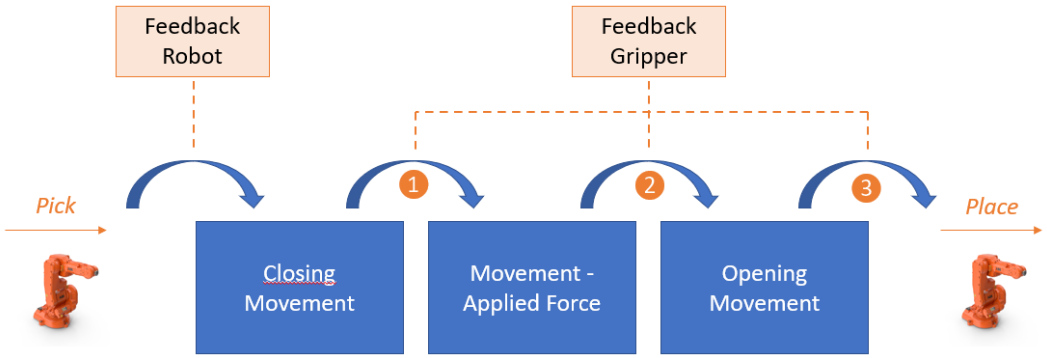


Figure 104 - Feedbacks requeridos na modelação de forma a permitir uma correta ordem nos movimentos

The different feedbacks present in the gripper structure model are assumed to be Boolean signals that admit the value of true when each of the movements is consummated. These Boolean signals are developed based on a set of logic models, whose state is defined by the angular values of the joints in the case of the closing and opening movements, and also by the force value in the damping spring component in the case of the force movement.

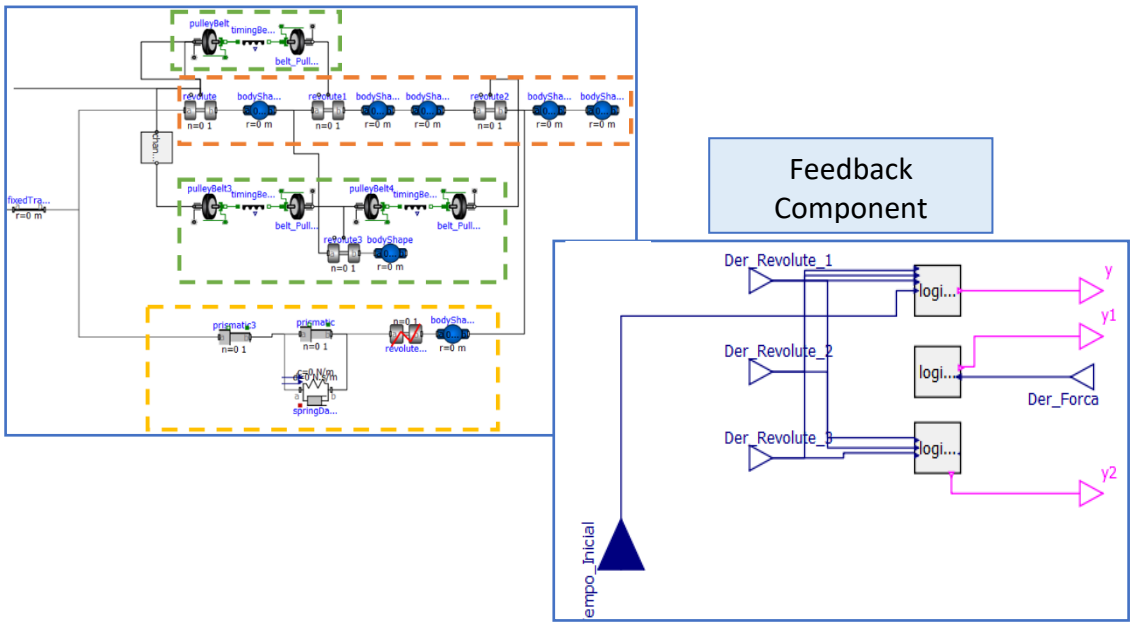


Figure 105 - Next to the modeling of the finger structure, a feedback section was developed

In Figure 105, regarding the feedback component, it is possible to observe the 4 inputs in blue, which relate to the derivatives of the angular position of the 3 joints along the finger and also the derivative of the force applied when the movement to grab the box is initiated. These 4 values work as inputs to the logic models, which will be analyzed next, and at the output of each of them there are Boolean feedbacks (in pink) that portray the state of the end of each movement.

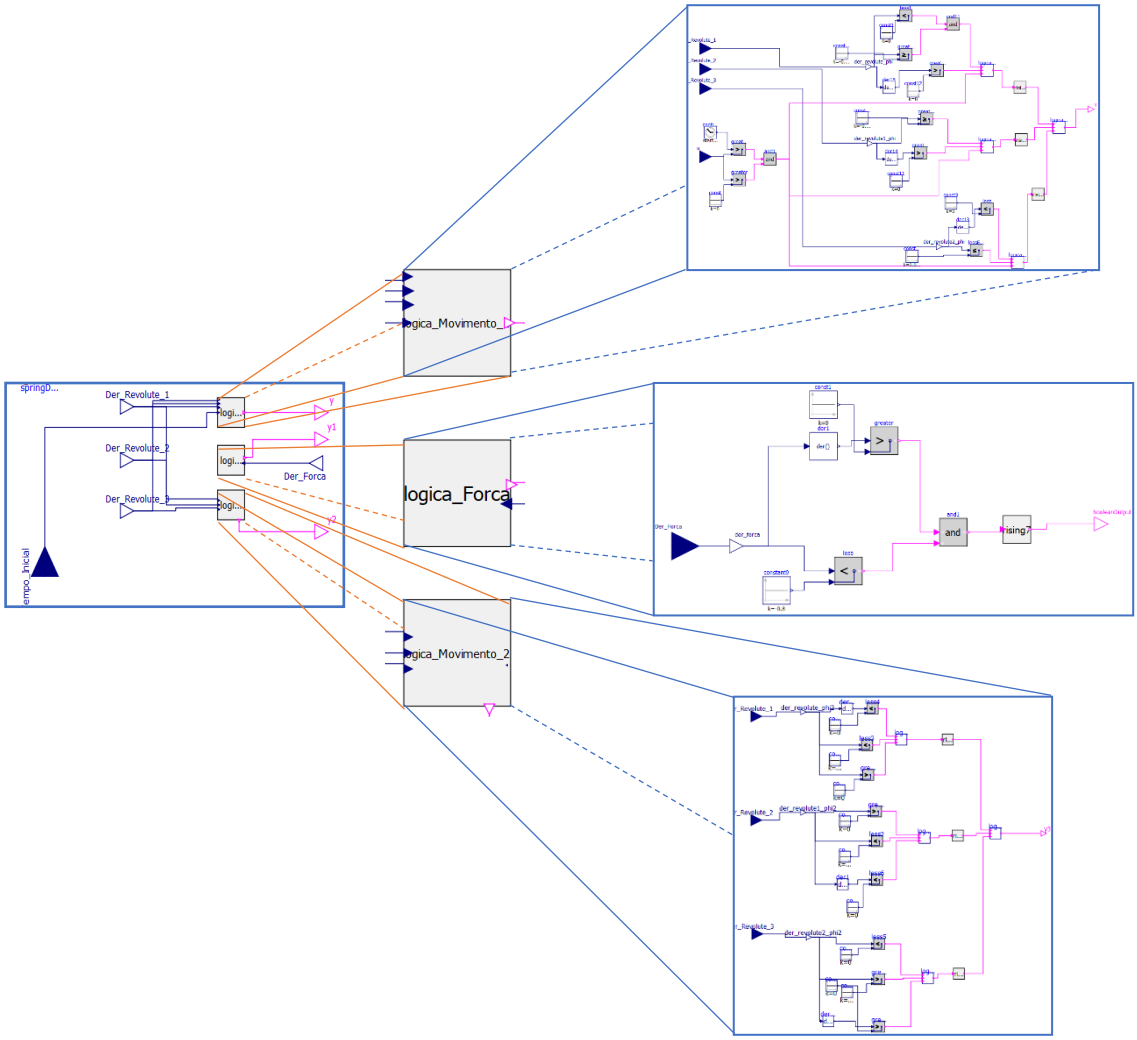


Figure 106 - The 3 models associated to the logic blocks of each movement executed by the gripper

In Figure 106, it can be seen the 3 modeling inherent to the programmed logic of the 3 output feedbacks. For this purpose are used blocks referring to the mathematical expressions of "*greater than (or equal to)*" and "*less than (or equal to)*", blocks of the logical condition *and* for both 2 and 3 expressions (*and* block for 3 expressions is created from scratch, given its absence in the library) and also a block created from scratch, which expresses a *rising edge* behavior in which a value of true at the output of the block is adopted when a transition false → true is verified. The state of the output also has the particularity of becoming true indefinitely after a *rising edge* is realized. The significance of this block is thus expressed by an interest in a single *rising edge* to be verified, thus presenting a neutral position with respect to later *rising edges*.

Table 17 - Logical states associated with the modeling of the closing movement logic block

Model	Variable	Logic				
		Condition 1	**Condition 2**	**Condition 3**	**Condition 4**	**Condition 5**
Closing Movement	Derivative Joint 1 (x1)	(x1) < 0	*and*	*and*	*Rising Edge*	*and*
		(x1) ≥ -0.0002				
		Der(x1) > 0				
	Clock (c)	(c) > (ti)	*and*			
	Start Time (ti)	(ti) > 0				
	Derivative Joint 2 (x2)	(x2) ≥ -0.001	*and*	*and*	*Rising Edge*	
		Der(x2) > 0				
	Clock (c)	(c) > (ti)	*and*			
	Start Time (ti)	(ti) > 0				
	Derivative Joint 3 (x3)	(x3) ≤ 0.0004	*and*	*and*	*Rising Edge*	
		Der(x3) < 0				
	Clock (c)	(c) > (ti)	*and*			
	Start Time (ti)	(ti) > 0				

Table 18 - Logical states associated with the modeling of the force movement logic block

Model	Variable	Logic		
		Condition 1	Condition 2	Condition 3
Movement - Applied Force	Derivative Force (x)	Der(x) > 0	*and*	*Rising Edge*
		(x) < -0.8		

Table 19 - Logical states associated with the modeling of the opening movement logic block

Model	Variable	Logic				
		Condition 1	Condition 2	Condition 3	Condition 4	Condition 5
Opening Movement	Derivative Joint 1 (x1)	(x1) < 0.001	*and*	*and*	*Rising Edge*	*and*
		(x1) > 0				
		Der(x1) < 0				
	Derivative Joint 2 (x2)	(x2) > 0	*and*	*and*	*Rising Edge*	
		(x2) < 0.001				
		Der(x2) < 0				
	Derivative Joint 3 (x3)	(x3) < 0	*and*	*and*	*Rising Edge*	
		(x3) > -0.002				
		Der(x3) > 0				

2.3.4. Simulations Study – Gripper

Some results derived from the simulations and arising from the gripper are presented here.

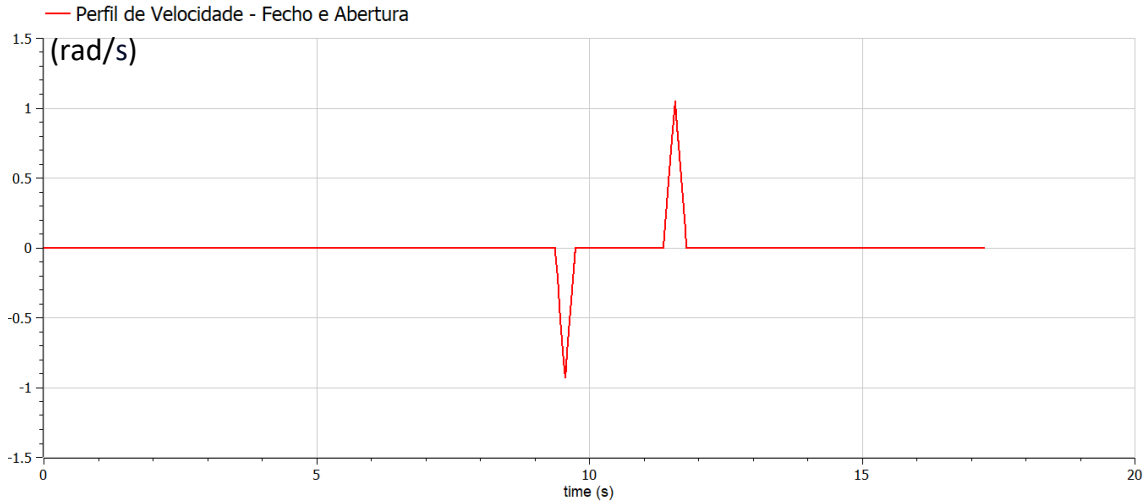


Figure 107 - Generated speed profiles for closing and opening the gripper

The velocity profile calculated in the trajectory generation models within the gripper control system adopts a triangular velocity profile. It can be considered that the displacement performed in the two phases is not large enough to reach the defined maximum velocity (2.5 rad/s) and make up a trapezoidal profile.

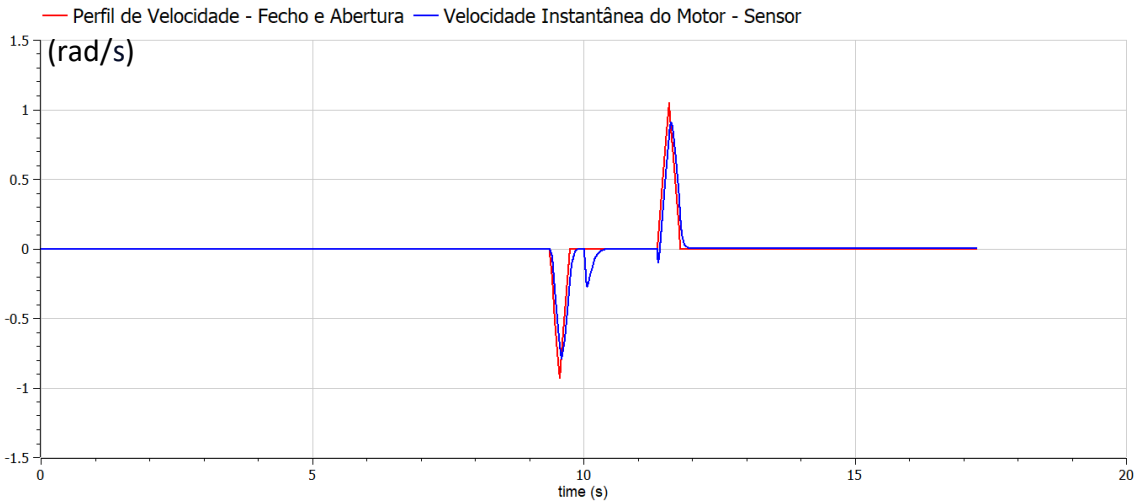


Figure 108 - Generated and measured speed profile

In Figure 108 it is possible to observe the profile of closing and opening speeds and the speed measured in the motor. It is easy to see a certain gap between the desired speed and the speed obtained, the result of a control focused exclusively on speed and understood to be sufficient for the desired purpose. Between the two triangular profiles it is also possible to observe a variation of speed, resulting from the phase intended for the force control, in which a small angular displacement is guaranteed so that the package is gripped firmly.

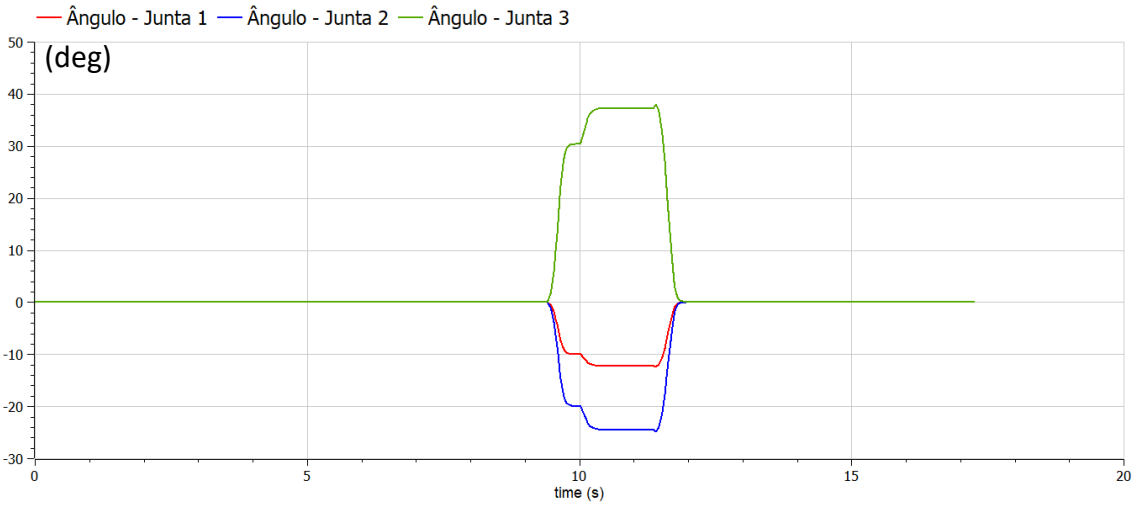


Figure 109 - Angular variation of the 3 joints present in each finger

Observing the angular variation values of the different joints in the gripper it is possible to observe a variation in an initial phase, coinciding with the closure of the gripper. When the speed is close to absolute 0 and the reference position is reached, then there is a new variation, this time of smaller size, representative of the force profile to be applied. Once the predefined acting force is reached, a period of zero variation follows, in which the package is duly transported by the robotic arm and, finally, placed on the exit conveyor through the gripper opening, in which an angular variation, in the opposite direction to the previous phases, is verified.

Between the transport phase/zero angular variation and the gripper opening, it is possible to observe a little protuberance, easily explained by the abrupt change in the spring properties, modeled with the intention of replicating the force to be applied on the package to be transported.

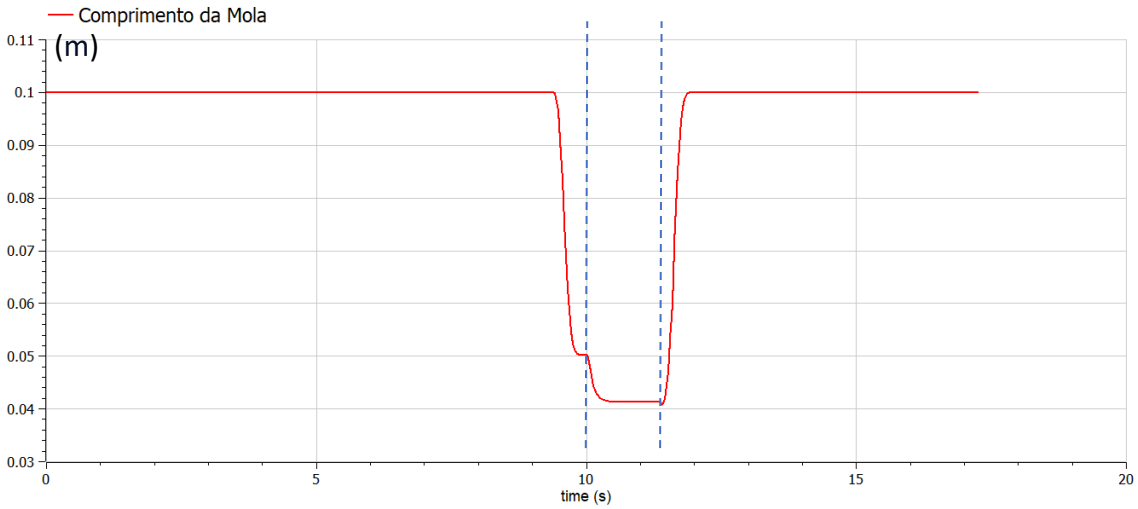


Figure 110 - Variation of the damping spring length

Variation of the spring length, where the initial and final moments of length variation may be discarded, as they correspond to the moments when the physical properties of the spring are equal to 0 and thus try to replicate a free movement of the joints and phalanges, without any elastic constriction or damping caused by the spring.

3. SIMULATION OF THE PHYSICAL ENVIRONMENT

Once the modeling of the overall system is finished, the presentation of the physical simulation for an operation cycle will be presented, fractioned into different steps. Before going on to a set of steps that allow the understanding of the simulation environment, the description with the set of constituent components of the system will be illustrated.

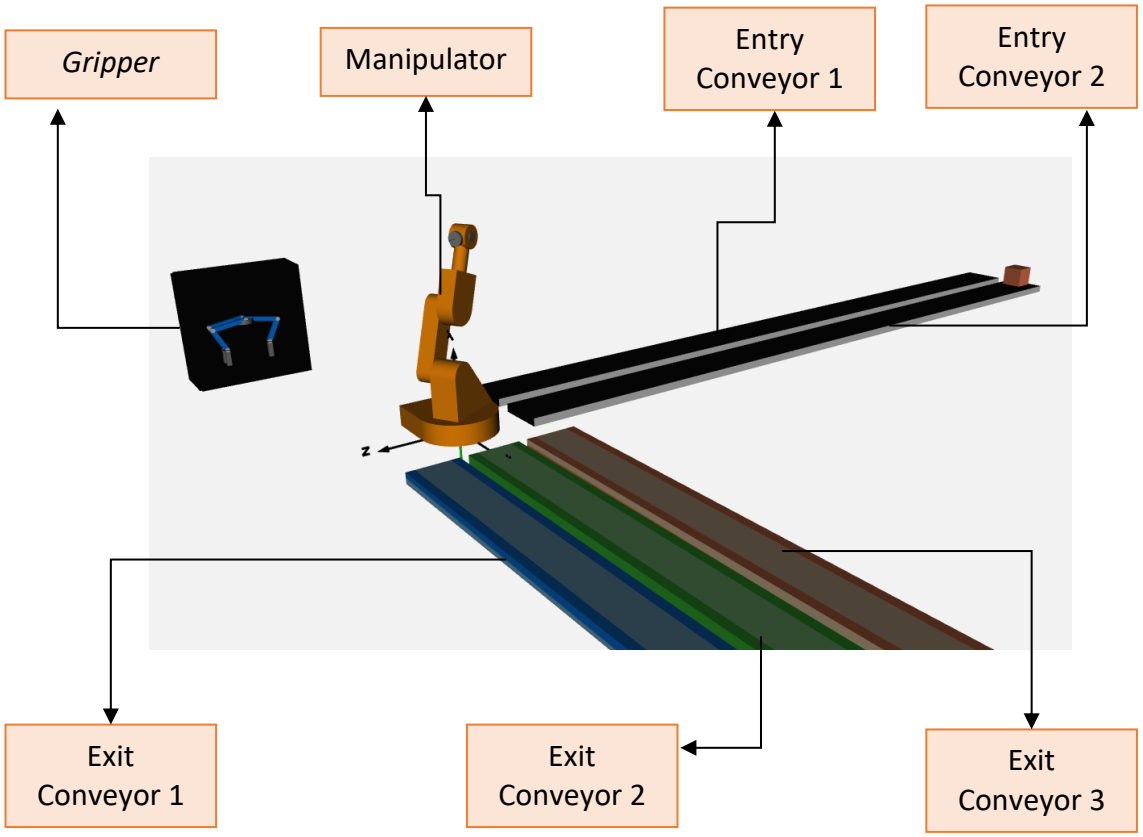


Figure 111 - "Animated" modeled components. Physical representation of the components in the simulation.

3.1. Box Transportation - Entry Conveyor Belt

Presentation of the set of steps involved in the operation of the Pick & Place system entry conveyor, responsible for transporting the package to the collection point.

- Start of the transport process.

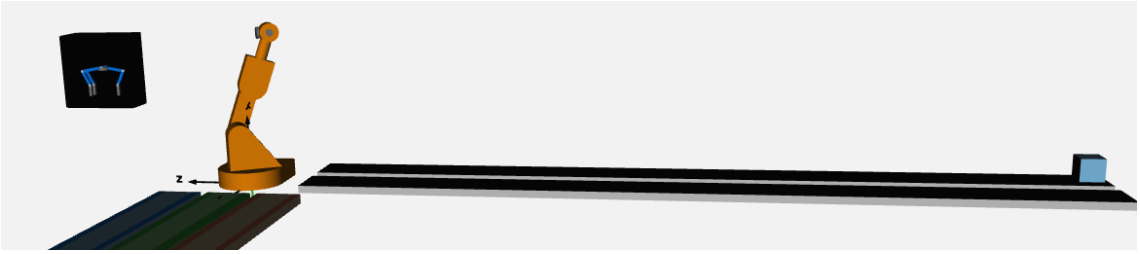


Figure 112 - Entry Conveyor 1 - Box 1 - Initial Rest

- Carrying the Package along the entire length of the conveyor.

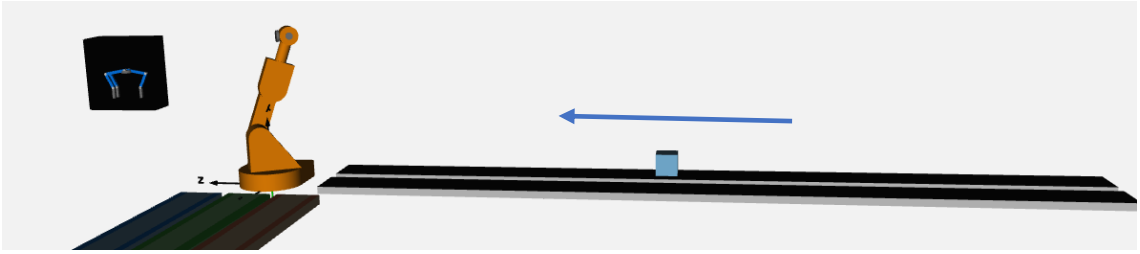


Figure 113 - Entry Conveyor 1 - Box 1 - Motion State

- End of transportation, with packaging allocation at the collection point.

Figure 114 - Entry Conveyor 1 - Box 1 - Final Rest

3.2. Movement of the Manipulator – *Pick*

Presentation of the set of steps for the package pickup by the manipulator, with a movement of the different axes dictated by the inverse kinematics of the pickup/ collection point.

- Start of the Pick motion still with the manipulator at the initial instant, defined by a random spatial configuration.

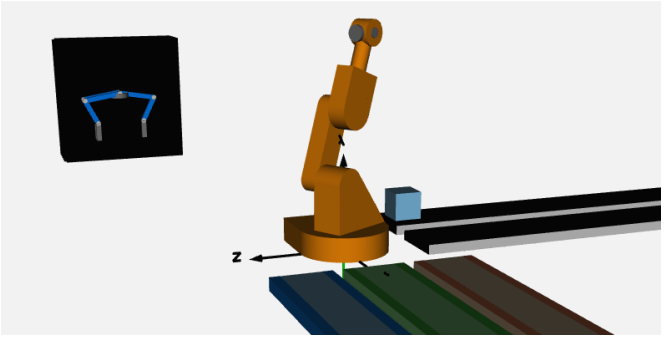


Figure 115 - *Pick* - Conveyor 1 - Box 1 – Initial Rest

- *Pick* motion with variation of the spatial configuration over time, through the angular variation of the different axes.

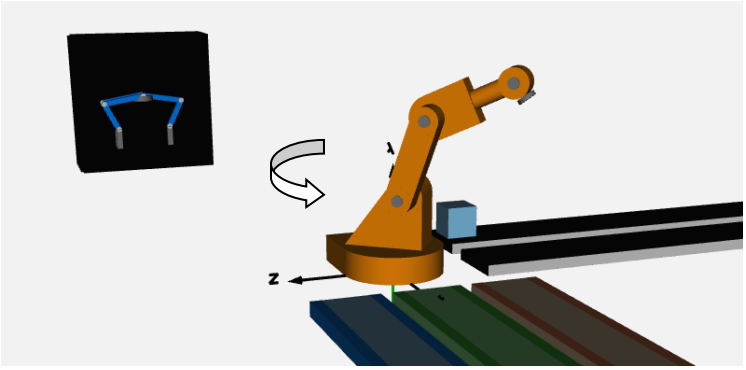


Figure 116 - *Pick* - Conveyor 1 - Box 1 - Motion

- Final configuration for collection achieved.

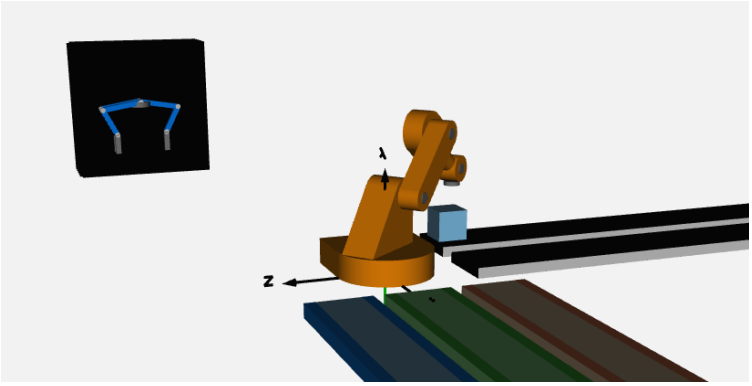


Figure 117 - *Pick* - Conveyor 1 - Box 1 - Final Rest

3.3.Gripper Movement – Closing

Presentation of the set of steps for closing the gripper and the respective collection of the package.

- Gripper still in its initial resting state, without the closure having been started yet.

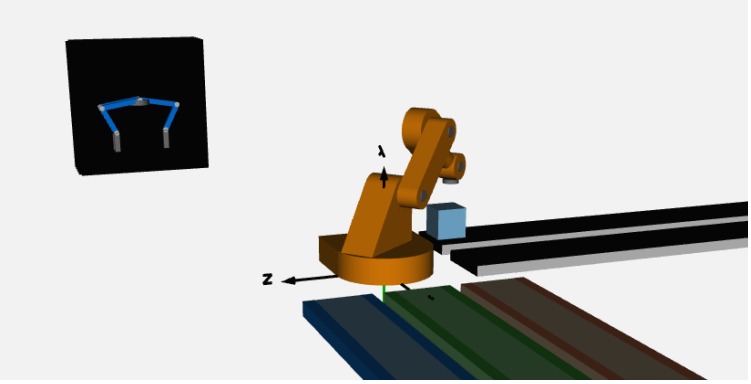


Figure 118 – Gripper Movement - Start Position

- Closure movement initiated by the gripper

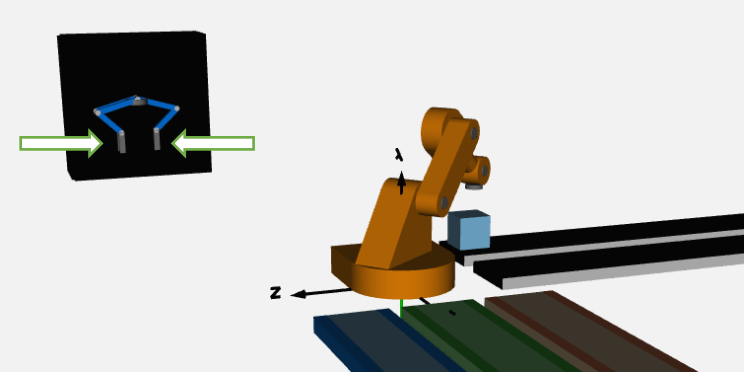


Figure 119 – Gripper Movement - Closing

- Force component initiated in the gripper allowing the package to be gripped.

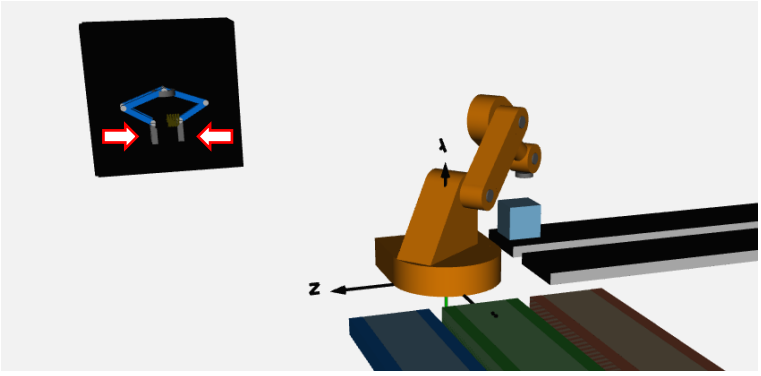


Figure 120 - *Gripper* – Power movement

3.4.Movement of the Manipulador - *Place*

Presentation of the set of steps for the package delivery by the manipulator, with a movement of the different axes dictated by the inverse kinematics involved between the pickup/collection point and the deposition point.

- Start of the Place movement with the package being grabbed and transported. In this phase the package visually disappears in the simulation, avoiding any kind of incompatibility to the understanding of the operation.

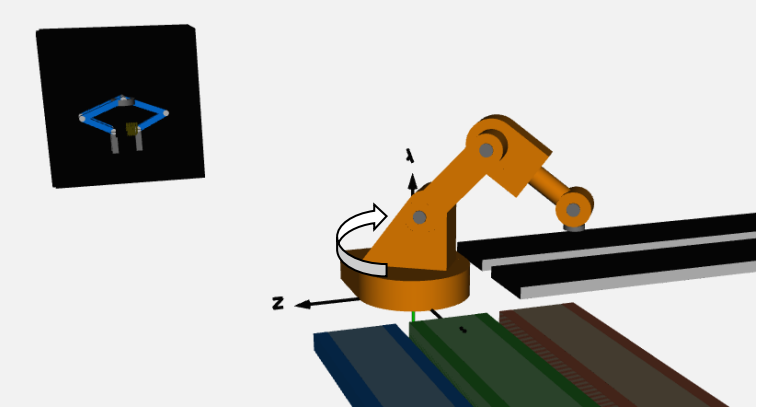


Figure 121 - *Place* - Conveyor 1 - Box 1 - Motion

- Place movement completed.

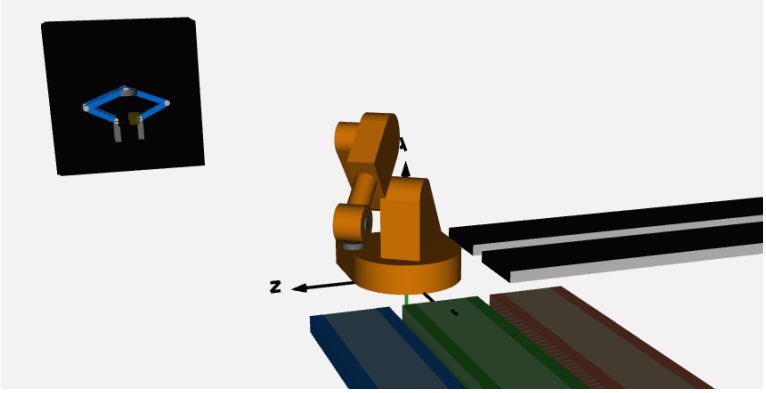


Figure 122 - *Place* - Conveyor 1 - Box 1 –Final Rest

3.5.Gripper Movement – Opening

Presentation of the set of steps for opening the gripper and the respective deliver the package.

- Opening movement initialized.

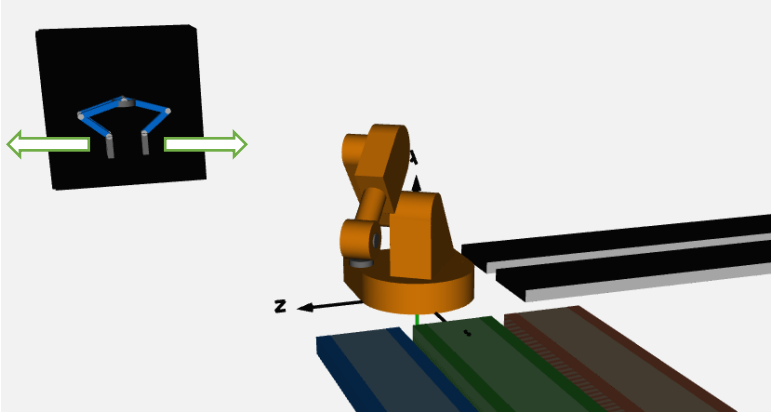


Figure 123 – Gripper Movement - Opening

- The opening movement is completed by depositing the package on the output conveyor.

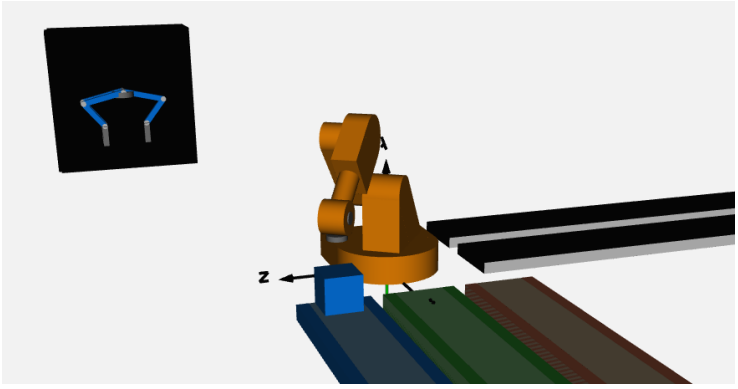


Figure 124 - Gripper - Finished opening and deposition of the packaging

3.6. Box Transportation – Exit Conveyor Belt

Presentation of the set of steps involved in the operation of the Pick & Place system exit conveyor, responsible for transporting the package to the deposition point.

- Start of exit transportation process.

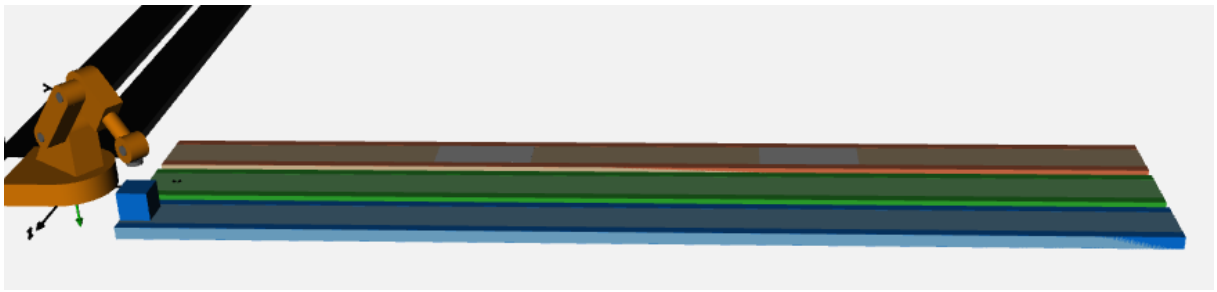


Figure 125 – Exit Conveyor 1 - Box 1 - Initial Rest

- Carrying the Package along the entire length of the conveyor.

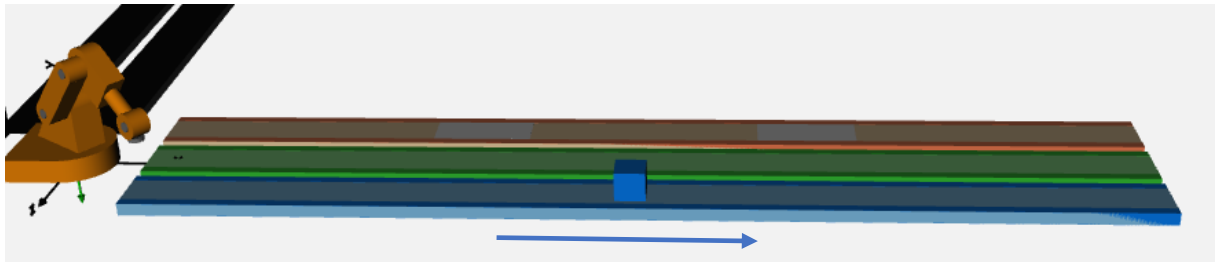


Figure 126 – Exit Conveyor 1 - Box 1 – Motion State

- End of transportation, with allocation of the package off the conveyor in its entirety, allowing it to be deposited for further processing. At this stage the simulation ends.

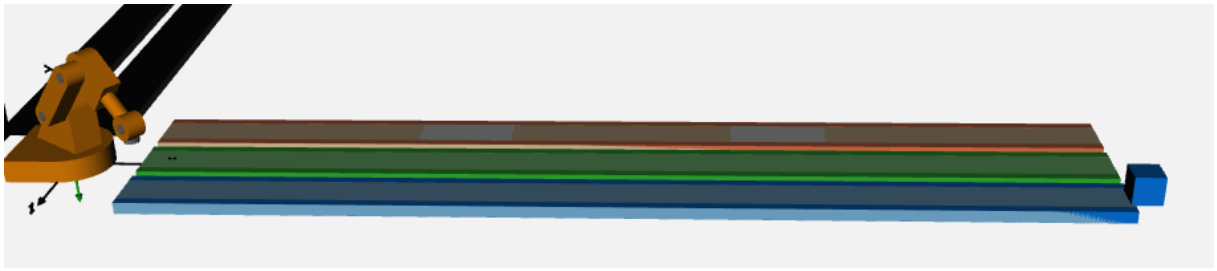
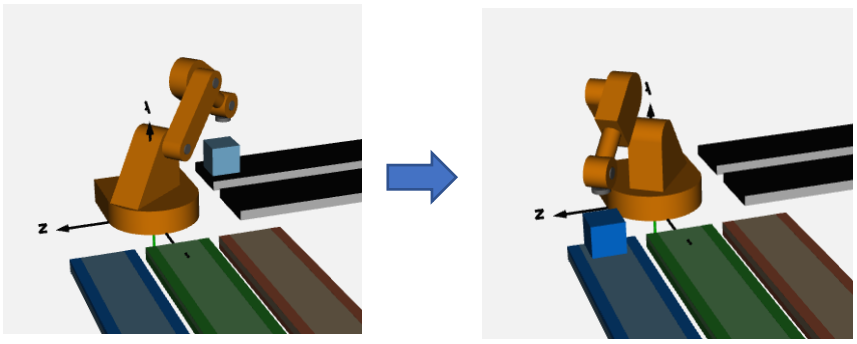
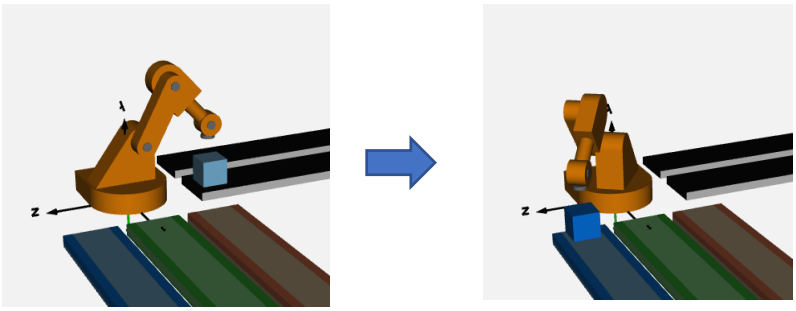
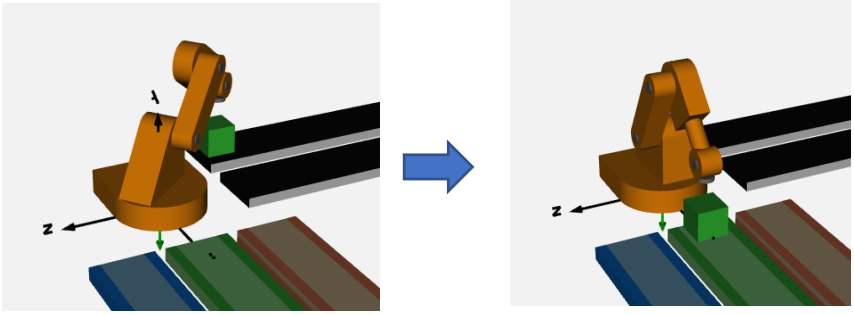
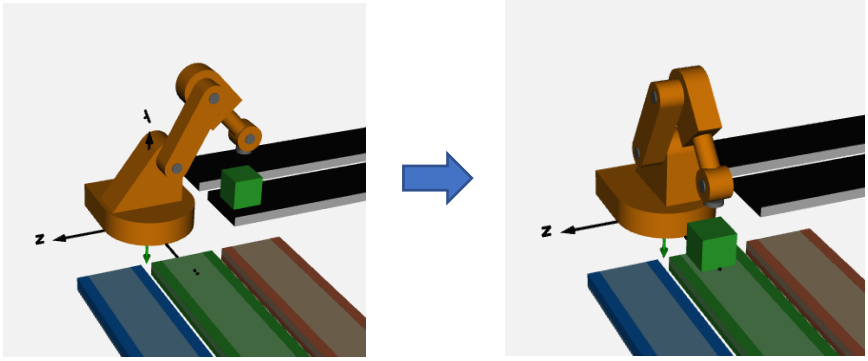
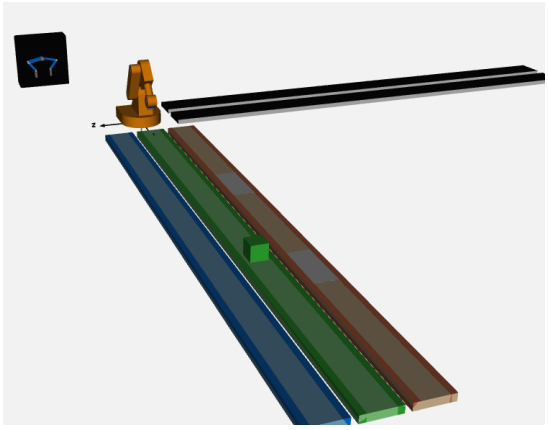
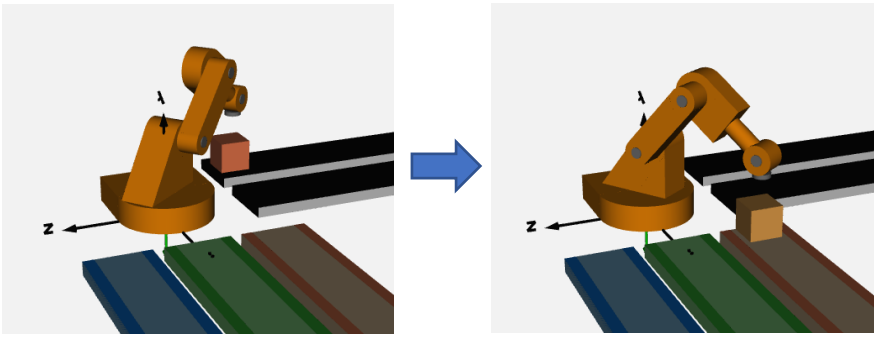
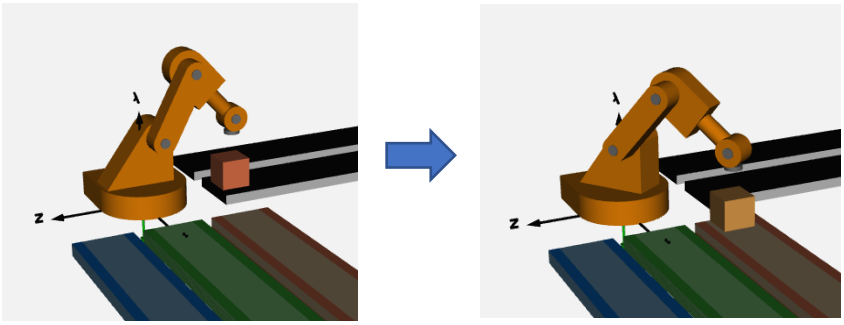


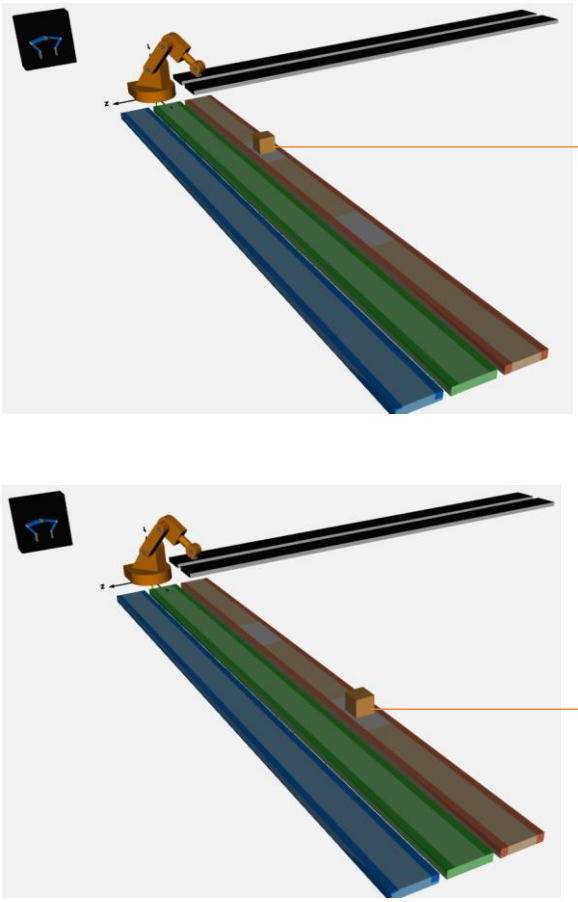
Figure 127 – Exit Conveyor 1 - Box 1 – Final Deposition

With two entry conveyors and three exit conveyors, the combination of possible *Pick&Place* trajectories amounts to 6 distinct trajectories (which extends to more if the option with *Via Points* is considered). The illustrations inserted in the following table try to give a synthesized portrait of the characteristic route for each combination.

Table 20 - *Prints* illustrating the routes involved in other options provided

Process	Simulation *Prints*
Pick&Place **Entry Conveyor 1** **Box 1**	
Pick&Place **Entry Conveyor 2** **Box 1**	
Pick&Place **Entry Conveyor 1** **Box 2**	
Pick&Place **Entry Conveyor 2** **Box 2**	

Exit Conveyor **Box 2**	
Pick&Place **Entry Conveyor 1** **Box 3**	
Pick&Place **Entry Conveyor 2** **Box 3**	

Exit Conveyor **Box 3**	

It is possible to verify in the graphs below the relative duration for each of the package transport phases. The duration of the entry conveyor phase presents a longer duration in entry conveyor 2, due to the losses considered. The *Pick&Place* transport duration presents a longer execution duration when both the entry conveyor 1 and the package 1 are used, due to a longer distance involved in the total route. The exit conveyor duration, on the other hand, exhibits a longer time when package 3 is transported and exit conveyor 3 is actuated, due to the stops involved.

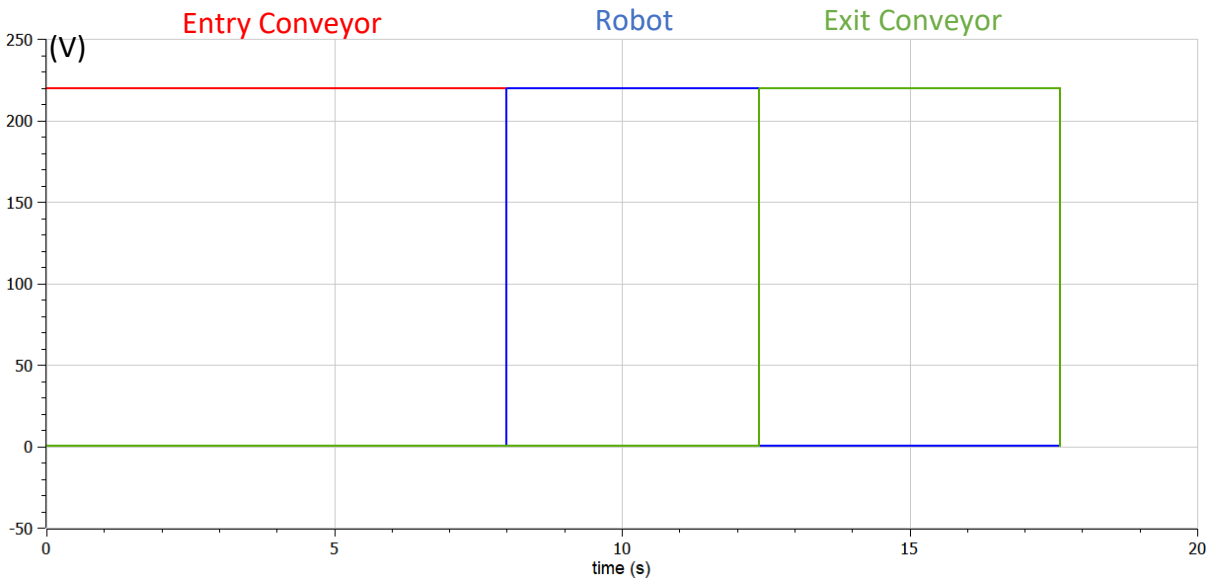


Figure 128 - Voltages present in the signal lamps associated with each of the conveyor phases. Entry conveyor belt 1 and box 1

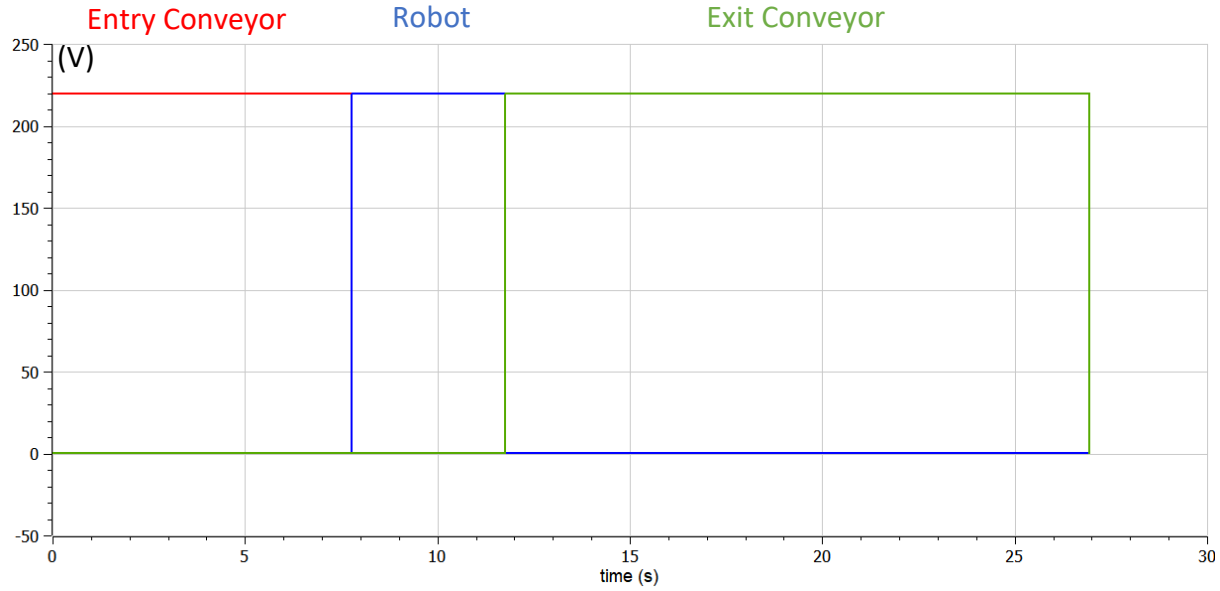


Figure 129 - Voltages present in the signal lamps associated with each of the conveyor phases. Entry conveyor belt 1 and box 3

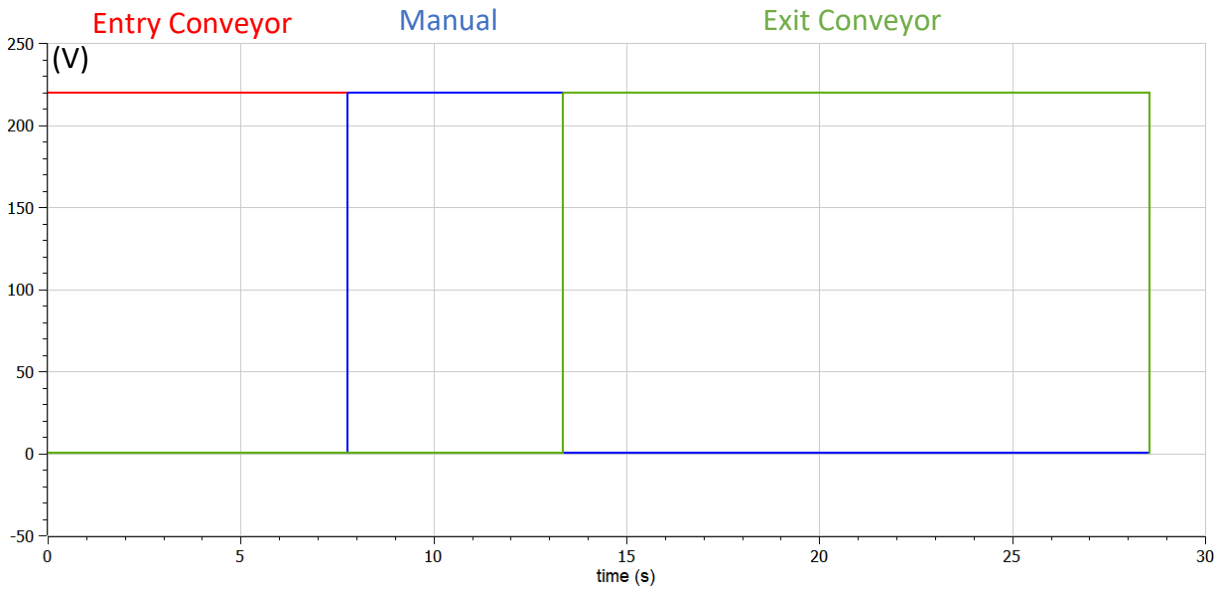


Figure 130 - Voltages present in the signal lamps associated with each of the transport phases. Entry conveyor belt 1 and Box 3 with manual *Pick&Place* transport

4. CONCLUSIONS AND FUTURE WORK

4.1. Conclusions

Developed Project and *Modelica* Language

The developed project allowed to enhance the intrinsic potentialities of object-oriented modeling. The comparison between a system simulated in *Matlab/Coppelia* (user's manual) and in *OpenModelica*, allowed us to understand the different types of simulations that can be reproduced that portray an industrial process, as well as bring more relevance to the aspects that can be explored with the *Modelica* language.

One of the main aspects to retain is undoubtedly the versatility presented by the language, where throughout the elaboration of the system, according to an object-oriented modeling, the broad spectrum of possible underlying configurations is clearly evident, revealing a very complete study. The intention to model a multi-physics system using *Modelica* has been achieved, allowing the ability to bring together:

- A very comprehensive branch of different areas.
- Development of both physical design and control design in a single language, projecting not only the ability to develop diagrams, but also programmed code. At the most fundamental level possible, the integrity of the modeling ends up being all programmed code, however, the ability to instantiate models makes it possible to simplify and shorten the direct use of that same code, since significant portions of the code are already gathered in the form of blocks/models, and the elaboration of diagrams through the association of those models allows to focus only on equations that associate the models, through the transmission of variables.
- A hybrid conjecture that admits states or discretizations, as well as continuous functions, allowing a much wider range of development, such as a much more complete and sophisticated control capability.
- Allied to the hierarchization and organization of the theoretical principles involved in the modeling, as well as the presentation of the values of the variables over time, it is possible to physically simulate the environment. Although the simulation has some limitations compared to other existing software programs, it is still an important aspect, since it highlights the visual aspects involved in the physical process.

- Presentation of graphical results, which determine the behavior of the different variables present in the modeled system. This particularity, however, greatly determined the approach taken in refining the modeling. The opportunity to simulate and analyze the variation of the variables values, simultaneously with the design of the models, brings a greater insight in the development of the system. Modeling errors or process interpretation errors are easier to detect based on a study of the graphical evolution of the different variables. The theoretical assumption of the system's behavior allows the detection of a misplaced evolution, resulting from a modeling with some lapses. A previous understanding of the system's behavior thus allows not only a more realistic design, having as a basis for comparison the real replicated system, but also a greater ability to detect dysfunctions in the simulation due to projected modeling errors.

Developed *Pick&Place* System

The development of the modeling of a *Pick&Place* system using the *Modelica* language, as basis for starting studies, proved to be, on the one hand, a complex task due to the multi-polarity of domains involved in the modeling, whose complexity is increasing as a function of the rigorousness sought in the development of the different models, or a broader scope of the engineering domains to be quantified and on the other hand, the use of the language allowed to counterbalance these demands by its capacity to support not only the design of the physical system, without imposing any kind of artificial limits that restrict the multi-domain use, but also the design of the control system.

As for the manipulator, regarding the analysis of the results obtained in the different subsystems, the differences that stand out in relation to the reference model, taken from the library present in the language, dictate fundamentally a more refined modeling of the manipulator control (replicating a movement delineation, proper of the associated main computer), allowing a closer study of the paths involved in an industrial environment, as well as the study of all variables involved in the movement. The torques, and consequently the heating of the motors resulting in thermal dissipation, present peaks for higher values of tool mass (gripper), as expected, and for a 5th order polynomial velocity profile, explained by high velocity variations.

In modeling the conveyor belts, it was important to make an exposition of the different modeling options for the same architecture, and each of the differences were implemented according to the specific function of each conveyor. The modeling of the losses involved is an important factor to take into consideration for its significant impact on performance and, in parallel with the conclusions drawn with the motion profiles implemented in the robot, it was possible to realize that the heat transfer, as a consequence of the total torque covered, sees its values varying depending on the type of motion profile implemented, and whose determining factors involved are the kinematic constrictions defined in the profiles as well as the total range of motion.

In modeling the gripper, the different commands to be admitted, in different instances in the control loop, was undoubtedly the most relevant aspect, compared to the control of the manipulator and conveyors. The modeling of the mechanical structure of each of the fingers was also highlighted by the presence of a set of important kinematic constraints to ensure a smooth operation of the package pickup and deposit.

4.2. Future Works

Although some relevant work is presented and important achievements were possible, there is, still, a wide range of studies that can be developed in this domain. Some of them – in a global and specific view - are:

- Redefine some models, trying to design instantiated models that can be used in the modeling of other systems. Most of the models were adapted exclusively to the modeling of the present system, so they lack some versatility in terms of subsequent use. The specificity of most of the models is thus high, but the modifications required are, however, not significant, pending only on details (fundamentally at the level of inputs, outputs and parameterization).
- More focus on the values involved that are parameterized throughout the model set. The focus of attention throughout the project was more on the correct development of models, often putting the values of the parameters involved to a secondary place. It is clear that no model, no matter how well designed, will translate into an approximate replication of a real process, if the parameters defined and that characterize the system are not well designed. A review of these values, accompanied

by a detailed study that addresses the limitations they may offer as well as existing standardizations, should be carried out.

- The manipulator-gripper connection should be solved, allowing a better framed study between the two systems and the implications that one presents to the other, mainly the weight of the gripper on the manipulator's structure.
- A general improvement of the subsystems, based on the outlined global architecture.

5. BIBLIOGRAPHYC REFERENCES

[1] A. Polenghi, L. Fumagalli e I. Roda, "Role of simulation in industrial engineering: focus on manufacturing systems," *IFAC PapersOnLine,* pp. 496-500, 2018.

[2] J. S. Smith, "Survey on the use of simulation for manufacturing system design and operation," *Journal of Manufacturing Systems,* pp. 157-171, 2003.

[3] A. Maria, "Introduction to modeling and simulation," *Proceedings of the 29th conference on Winter simulation,* pp. 7-13, 1997.

[4] J. Banks, Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, John Wiley & Sons, 1998.

[5] S. Ergasheva e A. Kruglov, "Software Development Life Cycle early phases and quality metrics: A Systematic Literature Review," *Journal of Physics: Conference Series,* vol. 1964, nº 012007, 2020.

[6] P. Fritzson, Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica, IEEE Press, 2011.

[7] "OpenModelica," [Online]. Available: https://openmodelica.org/. [Acedido em 10 April 2023].

[8] M. M.Tiller, "Modelica by Example - Electronic Interactive Book, first published in 2014 and continuously updated since," 2014. [Online]. Available: https://mbe.modelica.university/. [Acedido em 16 May 2022].

[9] H. B. P, S. R. Bharat e S. K. Srinivas, "Design of Pick and Place Robot Test Rig," *International Journal of Engineering Research & Technology (IJERT),* vol. 2, nº 12, pp. 982-991, 2013.

[10] Z. Lv e E. Fersman, Digital Twins: Basics and Applications, Springer, 2022.

[11] A. A. Jahromi e D. Kundur, "Fundamentals of Cyber-Physical Systems," em *Cyber-Physical Systems in the Built*, Springer, 2020, pp. 1-13.

[12] P. Fritzson e V. Engelson, "Modelica - A Unified Object-Oriented Language for System Modeling and Simulation," em *Lecture Notes in Computer Science*, Linköping, 1998.

[13] D. Brück, H. Elmqvist, S. E. Mattsson e H. Olsson, "Dymola for Multi-Engineering Modeling and Simulation," em *2nd International Modelica Conference*, Oberpfaffenhofen, 2002.

[14] P. Fritzson, P. Aronsson, H. Lundvall e K. Nyström, "The OpenModelica Modeling, Simulation, and Development Environment," em *In Simulation News Europe*, Linköping, 2005.

[15] ".Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.FullRobot," OpenModelica, [Online]. Available: https://build.openmodelica.org/Documentation/Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.FullRobot.html. [Acedido em 10 April 2023].

[16] E. B. Erlhagen, “Kinematic analysis of Industrial Serial Robots with 6 DoF,” em *Case study: Yaskawa Motoman MH5*, Guimarães, 2020.

[17] A. A. Hayat, R. Chittawadigi, A. D. Udai e S. K. Saha, “Identification of Denavit-Hartenberg Parameters of an Industrial Robot,” em *AIR '13: Proceedings of Conference on Advances In Robotics*, Pune, India, 2013.

[18] R. ABB AB, Product specification - IRB 140 (Revision: H), Västerås, 2004-2017.

[19] J. J. Craig, Introduction to Robotics : Mechanics and Control, Pearson Prentice Hall , 2005.

[20] B. Siciliano, L. Sciavicco , L. Villani e G. Oriolo , Robotics : Modelling, Planning and Control, Springer Science & Business Media, 2010.

[21] “.Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Utilities.PathPlanning6,” OpenModelica, [Online]. Available: https://build.openmodelica.org/Documentation/Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Utilities.PathPlanning6.html. [Acedido em 10 April 2023].

[22] M. Otter, “.Modelica.Blocks.Sources.KinematicPTP2,” OpenModelica, [Online]. Available: https://build.openmodelica.org/Documentation/Modelica.Blocks.Sources.KinematicPTP2.html. [Acedido em 10 April 2023].

[23] H. J. Yoon, S. Y. Chung, H. S. Kang e M. J. Hwang, “Trapezoidal Motion Profile to Suppress Residual Vibration of Flexible Object Moved by Robot,” *Electronics ,* vol. 8, 2019.

[24] R. Zhao, D. Sidobre e W. He, “Online Via-Points Trajectory Generation for Reactive Manipulations,” em *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Besançon, 2014.

[25] E. B. Erlhagen, “Trajectory Generation in robotic manipulators,” Guimarães, 2020.

[26] “IRB 140 - ABB,” ABB, [Online]. Available: https://new.abb.com/products/3HAC020536-001/irb-140. [Acedido em 10 April 2023].

[27] M. M.Tiller, “Multi-Domain Modeling - Conveyor System,” em *Introduction to Physical Modeling with Modelica*, Kluwer Academic Publishers, 2001, p. 232.

[28] “.Modelica.Electrical.Machines.Examples.DCMachines.DCPM_Cooling,” OpenModelica, [Online]. Available: https://build.openmodelica.org/Documentation/Modelica.Electrical.Machines.Examples.DCMachines.DCPM_Cooling.html. [Acedido em 10 April 2023].

[29] G. Ferretti, G. Magnani, P. Rocco e L. Viganò, “Modelling and simulation of a gripper with Dymola,” *Mathematical and Computer Modelling of Dynamical Systems,* vol. 12, 2006.

APPENDIX 1 – TRAPEZOIDAL PROFILE IMPLEMENTATION

Having studied the theoretical assumptions associated with the generation of trapezoidal kinematic profiles, it's now explained their practical implementation in the model. Basically, the script regarding the trapezoidal motion profile is the same present in the trajectory model already inserted in the provided global reference model. There is however, a total lack of information regarding the explanation of the code, so throughout this chapter it will be presented. The explanation will be elaborated by presenting, in order, different excerpts or sections of the script.

```
equation
  for i in 1:nout loop
    aux1[i] = p_deltaq[i] / p_qd_max[i];
    aux2[i] = p_deltaq[i] / p_qdd_max[i];
  end for;
  sd_max_inv = max(abs(aux1));
  sdd_max_inv = max(abs(aux2));
  if startTime == 0 then
    sd_max = 0;
    sdd_max = 0;
    Ta1 = 0;
    Ta2 = 0;
    noWphase = false;
    Tv = 0;
    Te = 0;
    Ta1s = 0;
    Ta2s = 0;
    Tvs = 0;
    Tes = 0;
    sd_max2 = 0;
    s1 = 0;
    s2 = 0;
    s3 = 0;
    s = 0;
```

Figure 131 - First excerpt of the code involved in the kinematic profile generation model

In an initial phase, 2 vectors are calculated for each element of each axis, one corresponding to the ratio between the angular difference and the maximum speed (thus consisting of the travel time if the maximum speed covered the entire travel time) and another concerning the ratio between the angular difference and the acceleration. From these 2 vectors, only the element with the highest value will be extracted later. Taking into account the required displacement and kinematic constraints for each joint, it is easy to deduce that the displacement times involved vary for each axis, in accordance with the corresponding constraints. However, a smooth motion is desired ensuring an equal travel time for all joints. The selection of the higher value in each of the 2 vectors mentioned, guarantees not only an equal travel time for all joints, but also ensures that this same time is sufficient for the consummation of the trapezoidal motion profile in all axes.

The remaining parameters present in the excerpt will be described in the following Table 21. Each of these parameters has a value of 0 if the *startTime* = 0 condition is verified, that is, if the instant at which it's wanted the trajectory command to be calculated, has not yet been reached.

Table 21 - Description of the parameters present in the model excerpt

Variable	Description
sd_max	Inverse of the ratio between angular amplitude and maximum speed, the value of this ratio being the maximum value acquired along the 6 axes
sdd_max	Inverse of the ratio between angular amplitude and maximum acceleration, the value of this ratio being the maximum value acquired along the 6 axes
Ta1	Acceleration time if the maximum speed is not reached, i.e. with a triangular velocity profile
Ta2	Acceleration time with maximum speed to be reached, i.e. with trapezoidal velocity profile and not triangular
noWphase	Boolean that when true portrays the existence of a triangular velocity profile, i.e., without constant phase
Tv	In the trapezoidal profile it equals the time along the section with constant velocity. In the triangular profile it equals the acceleration time
Te	Total travel time
Ta1s	Time instant at which the acceleration time (if maximum speed is not reached) ends, i.e. it is equivalent to acceleration time + *startTime*. Triangular profile.
Ta2s	Time instant at which the acceleration time (if maximum speed is reached) ends, i.e. it is equivalent to acceleration time + *startTime*. Trapezoidal profile.
Tvs	Time instant when the constant velocity ends (trapezoidal profile) or time instant when the acceleration phase ends (triangular profile, thus equaling ***Ta1s***)

Tes	Time instant at which the travel time ends, equals travel time + *startTime*
sd_max2	Maximum speed reached when facing a triangular speed profile
s1	Percentage of displacement occurred in the acceleration time
s2	Percentage of displacement that occurred in the time at which the constant speed is terminated
s3	Percentage of displacement occurred in the final time, which basically adopts the entire displacement, thus equivalent to the fixed value of 1
s	Percentage of displacement occurring over time and gradually adopting over time the values of de *s1, s2* and *s3* Thus varies between 0 and 1

In the event that the instant relative to the calculation is reached, the following excerpt shows the calculation developed for the parameters previously described.

```
else
  sd_max = 1 / max(abs(aux1));
  sdd_max = 1 / max(abs(aux2));
  Ta1 = sqrt(1 / sdd_max);
  Ta2 = sd_max / sdd_max;
  noWphase = Ta2 >= Ta1;
  Tv = if noWphase then Ta1 else 1 / sd_max;
  Te = if noWphase then Ta1 + Ta1 else Tv + Ta2;
  Ta1s = Ta1 + startTime;
  Ta2s = Ta2 + startTime;
  Tvs = Tv + startTime;
  Tes = Te + startTime;
  sd_max2 = sdd_max * Ta1;
```

Figure 132 - Second excerpt of the code involved in the kinematic profile generation model

For a better understanding of the calculations replicated in the excerpt above, they can be found in the following table.

Table 22 - Calculation of the parameters present in the model excerpt

Variable	Calculation
sd_max	$1 \Big/ \left(\frac{\Delta x}{v_{max}}\right)_{max}$
sdd_max	$1 \Big/ \left(\frac{\Delta x}{a_{max}}\right)_{max}$

Ta1	$\sqrt{\left(\frac{\Delta x}{a_{max}}\right)_{max}}$
Ta2	$\left(\frac{v_{max}}{a_{max}}\right)_{max}$
noWphase	True if $\left(\frac{v_{max}}{a_{max}}\right)_{max} > \sqrt{\frac{\Delta x}{a_{max}}}$
Tv	If ***noWphase*** then ***Tv = Ta1***, otherwise ***Tv =*** $\left(\frac{\Delta x}{v_{max}}\right)_{max}$
Te	If ***noWphase*** then ***Te*** = 2***Ta1***, otherwise ***Te*** = ***Tv*** + ***Ta2***
Ta1s ***Ta2s*** ***Tvs*** ***Tes***	***Ta1*** ***Ta2*** ***Tv*** ***Te*** ***+ startTime***
sd_max2	$\sqrt{\left(\frac{\Delta x}{a_{max}}\right)_{max}} \Big/ \left(\frac{\Delta x}{a_{max}}\right)_{max}$
s1	**Considerations** • ***s1 =*** $\frac{\boldsymbol{Distance\ traveled\ along\ the\ acceleration}}{\boldsymbol{Total\ Distance}}$ • If it's taken into account the trapezoidal velocity profile, the distance traveled is equivalent to the area of the triangular left half, so $d = \frac{1}{2} \times t_a \times v_{max}$ • It's also known that $v_{max} = a_{max} \times t_a$, so $d = \frac{1}{2} \times (t_a)^2 \times a_{max}$ Thus: If ***noWphase*** then, $\frac{\frac{1}{2} \times (t_a)^2 \times (a_{max})_{max}}{(\Delta x_{max})_{max}} = \frac{1}{2} \times (\boldsymbol{Ta1})^2 \times \boldsymbol{sdd\_max}$ Otherwise, $\frac{\frac{1}{2} \times (t_a)^2 \times (a_{max})_{max}}{(\Delta x_{max})_{max}} = \frac{1}{2} \times (\boldsymbol{Ta2})^2 \times \boldsymbol{sdd\_max}$

s2	• If it's taken into account the trapezoidal velocity profile, the distance traveled is equivalent to the rectangular area in the center of the trapezoidal section, so $d = t_{vel.const} \cdot v_{máx}$ If ***noWphase*** then, $$\frac{\sqrt{\left(\frac{\Delta x}{a_{max}}\right)_{max}}}{\left(\frac{\Delta x}{a_{max}}\right)_{max}} - \frac{\frac{1}{2}(a_{max})_{max}\,(\boldsymbol{Te}-\boldsymbol{Ta1})^2}{(\Delta x)_{max}}$$ Otherwise, $$\boldsymbol{s1} + \frac{(\boldsymbol{Tv}-\boldsymbol{Ta2})(v_{max})_{max}}{(\Delta x)_{max}}$$
s3	Note that for the calculation of s3, this corresponds to a section of the profile that only exists for the trapezoidal profile, and defines the third profile section. $$\boldsymbol{s2} + \left(\frac{v_{max}\cdot(t_d) - \frac{1}{2}\times(t_d)^2\times(a_{max})_{max}}{(\Delta x)_{max}}\right)$$ $$\boldsymbol{s2} + \left(\left(\frac{v_{max}}{\Delta x}\right)_{max} \times (\boldsymbol{Te} - \boldsymbol{Tv}) - \frac{\frac{1}{2}\times(\boldsymbol{Te}-\boldsymbol{Tv})^2\times(a_{max})_{max}}{(\Delta x_{max})_{max}}\right)$$ $$\boldsymbol{s2} + \left(\boldsymbol{sd_{max}} \times (\boldsymbol{Te} - \boldsymbol{Tv}) - \frac{1}{2} \times (\boldsymbol{Te} - \boldsymbol{Tv})^2 \times \boldsymbol{sdd\_max}\right)$$
s	Using the expressions of the functions shown in Table 3 and Table 4 and dividing them by the range of angular variation: **Trapezoidal Profile** If $\boldsymbol{t}(time) < \boldsymbol{Ta2s}$: $$\frac{\frac{1}{2}\times \boldsymbol{t}^2 \times \boldsymbol{a_{max}}}{(\Delta x)_{max}} = \frac{\boldsymbol{sdd\_max}}{\boldsymbol{2}} \times (time - startTime)^2$$ If $\boldsymbol{t}(time) > \boldsymbol{Ta2s}$ e $\boldsymbol{t}(time) < \boldsymbol{Tvs}$: $$\frac{\boldsymbol{p_2} + \boldsymbol{v_{max}} \times (\boldsymbol{t} - \boldsymbol{t_a})}{(\Delta x)_{max}} = \boldsymbol{s1} + \boldsymbol{sd\_max}\,(time - \boldsymbol{Ta2s})$$ If $\boldsymbol{t}(time) > \boldsymbol{Tvs}$ e $\boldsymbol{t}(time) < \boldsymbol{Tes}$: $$\frac{\boldsymbol{p_3} + \frac{\boldsymbol{1}}{\boldsymbol{2}} \times (\boldsymbol{t} - \boldsymbol{t_a} - \boldsymbol{t_{const}})^2 \times \boldsymbol{a_{max}}}{(\Delta x)_{max}}$$

$$= \frac{\boldsymbol{p_3} + \boldsymbol{v_{max}} \times (\boldsymbol{t} - \boldsymbol{t_a} - \boldsymbol{t_{const}}) - \frac{\boldsymbol{1}}{\boldsymbol{2}} \times (\boldsymbol{t} - \boldsymbol{t_a} - \boldsymbol{t_{const}})^{\boldsymbol{2}} \times \boldsymbol{a_{max}}}{(\Delta x)_{max}}$$

$$= \boldsymbol{s2} + \boldsymbol{sd\_max}(time - \boldsymbol{Tvs}) - \frac{\boldsymbol{sdd\_max}}{2 \times (time - \boldsymbol{Tvs})^2}$$

Triangular Profile

If $\boldsymbol{t}(time) < \boldsymbol{Ta1s}$:

$$\frac{\frac{1}{2} \times \boldsymbol{t^2} \times \boldsymbol{a_{max}}}{(\Delta x)_{max}} = \frac{\boldsymbol{sdd\_max}}{\boldsymbol{2}} \times (time - startTime)^2$$

If $\boldsymbol{t}(time) > \boldsymbol{Ta1s}$ and $\boldsymbol{t}(time) < \boldsymbol{Tes}$:

$$\frac{\boldsymbol{p_2} + \frac{\boldsymbol{1}}{\boldsymbol{2}} \times (\boldsymbol{t} - \boldsymbol{t_a'})^{\boldsymbol{2}} \times \boldsymbol{a_{max}}}{(\Delta x)_{max}}$$

$$= \frac{\boldsymbol{p_2} + \boldsymbol{v'_{max}} \times (\boldsymbol{t} - \boldsymbol{t_a'}) - \frac{\boldsymbol{1}}{\boldsymbol{2}} \times (\boldsymbol{t} - \boldsymbol{t_a'})^{\boldsymbol{2}} \times \boldsymbol{a_{max}}}{(\Delta x)_{max}}$$

$$= \boldsymbol{s1} + \boldsymbol{sd\_max2}(time - \boldsymbol{Ta1s}) - \frac{\boldsymbol{sdd\_max}}{2 \times (time - \boldsymbol{Ta1s})^2}$$

Once each of the reference positions (*s1*, *s2* and *s3*) of the configured space (between 0 and 1) have been formulated, they are implemented in the model.

```
s1 = sdd_max * (if noWphase then Ta1 * Ta1 else Ta2 * Ta2) / 2;
s2 = s1 + (if noWphase then sd_max2 * (Te - Ta1) - sdd_max / 2 * (Te - Ta1) ^ 2 else sd_max * (Tv - Ta2));
s3 = s2 + sd_max * (Te - Tv) - sdd_max / 2 * (Te - Tv) * (Te - Tv);
if time < startTime then
  s = 0;
elseif noWphase then
  if time < Ta1s then
    s = sdd_max / 2 * (time - startTime) * (time - startTime);
  elseif time < Tes then
    s = s1 + sd_max2 * (time - Ta1s) - sdd_max / 2 * (time - Ta1s) * (time - Ta1s);
  else
    s = s2;
  end if;
elseif time < Ta2s then
  s = sdd_max / 2 * (time - startTime) * (time - startTime);
elseif time < Tvs then
  s = s1 + sd_max * (time - Ta2s);
elseif time < Tes then
  s = s2 + sd_max * (time - Tvs) - sdd_max / 2 * (time - Tvs) * (time - Tvs);
else
  s = s3;
end if;
```

Position functions (configured space) - Trapezoidal profile

Position functions (configured space) - Triangular profile

Reference positions (1,2 and 3) - Trapezoidal and triangular profile.

Figure 133 - Third excerpt of the code involved in the kinematic profile generation model

Once calculated the relative positions s according to the configured space (variation between 0 and 1), it's now converted the path into a trajectory (definition of s as a function over time) by multiplying the values obtained by the angular variation in each of the axes.

```
    sd = der(s);
    sdd = der(sd);
    qdd = p_deltaq * sdd;
    qd = if Polinomial then vel_juntas elseif Polinomial_5th then vel_juntas_5th else p_deltaq * sd;
    q = if Polinomial then juntas elseif Polinomial_5th then juntas_5th else p_q_begin + p_deltaq * s;
    endTime = Tes;
  // report when axis is moving
    motion_ref = time < endTime;
    for i in 1:nout loop
      moving[i] = if abs(q_begin[i] - q_end[i]) > eps then motion_ref else false;
    end for;
    annotation( ...);
  end Cinematica;
```

Functions of position, velocity, and acceleration at different time instants

Figure 134 - Fourth excerpt of the code involved in the kinematic profile generation model

APPENDIX 2 – MODELING OF ROBOT CONTROL (STRUCTURE)

Once a control technology like *IRC5* has been analyzed, it is important to incorporate some of its inherent control principles into the modeling of the robotic arm control system to be designed. It is easy to see that modeling a system with this technology, following each of the sections and components discussed, requires a very detailed global model and a certain multi-domain complexity.

In order to preserve the multi-domain property that a control arm technology requires, but seeking to explore a more simplified set of models, the *Modelica* modeling of the robotic arm aimed at a series of modifications and simplifications, of which stand out:

- The *AC* motor will be replaced by a simplified *DC* motor, and with it, a downgrading of the power section, consisting of the rectifier, *DC - Link* and also *DC-AC* inverter.
- Current control using the *SVPWM* (Space Vector Pulse Width Modulation) modulation technique is replaced by a current control using operational amplifiers.
- A position control is added, thus allowing a simpler, but compact control.
- The resolver as well as the whole position calculation system present in the *SMB* is replaced by a position and speed sensor present in the *Modelica Standard Library*.
- Introduction of a current sensor.

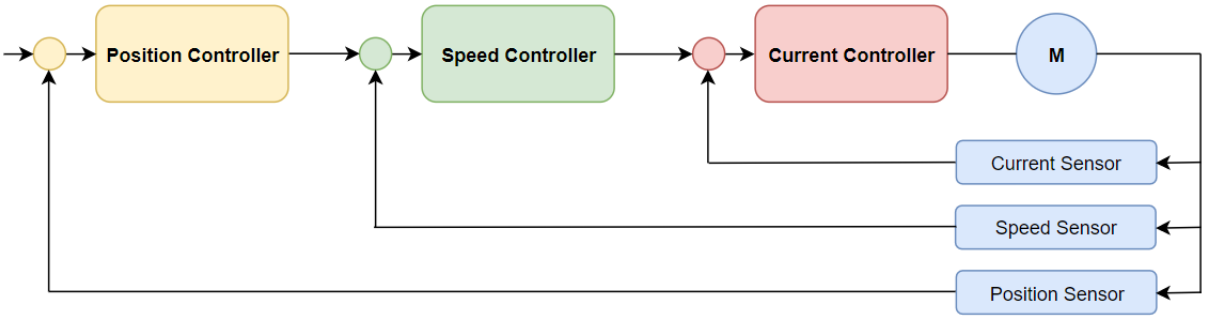


Figure 135 - Simplified control system schematization and easier adaptation to object-oriented language

APPENDIX 3 – AXIAL CONTROL AND ACTUATOR

To control the motor in accordance with the instructions provided by the robot computer, a servo motor driver is modeled. The driver modeled here could also be called an amplifier in the motion control industry. It accepts a command signal for position, speed or current and adjusts the voltage and current applied to the servo motor. The driver logic includes three types of cycles or *loops*: current, speed and position. Each of the loops uses feedback signals to adjust the values at the output of the loop and thereby produce the desired results.

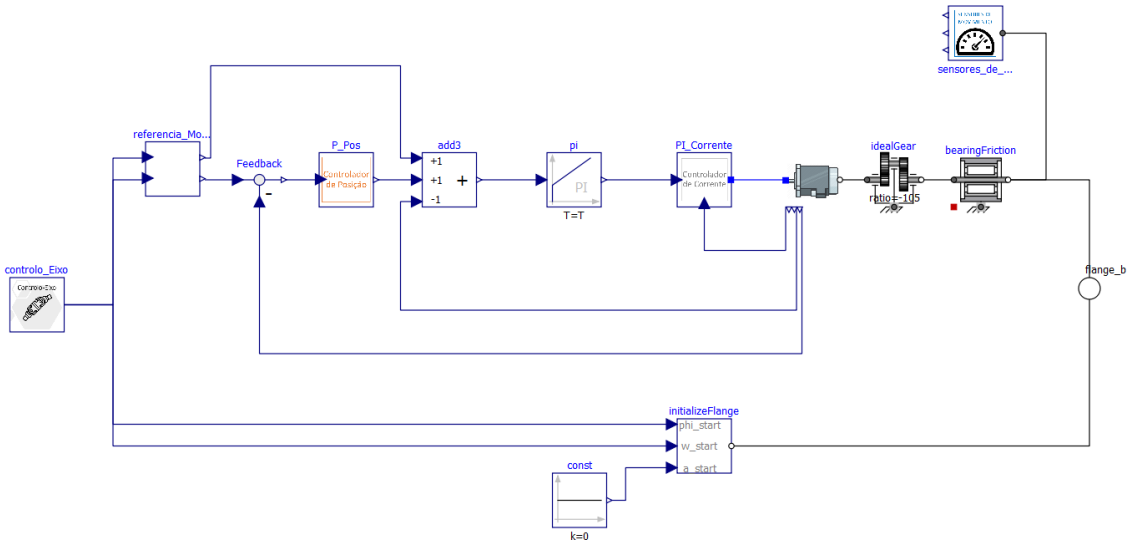


Figure 136 - Modeling of the axial control, servomotor and transmission

PI/PID control is a cornerstone in our automated world. From setting a desired temperature to controlling the pressure of a tool, *PI/PID* control is one of the most effective control methods. Throughout this chapter are examined in some detail the elements that make up the P - PI control that is built into the control of each axis of the arm. Initially, the general control theory will be covered, and will be proceeded by applying the same principles to operational amplifier circuits, which serve the same purpose, but by means of analog control. A block diagram for a general control system may be seen in Figure 137, where an input signal, $e(t)$, is sent to the controller block. Then the obtained control signal, $u(t)$, is transmitted to the plant block or physical system to be controlled. The feedback loop compares the output signal from the plant block, $h(t)$, with the input signal to the control system, $d(t)$. The difference between the output signal and the input signal gives the error term, $e(t)$, which feeds back to the controller. Mathematically, the error term may be calculated as:

$$e(t) = h(t) - d(t) \tag{57}$$

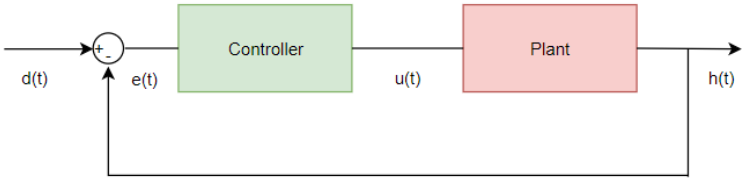


Figure 137 - Block Diagram - General Control

The definition of the control is made based on successive simulations of the system, allowing the analysis of its response and behavior.

POSITION CONTROL

For position control along each axis, an external position *loop* is inserted next to the speed *loop*, in what is also known as a position-speed cascade loop. The position loop determines an error that is the deviation between the instantaneous position and the reference position, and issues a speed command that aims to reduce or eliminate that same error. In a cascade system, the position cycle typically only uses a proportional gain as control, and this has been the adopted position control.

In the proportional control used, the error is multiplied by a proportionality constant, usually called K_p, serving the purpose of making the system react faster, since the error term is magnified. The control signal can be written as:

$$u(t) = K_p \times e(t) \tag{58}$$

Where $u(t)$ is the control signal over time and, likewise, $e(t)$ is assumed to be the position error obtained over time. The implementation of the position controller resorted to the use of a proportional block, whose gain assigned in the block is assumed to be the proportionality constant K_p.

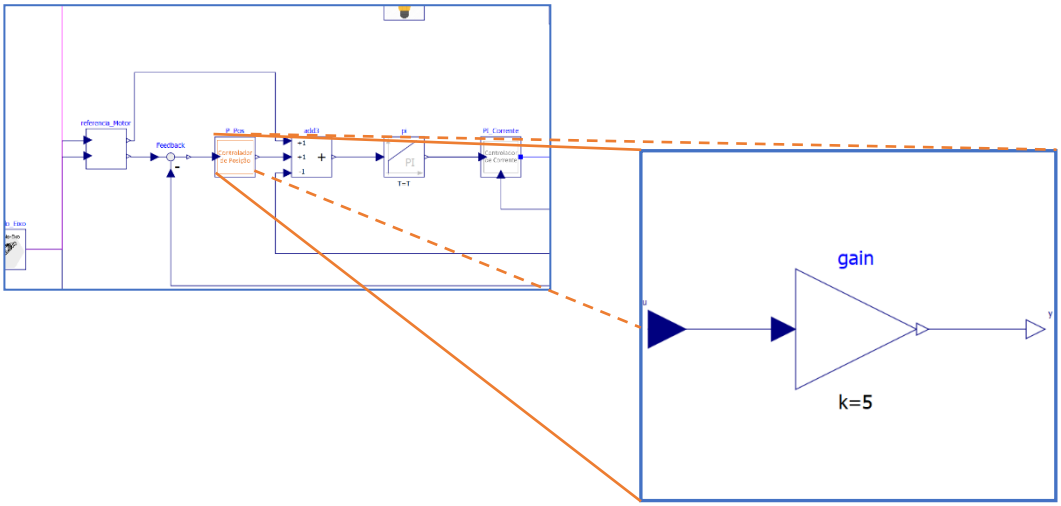


Figure 138 - Position control model with a proportional block

SPEED CONTROL

The speed *loop* is the most common *loop* for servo motor control. Similar to the position *loop*, the instantaneous values obtained are collected by the resolver by deriving the angular position values. The speed loop is assumed to be a PI controller, since it uses the proportional gain and the integrator gain to determine the correct command to be sent to the current control.

In PI control, the proportional control is expanded by adding a control term. The integrator constant K_i, is multiplied by the intergral of the error term. With the addition of the integral term, the control signal becomes:

$$u(t) = K_p \times e(t) + K_i \times \int e(t)\ dt \tag{59}$$

The PI speed control is implemented using a controller block provided by the main library, where the terms are defined directly.

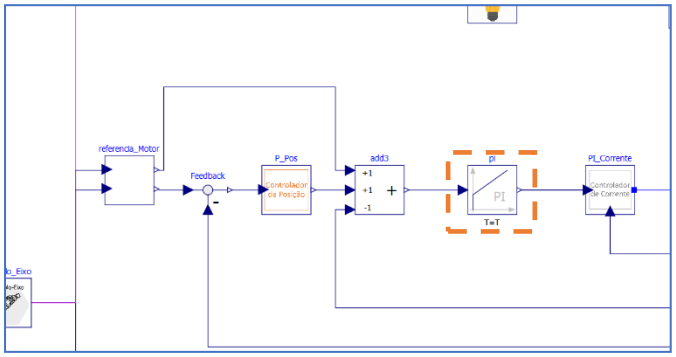


Figure 139 - Model of the speed control with a PI controller block

CURRENT CONTROL

To create a current control scheme, a circuit of operational amplifiers, which are analogous to the *PI - PID* control blocks, has been designed. This approach to current control shows the relationship between control theory and analog control circuits.

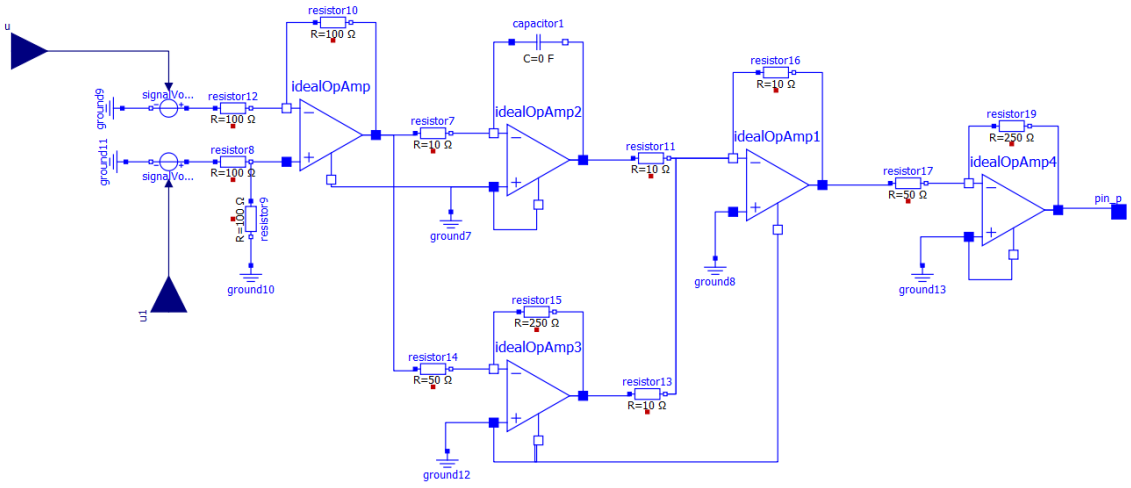


Figure 140 - Circuit with operational amplifiers used in current control

The first operational amplifier admits a subtraction between the value coming from the current sensor (which gets the instantaneous amperage drawn from the motor) and the current command value, coming from the external speed loop in the control system. Both current commands input to the model are a reference signal that takes the form of a voltage (given the proportionality exhibited between current and voltage).

It is important to note that, from the actual and not the model point of view, the command values transmitted between the respective position, speed and current controls are analog

values that translate into a voltage after being processed by a digital-to-analog converter (DAC).

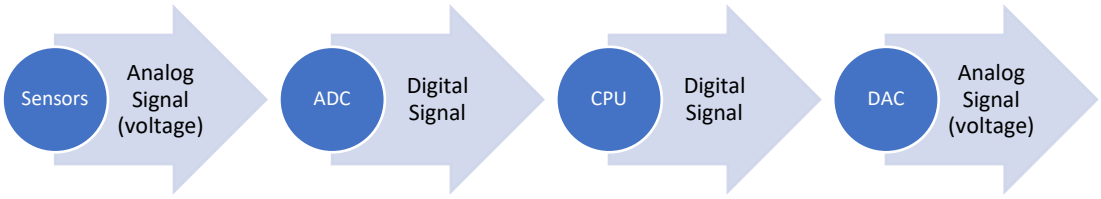


Figure 141 - Process of obtaining a reference command in the form of a voltage

The process by which a reference value in the form of a voltage is obtained meets the data acquisition application that the converters can provide for certain systems. For the case under study, the values of the properties to be measured are digitized by an ADC (analog-to-digital converter) and then sent to a processor. In the processor the control calculations are performed and then transmitted to a DAC (digital-to-analog converter) allowing an analog reference to be collected by the driver.

Once the reference voltage is received, as well as the voltage collected at the motor, a feedback process can be initiated and thereby also the current control. The analog control circuit can be divided into different stages. For one of these stages there is an operational amplifier performing a specific task.

Differential Operational Amplifier

In the first instance, a differential operational amplifier is used, which outputs a voltage signal that characterizes the error, inherent in the delay, between the reference voltage and the instantaneous voltage. Similar to a feedback block in a loop control diagram, this amplifier has the function of presenting the error between the desired value and the obtained value.

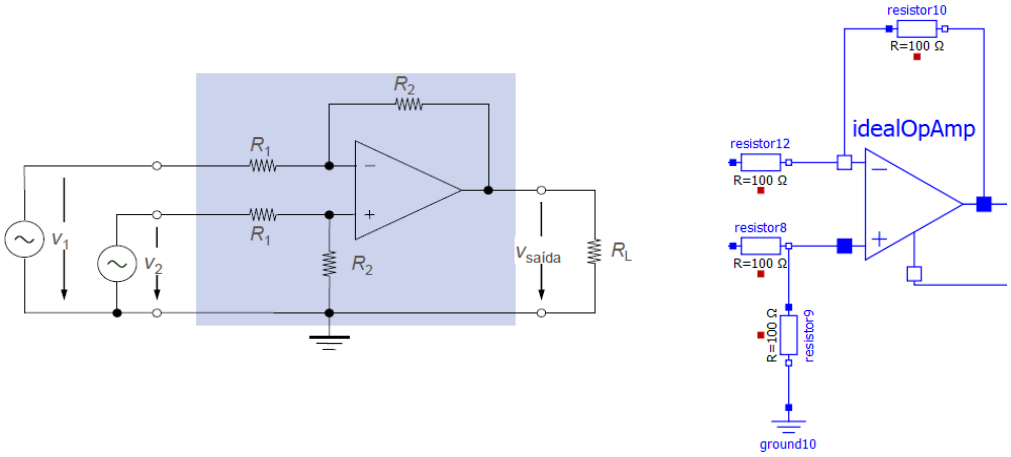


Figure 142 - Implemented differential amplifier and associated nomenclature

Based on the resistances shown in Figure 142, the output voltage is defined as follows:

$$V_{out} = V'_{out} + V''_{out} = -\frac{R_2}{R_1}V_1 + V_2\frac{R_2}{R_1} = \frac{R_2}{R_1}(V_2 - V_1) \quad (60)$$

Given the equation shown and the values assigned to the resistors, it follows that the output voltage is equal to the difference between the two inputs (the gain here is non-existent since the value assigned to the resistors is equal).

INVERTER OPERATIONAL AMPLIFIER (PROPORTIONAL)

It is possible to resort to using an operational amplifier for developing a proportional controller. The operational amplifier, also called an op-amp, follows a simple set of rules: the amplifier inputs have almost no electric current and the output is a positive voltage if $V_+ > V_-$, or a negative voltage if $V_+ < V_-$. Figure 143 shows an op-amp in which an input voltage is associated with the negative input of the amplifier, while the positive input of the amplifier is connected to ground. The amplifier does not allow current to be admitted to the inputs, so:

$$I_{R1} = I_{R2} \quad (61)$$

And since at the amplifier output will be any voltage required to allow the current of the two voltages to flow through R_2, the voltage drops can be written as

$$\frac{0 - V_{in}}{R_2} = \frac{V_{out} - 0}{R_1} \tag{62}$$

Which with a little rearrangement:

$$V_{out} = (V_{in})\frac{-R_2}{R_1} \tag{63}$$

$$\frac{V_{out}}{V_{in}} = \frac{-V_2}{V_1} = \frac{-R_2 I}{R_1 I} = -\frac{R_2}{R_1} \tag{64}$$

Visualizing equation 64, it is perceptible the ability to develop a proportional controller with an *op-amp* integrated into an electrical circuit, where V_{in} can be considered the error $e(t)$, and also $R_2 / R_1 = K_p$.

This amplifier thus allows the error, or difference between the desired and obtained value, to be subjected to a gain, thus corresponding to the proportional element of a PI controller.

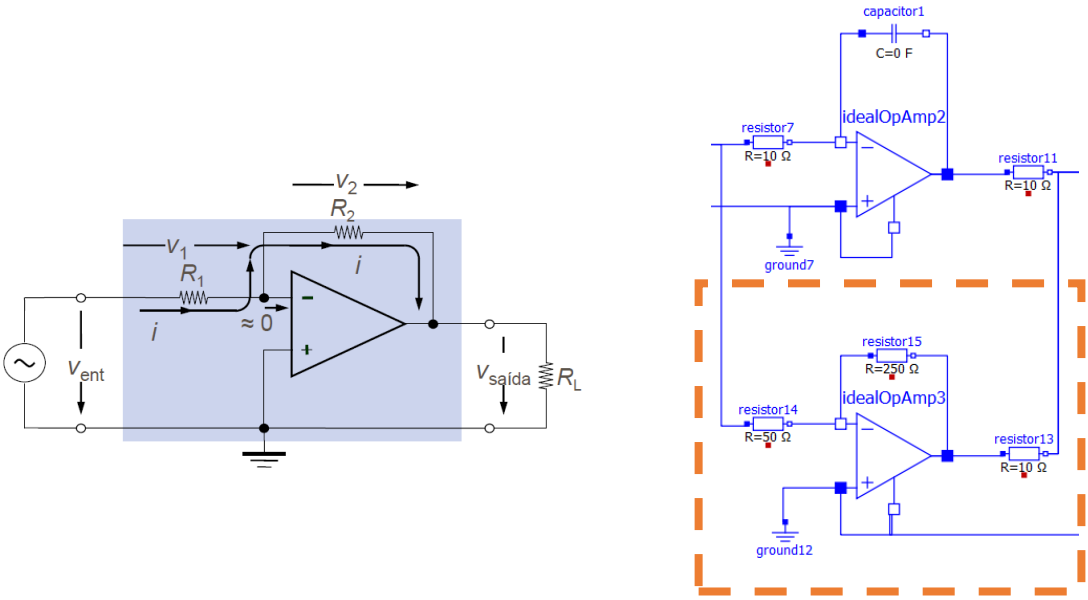


Figure 143- Implemented proportional inverter amplifier and associated nomenclature

INTEGRATOR OPERATIONAL AMPLIFIER

Amplifier that constitutes the integrator element of the PI controller, defined here in the form of operational amplifiers. The way the integrator control is implemented is by means of an *op-amp* like the one schematized in figure 144. The expression will now be designed keeping in mind that the current in R is equal to the current in C:

$$\frac{V_{in}}{R} = -C\frac{dV_{out}}{dt} \tag{65}$$

Isolating V_{out},

$$dV_{out} = \frac{-1}{RC} V_{in}\, dt \tag{66}$$

And finally,

$$V_{out} = \frac{-1}{RC}\int V_{in}(t)\, dt \tag{67}$$

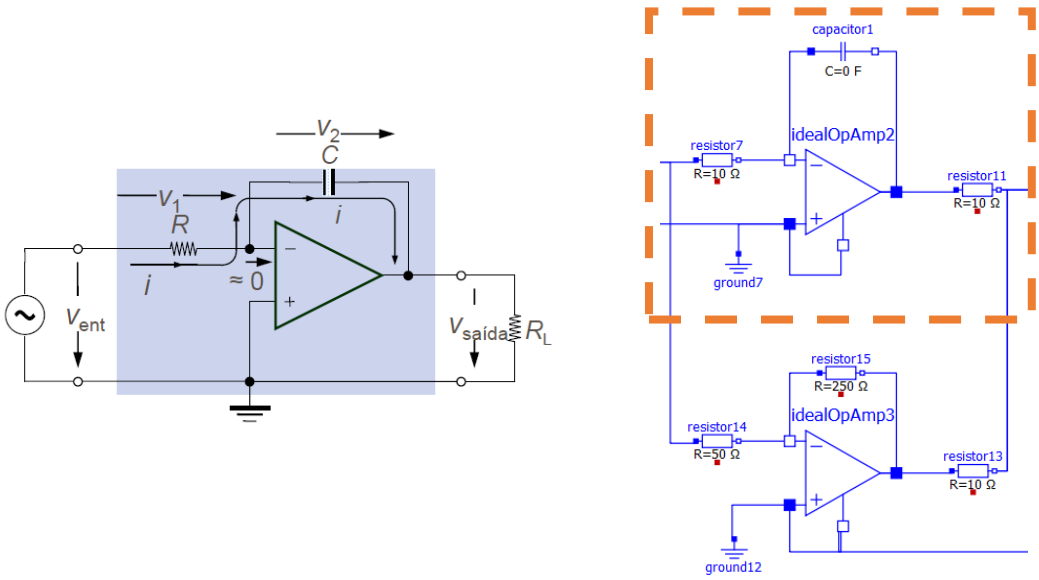


Figure 144 - Implemented integrator amplifier and associated nomenclature

SUMMING OPERATIONAL AMPLIFIER

Amplifier that aims to "bring together" both controller elements by summing them. The sum between the proportional element and the integrator element is thus performed, and for this

purpose no coefficient has been added to both terms, i.e. the value of the associated resistances is thus equal.

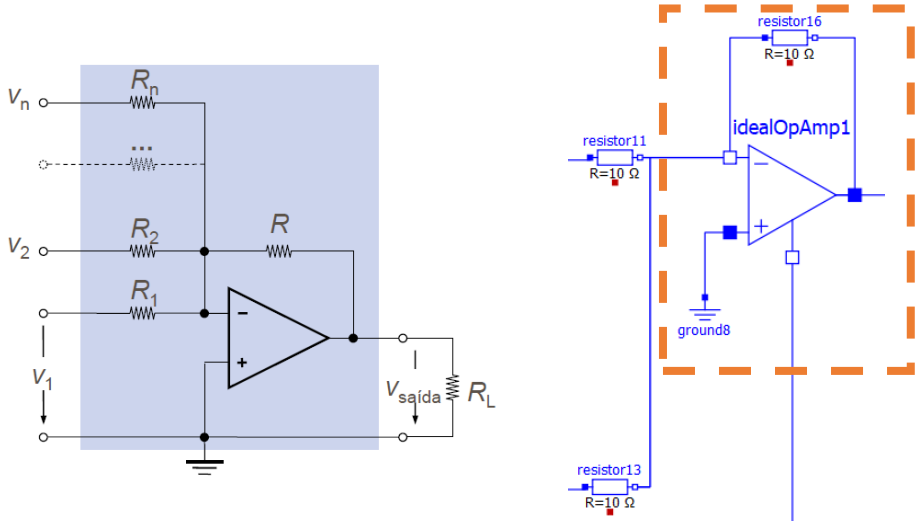


Figure 145 - Implemented summing amplifier and associated nomenclature

$$V_{out} = -\left(\frac{R}{R_1}V_1 + \frac{R}{R_2}V_2 + \cdots + \frac{R}{R_n}V_n\right) \tag{68}$$

INVERTER OPERATIONAL AMPLIFIER

At the driver's output there is also the concern of inverting the signal, from negative to positive, in order to adjust the order in the difference of values realized at the driver's input. The present amplifier, in addition to the gain it performs, allows this signal inversion to be done.

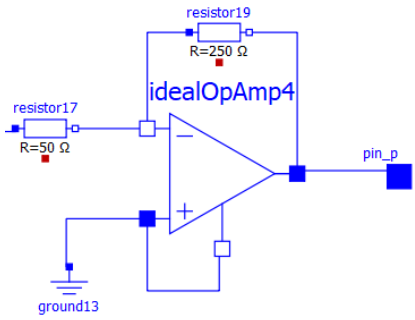


Figure 146 - Inverter amplifier implemented at the end of the circuit

ACTUATORS - SERVOMOTORS

An electric motor can be modeled with an equivalent electric circuit, as schematically represented in Figure 147. In this, a supply voltage that remains constant is fed into the circuit accompanied by a resistance, representative of the internal resistance of the copper wires used to create the coils. On the other hand, the inductor is representative of inductance, describing the tendency of the wires to oppose a change in the electric current flowing over them. Finally, a component where the electromotive force is described is inserted into the circuit.

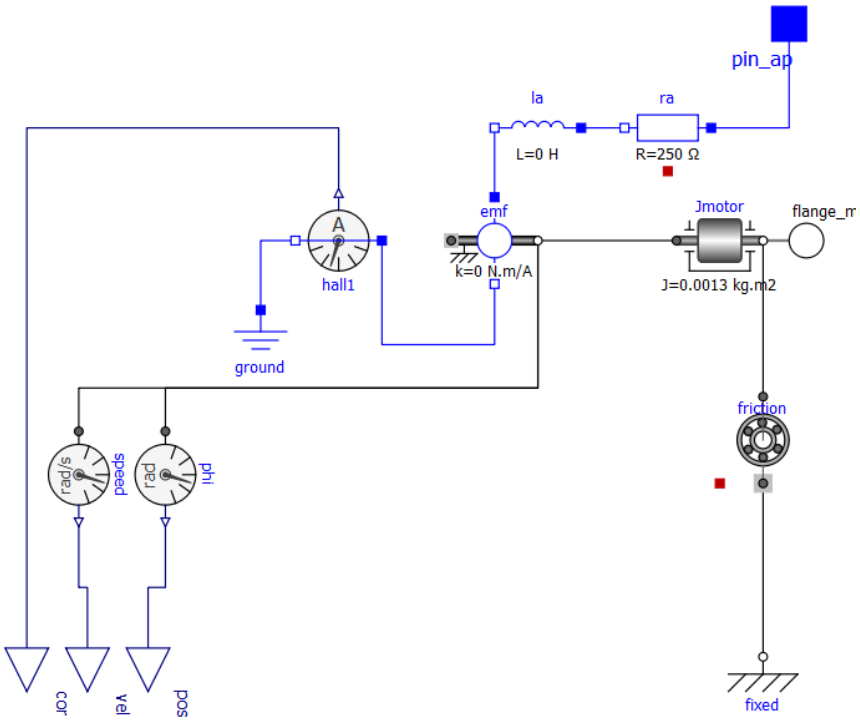


Figure 147 – Servomotor Model

Once the electrical component is elaborated, we are left with the mechanical component of the system in question. For this purpose, a rotary damper is added, allowing to absorb and slow down the rotary motion generated by the electromotive force of the motor, changing vibration, noise and wear of the components, thus founding a smoother mechanical movement. In addition, a moment of inertia is added to represent the energy need to be supplied by the induced electromotive force, so that the rotation of the shaft is verified. This parameter is directly related to the rotor of the electric motor.

TRANSFER FUNCTION – SERVOMOTOR

After analyzing the modeling underlying the motor and realizing its importance in the process involved in the control cycles, the transfer function representative of the servo motor is elaborated. This allows to approach the way in which the model development is faced, because it allows the simplification of the control process. Thus, the servomotor model can be replaced by a block designating the transfer function, in which all the equations and variables involved in the electrical and mechanical circuit components are underlying. For the model in question, the transfer function is not implemented, since for this particular situation it provided no benefit, and would also be incompatible with the current controller, since it is analog. However, the study is explored, allowing us to understand the main equations involved in the design of the servomotor and its operation.

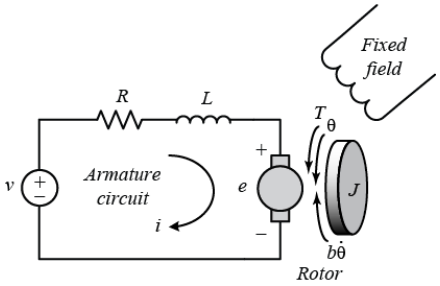
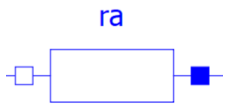
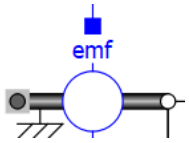
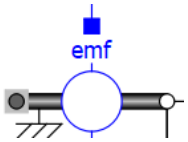
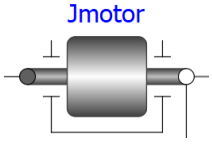


Figure 148 - Electrical circuitry incorporated in the servomotor

The transfer function is obtained through the ratio between the output variable to be studied and the manipulated input variable of the system. Thus, the function could be defined as the ratio between the speed at the output of the rotor shaft and the voltage supplied to the electric circuit incorporated in the armature of the servomotor, as can be seen in Figure 148. The set of equations and parameters involved in each of the motor subsystems can be found in the table.

Table 23 - Equations involved in the different servomotor domains

Component	Variable/ Parameter	Balance/Equation
pin_ap	E_a	**Electrical System** $E_a - R_a I_a - L_a S I_a - V_b = 0$ (69)

<table>
<tr><td> 
</td><td>R_a</td><td rowspan="3">E_a – Supplied voltage
R_a – Resistance
L_a – Inductance
V_b – Voltage at the motor</td></tr>
<tr><td> 
</td><td>L_a</td></tr>
<tr><td> 
</td><td>V_b</td></tr>
<tr><td> 
</td><td>V_b</td><td>Motor Constants
$T_m = K_t I_a \rightarrow I_a = \frac{T_m}{K_t}$ (70)
$V_b = K_b \dot{\theta}_m \rightarrow V_b = K_b S \theta_m$
$E_a - R_a\left(\frac{T_m}{K_t}\right) - L_a S\left(\frac{T_m}{K_t}\right) - K_b S \theta_m = 0$
T_m – Motor Torque
K_t – Torque Constant
K_b – Constant of the electromotive force of the motor</td></tr>
<tr><td> 
</td><td>J</td><td>Rotary System
$T_m - D\dot{\theta}_m = J\ddot{\theta}_m$ (71)
$T_m = J\ddot{\theta}_m + D\dot{\theta}_m$
$T_m = JS^2\theta_m + DS\theta_m$
$T_m = \theta_m(JS^2 + DS)$</td></tr>
</table>

Based on the equations in the table, more specifically the substitution in (69) of (70) and (71), equation 72 can be written.

$$E_a - \theta_m(JS^2 + DS)\left(\frac{R_a + L_a S}{K_t}\right) - K_b S \theta_m = 0 \quad (72)$$

Knowing that $S\theta_m = w_m$ and that $K_t = K_b$, then the final formulation of the transfer function is as follows

$$\frac{\boldsymbol{w_m}}{\boldsymbol{E_a}} = \frac{\boldsymbol{KS}}{(\boldsymbol{JL_aS^3} + (\boldsymbol{JR_a} + \boldsymbol{DL_a})\boldsymbol{S^2} + (\boldsymbol{DR_a} + \boldsymbol{K^2})\boldsymbol{S})} \tag{73}$$

The study of the transfer function thus requires a thorough understanding and analysis of the system in question, providing greater insight into the control and modeling options involved. When the "plant" or control object is known with all its parameters already well defined, and the study focuses more on the output variables and not necessarily on their manipulation, the design of a modeling with a transfer function presents itself as advantageous.

TRANSMISSION

Comprised between the shaft at the motor output and the shaft of each of the joints, the transmission elements are modeled. The first element characterizes a gearbox that has a shaft at the input that drives and a driven shaft at the output. The gears involved are ideal, where inertia, elasticity, damping and backlash are disregarded.

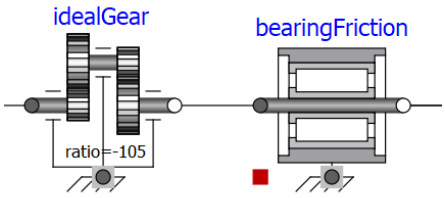


Figure 149 - Transmission components used

AUXILIARY MODELS

In order to maintain a correct control operation of the different variables measured at the actuators, it is relevant that the discrepancy between the angular variation along the axis of the servomotors rotor and the real angular variation verified along each axis at the arm, is tackled. Speaking specifically of position and angular velocity, these two properties have to be properly adjusted with the transmission ratio present in the transmission gears, and thus

generate a difference in the different feedbacks with values inserted in the same operational scale.

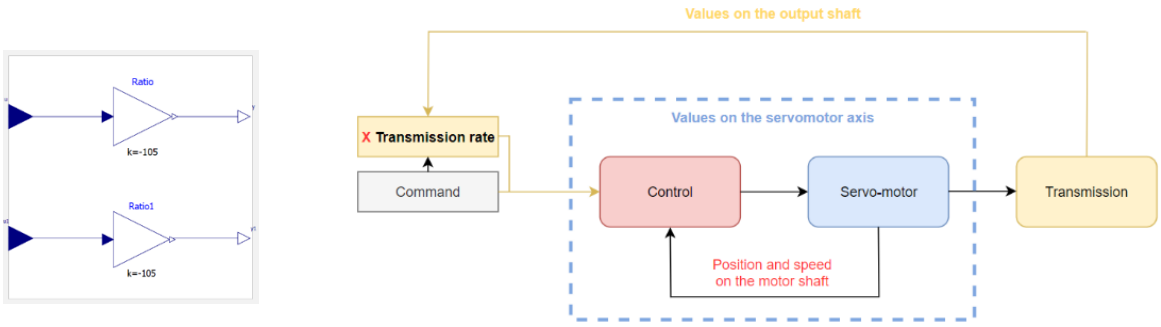


Figure 150 - Adjustment in the transmission rate, providing a correct functioning of the control

In the transmission, the values of the properties under study suffer a reduction and adopt a negative value, i.e., the direction of rotation is reversed at the output of the gears. This reduction and reversal of direction is balanced, with the multiplication of the reference values of position and speed by the transmission ratio at the controller input and the respective control cycles, and a model is developed for this purpose.

Associated with the output flange of the axial control model is a component responsible for assigning an initial angular position to the flange.

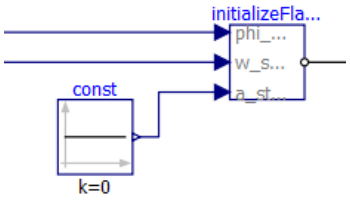


Figure 151 - *InitializeFlange* component responsible for assigning position, velocity and acceleration values to the output axis flange

This component thus enables the initial (defined) position of the robot to be guaranteed.

Associated with the axis, there is also a model with a resistor, thus simulating a small motion signal lamp, which turns on when the motion command is transmitted. In this way, the detection of a physical malfunction of the process is made easier, the absence of movement of the axis when the light is on allows the identification of a possible anomaly in the physical system.

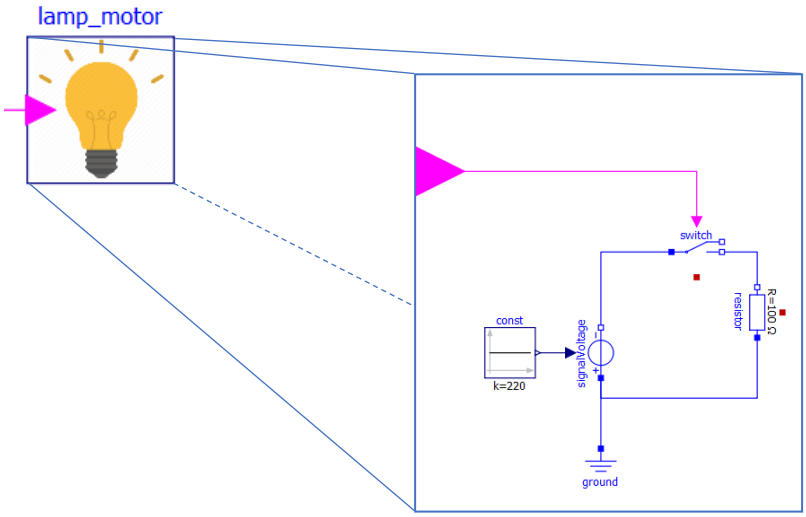


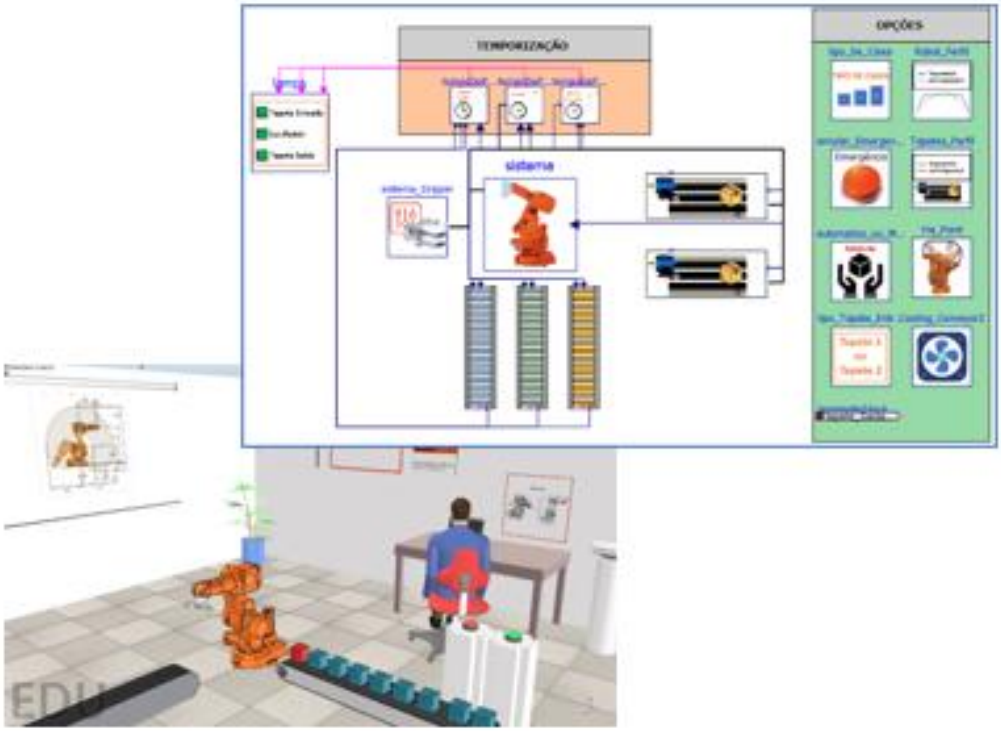
Figure 152 - Model that replicates a motion signal lamp on the shaft

APPENDIX 4 – USER'S MANUAL

Universidade do Minho
Escola de Engenharia

MANUAL DE DESENVOLVIMENTO

MODELAÇÃO E SIMULAÇÃO DE UM SISTEMA *PICK&PLACE* COM OPENMODELICA



MESTRADO INTEGRADO EM ENGENHARIA MECÂNICA

ALBERTO JOSÉ CIBRÃO MONTEIRO – A81599

ABRIL, 2023

MODELAÇÃO E SIMULAÇÃO DE UM SISTEMA *PICK&PLACE* COM OPENMODELICA

A escolha do tema, Modelação de um sistema *Pick&Place* utilizando a linguagem de modelação *Modelica* e o software de simulação *OpenModelica*, constituiu-se como uma oportunidade no aprofundamento do estudo de técnicas e modelos computacionais. Além de permitir uma familiarização e compreensão com diferentes *softwares*, alicerçados em técnicas que se projetam como cada vez mais indispensáveis na posição que representam, pela fusão cada vez mais inevitável entre a realidade física e o universo virtual.

A modelação de sistemas híbridos é uma tarefa que se poderá constituir como particularmente complexa, nomeadamente quando os sistemas em análise apresentam tecnologias diversas tais como a pneumática, hidráulica, sistemas mecânicos, sistemas elétricos, entre outros. A maior dificuldade inerente no estudo integral destes sistemas surge na medida em que os ambientes de simulação, que se complementam com linguagens de modelação, são escassos, mas surgem como cada vez mais necessários pois permitem um estudo aprofundado que embarga a totalidade do sistema em todas as suas divisões e partes conjuntas.

Com a linguagem de modelação *Modelica* e com o *software* de simulação *OpenModelica* essa área ainda em desenvolvimento poderá ser explorada, pelo que a sua aplicação constitui-se como uma mais valia quando conjugada com um futuro num âmbito que se projeta como bastante promissor.

Ao longo deste trabalho, as potencialidades apresentadas pela linguagem foram exploradas, reunindo o conceito de um manipulador robótico inserido na biblioteca principal de *Modelica* com o delineamento de um comando detalhado, em que os conceitos de cinemática inversa e *pick&place* se inserem. A incorporação com um sistema global também foi escrutinada, no qual o modelo do manipulador se encontra associado com um *gripper* e tapetes transportadores, projetados num sistema com sequenciamento lógico.

PALAVRAS-CHAVE

Modelação, Simulação, *Pick&Place,* Cinemática Inversa, Tapetes transportadores, *Gripper*

Índice

Índice de Figuras

Índice de Tabelas

1. Estudo de Mercado – Simulações em Ambiente Industrial

A importância da simulação em ambiente industrial, que veio mais tarde ganhar relevância em sistemas de engenharia ciberfísicos, foi já destacada pelos primeiros pioneiros em estudos de investigação de simulação, desde os anos 80, como uma metodologia de topo. Além disso, a simulação é classificada como a segunda metodologia mais utilizada para o campo de estudo de gestão de operações, apenas após a otimização. Esta visão da simulação como central na empresa e na investigação é demonstrada por muitos trabalhos de investigação e estudos empíricos, cujo número tem vindo a aumentar continuamente desde a primeira conceptualização de possíveis usos de simulação em ambiente industrial. Para calcular a quantidade de trabalhos realizados, alguns números são aqui fornecidos (Polenghi, Fumagalli, & Roda, 2018): de 2002 a 2013, o processo de pesquisa com as palavras-chave "simulação" mais "manufatura" apresenta cerca de 3000 artigos publicados, enquanto o processo global de investigação com outras palavras-chave, tais como "engenharia industrial" ou "operações", resulta em cerca de 12000 artigos publicados no mesmo período de tempo. Estes resultados são expressivos e são confirmados por outro inquérito realizado no mesmo ano em 2014, onde os resultados podem ser observados no gráfico 1 em baixo.

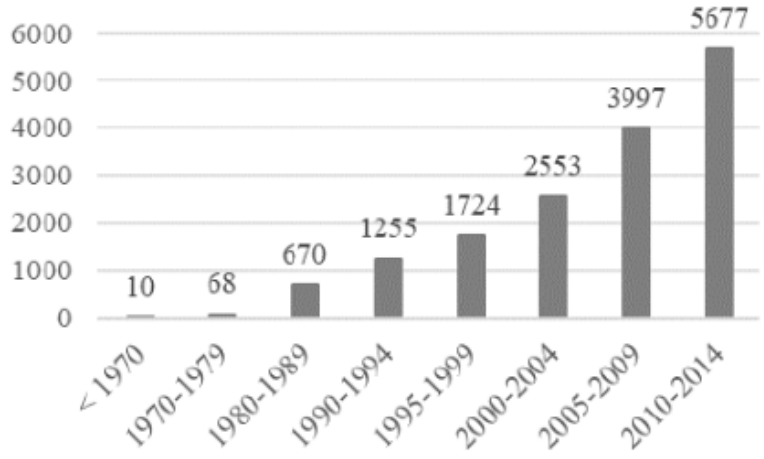


Figura 1 - Números de publicações relacionadas com Simulação (Polenghi, Fumagalli, & Roda, 2018)

Muitas definições de simulação consideram o envolvimento do computador como uma parte consistente da própria definição, contudo, esta definição é restrita porque limita a aplicação da simulação com um sistema informatizado: o sucesso da simulação está estritamente relacionado com o advento dos computadores (Smith, 2003), mas o conceito de simulação

existe de forma independente. Uma definição simples, mas significativa é fornecida por (Maria, 1997): "A simulação é uma forma de avaliar um sistema proposto para vários parâmetros dentro de um período de tempo específico". Uma outra definição completa, e talvez a mais notória, é: "A simulação é a imitação do funcionamento do processo ou sistema do mundo real ao longo do tempo" (Banks, 1998).

A segunda metade do século XX é caracterizada pela simulação, quer em estudos de investigação quer em campos industriais. A utilidade desta metodologia é bem reconhecida, uma vez que ajuda na maioria das fases do ciclo de vida do produto ou do processo, fornecendo assim um apoio importante no processo de tomada de decisão. Por conseguinte, a análise da literatura levou à individuação de duas tendências principais no uso da simulação na indústria. A primeira tendência é determinada pela mudança da visão da simulação, do desenho de fabrico apenas, para uma perspetiva mais ampla constituindo-se como um auxílio útil durante o ciclo de vida, para a gestão de operações; a segunda tendência é o aumento do número de campos de aplicação da simulação na indústria.

A aplicação da simulação na indústria pode dizer respeito à melhoria da fase de conceção (*BOL - Beginning-Of-Life*) do sistema: a possibilidade de simular o sistema antes da sua instalação favorece todas as decisões principalmente no que diz respeito à disposição, em termos de configuração da planta, e ao movimento de material, em termos de sistema de manipulação de materiais. A evolução subsequente foi o alargamento do âmbito da simulação para a fase operacional do sistema, como políticas de planeamento e manutenção da produção. Este aumento na utilização de simulação no MOL (*Middle-Of-Life*) do sistema foi impulsionado pelo aumento do poder computacional e, principalmente, pelo aumento da disponibilidade de dados em tempo real do chão de fábrica (ou "*close-to-real-time*").

Dentro da área industrial, as principais aplicações podem ser observadas na seguinte tabela (Polenghi, Fumagalli, & Roda, 2018).

Tabela 1 - Principais aplicações de simulação em ambiente industrial

Aplicação de Simulação em Ambiente Industrial
Projeto do Sistema e Projeto/*Layout* da Instalação
Projeto do Sistema e Manuseio de Materiais
Planejamento e Programação de Operações
Controlo em Tempo – Real

Políticas Operacionais
Análise de Desempenho
Projeto da Cadeia de Suprimentos
Projeto do Processo
Gestão de inventário
Gerenciamento de Manutenção
Alocação de Recursos
Compras
Design de Produto
Ergonomia
Gestão do Conhecimento

Tendo por base a modelação orientada a objetos, retira-se desde logo um maior foco nas três primeiras aplicações de simulação apresentadas na tabela 1 sendo que, todavia, a sua aplicabilidade poderá ser estendida às restantes aplicações.

2. SISTEMA

Quase tudo poderá ser classificado como um sistema. Uma definição consensual para sistema é que é um objeto ou conjunto de objetos cujas propriedades se quer ver estudadas. A definição daquilo que constitui um sistema é assim arbitrária, estando intrinsecamente dependente daquilo para o qual é usado. A exploração e estudo de sistemas, do ponto de vista de engenharia, surge da manifesta vontade em compreender e, consequentemente, construir.

Um sistema é constituído por:

- *Inputs,* ou seja, variáveis do ambiente que influenciam o comportamento do sistema.
- *Outputs,* variáveis determinadas pelo sistema e que podem influenciar o meio onde o sistema se incorpora.

2.1. Método Experimental

De forma a retirar informações sobre o comportamento de um sistema, a observação é um fator crucial. Observar os *outputs* é o que permite retirar pelo menos um entendimento básico sobre o sistema, sendo que o mesmo poderá ser refinado através do controlo dos diferentes *inputs*.

Um experimento é o processo ou conjunto de processos que permitem retirar informação de um sistema através da variação dos seus *inputs*. Há no entanto, um conjunto de problemas que se sobressaem através do exercício destes experimentos, dos quais se destacam as perturbações dos estados internos do sistema, os custos implicados, a perigosidade envolvida e o facto de que o sistema requerido para o experimento físico poderá ainda não existir.

Estes aspetos deficitários do método experimental conduziram ao surgimento do modelo conceptual, sendo que se este for realístico o suficiente então uma investigação cuidada e analítica do sistema poderá ser efetuada

3. MODELO

3.1.Conceito

> "Um modelo de um sistema é qualquer coisa em que um experimento possa ser aplicado, de forma a dar resposta a questões sobre esse mesmo sistema"

Esta definição de modelo (Fritzson, 2011), reunida com consensualidade dentro do seio de engenharia, implica que um modelo seja capaz de dar repostas a questões sem que sejam necessários experimentos com o sistema real, propriamente dito. Desta forma, experimentos conduzidos no modelo, que acaba por ser uma versão simplificada do sistema real, permitem refletir propriedades que poderão ser úteis e decisivas para um entendimento distinto desse mesmo sistema.

Dependendo da sua representação, os modelos poderão adotar diferentes tipos:

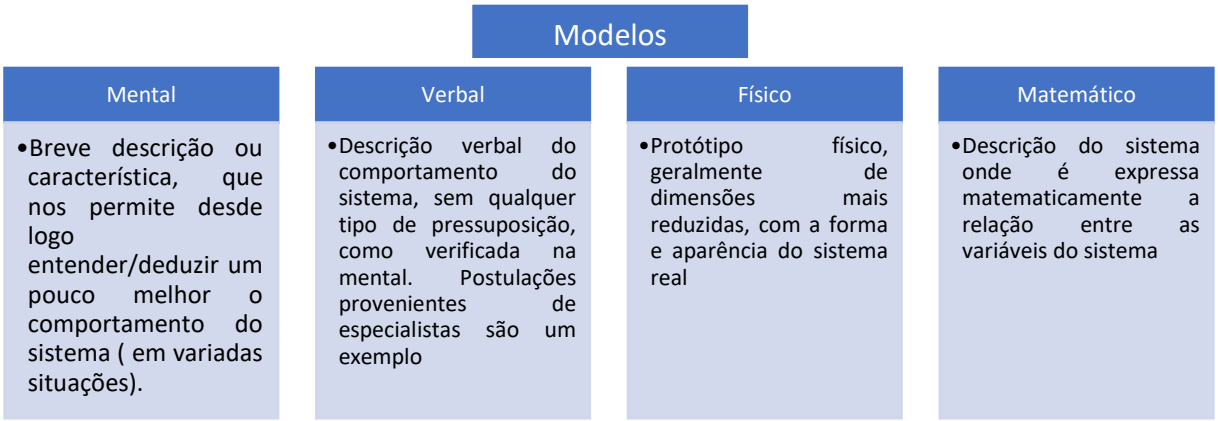


Figura 2 - Tipos de Modelos (Fritzson, 2011)

O tipo de modelos que são utilizados no *software OpenModelica* são do tipo matemático, sendo este tipo de modelo constituído de diferentes modos, desde equações e funções até

programas computacionais. Modelos matemáticos elaborados num computador são frequentemente denominados de modelos virtuais ou protótipos virtuais.

3.2. Construção de Modelos

Para a construção dos modelos, a utilização da linguagem *Modelica* permite a simplificação da sua construção, assim como a reutilização de modelos pré-existentes. A sua aplicação implica, os seguintes atributos por parte do utilizador.

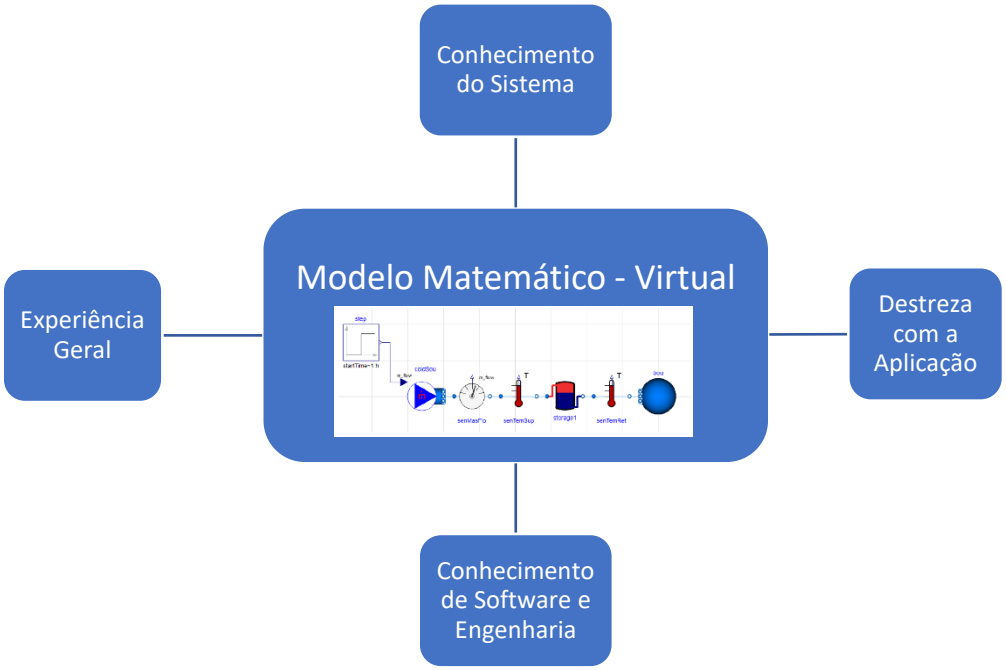


Figura 3 - Requisitos de utilização para modelação e simulação de modelos virtuais (Fritzson, 2011)

- **Experiência Geral –** conhecimentos recolhidos nos domínios de ciência e tecnologia encontrados na literatura e disponibilizados pelos diferentes especialistas nas áreas associadas.
- **Conhecimento do Sistema –** observações e familiarização com experimentos já conduzidos no sistema.
- **Destreza com a Aplicação –** dominar a área e técnicas de aplicação, permitindo exercer da melhor maneira possível as capacidades providenciadas pela aplicação na modelação de sistemas.
- **Conhecimento de *Software* e Engenharia –** conhecimento genérico sobre a definição, tratamento e representação de modelos e do *software.*

Assim, o projeto de concepção apropriado a ser tida em consideração no estabelecimento do processo de construção de modelos, de maneira a ir de encontro com a canalização e

sintetização das informações adquiridas, nas diferentes formas mencionadas em cima, é representada na seguinte esquematização (Figura 4):

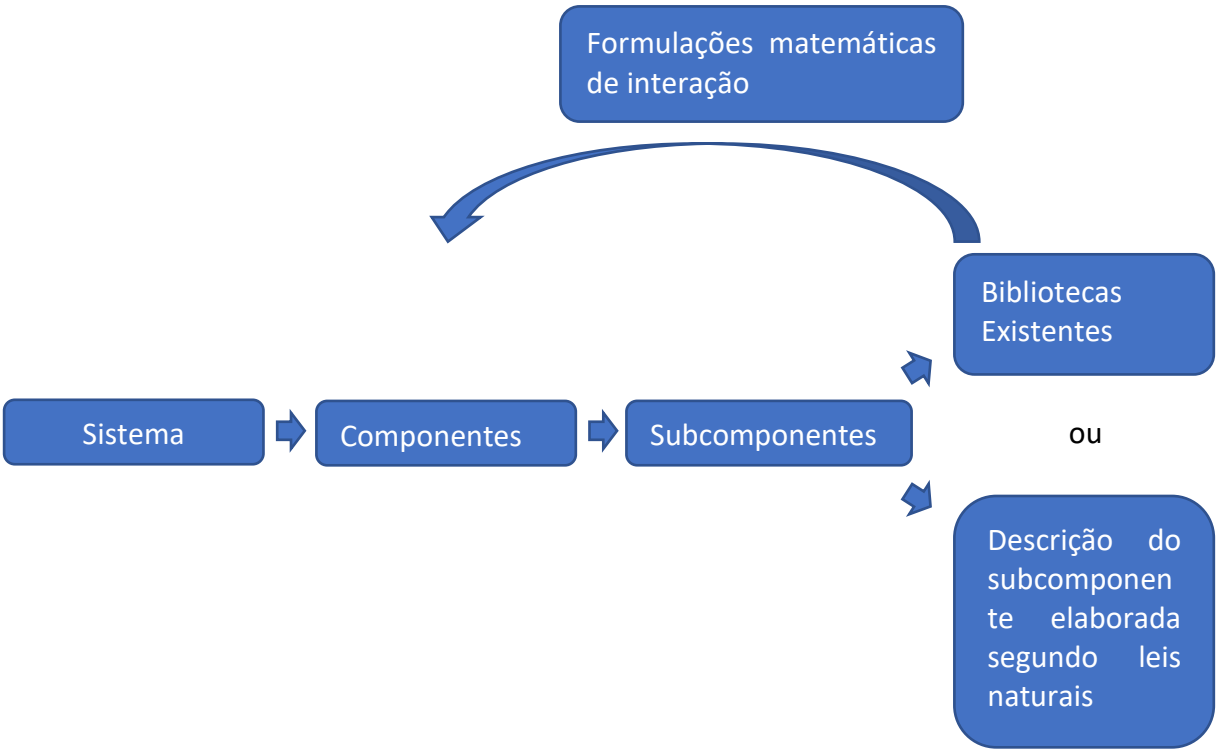


Figura 4 - Esquematização: processo de construção de modelos

3.3. Análise

Existem várias técnicas que permitem analisar os diferentes modelos e assim dar resposta sobre os sistemas. A mais convencionalmente usada é a simulação.

3.3.1. Simulação

À execução de experimentos com os diferentes modelos virtuais, uma vez arquitetados, dá-se o termo de simulação. A definição não exige uma atuação exclusiva em modelos matemáticos e computacionais.

Assim, o verdadeiro valor de uma simulação, no sentido em que ilustra devidamente o sistema em estudo, é completamente dependente no quão bem elaborado está o modelo, permitindo dar resposta às questões que se quer ver respondidas quanto ao comportamento do sistema. A significância da simulação poderá ser transparecida através das vantagens apresentadas de seguida, mas sem desvalorizar as suas limitações.

Tabela 2 - Simulação de um Modelo:Aspetos positivos e negativos (Fritzson, 2011)

Simulação de um Modelo	
Aspetos Positivos	**Aspetos Negativos**
• Experimentos físicos são dispendiosos, perigosos ou a sua construção ainda não foi concretizada. • A escala do tempo nas diferentes dinâmicas do sistema, não são compatíveis com a do utilizador real do sistema. Uma simulação poderá apresentar as mudanças no sistema, que de outra forma precisariam de uma escala de tempo maior para serem verificadas num experimento físico. • Encontra-se ao alcance de uma simulação, o estudo e controlo de todas as variáveis, até mesmo as que são inacessiveis no sistema real. • Manipulação fácil dos modelos assim como dos seus parâmetros. O valor de um parâmetro muda-se em segundos, enquanto que a mudanção do mesmo num sistema físico é mais demorada. • Isolar/retirar perturbações ou aspetos inevitáveis no sistema físico real e dessa forma ganhar um melhor entendimento sobre o sistema. • Supressão de efeitos de segunda ordem como não linearidades constadas em determinados componentes do sistema.	• Adoração e entusiasmo por um modelo, levando ao esquecimento das limitações envolvidas, ou seja, de que este apenas representa o sistema real sob certas circunstâncias. • Forçar a realidade dentro das restrições dos modelos. Poderão resultar numa visão simplista da realidade, ignorando aspetos importantes. • Esquecimento da precisão do modelo e das suas assunções simplificadas, podendo resultar numa incorreta interpretação dos resultados.

3.3.2. Análise Sensitiva

Análise à sensibilidade do sistema para com a mudança de parâmetros. Há situações em que reduzidas variações aleatórias dos parâmetros poderão gerar grandes variações aleatórias no comportamento do sistema. Sendo que cuidados redobrados devem ser tidos para com estas situações. Da mesma forma, quando há uma ausência de sensibilidade ou uma sensibilidade reduzida por parte do sistema como resposta a variações substanciais dos parâmetros, a atenção e cuidados devem ser estendidos. A procura por uma resposta robusta, é indiciada

por uma proporção razoável entre a variação dos parâmetros e a resposta do sistema (Fritzson, 2011).

3.3.3. Diagnóstico Baseado em Modelo

Técnica que de certa forma se relaciona com a análise sensitiva. Analisa os diferentes intervalos dentro de determinados valores específicos, que providenciam boas condições de operação. Valores computados que se distanciam desse intervalo, geram erros operacionais no sistema. Esta técnica visa assim, procurar entender os valores específicos dentro dos quais um intervalo funcional é verificado (Fritzson, 2011).

3.3.4. Verificação e Validação do Modelo

As técnicas que permitem averiguar a utilidade, pelo menos parcialmente, de um modelo serão citadas de seguida:

- Rever de forma crítica as assunções e aproximações inerentes ao modelo, com base em informação disponível.
- Comparar, quando possível, com resultados experimentais.
- Comparar, para casos especiais, variações simplificadas do modelo com soluções analíticas.
- Verificar a compatibilidade das dimensões e unidades ao longo das equações elaboradas.
- Análises de sensibilidade (respostas comportamentais) do modelo.

4. OPENMODELICA

A utilização da ferramenta *open-source OpenModelica* recorrendo à linguagem *Modelica* oferece uma excelente solução para a simulação e otimização de sistemas complexos de engenharia. *OpenModelica* é uma ferramenta de modelação. Com esta ferramenta computacional os respetivos utilizadores podem criar componentes individuais ou modelos de subsistemas. Estes modelos podem depois ser utilizados para prever o comportamento do sistema físico associado. Além de ser uma ferramenta que permite a modelação de sistemas, esta é ainda uma plataforma capaz de desempenhar simulações.

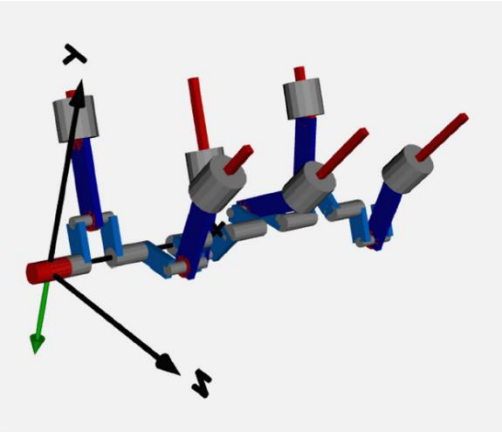


Figura 5 – Exemplo de sistema passível de simulação

Incluídos na plataforma de simulação do *OpenModelica*, o programa apresenta, depois de processados os diferentes modelos, a capacidade de *plot*, análise e "animar" o processo físico adjacente ao modelo. Esta capacidade de modelação e simulação, permite aos utilizadores da ferramenta computacional, explorar o ambiente de *design* e confirmar a sua *performance* expectável sem a necessidade de construir protótipos físicos dispendiosos. *OpenModelica* também suporta colaboração com outras ferramentas, através da interoperabilidade adjacente a um conjunto de *standards* (OpenModelica, s.d.). Estes *standards* são:

- Interface *standard* **FMI** *(Interface Mock Up Funcional*) ou *Funcional Mock up Interface*, que permite a troca de modelos e co simulação, fazendo com que seja uma interface cada vez mais adotada por diferentes vendedores de *software* e indústrias.
- *System Structure and Parameterization* (SSP)
- *Distributed Co-Simulation Protocol* (DCP)

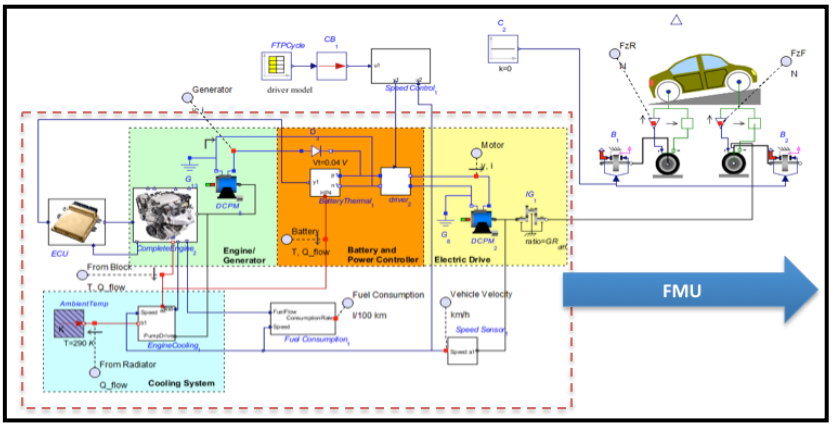


Figura 6 - Interface Standard FMI

OpenModelica suporta de forma completamente integral o standard *FMI* permitindo dessa forma, construir e exportar *fmu's* (Unidades funcionais *Mock up*), ou seja, ficheiros que contêm um modelo de simulação que adere ao *FMI*, e dessa forma poderem ser utilizados por outras ferramentas computacionais.

4.1. Desenvolvimento de Modelos

Na maioria dos casos os modelos no programa são criados através de um "*Modus Operandi*" bastante simples: arrastar, largar e conectar modelos de componentes, permitindo o desenvolvimento de diagramas esquemáticos.

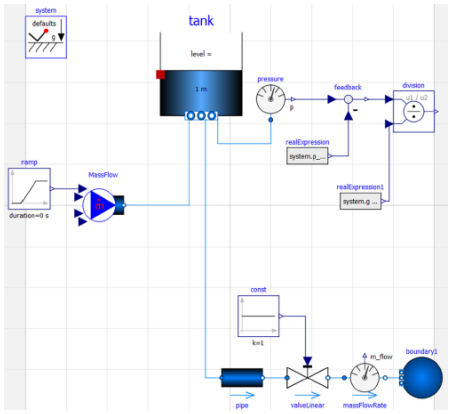


Figura 7 - Diagrama Esquemático

Quando o sistema a ser modelado ganha uma certa complexidade, é possível reorganizar o sistema criando subsistemas. Os componentes que são providenciados pela biblioteca principal (*standard*) estabelecida em *Modelica*, abrangem um leque de áreas bastante diversificado, distribuídos por diferentes domínios de engenharia. Em adição, um número alargado de bibliotecas é também disponibilizado, como por exemplo "*VehicleDynamics", "Smart Eletric Drives", "AirConditioning",* etc. Mas um dos aspetos que torna o *OpenModelica* um ambiente de modelação e simulação de distinção é a sua extensibilidade, característica essa que é capacitada pela programação de modelação utilizando *Modelica*.

4.2. Modelica

A linguagem que serve de referência para qualquer tipo de trabalho a ser elaborado com *OpenModelica*, é a linguagem de modelação *Modelica*, sendo que esta:

- Descreve o comportamento de sistemas complexos.

- É uma forma de representação usada para todos os modelos desenvolvidos no programa.

Aqui é apresentado um esquema ilustrativo das etapas realizadas pelo programa, recorrendo à linguagem, desde a modelação do sistema até à sua simulação propriamente dita.

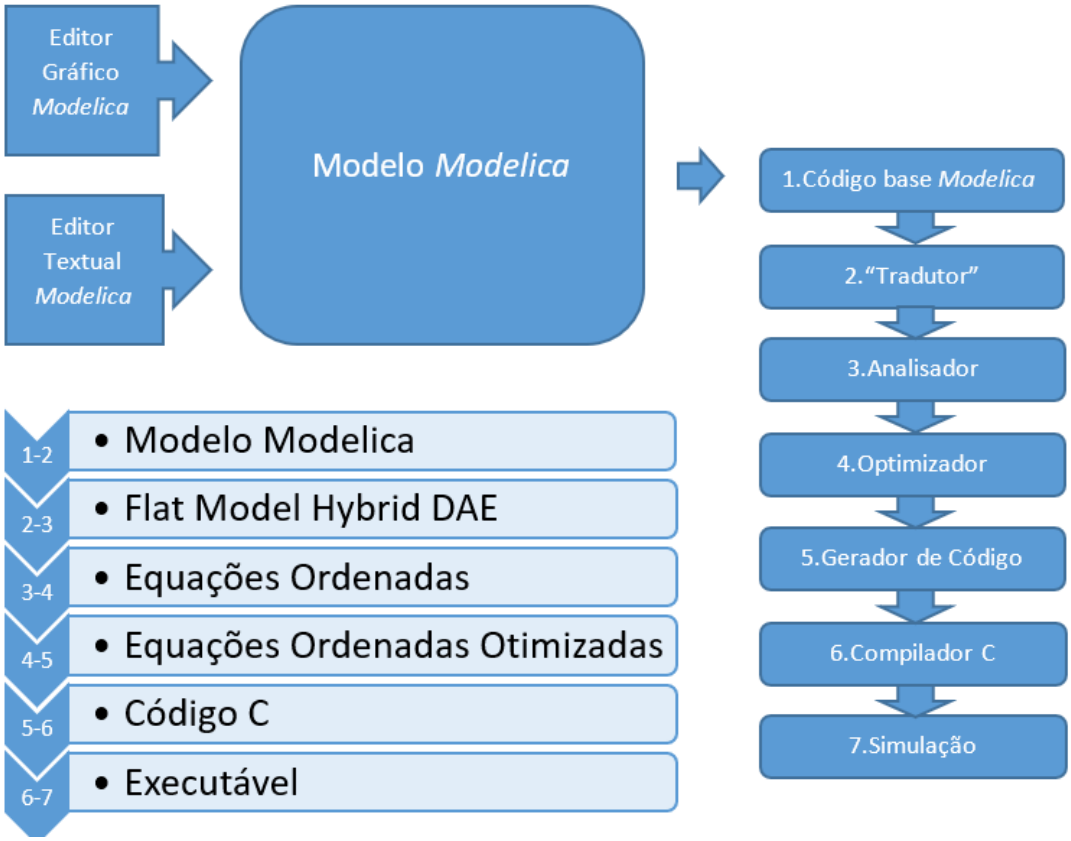


Figura 8 - Etapas realizadas pelo programa *OpenModelica* (ModelicaShortCourse, s.d.)

Modelica é uma linguagem declarativa de alto nível que permite descrever comportamentos matemáticos (M.Tiller, 2019). É usualmente aplicada em sistemas de engenharia e pode ser usada para facilmente descrever o comportamento de diferentes tipos de componentes em áreas específicas. Estes componentes por sua vez, podem ser combinados em sistemas, subsistemas e diferentes arquiteturas que podem daí advir. Relativamente aos aspetos que destacam esta linguagem relativamente às demais, sublinha-se a capacidade em descrever, de forma conjunta, comportamentos contínuos e comportamentos discretos num sistema, o facto de suportar aproximações casuais e acasuais dentro do mesmo modelo e por fim o facto de ser uma linguagem "aberta" cuja especificação é facilmente adquirível.

Modelica – Breve Evolução da Linguagem

Primeiro encontro de pessoas relativo ao design com recurso à ferramenta *Modelica* decorreu em 1996. Este encontro reuniu pessoas com alto nível de especialização no desenvolvimento de linguagem assim como em modelação física.

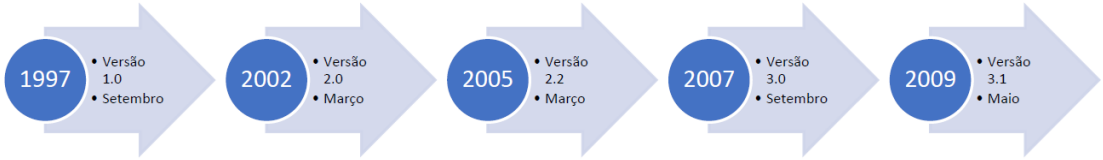


Figura 9 - Histórico das primeiras versões *Modelica* (OpenModelica, s.d)

A linguagem *Modelica* é multi-domínio. Esta foi concebida para dar suporte a todos os domínios de Engenharia, não apenas um. *OpenModelica* inclui a *Modelica Standard Library*, sendo que a sua versão mais recente apresenta:

Modelica Standard Library	**1288**	Modelos de Componentes e Blocos
	404	Exemplos de Modelos
	1227	Funções

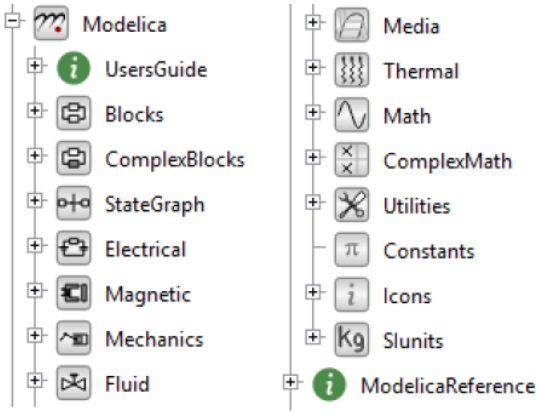


Figura 10 - *Modelica Standard Library* (OpenModelica, s.d)

São providenciadas ainda um conjunto de bibliotecas especializadas, passíveis de serem integradas na modelação e que se focam em sistemas específicos de engenharia.

Modelica e a robustez de um processo de engenharia

Um processo de engenharia robusto, pode ser definido como um processo em que os erros, devido aos diferentes engenheiros usarem diferentes tipos de informação, não ocorrem. Não ocorre gasto em tempo na execução de cálculos ao mover um modelo de uma ferramenta

para outra (como por exemplo CAD para um modelo matemático ou vice-versa) e o modelo otimizado é atingido sem que a mudança de assunções e objetivos durante o processo se interponham e gerem incongruências conjunturais.

Esta robustez pode ser atingida com a utilização da linguagem *Modelica* na exploração de processos concorrentes, em que a constante tradução entre diferentes formatos de informação adjacentes à construção de diferentes modelos e respetiva análise de forma sequencial, possa ser evitada num sistema complexo maior.

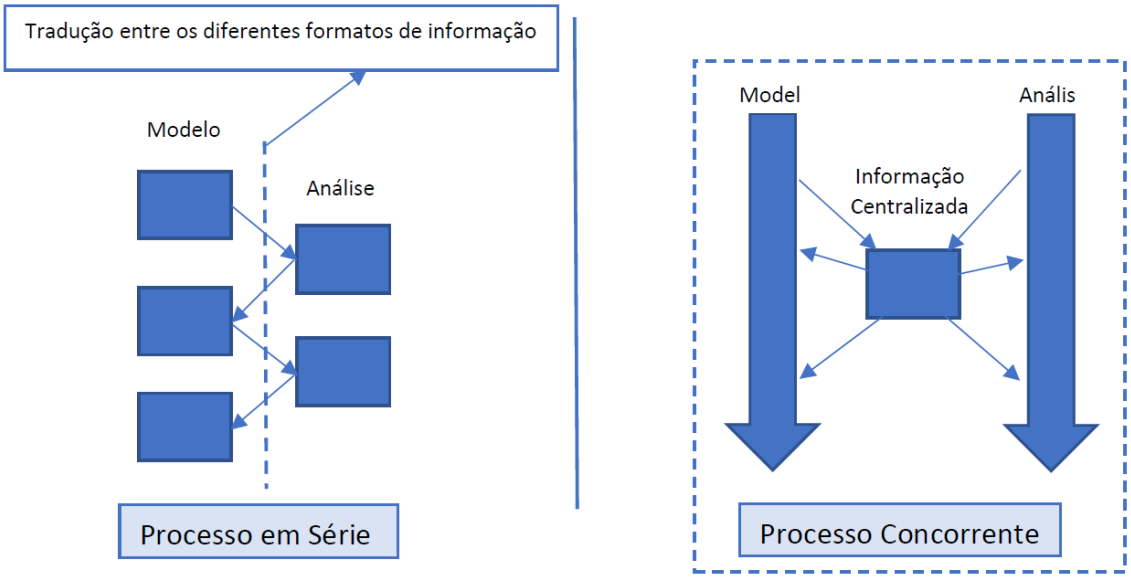
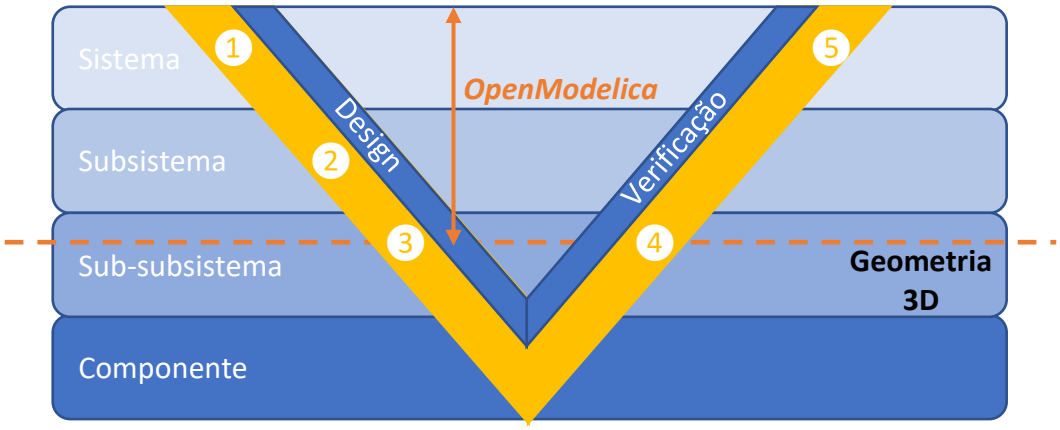


Figura 11 - Processo de Engenharia em Série e Concorrente

A significância desta particularidade, advém do facto de que esta linguagem pode ser usada também para a gestão ao longo do ciclo de vida do produto e da sua produção. Se se tiver em consideração por exemplo a indústria automotiva, onde *Modelica* encontra enorme aplicabilidade, esta pode ser utilizada para centralizar informação e albergar assim as diferentes áreas de engenharia que estão a ser trabalhadas. É sabido que para a simulação de um sistema hidráulico, sistema aerodinâmico, desenvolvimento de um sistema de controlo, etc, diferentes ferramentas de simulação computacional podem ser utilizadas respetivamente, pelo que desfasamentos entre informações cruciais e de importância generalizada possam falhar. Desta forma, a linguagem *Modelica* pode ser utilizada para certificar que os diferentes departamentos estão a trabalhar sob as mesmas assunções, através da geração de código de simulação e o envio às diferentes plataformas.

5. ENQUADRAMENTO DO PROGRAMA *OPENMODELICA* NUM SISTEMA DE ENGENHARIA

Dentro do processo associado a um sistema de engenharia, o programa *OpenModelica* poderá ser utilizado em diferentes estágios do mesmo.



1. Estudos de Arquitetura
2. Otimização do Design
3. Condições de Fronteira
4. Redução do Modelo/Calibração
5. Hardware em Loop

Figura 12 - Diagrama em V de um sistema de engenharia e os estágios constituintes

Cada um destes estágios será examinado minuciosamente tendo por base a modelação orientada a objetos, em que *OpenModelica* se baseia.

1 **Avaliação do conceito e estudos de arquitetura –** Um dos estágios de um sistema de engenharia no qual o *OpenModelica* se poderá ocupar é ao nível do Sistema. É aqui que o processo do desenvolvimento do produto se inicia e é comum explorar diferentes potenciais configurações ou arquiteturas. As diferentes soluções são avaliadas procurando definir uma que satisfaça na plenitude os interesses do cliente, desde a *performance*, economia, funcionalidade, sustentabilidade, etc.

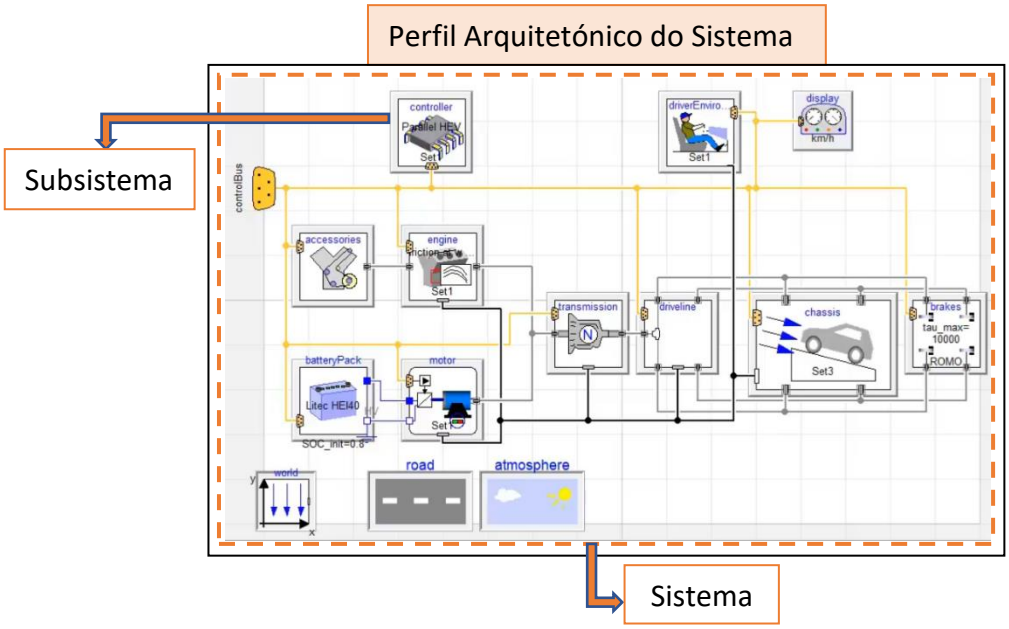


Figura 13 - Arquitetura de um Sistema Modelado

Dentro de cada subsistema é possível explorar diferentes opções e dessa forma perceber e avaliar as diferentes configurações que podem ser implementadas no sistema.

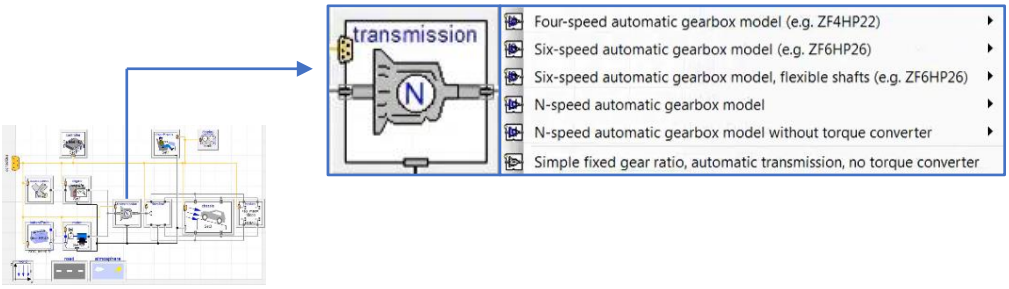


Figura 14 - Diferentes Configurações dentro de um Subsistema. Interface de escolha avançada disponibilizada noutros programas que não são *opensource*

2 **Conceção de princípio e otimização de conceito –** Estabelecida a conjuntura apropriada para o sistema, o processo evoluirá para a caracterização específica de cada subsistema. E é nesta fase que a caracterização o mais preditiva possível do subsistema a ser modelado é requerida. Com essa projeção em mente, é essencial que a aproximação feita pela modelação seja baseada numa série de princípios basilares como a conservação de quantidades físicas como momento, energia, massa, etc.

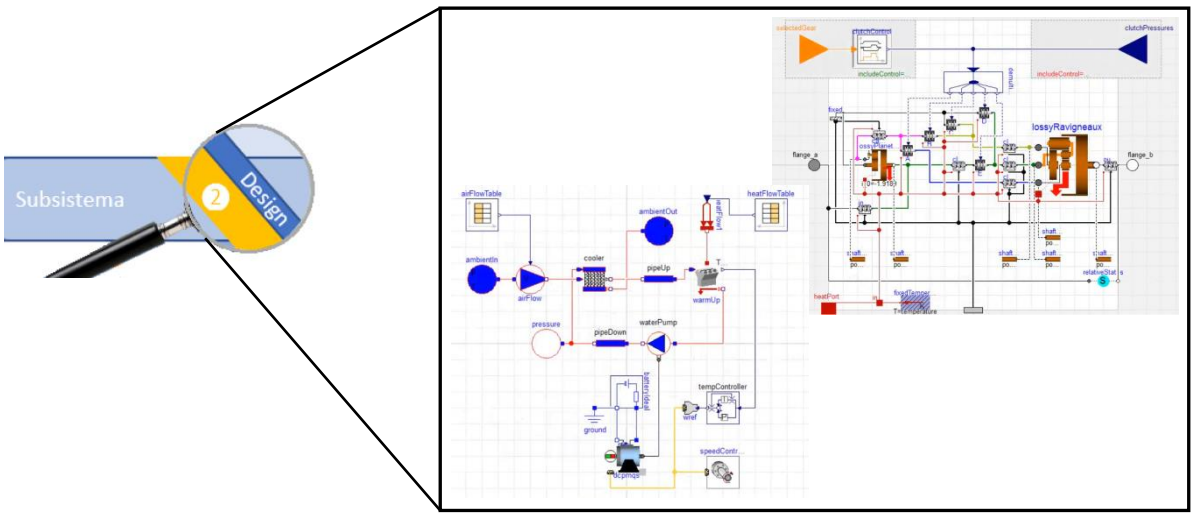


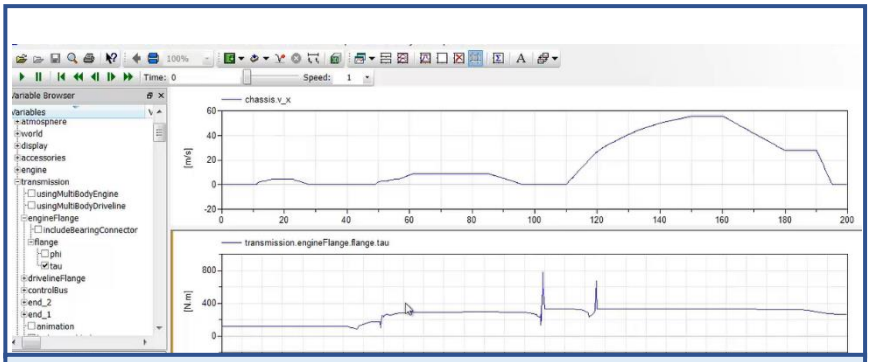
Figura 15 - Diagramas dentro de Subsistema

Uma boa formulação dos modelos adjacentes aos subsistemas estará assente nos seus diferentes parâmetros. Um bom modelo terá assim uma capacidade de previsão relativamente precisa independentemente dos parâmetros e, de forma consequente, do design final associado.

Finalizada a conceção dos princípios de desenvolvimento dentro de um modelo do subsistema, como por exemplo os de conservação, a próxima etapa redireciona-se para a otimização do design desse subsistema. A otimização poderá ser explorada com ferramentas específicas do programa *OpenModelica* (bibliotecas), onde são delineados os objetivos de performance, os parâmetros e os constrangimentos finais exigidos. Desta forma, valores otimizados são concebidos pelo programa, dando resposta às exigências postuladas.

3 **Geração das condições fronteira –** Por esta altura do processo, encontra-se no limite daquilo que é possível ser concebido com a aproximação ao sistema de engenharia segundo o modelo concentrado.

Aspetos adicionais do projeto precisam de incluir modelações de geometria 3D de forma a avaliar, de forma precisa, a performance de subsistemas em determinados componentes. Desta forma, o software *OpenModelica* permite a geração de condições fronteira, para o sistema ou para os subsistemas, pois esta é necessária para a subsequente avaliação e análise dos diferentes componentes pertencentes ao sistema.



No caso presenteado em cima tem-se o estudo do eixo de transmissão de um automóvel, sujeito a um torque proveniente do motor. Este torque que será fornecido pelo motor dependerá de uma série de fatores, fatores estes que poderão ser encapsulados no modelo associado à transmissão. Assim, não só nos será apresentado o torque médio à saída do motor, mas também como o torque varia ao longo do tempo em função de diferentes condições de operação. Estas condições fronteira a serem utilizadas, posteriormente, nos diferentes componentes 3D modelados, definirão assim um conjunto de valores mais fieis a serem utilizados, ao invés de um valor fixo.

Figura 16 - Valores mais fiéis em função de Condições Fronteira

Usando a geometria 3D de componentes assim como condições fronteira, o processo de análise do projeto ao nível do componente, progride para o ponto em que este se encontra completo.

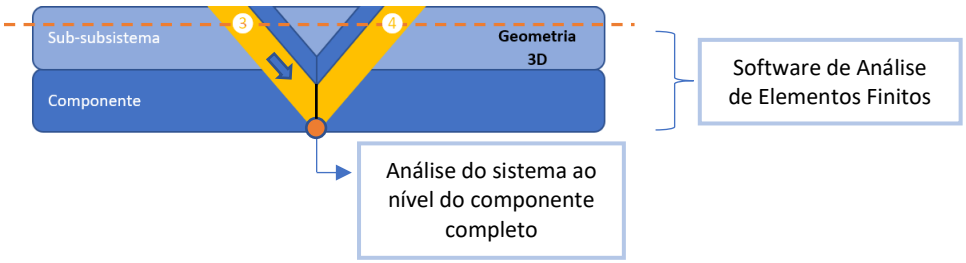


Figura 17 - Fase final do Diagrama em V

4 **Redução do Modelo e Calibração –** Analisado o sistema ao nível do componente, entra-se numa fase de verificação ao longo do processo da engenharia de sistemas. Esta fase começa

ao nível do componente, onde através da sua modelação procura-se averiguar se o componente atinge todos os seus requisitos de *performance*. A utilização de todas as modelações geométricas ao longo de todo o processo de verificação seria do ponto de vista computacional, incrivelmente dispendioso. É nesse sentido que a utilização dos modelos concentrados (modelos estes utilizados na fase de *Design*, no lado esquerdo do diagrama da engenharia de sistemas) é o ideal a ser posto em prática para continuar o processo de verificação, presente no lado direito do diagrama em V. Nos processos concentrados elaborados na fase de Design, são apresentadas as caracterizações que melhor definem as previsões para o comportamento final e real do componente. No entanto para a verificação, as previsões devem ser baseadas em caracterizações mais precisas, alicerçando-as nas diferentes modelações 3D dos componentes, previamente elaboradas. Essas características que melhor definem o componente podem ser extraídas projetando diferentes experimentos que permitem avaliar e analisar essas mesmas características do componente recorrendo para isso a um modelo de redução ou processo de calibração. Uma vez extraídas, elas podem ser inseridas de volta nos modelos concentrados já concebidos e aí serem realizadas as diferentes tarefas de verificação desejadas, de forma relativamente rápida, utilizando os modelos concentrados que agora contêm representações especificas do nosso componente e subsistema. A biblioteca de otimização providenciada pelo *OpenModelica* pode ser utilizada para auxiliar todos estes esforços no estudo de engenharia. A ferramenta de calibração na biblioteca permite importar dados de testes e simulações de componentes ou subsistemas, e a partir daí servir de base para a resposta que se quer observar nos modelos concentrados.

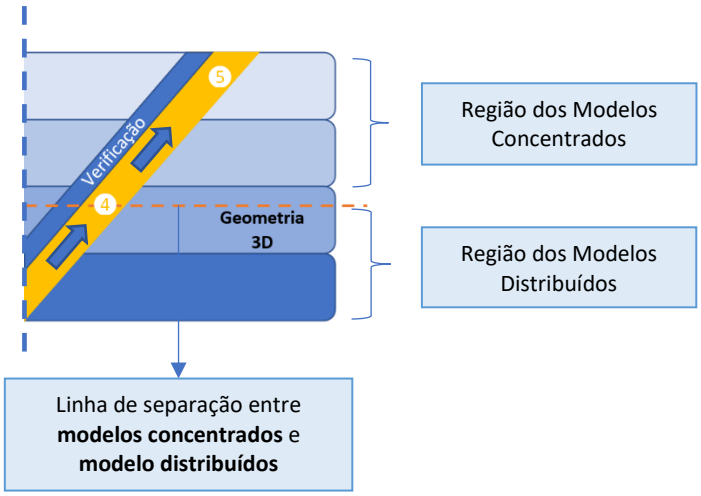


Figura 18 - Modelos Concentrados e Modelos Distribuídos

5 **Interface Mock-Up (FMI) funcional –** No contexto ainda da Engenharia de Sistemas, a *FMI* permite a interoperabilidade de modelos compilados entre ferramentas computacionais. Isto torna-se relevante no contexto da engenharia de sistemas porque dá suporte a situações com bastante utilidade. Por exemplo, *FMI* é uma excelente ferramenta para combinar modelos do controlador com o comportamento do sistema físico ou subsistema. *FMI* pode ser utilizado para dar suporte ao *Sotware in the Loop*, *Model in the Loop* e *Hardware in the Loop* (procedimentos de verificação), além de permitir apresentar interoperabilidade e co-simulação. Por fim, FMI pode ser utilizado para agregar modelos concentrados e distribuídos.

6. Etapas para Modelação de um Sistema Híbrido

Passos a serem consideradas na projeção de um modelo de um sistema híbrido em *OpenModelica* (OpenModelica, s.d):

- **Definir conceptualmente o sistema**

Conhecimento profundo sobre o sistema híbrido a ser modelado, ou seja, como os componentes dos diferentes domínios de engenharia, como por exemplo motores de combustão interna e motores elétricos, se combinam.

Figura 19 - Exemplo de sistema híbrido

- **Iniciar de forma simples a sua modelação**

Proceder à modelação de um domínio específico do sistema de cada vez, começando com aqueles que poderão ser considerados os mais importantes ou determinantes no sistema.

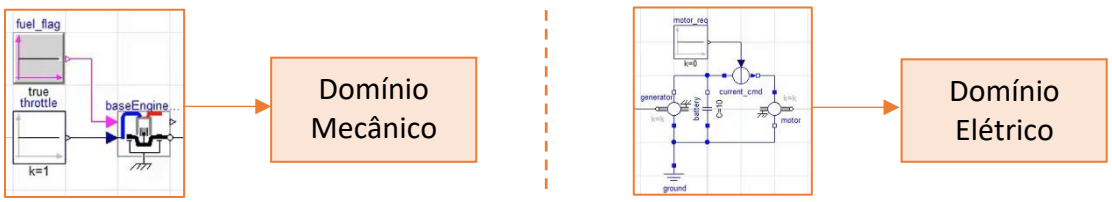


Figura 20 - Diferentes Domínios do Sistema

- **Refinar o modelo**

Englobar no sistema total, aspetos de igual importância como por exemplo transmissão, que serve de interação entre o domínio mecânico e elétrico, ou ainda perdas associadas.

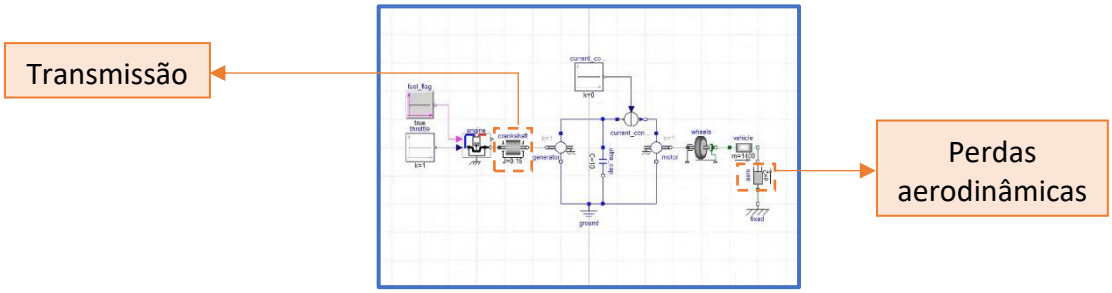


Figura 21 - Modelo Geral

- **Arquitetar o modelo de maneira a simplificar a gestão da sua configuração**

Decomposição em subsistemas. Reorganização do diagrama de forma a facilitar a gestão do mesmo, permitindo uma maior facilidade na sua reconfiguração e, de forma consequente, na capacidade de uma melhorar e testar as suas diferentes configurações.

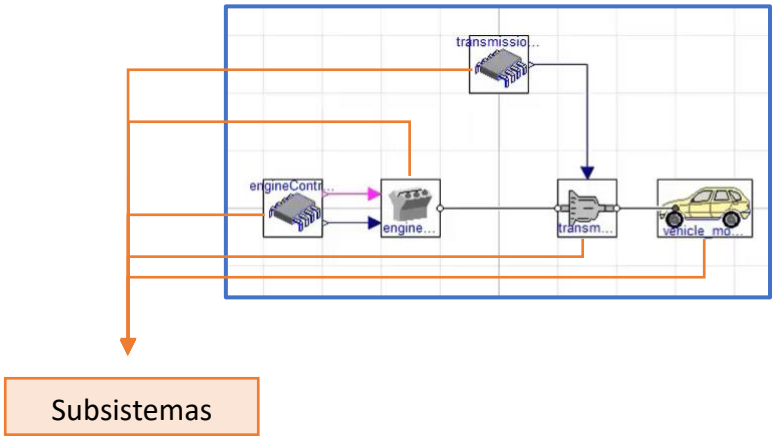


Figura 22 - Reorganização em Subsistemas

Em cada um dos subsistemas, pode-se elaborar uma interface em que as entradas e as saídas dos subsistemas ficam definidas. A partir dessa interface, diferentes implementações podem ser estabelecidas, possibilitando testar o modelo geral com uma diferente configuração dentro de um subsistema, sem que para isso um novo modelo geral seja construído de raiz.

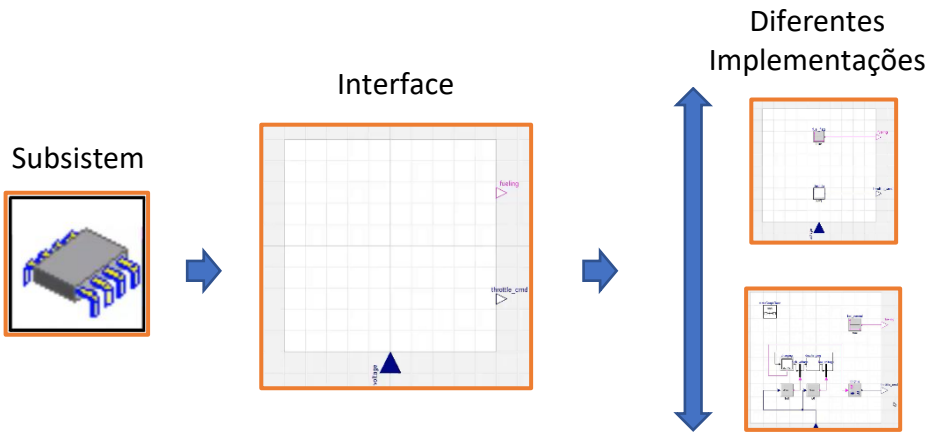


Figura 23 - Interface e Diferentes Implementações dentro de um Subsistema

- **Considerar diferentes estratégias de controlo**

Com a realização de diferentes implementações dentro das interfaces do subsistema, fica-se apto a poder rearranjar e definir diferentes modos de controlo do sistema, passando por exemplo, de um ciclo aberto para um ciclo fechado.

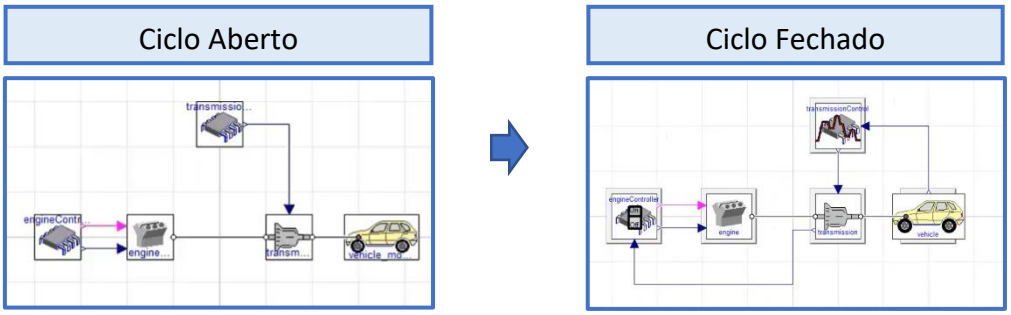


Figura 24 - Ciclo Aberto e Ciclo Fechado

7. SISTEMA HÍBRIDO ESCOLHIDO

Os sistemas híbridos surgem na indústria em áreas como controlo de tráfico, controlo de processos e automação de plantas (teoria de controlo), projeto de aeronaves, robôs e planeamento de percursos. A definição mais consensual de sistema híbrido em teoria de controlo apresenta-se como a interação entre sistemas de variáveis contínuas (por exemplo sistemas que podem ser descritos por um sistema de diferenças ou por equações diferenciais) e sistemas de eventos discretos, ou seja, sistemas assíncronos em que as transições de estado são iniciadas por eventos (R.K. Boel, 1999).

A necessidade de customização de produtos impulsiona a indústria de manufatura a buscar um método eficaz e eficiente para o desenvolvimento de células flexíveis de manufatura. Ao contrário de um sistema de produção em massa que produz apenas algumas variantes do produto, a indústria exige atenção a muitas variantes do produto que são produzidas por um curto período de tempo. Uma célula de manufatura é, desta forma, um sistema mecatrónico complexo, que exige um rigor de controlo mecânico, elétrico e automático. Toda a abrangência de domínios requerida no desenvolvimento de células de fabricação com um grau de personalização considerável, implica uma tarefa complexa, com muitos obstáculos, na conceção do sistema.

Um dos métodos que trouxe um impacto significativo na forma de lidar com todas as dificuldades supramencionadas, e que tem vindo a ser uma aproximação recorrente no ramo industrial, é a modelação e simulação. Esta, no entanto, peca por não fazer um enquadramento multi-domínio, na maioria da literacia científica até à data, sendo que o foco tem sido num domínio físico específico. Esta abordagem específica do ponto de vista disciplinar, torna os processos de análise e otimização relativamente dispendiosos, e é nesse sentido que as ferramentas de simulação multi-domínio surgem como uma solução particularmente interessante a explorar.

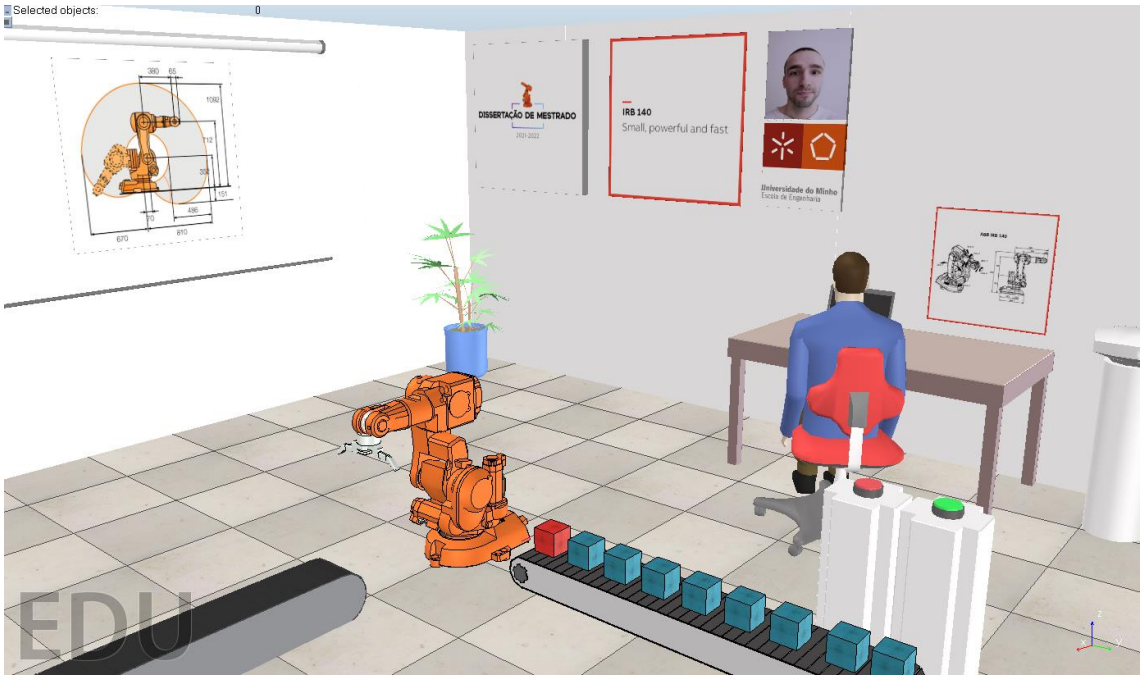


Figura 25 - Simulação de um sistema *Pick&Place* recorrendo ao *CoppeliaSim*

Na Figura 25 encontra-se apresentado o sistema escolhido. O mesmo caracteriza-se como um sistema *Pick and Place* industrial, constituído por 1 braço robótico e 2 *conveyors* ou tapetes de transporte. Por forma a complementar todo o processo de modelação e simulação computacional, orientada a objetos, dos componentes de maior importância, envolvidos em todo o processo do sistema, uma simulação do ambiente físico foi desenvolvida, paralelamente, permitindo desde logo integrar conhecimentos no âmbito da cinemática envolvida com manipuladores industriais e ainda desenvolver competências nos *Softwares* envolvidos. O ambiente de simulação utilizado foi o *CoppeliaSim* e o programa de desenvolvimento do *script* de simulação, foi o *Matlab*.

- **Funcionamento do Sistema**

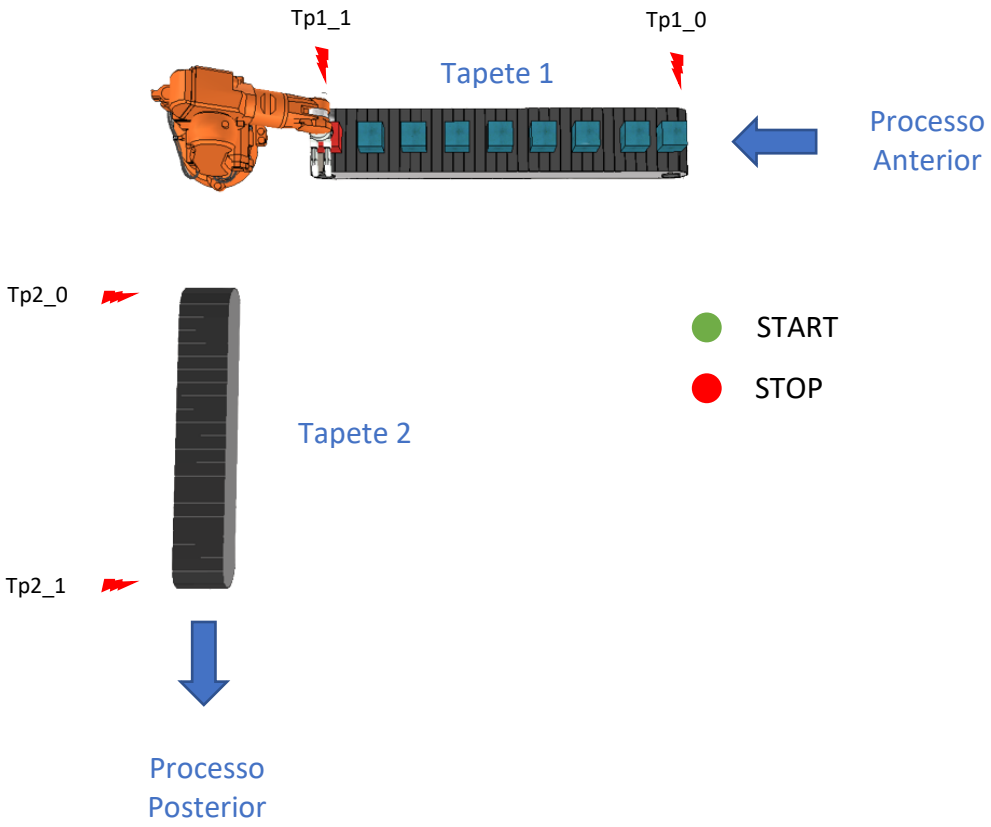


Figura 26 - Ilustração relativa a um sistema de funcionamento *Pick&Place* com um manipulador industrial

O sistema representado na Figura 26 esquematiza um processo industrial de produção, que tem como objetivo o transporte *Pick and Place* de uma embalagem, de uma linha de produção para outra. A sucessão de eventos ao longo deste processo de produção pode ser descrito do seguinte modo:

- O botão START inicia o processo, o motor que move o tapete 1 é acionado e começa a movimentação da caixa ao longo da linha. Caso nenhuma caixa tenha sido detetada, o acionamento do motor não é realizado.
- Quando a presença da caixa é detetada na zona de recolha, ou seja, na secção final do tapete 1, o braço robótico inicia o seu movimento *Pick and Place,* partindo da *Pose Position.*
- Uma vez transportada e colocada a caixa na secção inicial do tapete 2, esta é detetada permitindo o acionamento do motor associado ao tapete 2. Quando a caixa se encontra localizada no final do tapete, o movimento do tapete é interrompido, sendo a caixa posteriormente recolhida para uma tarefa a ser realizada numa fase posterior.

Um estudo mais detalhado, em que uma análise das entradas e saídas do sistema é efetuado, assim como a elaboração do *Grafcet* do processo, poderá ser encontrado em apêndice (1 e 2). O estudo do Sistema estendeu-se assim a várias frentes, procurando torná-lo o mais completo possível. Esquematizado em baixo (27) encontra-se sequenciado o estudo do sistema desenvolvido, atribuindo maior foco à última fase, em que a modelação orientada a objetos foi executada.

Figura 27 - Divisão do estudo do sistema

Relativamente à modelação orientada a objetos do sistema, esta procurou escrutinar cada um dos seus componentes do sistema, tendo por base os diferentes domínios de engenharia nele integrados. Com foco nos diferentes componentes do sistema, a abordagem da modelação orientada a objetos poderá ser projetada da seguinte forma.

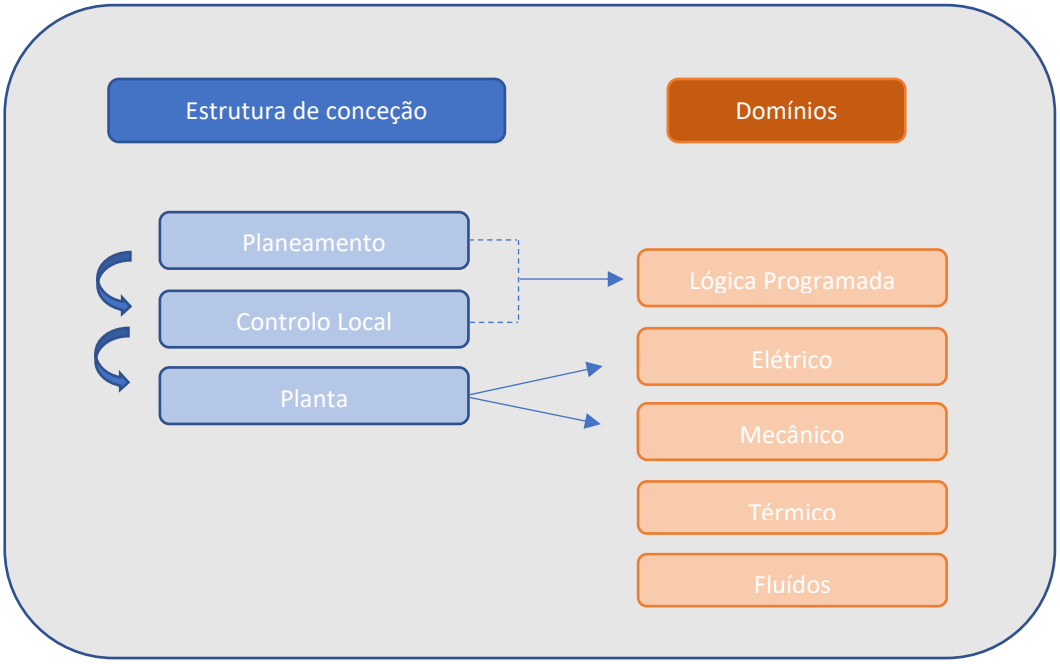


Figura 28 - Modelação orientada a objetos de um sistema *Pick&Place* - Estrutura de Conceção e diferentes áreas de estudo

O modelo global estudado, debruçar-se-á, tendo em perspetiva a estrutura de controlo apresentada, no planeamento ao longo de um ciclo de operação, concebendo para isso algoritmos de percurso, no controlo local, projetando para o efeito controladores *PI – PID* e por fim na planta, onde se inclui os atuadores assim como os processos físicos inerentes.

8. MODELICA – PROJETO DESENVOLVIDO

No sentido de levar a cabo a modelação orientada a objetos de um sistema de funcionamento *Pick&Place*, um estudo pormenorizado de um modelo já existente na *Modelica Standard Library* foi realizado (diagrama 1 na Figura 29). Este modelo (.Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.FullRobot, s.d.) foi o ponto de partida do projeto, na medida em que parte da sua estrutura serviu de base para a elaboração do modelo do manipulador e a partir do qual uma reestruturação e integração com um sistema global foi levada a cabo. Na Figura 29 é possível observar a ideia subjacente, no qual pode-se observar uma remodelação do sistema destinado ao robot/manipulador (diagrama 2 da Figura 29), com a introdução de novos princípios no domínio de comando assim como na modelação do processo físico do manipulador em si. Um modelo do manipulador foi elaborado, assim, de raiz, com o propósito não só de integrar alguns princípios de modelação pré-desenvolvidos como também articulá-los com novos modelos através de lógica programada, e dessa forma criar um sistema mais dinâmico, mais intuitivo, com a projeção de uma maior simplicidade na interação com o utilizador, e ainda que sirva o propósito complexo de estudar um processo *Pick&Place*, que encontra a sua enorme utilidade em processos industriais.

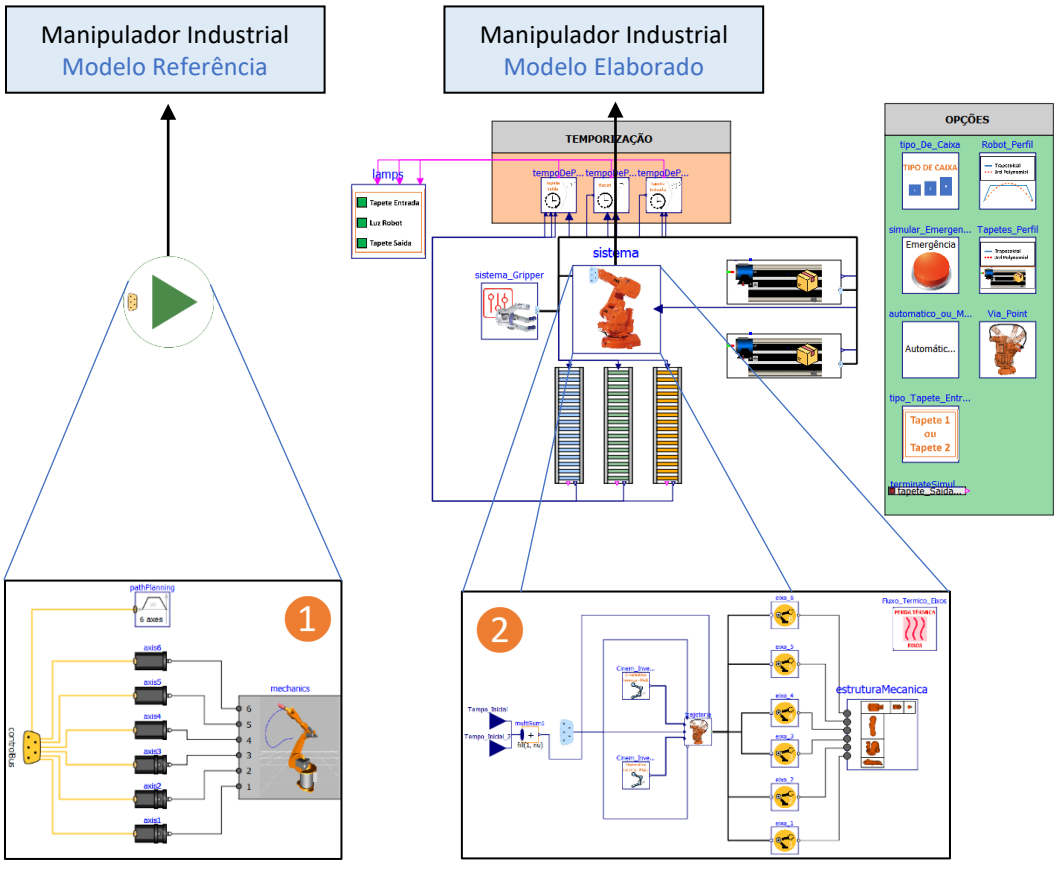


Figura 29 - Modelo de Referência do manipulador à esquerda. Modelo do manipulador elaborado, à direita, incorporado num sistema global *Pick&Place* desenvolvido

O desenvolvimento do manipulador industrial, além de um estudo detalhado envolvido no modelo de referência, já desenvolvido no *Modelica Standard Library*, permitiu acrescentar:

- Introdução do conceito de Cinemática Inversa, melhorando a logística de funcionamento e simulação do manipulador, para tarefas específicas (entre as quais, a de *Pick&Place*).

- Complementaridade apresentada no estudo do perfil do movimento, projetando não só um perfil de velocidades trapezoidal, como também um perfil de velocidades polinomial (de 3ª e 5ª ordem).

- Conceber um *Path Planning* totalmente projetado para uma tarefa *Pick&Place*, com todas as restruturações e melhorias que o mesmo implica.

- Desenvolvimento de um modo com *Via Points,* procurando retratar ao máximo um sistema de recolha e entrega observado no mundo industrial.

- Estudo das perdas térmicas nos eixos verificadas ao longo do funcionamento do manipulador.

- Restruturação hierárquica do controlador, tornando o sistema de controlo em cascata mais percetível e dessa forma facilitar eventuais melhorias.

Sendo que, por outro lado, o desenvolvimento de um Sistema Global no qual o modelo do robô se encontra inserido, permitiu acrescentar:

- Conceptualização e desenvolvimento de um *Gripper* a ser acoplado ao manipulador, permitindo desde logo uma projeção mais realística do funcionamento de um robot industrial.

- Conceptualização e desenvolvimento de tapetes transportadores, com um sistema de arrefecimento integrado.

- Modelação com sequência de funcionamento Tapete (entrada) → Robot/Gripper → Tapete(saída) com recurso a lógica programada. Modos de operação foram ainda desenvolvidos com a definição distinta de parâmetros funcionais.

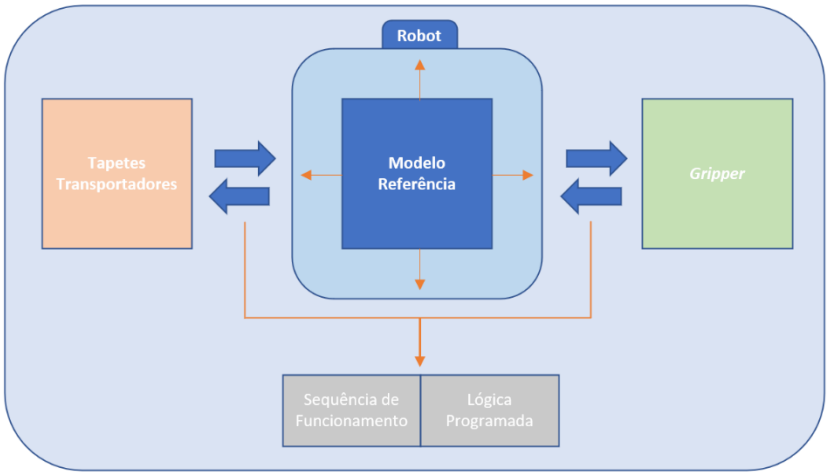


Figura 30 - Sequência de funcionamento estabelecida com recurso a troca de informações entre os modelos dos tapetes, do manipulador e ainda do gripper

Simulado o ambiente físico do sistema em *CoppeliaSim*, permitindo uma maior assimilação dos conceitos gerais inerentes ao sistema, do qual se destaca uma compreensão exímia da cinemática envolvida no movimento do braço, assim como estudada uma forma simples de controlo do braço que possa ser implementada, foi realizada de seguida a modelação orientada a objetos do sistema global, sendo que neste, certas funções foram exploradas de forma mais clara, dos quais se destacam:

- A modelação do braço robótico, com um modelo de cinemática inversa incorporado, permitindo dessa forma o movimento do braço ao longo da tarefa *Pick and Place*, através das coordenadas cartesianas inseridas, na extremidade do braço.

- A modelação de tapetes/*conveyors,* responsáveis pelo transporte da embalagem ao longo do sistema. O sistema é constituído por um tapete de entrada e um tapete de saída, sendo que o tapete de entrada respetivo apresenta uma modelação estabelecida em função das opções do utilizador. Para efeitos do tapete de entrada, existem 2 tapetes possíveis cuja modelação é distinta. Num é apresentado um perfil de modelação mais simplificado, em contraste com o tapete adjacente, cuja modelação do sistema físico é ligeiramente mais complexa, envolvendo um maior número de variáveis em estudo. Esta distinção permite desde logo comparar e avaliar diferentes *performances* para diferentes modelações do mesmo subsistema.

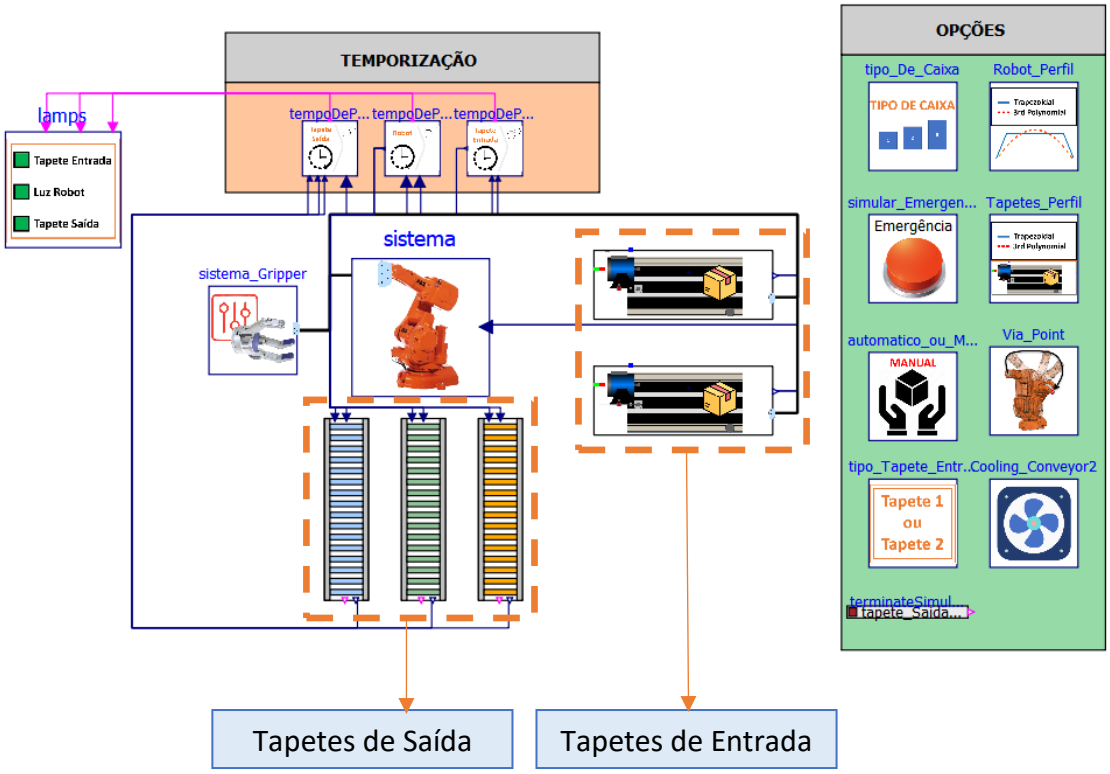


Figura 31 - Modelo do Sistema Global

Relativamente aos tapetes de saída, estes diferenciam-se pelo tipo de embalagem/caixa que transportam, sendo que no caso do último tapete (tapete de saída 3) é apresentado um percurso do transporte da caixa diferenciado com 2 paragens sucessivas, para diferentes células de operação, respetivamente.

8.1.Robot

Neste subcapítulo, a conceção do comando do manipulador, no qual se inclui o cálculo da cinemática inversa assim como os perfis de trajetória, será estudada numa fase inicial, acompanhada pela sua implementação no sistema. De seguida, o controlo dos diferentes eixos, por ciclos de *feedback,* é estudado e implementado, precedido por uma análise cuidada da tecnologia de controlo implementada no modelo industrial *ABB IRB 140* e os reducionismos implicados numa modelação orientada a objetos. Por fim, a modelação da estrutura mecânica é projetada.

Neste capítulo será apresentada a modelação do modelo do braço robótico global.

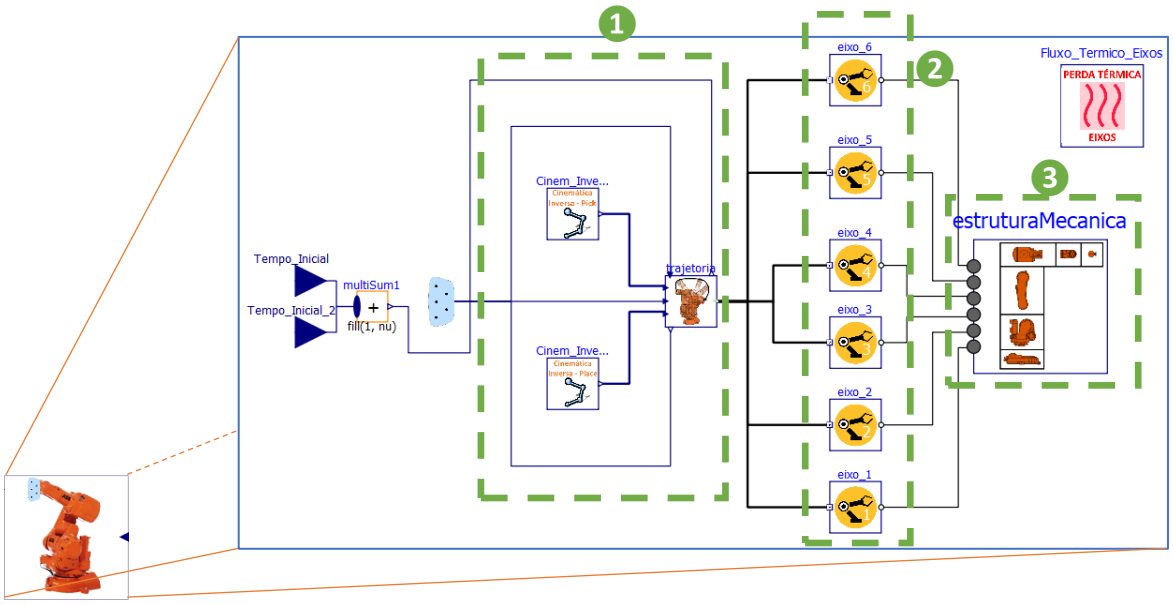


Figura 32 - Modelo do manipulador constituído por diferentes secções

O modelo de hierarquia superior elaborado, em relação ao braço robótico, pode ser dividido em diferentes etapas:

- 1 - Uma etapa destinada à delineação do percurso que deve ser efetuado pelo braço, no qual se inclui o cálculo da cinemática inversa resultando nos valores angulares das juntas a serem alcançados ao longo do tempo e, ainda, desenvolver através desses valores angulares os perfis de posição e velocidade ao longo dos diferentes eixos.
- 2 - Uma etapa de controlo e transmissão do movimento. Em que os perfis calculados anteriormente servirão como um conjunto de valores de referência em controladores. Servomotores são os respetivos atuadores associados aos controladores, funcionando como a planta do sistema de controlo, onde um sistema de transmissão é ainda acoplado.
- 3 - Uma etapa destinada exclusivamente à modelação da estrutura mecânica do braço, em que a estrutura de referência dimensional será a do manipulador industrial *IRB 140* da *ABB*.

8.1.1. Cinemática Inversa

O primeiro passo para que o movimento do manipulador ao longo da operação ocorra como desejado, é um correto cálculo dos valores envolvidos que propiciam o seu movimento cinemático, nomeadamente uma precisão na apresentação dos valores das juntas de cada eixo (valores das juntas iniciais e valores das juntas finais). Para o efeito, um modelo de cinemática inversa foi desenvolvido, visando calcular a variação dos valores angulares das juntas, para que o percurso implicado na operação ocorra.

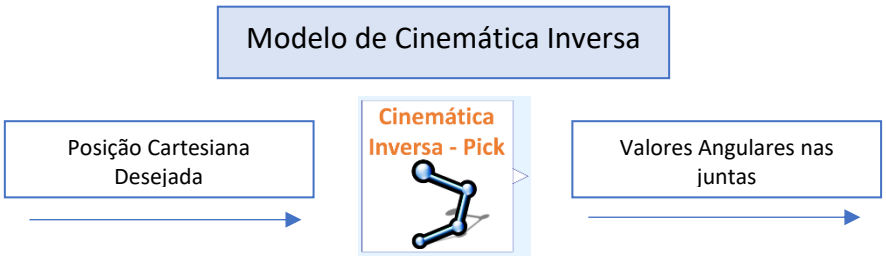


Figura 33 - Objetivo do modelo de Cinemática Inversa

A elaboração deste modelo foi realizada, na sua totalidade, recorrendo ao *text view editor,* ou seja, o cálculo da variação angular das juntas foi inteiramente programado em código. Tendo a concretização do mesmo sido elaborada utilizando as equações de cinemática inversa já previamente calculadas em *Matlab* (apêndice 2).

Comparando *scripts* realizados em *Matlab* e *Modelica* relativos à Cinemática Inversa, salta em evidência o facto de que em *Modelica* uma "fragmentação" do código ser necessária, tornando por vezes pouco fluída a realização do mesmo. A divisão, entre parâmetros/inputs e o algoritmo propriamente dito, é necessária em *Modelica*, ao contrário de *Matlab,* em que tudo poderá ser efetuado de forma sequencial, tornando o acompanhamento do algoritmo mais linear e intuitivo.

Este modelo permite simplificar desde logo o sistema, uma vez que é mais fácil ao utilizador saber a posição cartesiana final da ponta do braço, de maneira a realizar o *Pick,* do que saber a variação do valor angular de todas as juntas para que o movimento ocorra como desejado (como sugerido no modelo de referência) e a configuração final seja atingida.

Visto que para um ciclo de funcionamento, o transporte implicado deve assegurar a recolha da embalagem num tapete de entrada e o largar a embalagem noutro tapete de saída, pelo menos dois movimentos deverão ser assegurados. Nesse sentido, dois modelos de cinemática inversa foram utilizados, um para o *Pick* e outro para o *Place*, onde os únicos elementos

diferenciadores entres estes modelos acabam por ser as coordenadas cartesianas definidas (iniciais e finais).

Ao longo desta secção tem-se a apresentação da componente teórica (Erlhagen, 2020) destinada ao cálculo da cinemática inversa, fazendo-se esta acompanhar de forma simultânea pela implementação no modelo de cinemática inversa em *OpenModelica*.

8.1.1.1. Teoria geral

8.1.1.1.1. Atribuição de Referências

Esta atribuição de referências servirá de ponto de partida para a determinação dos parâmetros de *Denavit – Hartenberg* (Hayat, Chittawadigi, Udai, & Saha, 2013), que se baseia nas dimensões apresentadas pelo manipulador industrial IRB – 140 (ABB AB, 2004-2017). Os diferentes referenciais ao longo da estrutura definem-se através da atribuição de um conjunto de eixos coordenados em cada elo da estrutura.

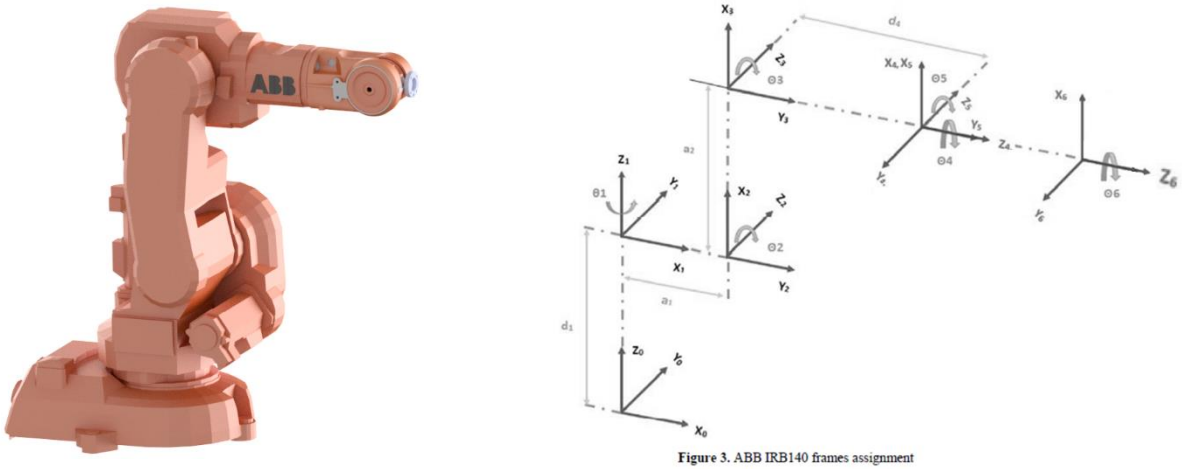


Figura 34 - Manipulador IRB 140 e conjunto de eixos coordenados

Sendo que cada uma das dimensões apresentadas na Figura 34 ditam os seguintes valores (Tabeça 3).

Tabela 3 - Dimensões envolvidas nas translações entre os diferentes referenciais

	(m)
d1	0.352
a1	0.07
a2	0.36

d4	0.38
d6	0.065

Estas dimensões apresentam a relevância de virem a ser utilizadas na definição dos parâmetros de *Denavit – Hartenberg*, determinando a posição dos diferentes pontos de referência, no entanto caso se foque em todas as dimensões ao longo da estrutura, e que terão a sua utilidade no desenvolvimento da cinemática inversa, as mesmas poderão ser descritas segundo o vetor {0.352; 0.07; 0.177; 0.36; 0.177; 0.38; 0.065}.

8.1.1.1.2. Parâmetros de Denavit – Hartenberg

Tendo-se por base a Figura 34 é possível determinar os parâmetros de *Denavit-Hartenberg*, que se encontram presentes na tabela 4. Para cada referencial existem 4 parâmetros, que originam uma matriz transformação (Craig, 2005) (Siciliano, Sciavicco , Villani, & Oriolo , 2010). Esta estabelece a relação entre o referencial anterior (i-1) e o seguinte (i). Os parâmetros a_i e α_i referem-se ao deslocamento e à rotação no eixo x, respetivamente, tendo em conta o referencial seguinte. Os parâmetros d_i e θ_i referem-se ao deslocamento e à rotação no eixo z, respetivamente, tendo em conta também o referencial seguinte.

Tabela 4 - Parâmetros de *Denavit-Hartenberg* relativos à esquematização dos eixos coordenados distribuídos na estrutura do IRB 140

${}^{i-1}T_i$	a_i	α_i	d_i	θ_i
${}^{0}T_1$	0	0	-90	d1
${}^{1}T_2$	90	a1	90	0
${}^{2}T_3$	0	a2	0	0
${}^{3}T_4$	90	0	0	d4
${}^{4}T_5$	-90	0	0	0
${}^{5}T_6$	90	0	0	d6

8.1.1.1.3. Matrizes de transformação individuais

Obtidos os parâmetros de *Denavit – Hartenberg* pode-se recorrer à matriz apresentada (35) de maneira a obter-se cada uma das matrizes de transformação individuais.

$$ {}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \cos(\alpha_{i-1}).\sin(\theta_i) & \cos(\alpha_{i-1}).\cos(\theta_i) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1}).d_i \\ \sin(\alpha_{i-1}).\sin(\theta_i) & \sin(\alpha_{i-1}).\cos(\theta_i) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1}).d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Figura 35 - Matriz de transformação individual

Por outro lado, a matriz de transformação global do manipulador é obtida pelo produto entre as matrizes de transformação individuais apresentadas em cima, segundo a equação (1).

$$ {}^{0}T_6 = {}^{0}T_1 * {}^{1}T_2 * {}^{2}T_3 * {}^{3}T_4 * {}^{4}T_5 * {}^{5}T_6 \quad (1) $$

A matriz global apresentada, exibe na sua quarta coluna a posição do *end effector* em relação à base (P_{06}).

$$ {}^{0}T_6 = \begin{bmatrix} & & & {}^{0}x_6 \\ & & & {}^{0}y_6 \\ & & & {}^{0}z_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad P_{06} $$

Figura 36 - Posição do *end effector* em relação à base extraída da matriz global

A partir da matriz global é possível determinar não só a posição do *end effector* mas também a orientação do mesmo em relação à base. Esta orientação é definida pelas três primeiras linhas e três primeiras colunas presentes na matriz global, definindo assim a matriz RPY que equivale à matriz de rotação R_{06}, que define a orientação final do *end-effector* relativamente à base.

$$ RPY = (r_{ij})_{i,j=1,2,3} = \mathrm{Rot}(z,\phi_t)\,\mathrm{Rot}(y,\psi_t)\,\mathrm{Rot}(x,\theta_t) $$

$$ = \begin{bmatrix} c\phi_t\, c\psi_t & -s\phi_t\, c\theta_t + c\phi_t\, s\psi_t\, s\theta_t & s\phi_t\, s\theta_t + c\phi_t\, s\psi_t\, c\theta_t \\ s\phi_t\, c\psi_t & c\phi_t\, c\theta_t + s\phi_t\, s\psi_t\, s\theta_t & -c\phi_t\, s\theta_t + s\phi_t\, s\psi_t\, c\theta_t \\ -s\psi_t & c\psi_t\, s\theta_t & c\psi_t\, c\theta_t \end{bmatrix} $$

$$ \hat{x}_6 \qquad \hat{y}_6 \qquad \hat{z}_6 $$

Figura 37 - Matriz de rotação R_{06} ou *RPY*, baseada nos ângulos *Roll*, *Pitch*, *Yaw*, ditam a orientação final do pulso

A matriz global presente na Figura 36 possui, também, a orientação do *end effector* nas suas primeiras três linhas e colunas, em que as colunas correspondem, respetivamente, à orientação em x ($\hat{x}_6$), y ($\hat{y}_6$) e z ($\hat{z}_6$).

É possível, também, determinar, com base na matriz de rotação e através das equações 3, 4 e 5 o valor dos ângulos que definem a orientação, *Roll* (Φ), *Pitch* (Ψ)e *Yaw* (θ) segundo a nomenclatura da equação 2

$$RPY = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (2)$$

$$\Phi = atan_2(r_{21}, r_{11}) \quad (3)$$

$$\Psi = atan_2\left(-r_{31}, \sqrt{{r_{32}}^2 + {r_{33}}^2}\right) \quad (4)$$

$$\theta = atan_2(r_{32}, r_{33}) \quad (5)$$

8.1.1.2. Determinação angular e Implementação no Modelo

O cálculo de cada um dos ângulos passará agora a ser analisado do ponto de vista teórico com a respetiva execução prática recorrendo a *Modelica*, mais concretamente à secção *Text View* para o desenvolvimento do modelo.

Antes de se debruçar detalhadamente em cada um dos ângulos, as matrizes em (1) e Figura 36 descritas na secção anterior serão implementadas. O conjunto de passos realizados ao longo da implementação serão devidamente descritos.

- Definição do comprimento dos elos, com base no vetor L e onde se incluem os valores das dimensões $d1, a1, a2, d4$ e $d6$ implicados nos parâmetros de *Denavit – Hartenberg*.
- Definição do vetor $Pose\_tip$, em que se incorporam nos 3 valores iniciais as coordenadas da posição final do *end-effector* (P_{06}) e nos 3 valores finais os ângulos *Roll*, *Pitch* e *Yaw* (RPY), sendo que os mesmos são arbitrados com valor de 0.
- Declaração da matriz rotação R_{06}, com base nos valores declarados para os 3 ângulos envolvidos na orientação do *end-effector* (RPY) e ainda a matriz associada ao *end-*

effector RPY_{tip} que define a matriz transformada entre o referencial base e o referencial do *end-effector*.

- Definição da matriz global T_{06}, com base na matriz de rotação R_{06} e no vetor de posição cartesiana do *end-effector* (P_{06}).

Sendo cada um dos passos observáveis no seguinte print.

```
  parameter Real L[:] = {0.352, 0.07, 0.177, 0.36, 0.177, 0.38, 0.065};
  parameter Real d1 = L[1];
  parameter Real a1 = L[2];
  parameter Real a2 = L[4];
  parameter Real d4 = L[6];
  parameter Real d6 = L[7];
  output Real joints[6];
  parameter Real Pose_tip[6, 1] = [0; 0.4965; 0.35; 0; 0; 0];
  Real P06[3, 1] = [Pose_tip[1]; Pose_tip[2]; Pose_tip[3]];
  Real RPY[1, 3] = pi / 180 * [Pose_tip[4], Pose_tip[5], Pose_tip[6]];
  Real yaw_x = RPY[1, 1];
  Real pitch_y = RPY[1, 2];
  Real roll_z = RPY[1, 3];
  Real R06[3, 3] = [cos(roll_z) * cos(pitch_y), (-sin(roll_z) * cos(yaw_x)) + cos(roll_z) * sin(pitch_y) * sin(yaw_x),
sin(roll_z) * sin(yaw_x) + cos(roll_z) * sin(pitch_y) * cos(yaw_x); sin(roll_z) * cos(pitch_y), cos(roll_z) *
cos(yaw_x) + sin(roll_z) * sin(pitch_y) * sin(yaw_x), (-cos(roll_z) * sin(yaw_x)) + sin(roll_z) * sin(pitch_y) *
cos(yaw_x); -sin(pitch_y), cos(pitch_y) * sin(yaw_x), cos(pitch_y) * cos(yaw_x)];
  Real RPY_tip[3, 3] = [0, 1, 0; 1, 0, 0; 0, 0, -1];
  Real RPY_tip_z[3, 1] = [RPY_tip[1, 3]; RPY_tip[2, 3]; RPY_tip[3, 3]];
  Real T06[4, 4] = [R06[1, 1], R06[1, 2], R06[1, 3], P06[1]; R06[2, 1], R06[2, 2], R06[2, 3], P06[2]; R06[3, 1],
R06[3, 2], R06[3, 3], P06[3]; 0, 0, 0, 1];
  Real P05_x = P05[1, 1];
  Real P05_y = P05[2, 1];
  Real P05_z = P05[3, 1];
```

Figura 38 - Primeiro *print* do código envolvido no modelo de cinemática inversa

- Cálculo da posição do pulso (P_{05}), equivalente à subtração entre a posição especificada para o *end-effector* (P_{06}) e a distância entre o *end-effector* e o pulso (junta 5) multiplicada pela orientação em z da matriz de rotação RPY (RPY_{tip_z}).

```
Real P05_x = P05[1, 1];
Real P05_y = P05[2, 1];
Real P05_z = P05[3, 1];
Real theta1_L;
Real theta1_R;
Real theta1;
Real theta1_deg = theta1 * 180 / 3.14159265358793238;
Real P05[3, 1] = P06 - (d6 + 0.0238) * RPY_tip_z;
```

Figura 39 - Segundo *print* do código envolvido no modelo de cinemática inversa

8.1.1.2.1. Determinação de θ_1 geométrico

A determinação prévia das coordenadas do pulso (P_{05}), permite obter geometricamente o valor de θ_1, tendo por base a projeção do vetor p_w no plano xy (Figura 40), através da expressão (6). Nesta expressão destaca-se a importância da utilização da função trigonométrica $arctan_2$, para que se tenha em conta a localização espacial do robô no cálculo de θ_1.

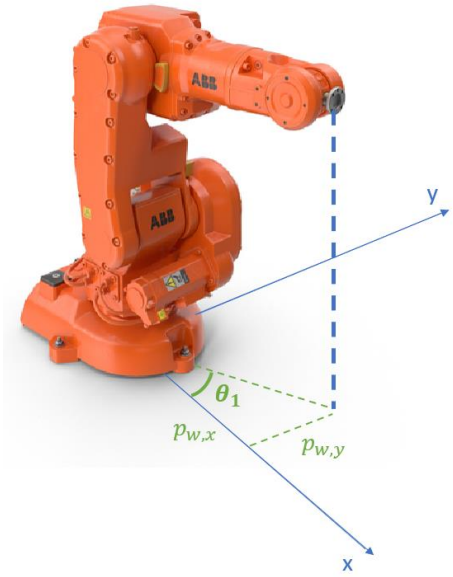


Figura 40 - Projeção do vetor p_w no plano xy

$$\theta_1 = \begin{cases} arctan_2(p_{w,y}, p_{w,x}) & se \quad p_{w,y} \neq 0 \\ 0 & se \quad p_{w,y} = 0 \end{cases} \tag{6}$$

Esta não é a única solução para θ_1, uma vez que este ângulo é de rotação sobre z. Deste modo, a segunda solução apresenta um cálculo muito simples:

$$\theta_1{}' = \theta_1 + 180^{\circ} \tag{7}$$

8.1.1.2.2. Cálculo de θ_1 - Implementação

Os ângulos responsáveis pelo posicionamento do pulso são o θ_1, θ_2 e θ_3. Pode-se ter a consideração que o ângulo 1 define o posicionamento do pulso ao longo do plano xy, enquanto o ângulo 2 e o ângulo 3 definem o posicionamento ao longo do plano xz. O ângulo 1 é definido como o arc-tangente da posição do pulso ao longo do plano xy. Se a variação verificada ao longo do eixo Y for próxima de 0, o valor do ângulo poderá ser desconsiderado e atribuída uma aproximação a 0.

- Definição dos valores das 2 soluções apresentadas para θ_1, $theta1\_L$ (quando o "cotovelo" esquerdo é utilizado) e $theta1\_R$ (quando o "cotovelo" direito é utilizado).

```
equation
  if abs(P05_y) <= 0.01 then
    theta1_L = 0;
  else
    theta1_L = atan2(P05_y, P05_x);
  end if;
  theta1_R = theta1_L + pi;
  if theta1_L < (-pi) or theta1_L > pi then
    theta1 = theta1_R;
  elseif theta1_R < (-pi) or theta1_R > pi then
    theta1 = theta1_L;
  else
    theta1 = theta1_L;
  end if;
```

Figura 41 - Terceiro *print* do código envolvido no modelo de cinemática inversa

8.1.1.2.3. Cálculo de θ_2 e θ_3

Para o cálculo de θ_2 e θ_3 o problema pode ser simplificado a um robot planar de 2 elos, em que um vetor entre o ombro e o pulso são definidos. Assim, a determinação de θ_2, à semelhança de θ_1 depende das coordenadas do *shoulder* (s) e do *wrist* (w), previamente determinadas, sendo o método de obtenção puramente geométrico.

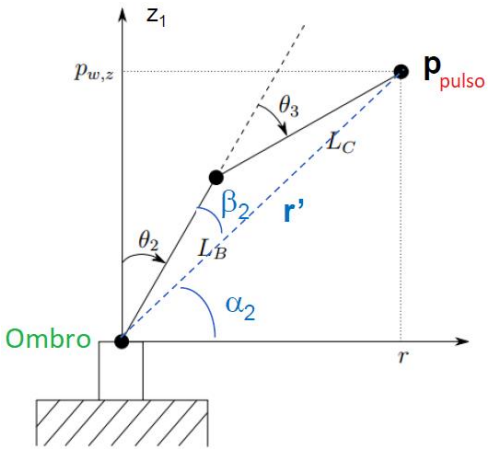


Figura 42 - Problema simplificado a um robot planar de 2 elos

Através da esquematização apresentada e do teorema do cosseno é possível extrair os ângulos β_2 e, de forma consequente, θ_2.

$$\begin{cases} \theta_2 = 90^o - \alpha_2 - \beta_2 \\ \alpha_2 = \arctan({}^{B}p_{ws,z}, r) \\ L_c^2 = L_B^2 + r'^2 - 2.L_B.r'.\cos(\beta_2) \end{cases} \quad (8)$$

Sendo que,

$$\beta_2 = \text{acos}\left(\frac{L_B^2 + r'^2 - L_c^2}{2\,.\,L_B\,.r'}\right) \quad (9)$$

Apenas existe se,

$$-1 \leq \frac{L_B^2 + r'^2 - L_c^2}{2\,.\,L_B\,.\,L_C} < 1 \quad (10)$$

Pelo teorema dos cossenos é ainda possível determinar β_3. Este assume-se como um ângulo considerado que auxilia na obtenção de θ_3. Salientando-se ainda que $r = \sqrt{p_{pulso,x}{}^2 + p_{pulso,y}{}^2}$.

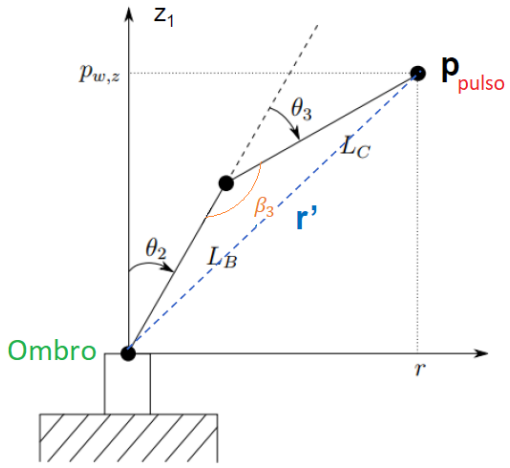


Figura 43 - Problema simplificado a um robot planar de 2 elos (β_3 presente em vez de β_2).

$$\beta_3 = \text{acos}\left(\frac{L_B^2 + L_c^2 - r'^2}{2\,.\,L_B\,.\,L_C}\right) \quad (11)$$

Com β_3 definido, e usando as propriedades geométricas do esboço, é possível obter a equação para o ângulo θ_3:

$$\theta_3 = \beta_3 - 90 \quad (12)$$

8.1.1.2.4. Cálculo de θ_2 e θ_3 – Implementação

- Cálculo do comprimento do vetor entre o ombro e o pulso r' ($r3$ no *script*), segundo a seguinte formulação $r3 = \sqrt{{P_{25_x}}^2 + {P_{25_y}}^2 + {P_{25_z}}^2}$. Precedido deste cálculo é apresentado o cálculo de P_{25} através da subtração entre o vetor entre a base e o pulso P_{05} e o vetor P_{02} definido entre a base e o segundo eixo do manipulador, sendo $P_{25} = \{P_{25_x},\ P_{25_y},\ P_{25_z}\}$.
- Cálculo de $\beta_3 (beta3)$ segundo a equação 11, precedido pelo cálculo do argumento $(arg3)\ \frac{L_B^2 + L_C^2 - r'^2}{2\ .\ L_B\ .\ L_C}$ em que L_B e L_C se igualam às dimensões $a2$ e $d4$, respetivamente.
- Cálculo das duas soluções de θ_3 ($theta3\_L$ e $theta3\_R$). A solução que perfila o cotovelo esquerdo dá-se pela equação 12, enquanto que a solução estabelecida pelo cotovelo direito é calculada através da seguinte fórmula.

$$\theta_3 = 270 - \beta_3 \tag{13}$$

- Cálculo de r' projetado ao longo do plano xy (r)
- Cálculo de α_2 ($alfa2$) recorrendo à cota do vetor P_{25} ($P25\_Z$) e r
- Cálculo de $\beta_2 (beta2)$ segundo a equação 9, precedido pelo cálculo do argumento $(arg2)\ \frac{L_B^2 + r'^2 - L_C^2}{2\ .\ L_B\ .r'}$.
- Cálculo das duas soluções de θ_2 ($theta2\_L$ e $theta2\_R$). A solução que perfila o cotovelo esquerdo dá-se pela equação 8, enquanto que a solução estabelecida pelo cotovelo direito é calculada através da seguinte fórmula.

$$\theta_2 = 90^o - \alpha_2 + \beta_2 \tag{14}$$

```
Real P02[3, 1] = [a1 * cos(theta1); a1 * sin(theta1); d1];
Real P25[3, 1] = P05 - P02;
Real P25_x = P25[1, 1];
Real P25_y = P25[2, 1];
Real P25_z = P25[3, 1];
Real r3 = sqrt(P25_x ^ 2 + P25_y ^ 2 + P25_z ^ 2);
Real LB = a2;
Real LC = d4;
Real arg3 = (LB ^ 2 + LC ^ 2 - r3 ^ 2) / (2 * LB * LC);
Real beta3 = acos(arg3);
Real theta3_L = beta3 - 3.141592653589793238 / 2;
Real theta3_R = 3 * (3.141592653589793238 / 2) - beta3;
Real theta3 = theta3_L;
Real theta3_deg = theta3 * 180 / 3.141592653589793238;
Real r = sqrt(P25_x ^ 2 + P25_y ^ 2);
Real alfa2 = atan2(P25_z, r);
Real arg2 = (LB ^ 2 + r3 ^ 2 - LC ^ 2) / (2 * LB * r3);
Real beta2 = acos(arg2);
Real theta2_L = -(3.141592653589793238 / 2 - alfa2 - beta2);
Real theta2_R = -(3.141592653589793238 / 2 - alfa2 + beta2);
Real theta2 = theta2_L;
Real theta2_deg = theta2 * 180 / 3.141592653589793238;
```

Figura 44 - Quarto *print* do código envolvido no modelo de cinemática inversa

8.1.1.2.5. Cálculo de θ_4 , θ_5 e θ_6

Através dos parâmetros de *Denavit – Hartenberg*, é possível calcular as 3 primeiras matrizes de transformação individual e a partir daí extrair a matriz de rotação correspondente ${}^{0}R_3$. Considerando ${}^{0}R_6$, a matriz de rotação RPY (que é conhecida), é possível resolver a equação para ${}^{3}R_6$.

$$ {}^{3}R_6 = {}^{3}R_0 \,.\, {}^{0}R_6 = inv\left({}^{0}R_3\right).\, {}^{0}R_6 \qquad (15) $$

Obtidos os valores finais da matriz ${}^{3}R_6$ e sabendo que os elementos da mesma poderão ser definidos da forma ilustrada na Figura 45, pode-se partir para o cálculo de θ_4, θ_5 e θ_6.

$$ {}^{3}\boldsymbol{R}_{\boldsymbol{6}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \cos(\theta_4)\cos(\theta_5)\cos(\theta_6) - \sin(\theta_4)\sin(\theta_6) & -\cos(\theta_6)\sin(\theta_4) - \cos(\theta_4)\cos(\theta_5)\sin(\theta_6) & -\cos(\theta_4)\sin(\theta_5) \\ \sin(\theta_5)\cos(\theta_6) & -\sin(\theta_5)\sin(\theta_6) & \cos(\theta_5) \\ -\cos(\theta_4)\sin(\theta_6) - \cos(\theta_5)\cos(\theta_6)\sin(\theta_4) & \cos(\theta_5)\sin(\theta_4)\sin(\theta_6) - \cos(\theta_4)\cos(\theta_6) & \sin(\theta_4)\sin(\theta_5) \end{bmatrix} $$

Figura 45 - Matriz de rotação ${}^{3}R_6$

Recorrendo ao elemento r_{23} e sabendo que $\sin(\theta) = \pm\sqrt{1 - \left(cos(\theta)\right)^2}$, a determinação de θ_5 torna-se possível, cuja solução dupla pode ser retirada da seguinte equação.

$$\theta_5 = arctan_2\left(\pm\sqrt{1-(r_{23})^2}, r_{23}\right) \quad (16)$$

Do mesmo modo, o uso das formulações retiradas da matriz (45) permite, com base nos elementos r_{33} e r_{13}, extrair o valor de θ_4 assim como, recorrendo a r_{22} e r_{21}, o valor de θ_6 .

$$\begin{cases}\theta_4 = arctan2\,(r_{33}, -r_{13}\,)\\ \theta_6 = arctan2\,(-r_{22}, r_{21}\,)\end{cases} \quad (17)$$

As deduções destes valores apresentam, no entanto, a condição de $\sin(\theta_5) \neq 0$ visando evitar singularidades. É nesse sentido que a seguinte definição condicional para ambos os ângulos θ_4 e θ_6 é ainda considerada.

$$\begin{cases}\theta_4 = 0 \ \wedge \theta_6 = arctan_2\,(r_{12}, r_{32}) & se\ \theta_5 = 0\\ \theta_4 = 0 \ \wedge \theta_6 = -\,arctan_2(-r_{12}, -r_{32}) & se\ \theta_5 = \pi\end{cases} \quad (18)$$

8.1.1.2.6. Cálculo de θ_4 , θ_5 e θ_6 – Implementação

- Declaração dos parâmetros de *Denavit – Hartenberg* a_i , α_i , d_i e θ_i para os diferentes referenciais ($DH\_alpha,\ DH\_a, DH\_theta, DH\_d$).
- Cálculo das 3 primeiras matrizes de transformação individual ${}^0T_1$, ${}^1T_2$ e ${}^2T_3$ ($T01$, $T12$ e $T23$).
- Cálculo da matriz de transformação ${}^0T_3$ ($T03$) e consequente extração da matriz de rotação ${}^0R_3$ ($R03$) envolvida nos 3 primeiros referenciais, procedida pela sua inversa ${}^3R_0$ ($R30$) .
- Cálculo da matriz ${}^3R_6$ ($R36$), necessária para o cálculo dos 3 ângulos. Com base na definição desta matriz, cada um dos elementos constituintes é posteriormente determinado (r_{11}, r_{12}, r_{13}, etc).

```
  Real DH_alpha[6, 1] = 3.141592653589793238 / 180 * [0; 90; 0; 90; -90; 90];
  Real DH_a[6, 1] = [0; a1; a2; 0; 0; 0];
  Real DH_theta[6, 1] = 3.141592653589793238 / 180 * [-90; 90; 0; 0; 0; 0];
  Real DH_d[6, 1] = [d1; 0; 0; d4; 0; d6];
  Real T01[4, 4] = [cos(DH_theta[1, 1] + theta1), -sin(DH_theta[1, 1] + theta1), 0, DH_a[1, 1]; cos(DH_alpha[1, 1]) *
sin(DH_theta[1, 1] + theta1), cos(DH_theta[1, 1] + theta1) * cos(DH_alpha[1, 1]), -sin(DH_alpha[1, 1]), -
sin(DH_alpha[1, 1]) * DH_d[1, 1]; sin(DH_alpha[1, 1]) * sin(DH_theta[1, 1] + theta1), sin(DH_alpha[1, 1]) *
cos(DH_theta[1, 1] + theta1), cos(DH_alpha[1, 1]), cos(DH_alpha[1, 1]) * DH_d[1, 1]; 0, 0, 0, 1];
  Real T12[4, 4] = [cos(DH_theta[2, 1] + theta2), -sin(DH_theta[2, 1] + theta2), 0, DH_a[2, 1]; cos(DH_alpha[2, 1]) *
sin(DH_theta[2, 1] + theta2), cos(DH_theta[2, 1] + theta2) * cos(DH_alpha[2, 1]), -sin(DH_alpha[2, 1]), -
sin(DH_alpha[2, 1]) * DH_d[2, 1]; sin(DH_alpha[2, 1]) * sin(DH_theta[2, 1] + theta2), sin(DH_alpha[2, 1]) *
cos(DH_theta[2, 1] + theta2), cos(DH_alpha[2, 1]), cos(DH_alpha[2, 1]) * DH_d[2, 1]; 0, 0, 0, 1];
  Real T23[4, 4] = [cos(DH_theta[3, 1] + theta3), -sin(DH_theta[3, 1] + theta3), 0, DH_a[3, 1]; cos(DH_alpha[3, 1]) *
sin(DH_theta[3, 1] + theta3), cos(DH_theta[3, 1] + theta3) * cos(DH_alpha[3, 1]), -sin(DH_alpha[3, 1]), -
sin(DH_alpha[3, 1]) * DH_d[3, 1]; sin(DH_alpha[3, 1]) * sin(DH_theta[3, 1] + theta3), sin(DH_alpha[3, 1]) *
cos(DH_theta[3, 1] + theta3), cos(DH_alpha[3, 1]), cos(DH_alpha[3, 1]) * DH_d[3, 1]; 0, 0, 0, 1];
  Real T03[4, 4] = T01 * T12 * T23;
  Real R03[3, 3] = T03[1:3, 1:3];
  Real R30[3, 3] = inv(R03);
  Real R36[3, 3] = R30 * R06;
  Real r11 = R36[1, 1];
  Real r12 = R36[1, 2];
  Real r13 = R36[1, 3];
  Real r21 = R36[2, 1];
  Real r22 = R36[2, 2];
  Real r23 = R36[2, 3];
  Real r31 = R36[3, 1];
  Real r32 = R36[3, 2];
  Real r33 = R36[3, 3];
```

Figura 46 - Quinto *print* do código envolvido no modelo de cinemática inversa

- Cálculo das duas soluções para θ_5, recorrendo aos elementos presentes na equação 16.

```
Real theta5_R = atan2(sqrt(1 - r23 ^ 2), r23);
Real theta5_L = atan2(-sqrt(1 - r23 ^ 2), r23);
Real theta5 = theta5_L;
Real theta5_deg = theta5 * 180 / 3.141592653589793238;
Real theta4;
Real theta6;
Real theta4_deg;
Real theta6_deg;
```

Figura 47 - Sexto *print* do código envolvido no modelo de cinemática inversa

- Cálculo das soluções genéricas para θ_4 e θ_6, assim como das soluções condicionais dependentes do valor de θ_5.
- Definição final do vetor $joints$ constituído pelos 6 ângulos.

```
if abs(theta5) < 0.01 then
  theta4 = 0;
  theta6 = atan2(r12, r32);
elseif abs(theta5 - pi) < 0.01 then
  theta4 = 0;
  theta6 = -atan2(-r12, -r32);
else
  theta4 = atan2(r33 / sin(theta5), -r13 / sin(theta5));
  theta6 = atan2(-r22 / sin(theta5), r21 / sin(theta5));
end if;
theta4_deg = theta4 * 180 / 3.141592653589793238;
theta6_deg = theta6 * 180 / 3.141592653589793238;
joints = {theta1, theta2, theta3, theta4, theta5, theta6};
```

Figura 48 - Sétimo *print* do código envolvido no modelo de cinemática inversa

8.1.2. Definição de Trajetória

Desenvolvido o código que dita a obtenção dos ângulos pretendidos, tem-se a entrada desses mesmos ângulos no modelo destinado ao cálculo de trajetória ou também denominado de "motion planning". Em sistemas de controlo robóticos, o "*motion planning*" é o problema computacional responsável por delinear o percurso a ser executado, livre de obstáculos, através da sequência de um conjunto de configurações válidas que movem o objeto da origem até ao destino perfilado. Para uma melhor explicação desta secção, começa-se pela primeira aproximação tida no desenvolvimento deste modelo, em que um *Pick&Place* sem modo *via point*, foi considerado. Numa fase posterior, porém, a melhoria com o modo *via point* é implementada, trazendo assim aspetos complementares ao modelo inicialmente concebido.

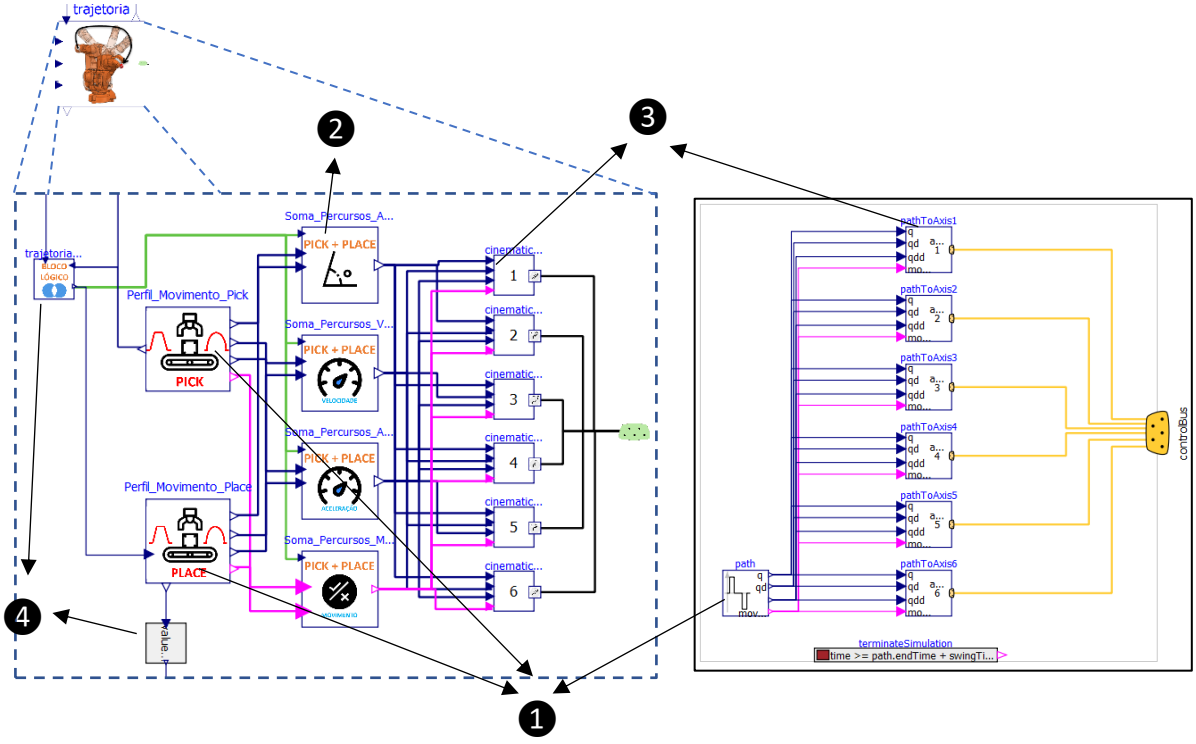


Figura 49 - À esquerda, o modelo elaborado envolvido na definição de trajetória. À direita, o modelo de referência utilizado

Na Figura 49 apresentada, é possível observar à esquerda o modelo inicialmente concebido do planeamento do trajeto e, analogamente, o modelo (.Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Utilities.PathPlanning6, s.d.) com o mesmo propósito inserido no modelo de referência presente no *Modelica Standard Library*. Destaca-se desde logo alguns aspetos introduzidos, como a existência (1) de 2 modelos de definição de trajetória, ou perfil de movimento, contrastando com 1 no modelo de referência, a existência de uma secção (2) que recolhe e soma ambos os perfis de

movimento ao longo do tempo, provenientes dos 2 modelos anteriores, definindo assim um perfil de movimento único, e apresentando ainda a especificidade de dividir o estado do mesmo em posição, velocidade, aceleração e ainda estado de movimento, através da elaboração de 4 modelos distintos. Uma terceira secção (3) presente em ambos os modelos, responsável por distribuir cada uma das informações provenientes do perfil de movimento ao longo dos 6 eixos e, por fim, sucedeu-se ainda a elaboração de pequenos modelos lógicos (4), que permitem associar o modelo com um processo precedente e posterior, em que a atualização de estados ao longo do tempo possibilita uma execução dos restantes blocos no modelo, apenas para um intervalo de tempo específico.

Devido à quantidade de cálculos exigida para este processo, o modelo global de definição de trajetória assume-se como denso a nível de modelos internos. De maneira a esclarecer a função de cada um com maior clareza, a divisão já referida será acompanhada por uma explicação mais clara das secções.

8.1.2.1. Modelos para o Perfil Cinemático

Através das entradas, que caracterizam o ângulo inicial e final da junta, assim como restrições cinemáticas, nomeadamente velocidade e aceleração angular máxima, passível de ser atingida pelo eixo, o algoritmo de trajetória processa as informações respetivas e define as variáveis de saída. Estas correspondem aos valores instantâneos de posição, velocidade e aceleração, existindo ainda uma variável booleana de saída que toma o valor positivo quando existe movimento angular (evolução do algoritmo a partir das entradas).

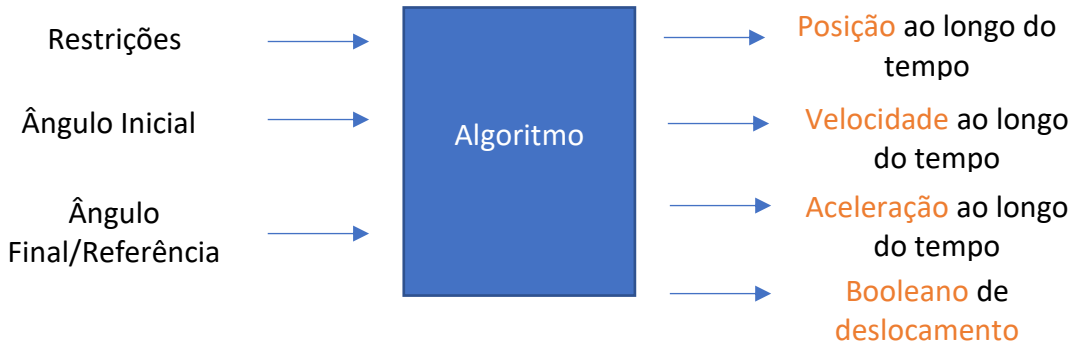


Figura 50 - Entradas e Saídas do modelo destinado ao perfil de movimento

O objetivo é o movimento mais rápido possível da posição inicial até à posição final sob determinadas constrições cinemáticas. Este bloco gera posição, velocidade e aceleração como valores de saída. Os sinais estão construídos de maneira a que não seja possível um movimento mais rápido, devido à máxima velocidade e aceleração permitidas que lhe são atribuídas.

O movimento cinemático inicalmente definido (trapezoidal), apresenta um movimento inicial de aceleração, procedido de velocidade constante e numa fase final é caracterizado por uma desaceleração, estabelecida quando o ponto final está próximo de ser atingido. Desta forma, os valores de referência a serem atingidos pelo manipulador serão continuamente fornecidos aos eixos, de forma a serem posteriormente processados e desencadeiem, assim, o movimento desejado.

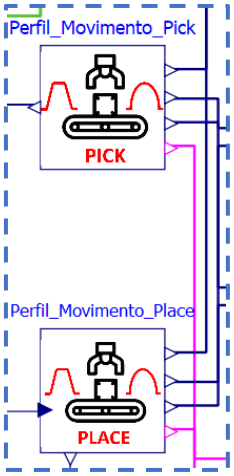


Figura 51 - Modelo de Geração do Perfil de Movimento do Pick (em cima) e do Place (em baixo)

Antes de uma perceção mais clara relativa ao algoritmo, as entradas e saídas ao longo da secção de geração do perfil cinemático serão explicadas. Para cada modelo de geração, são definidas as características cinemáticas assim como o instante de tempo em que o algoritmo e o cálculo do respetivo perfil é iniciado. Para o caso do modelo da geração de perfil *Pick* os valores angulares iniciais das juntas são os valores considerados para a posição inicial do robot que é independente do tipo de percurso de transporte a ser executado, enquanto que os valores angulares finais serão calculados pelo modelo de cinemática inversa. As constrições cinemáticas, relativas à velocidade e aceleração, são parâmetros estabelecidos pelo utilizador, ou seja, são valores fixos pré-definidos. O instante de tempo em que o cálculo do perfil se

inicia, é uma variável dependente do tempo de transporte da embalagem executado pelo tapete transportador à entrada do sistema global.

No modelo de geração do perfil *Place,* os valores angulares iniciais correspondem aos valores angulares finais do perfil *Pick,* e os valores angulares finais admitem os valores calculados pelo modelo de cinemática inversa correspondente. O instante de tempo em que o cálculo do perfil se inicia, depende do tempo de operação do *gripper* em agarrar a embalagem presente no tapete transportador. Desta forma, o instante de tempo corresponde ao instante em que o movimento *Pick* é finalizado mais o tempo de operação do *gripper*.

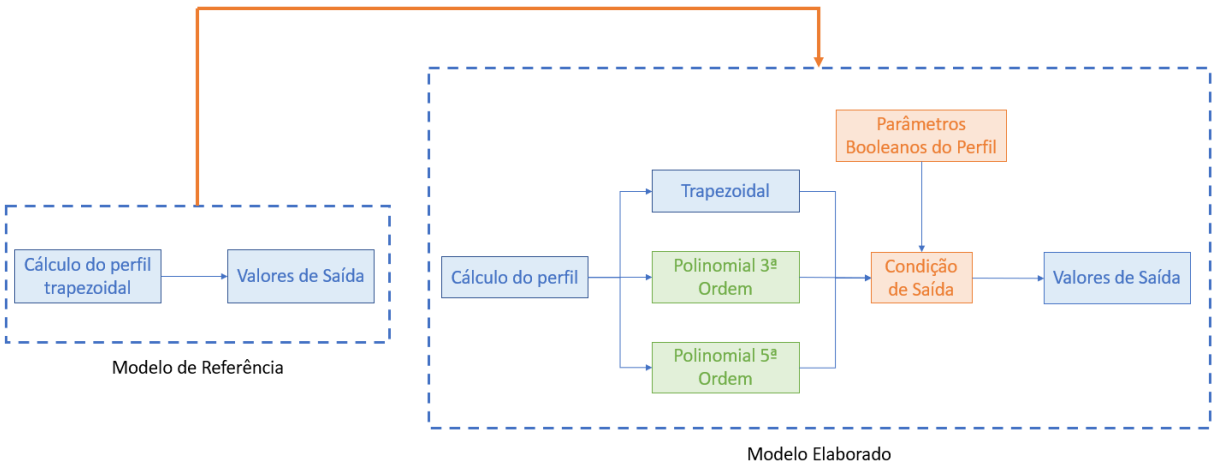


Figura 52 - Estrutura do código envolvido na geração do perfil do movimento

Na Figura 52 a estrutura do código destinado à geração do perfil de movimento presente nos modelos, é apresentada. Esta estrutura foi alargada de modo a admitir perfis de movimento adicionais, caso estes sejam selecionados. Além do perfil de velocidade trapezoidal, concebido já no modelo de referência presente na biblioteca (.Modelica.Blocks.Sources.KinematicPTP2, s.d.), o cálculo dos perfis polinomiais de 3ª e 5ª ordem foram introduzidos, procurando tornar a configuração da trajetória mais completa, de acordo com as pretensões do utilizador. Os diferentes tipos de perfis são calculados simultaneamente no modelo, sendo que apenas um, posteriormente, admite os valores finais a serem transmitidos.

8.1.2.2. Perfil Cinemático Trapezoidal

8.1.2.2.1. Perfil Trapezoidal - Teoria

Para a geração do algoritmo envolvido na trajetória cinemática do braço, foi necessário definir primeiro o tipo de perfil cinemático a ser adotado. Tendo como auxílio, o exemplo de braço robótico disponibilizado pelo *OpenModelica,* o perfil cinemático definido, inicialmente, foi o perfil de movimento trapezoidal, sendo que o seu estudo será levado a cabo (Yoon, Chung, Kang, & Hwang, 2019), antes da implementação subsequente de outros perfis. O perfil de velocidade trapezoidal confere desde logo, em comparação com uma trajetória polinomial de 3ª ordem, uma menor duração de movimento para uma velocidade máxima e deslocamento angular igual.

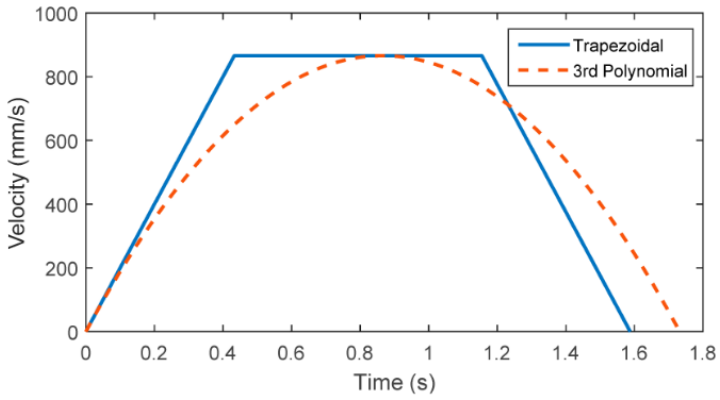


Figura 53 - Movimento trapezoidal e movimento polinomial de 3ª ordem, para a mesma velocidade máxima

Um simples exemplo deste perfil de velocidade para um deslocamento *point-to-point* poderá ser observado na Figura 54 onde se pode observar uma aceleração e desaceleração suave entre o ponto inicial e o ponto de referência a ser alcançado, sendo que uma velocidade constante é admitida entre a aceleração e desaceleração.

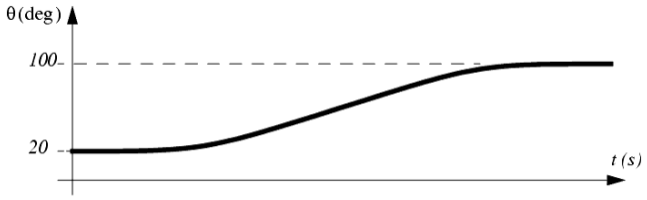


Figura 54 - Variação da posição ao longo de um perfil de velocidade trapezoidal

O deslocamento angular, entre dois pontos, pode ser especificado por um número de parâmetros, que de forma conjunta definem o perfil de movimento. Para o perfil trapezoidal simples, estes parâmetros são a distância, velocidade máxima, aceleração máxima e desaceleração. O perfil trapezoidal encontra-se dividido em 3 fases: aceleração, velocidade constante e desaceleração. Se o deslocamento for positivo, a aceleração é positiva e constante

na primeira fase. A velocidade é, ainda nesta fase, uma função linear no tempo e a posição é uma curva parabólica no tempo. Na secção 2 do perfil, a aceleração é zero e a velocidade apresenta um valor constante. Dessa forma, a posição é uma função linear no tempo nesta secção. Por fim, uma constante negativa de aceleração é imposta na 3ª secção e a velocidade é linearmente reduzida. A Figura 55 apresenta a posição, velocidade e aceleração de um perfil de movimento trapezoidal, em que os 3 gráficos são ilustrados.

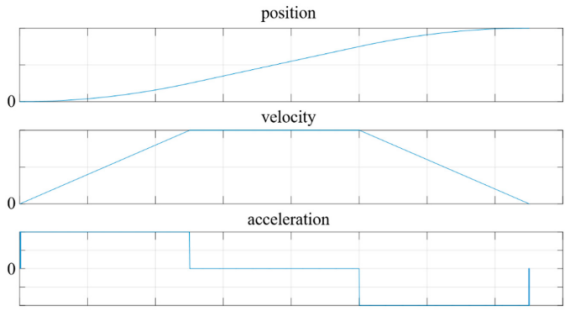


Figura 55 - Posição, velocidade e aceleração de um perfil de movimento trapezoidal

Para conceber um perfil de deslocamento trapezoidal, os parâmetros cinemáticos devem ser pré-definidos e o deslocamento dado. Se a velocidade e aceleração máxima são dadas, a duração da aceleração e deslocamento podem ser derivadas. Normalmente, o operador pode atribuir os valores máximos de velocidade e aceleração para definir os parâmetros de movimento de um robot industrial. É assumido que as velocidades nos pontos iniciais e finais são nulas, como se pode observar na Figura 56 e que os tempos de aceleração e desaceleração são idênticos.

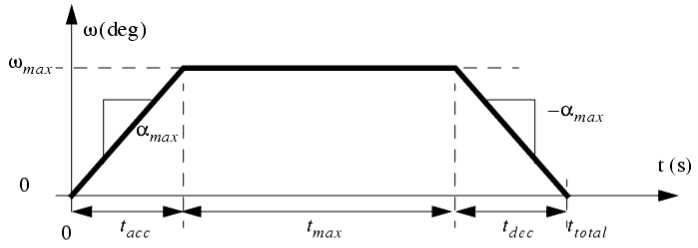


Figura 56 - Perfil de movimento trapezoidal, com a descrição dos tempos de aceleração, velocidade constante e desaceleração

Se a velocidade máxima for $v_{máx}$ e a aceleração máxima $a_{máx}$, o tempo de aceleração t_a pode ser derivado da seguinte forma.

$$t_a = \frac{v_{máx}}{a_{máx}} \tag{19}$$

O deslocamento é definido como Δx e é assumido como positivo. O tempo requerido para alcançar o alvo, através do deslocamento Δx, é t_f e pode ser computado com a seguinte fórmula

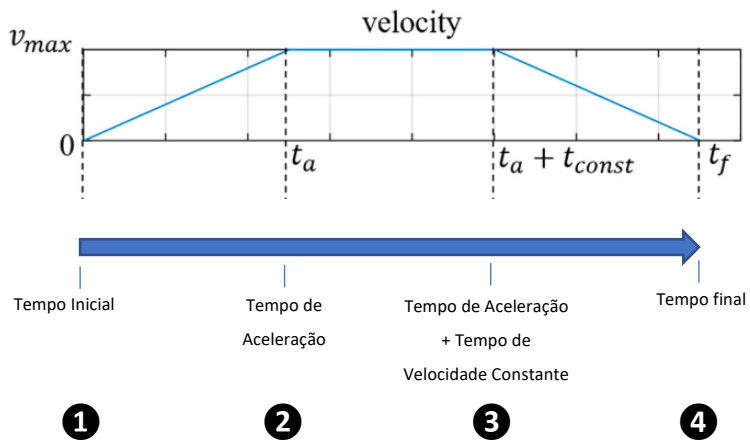
$$t_f = \frac{\Delta x}{v_{máx}} + t_a \tag{20}$$

Para uma simetria estabelecida entre a aceleração e desaceleração, a duração de tempo em que o eixo adota uma velocidade constante é calculado do seguinte modo.

$$t_{const} = t_f - 2t_a \tag{21}$$

No sentido de definir as diferentes posições verificadas ao longo do tempo e que terão especial importância no código a ser desenvolvido posteriormente, o perfil de velocidade trapezoidal apresenta a sua relevância, uma vez que as posições sujeitas a cálculo podem se definir diretamente através de equações que definem a área do perfil em diferentes instantes no tempo. Tendo por base este princípio, a seguinte tabela 5 foi realizada. Nesta, o cálculo das posições absolutas é apresentado, para os diferentes tempos de referência exibidos na imagem da tabela assim como ao longo dos instantes presentes entre os instantes de referência, e cujo cálculo se declara através de funções.

Tabela 5 - Cálculo das posições ao longo do perfil de velocidade trapezoidal



Posição	Valor/Formula
Posição 1 (Inicial)	$\boldsymbol{p_1 = 0}$

Função de Posição (1 → 2)	$$p(t) = \frac{1}{2} * t * v_{máx}$$ **ou** $$p(t) = \frac{1}{2} * t^2 * a_{max}$$
Posição 2 (posição em t_a)	$$p_2 = \frac{1}{2} * t_a * v_{máx}$$ **ou** $$p_2 = \frac{1}{2} * {t_a}^2 * a_{max}$$
Função de Posição (2 → 3)	$$p(t) = p_2 + v_{máx} * (t - t_a)$$
Posição 3 (posição em $t_a + t_{const}$)	$$p_3 = p_2 + v_{máx} * t_{const}$$
Função de Posição (3 → 4)	$$p(t) = p_3 + \frac{1}{2} * (t - t_a - t_{const}) * v_{máx}$$ **ou** $$p(t) = p_3 + \frac{1}{2} * (t - t_a - t_{const})^2 * a_{max}$$
Posição 4 (posição final t_f)	$$p_4 = p_3 + \frac{1}{2} * t_d * v_{máx}$$ $$p_4 = p_3 + \frac{1}{2} * {t_d}^2 * a_{max}$$ **Sendo que** $t_d = t_a$

As formas de cálculo apresentadas na tabela 5 apresentam a sua significância ao longo do desenvolvimento do código, no entanto os valores finais passíveis de obtenção através dos

cálculos podem ser desconsiderados, uma vez que as fórmulas de cálculo se devem auxiliar por uma escala de tempo, sendo que a mesma passará a ser explicada.

A configuração dos 6 eixos de um manipulador como uma função do tempo é denominada de trajetória.

Trajetória: $\theta(t), t \in [0, T]$ **Percurso**: $\theta(s), s \in [0,1]$

Figura 57 - Trajetória e caminho ao longo de um sistema com 2 eixos

Em que T se assume como o valor do tempo total de deslocamento. No entanto o interesse do desenvolvimento da configuração nem sempre se encontra inicialmente dependente do tempo, mas sim no espaço configurado. Ou seja, a curva observada na Figura 57 em que se pode observar a evolução dos ângulos de 2 eixos, ao invés de evoluir de acordo com o tempo t, evolui de acordo com uma função cujo parâmetro s varia entre 0 (configuração inicial) e 1(configuração final), definindo assim o percurso.

Um percurso poderá ser transformado numa trajetória, por sua vez, definindo s como uma função ao longo do tempo. Deste modo, a trajetória poderá ser definida:

$$\theta\big(s(t)\big), s : [0, T] \rightarrow [0,1]$$

A definição de um percurso, através da implementação de uma trajetória é assim denominado de escala do tempo (*time scaling*), em que se estabelece a rapidez com que o percurso é percorrido. Deste modo, a implementação final requerida implicará o uso das fórmulas apresentadas na tabela em conjunto com o deslocamento angular delineado entre os pontos do percurso, fazendo com que os valores finais de posição variem entre 0 e 1, decorrentes de um posicionamento relativo e não absoluto.

Na equação 21, t_{const} é positivo quando t_f é maior que $2t_a$. Isto significa que t_{const} pode ser zero se $t_f$for equivalente a $2t_a$. Isto acontece quando o deslocamento Δx é relativamente

pequeno. A zona de velocidade constante, no perfil, poderá não existir, dando assim lugar a um perfil de velocidade triangular em que apenas as zonas de aceleração e desaceleração aparecem, como perfilado na Figura 58.

Para um algoritmo de cinemática que ofereça uma maior consistência, este perfil de velocidade é também definido, na eventualidade de o deslocamento angular de um determinado eixo ser reduzido.

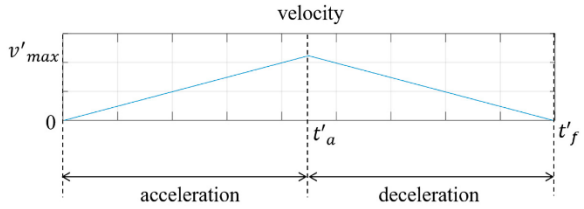


Figura 58 - Perfil de velocidade triangular

Para este perfil, a velocidade máxima não é alcançada quando a aceleração máxima $a_{máx}$, é imposta. É nesse sentido que as variáveis t_a, t_f e $v_{máx}$ necessitam de ser recalculadas, dando lugar a t'_a, t'_f e $v'_{máx}$.

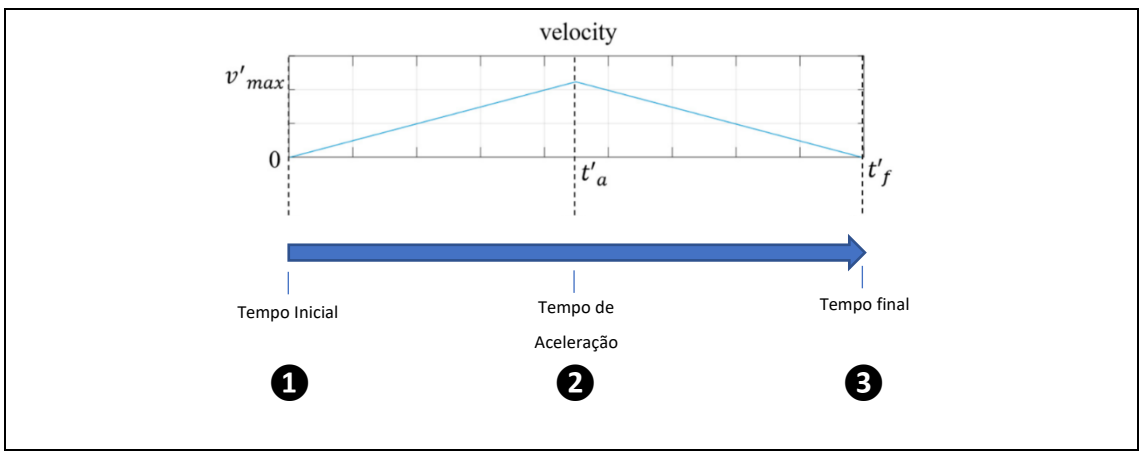
$$t'_a = \sqrt{\frac{\Delta x}{a_{máx}}} \tag{22}$$

$$v'_{máx} = a_{máx} * t'_a \tag{23}$$

$$t'_f = 2t'_a \tag{24}$$

As posições e funções de posição são novamente calculadas, desta vez para o perfil triangular.

Tabela 6 - Cálculo das posições ao longo do perfil de velocidade triangular



Posição	Valor/Formula
Posição 1 (Inicial)	$p_1 = 0$
Função de Posição (1 → 2)	$p(t) = \frac{1}{2} * t * v'_{máx}$ $p(t) = \frac{1}{2} * t^2 * a_{max}$
Posição 2 (posição em t'_a)	$p_2 = \frac{1}{2} * t'_a * v'_{máx}$ $p_2 = \frac{1}{2} * t'^2_a * a_{max}$
Função de Posição (2 → 3)	$p(t) = p_2 + \frac{1}{2} * (t - t'_a) * v'_{máx}$ $p(t) = p_2 + \frac{1}{2} * (t - t'_a)^2 * a_{max}$
Posição 3 (posição em $t'_a + t'_d$)	$p_3 = p_1 + v_{máx} * t_{const}$

8.1.2.2.2. Perfil Trapezoidal – Implementação

Estudados os pressupostos teóricos associados à geração dos perfis cinemáticos trapezoidais, passa-se agora a explicar a sua implementação prática no modelo. No fundo, o *script* relativo ao perfil de movimento trapezoidal é o mesmo presente no modelo de trajetória já inserido no modelo global de referência providenciado. Há no entanto, uma carência total de informação relativa à explicação do código, pelo que ao longo deste capítulo a mesma será apresentada. A explicação será elaborada através da apresentação, por ordem, de diferentes excertos ou secções do *script*.

```
equation
  for i in 1:nout loop
    aux1[i] = p_deltaq[i] / p_qd_max[i];
    aux2[i] = p_deltaq[i] / p_qdd_max[i];
  end for;
  sd_max_inv = max(abs(aux1));
  sdd_max_inv = max(abs(aux2));
  if startTime == 0 then
    sd_max = 0;
    sdd_max = 0;
    Ta1 = 0;
    Ta2 = 0;
    noWphase = false;
    Tv = 0;
    Te = 0;
    Ta1s = 0;
    Ta2s = 0;
    Tvs = 0;
    Tes = 0;
    sd_max2 = 0;
    s1 = 0;
    s2 = 0;
    s3 = 0;
    s = 0;
```

Figura 59 - Primeiro *print* do código envolvido no modelo de geração do perfil cinemático

Numa fase inicial, são calculados 2 vetores para cada elemento de cada eixo, um correspondente à razão entre a diferença angular e a velocidade máxima (consistindo assim no tempo de deslocamento se a velocidade máxima abrangesse todo o tempo de deslocamento) e outro relativo à razão entre a diferença angular e a aceleração. Destes 2 vetores, apenas o elemento com o valor mais elevado será extraído posteriormente. Tendo em consideração o deslocamento requerido e as restrições cinemáticas para cada junta, é fácil deduzir que os tempos de deslocamento envolvidos variam para cada eixo, em conformidade com as restrições correspondentes. No entanto, é desejado um movimento suave garantindo um tempo de percurso igual para todas as juntas. A seleção do valor mais elevado em cada um dos 2 vetores referidos, garante não só um tempo igual de percurso para todas as juntas, como garante que esse mesmo tempo seja suficiente para a consumação do perfil de movimento trapezoidal em todos os eixos.

Os restantes parâmetros presentes no excerto serão descritos na seguinte tabela 7. Cada um destes parâmetros apresenta o valor 0 caso a condição de *startTime* = 0 seja verificada, ou seja, caso o instante em que se quer que o comando de trajetória seja calculado ainda não tenha sido atingido.

Tabela 7 - Descrição dos parâmetros presentes no excerto do modelo

Variável	Significado

sd_max	Inverso da razão entre a amplitude angular e a velocidade máxima, sendo o valor desta razão o valor máximo adquirido ao longo dos 6 eixos.
sdd_max	Inverso da razão entre a amplitude angular e a aceleração máxima, sendo o valor desta razão o valor máximo adquirido ao longo dos 6 eixos.
Ta1	Tempo de aceleração caso a velocidade máxima não seja atingida, ou seja, perante um perfil de velocidade triangular.
Ta2	Tempo de aceleração com velocidade máxima a ser atingida, ou seja, com perfil trapezoidal de velocidade e não triangular.
noWphase	Booleano que quando verdadeiro retrata a existência de um perfil de velocidade triangular, ou seja, sem fase constante.
Tv	No perfil trapezoidal diz respeito ao tempo ao longo da secção com velocidade constante. No perfil triangular iguala-se ao tempo de aceleração.
Te	Tempo total de deslocamento.
Ta1s	Instante de tempo em que o tempo de aceleração (caso a velocidade máxima não seja atingida) finaliza, ou seja, é equivalente ao tempo de aceleração + *startTime.* Perfil triangular.
Ta2s	Instante de tempo em que o tempo de aceleração (caso a velocidade máxima seja atingida) finaliza, ou seja, é equivalente ao tempo de aceleração + *startTime.* Perfil trapezoidal.
Tvs	Instante de tempo em que a velocidade constante termina (perfil trapezoidal) ou instante de tempo em que a fase de aceleração termina (perfil triangular, igualando-se assim a ***Ta1s***).
Tes	Instante de tempo em que o tempo de deslocamento finaliza, é equivalente ao tempo de deslocamento + *startTime*
sd_max2	Velocidade máxima atingida quando se está perante um perfil triangular de velocidade.
s1	Percentagem de deslocamento ocorrido no tempo de aceleração.

s2	Percentagem de deslocamento ocorrido no tempo de em que a velocidade constante é finalizada.
s3	Percentagem de deslocamento ocorrido no tempo final, que no fundo adota a totalidade do deslocamento, equivalendo assim ao valor fixo de 1.
s	Percentagem de deslocamento ocorrido ao longo do tempo e que adota de forma gradual ao longo do tempo os valores de *s1, s2* e *s3* Varia assim entre 0 e 1.

Na eventualidade de o instante relativo ao cálculo ser atingido, está presente no seguinte excerto o cálculo desenvolvido para os parâmetros previamente descritos.

```
s1 = sdd_max * (if noWphase then Ta1 * Ta1 else Ta2 * Ta2) / 2;
s2 = s1 + (if noWphase then sd_max2 * (Te - Ta1) - sdd_max / 2 * (Te - Ta1) ^ 2 else sd_max * (Tv - Ta2));
s3 = s2 + sd_max * (Te - Tv) - sdd_max / 2 * (Te - Tv) * (Te - Tv);
if time < startTime then
  s = 0;
elseif noWphase then
  if time < Ta1s then
    s = sdd_max / 2 * (time - startTime) * (time - startTime);
  elseif time < Tes then
    s = s1 + sd_max2 * (time - Ta1s) - sdd_max / 2 * (time - Ta1s) * (time - Ta1s);
  else
    s = s2;
  end if;
elseif time < Ta2s then
  s = sdd_max / 2 * (time - startTime) * (time - startTime);
elseif time < Tvs then
  s = s1 + sd_max * (time - Ta2s);
elseif time < Tes then
  s = s2 + sd_max * (time - Tvs) - sdd_max / 2 * (time - Tvs) * (time - Tvs);
else
  s = s3;
end if;
```

Figura 60 - Segundo *print* do código envolvido no modelo de geração do perfil cinemático

Para uma melhor perceção dos cálculos replicados no excerto em cima, os mesmos encontram-se presentes na seguinte tabela.

Tabela 8 - Cálculo dos parâmetros presentes no excerto do modelo

Variável	Cálculo
sd_max	$1 \Big/ \left(\frac{\Delta x}{v_{máx}}\right)_{máx}$
sdd_max	$1 \Big/ \left(\frac{\Delta x}{a_{máx}}\right)_{máx}$
Ta1	$\sqrt{\left(\frac{\Delta x}{a_{máx}}\right)_{máx}}$

Ta2	$\left(\frac{v_{máx}}{a_{máx}}\right)_{máx}$
noWphase	Verdadeiro se $\left(\frac{v_{máx}}{a_{máx}}\right)_{máx} > \sqrt{\frac{\Delta x}{a_{máx}}}$
Tv	Se ***noWphase*** então ***Tv = Ta1***, de outra forma ***Tv =*** $\left(\frac{\Delta x}{v_{máx}}\right)_{máx}$
Te	Se ***noWphase*** então ***Te*** = 2***Ta1***, de outra forma ***Te*** = ***Tv*** + ***Ta2***
Ta1s ***Ta2s*** ***Tvs*** ***Tes***	***Ta1*** ***Ta2*** ***Tv*** ***Te*** ***+ startTime***
sd_max2	$\sqrt{\left(\frac{\Delta x}{a_{máx}}\right)_{máx}} \Big/ \left(\frac{\Delta x}{a_{máx}}\right)_{máx}$
s1	**Considerações** • ***s1 =*** $\frac{\textbf{\textit{Distância Percorrida ao londo da aceleração}}}{\textbf{\textit{Distância Total}}}$ • Caso se tenha em consideração o perfil trapezoidal de velocidades, tem-se que a distância percorrida é equivalente à área da metade esquerda triangular, pelo que $d = \frac{1}{2} . t_a . v_{máx}$ • Sabe-se também que $v_{máx} = a_{máx} . t_a$, pelo que $d = \frac{1}{2} . (t_a)^2 . a_{máx}$ Assim, tem-se: Se ***noWphase*** então, $\frac{\frac{1}{2} . (t_a)^2 . (a_{máx})_{máx}}{(\Delta x_{máx})_{máx}} = \frac{1}{2} . (\boldsymbol{Ta1})^2 . \boldsymbol{sdd\_max}$ De outra forma, $\frac{\frac{1}{2} . (t_a)^2 . (a_{máx})_{máx}}{(\Delta x_{máx})_{máx}} = \frac{1}{2} . (\boldsymbol{Ta2})^2 . \boldsymbol{sdd\_max}$

s2	• Caso se tenha em consideração o perfil trapezoidal de velocidade, tem-se que a distância percorrida é equivalente à área retangular no centro da secção trapezoidal, pelo que $d = t_{vel.const} . v_{máx}$ Se ***noWphase*** então, $$\frac{\sqrt{\left(\frac{\Delta x}{a_{máx}}\right)_{máx}}}{\left(\frac{\Delta x}{a_{máx}}\right)_{máx}} - \frac{\frac{1}{2}(a_{máx})_{máx}\,(\boldsymbol{Te}-\boldsymbol{Ta1})^2}{(\Delta x)_{máx}}$$ De outra forma, $$\boldsymbol{s1} + \frac{(\boldsymbol{Tv}-\boldsymbol{Ta2})(v_{máx})_{máx}}{(\Delta x)_{máx}}$$
s3	Repare-se que para o cálculo de s3, este corresponde a uma secção do perfil apenas existente para o perfil trapezoidal, e que define a terceira secção do perfil. Aqui tem-se: $$\boldsymbol{s2} + \left(\frac{v_{máx}.(t_d) - \frac{1}{2}.(t_d)^2.(a_{máx})_{máx}}{(\Delta x)_{máx}}\right)$$ $$\boldsymbol{s2} + \left(\left(\frac{v_{máx}}{\Delta x}\right)_{máx}.(\boldsymbol{Te} - \boldsymbol{Tv}) - \frac{\frac{1}{2}.(\boldsymbol{Te}-\boldsymbol{Tv})^2.(a_{máx})_{máx}}{(\Delta x_{máx})_{máx}}\right)$$ $$\boldsymbol{s2} + \left(\boldsymbol{sd_{max}}.(\boldsymbol{Te} - \boldsymbol{Tv}) - \frac{1}{2}.(\boldsymbol{Te} - \boldsymbol{Tv})^2.\boldsymbol{sdd\_max}\right)$$
s	Recorrendo às expressões das funções apresentadas na tabela 5 e 6 e dividindo-as pela extensão de variação angular fica-se com: **Perfil trapezoidal** Se $\boldsymbol{t}(time) < \boldsymbol{Ta2s}$: $$\frac{\frac{1}{2} * \boldsymbol{t}^2 * \boldsymbol{a_{max}}}{(\Delta x)_{máx}} = \frac{\boldsymbol{sdd\_max}}{\boldsymbol{2}} * (time - startTime)^2$$ Se $\boldsymbol{t}(time) > \boldsymbol{Ta2s}$ e $\boldsymbol{t}(time) < \boldsymbol{Tvs}$: $$\frac{\boldsymbol{p_2} + \boldsymbol{v_{máx}} * (\boldsymbol{t} - \boldsymbol{t_a})}{(\Delta x)_{máx}} = \boldsymbol{s1} + \boldsymbol{sd\_max}\,(time - \boldsymbol{Ta2s})$$

Se $\boldsymbol{t}(time) > \boldsymbol{Tvs}$ e $\boldsymbol{t}(time) < \boldsymbol{Tes}$:

$$\frac{\boldsymbol{p_3} + \frac{\mathbf{1}}{\mathbf{2}} * (\boldsymbol{t} - \boldsymbol{t_a} - \boldsymbol{t_{const}})^{\mathbf{2}} * \boldsymbol{a_{max}}}{(\Delta x)_{máx}}$$

$$= \frac{\boldsymbol{p_3} + \boldsymbol{v_{máx}} * (\boldsymbol{t} - \boldsymbol{t_a} - \boldsymbol{t_{const}}) - \frac{\mathbf{1}}{\mathbf{2}} * (\boldsymbol{t} - \boldsymbol{t_a} - \boldsymbol{t_{const}})^{\mathbf{2}} * \boldsymbol{a_{max}}}{(\Delta x)_{máx}}$$

$$= \boldsymbol{s2} + \boldsymbol{sd\_max}(time - \boldsymbol{Tvs}) - \frac{\boldsymbol{sdd\_max}}{2.(time - \boldsymbol{Tvs})^2}$$

Perfil triangular

Se $\boldsymbol{t}(time) < \boldsymbol{Ta1s}$:

$$\frac{\frac{1}{2} * \boldsymbol{t}^{\mathbf{2}} * \boldsymbol{a_{max}}}{(\Delta x)_{máx}} = \frac{\boldsymbol{sdd\_max}}{\mathbf{2}} * (time - startTime)^2$$

Se $\boldsymbol{t}(time) > \boldsymbol{Ta1s}$ e $\boldsymbol{t}(time) < \boldsymbol{Tes}$:

$$\frac{\boldsymbol{p_2} + \frac{\mathbf{1}}{\mathbf{2}} * (\boldsymbol{t} - \boldsymbol{t_a'})^{\mathbf{2}} * \boldsymbol{a_{max}}}{(\Delta x)_{máx}}$$

$$= \frac{\boldsymbol{p_2} + \boldsymbol{v'_{máx}} * (\boldsymbol{t} - \boldsymbol{t_a'}) - \frac{\mathbf{1}}{\mathbf{2}} * (\boldsymbol{t} - \boldsymbol{t_a'})^{\mathbf{2}} * \boldsymbol{a_{max}}}{(\Delta x)_{máx}}$$

$$= \boldsymbol{s1} + \boldsymbol{sd\_max2}(time - \boldsymbol{Ta1s}) - \frac{\boldsymbol{sdd\_max}}{2.(time - \boldsymbol{Ta1s})^2}$$

Formuladas cada uma das posições de referência (*s1*, *s2* e *s3*) do espaço configurado (entre 0 e 1), as mesmas são implementadas no modelo.

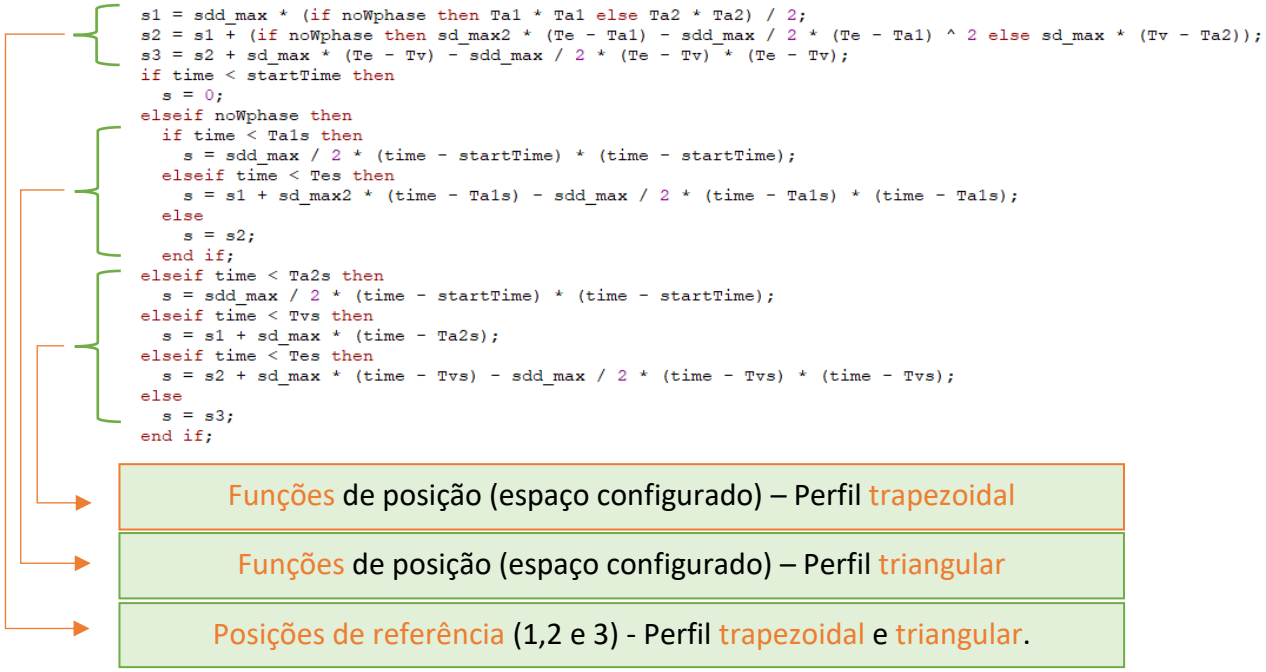


Figura 61 - Terceiro *print* do código envolvido no modelo de geração do perfil cinemático

Calculadas as posições relativas *s* segundo o espaço configurado (variação entre 0 e 1). Converte-se agora o percurso em trajetória (definição de *s* como uma função ao longo do tempo), multiplicando os valores obtidos pela variação angular em cada um dos eixos.

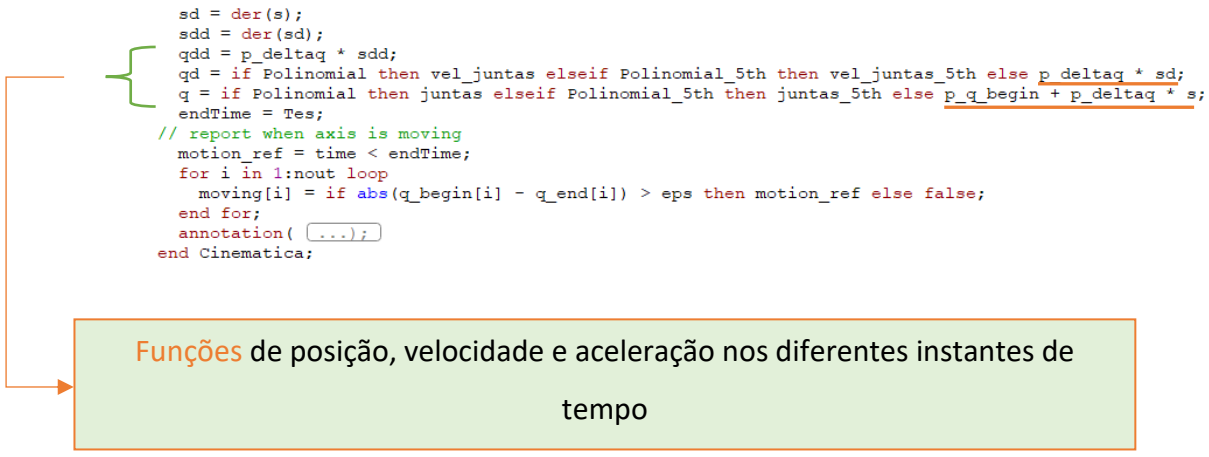


Figura 62 - Quarto *print* do código envolvido no modelo de geração do perfil cinemático

8.1.2.3. Perfil Cinemático Polinomial 3ª Ordem

8.1.2.3.1. Perfil Polinomial 3ª Ordem - Teoria

De forma a poder explorar diferentes perfis cinemáticos, foi elaborado, ao longo dos modelos cinemáticos de *Pick* e *Place*, um perfil polinomial de movimento. As equações de cálculo do

polinómio foram inseridas nos modelos cinemáticos e dessa forma, permitir comparar o perfil trapezoidal com um polinomial de 3ª ordem. É dada ainda, como parâmetros de entrada no sistema global, a opção ao utilizador de escolher qual dos dois perfis cinemáticos deseja para a delineação das trajetórias angulares ao longo dos eixos do robot.

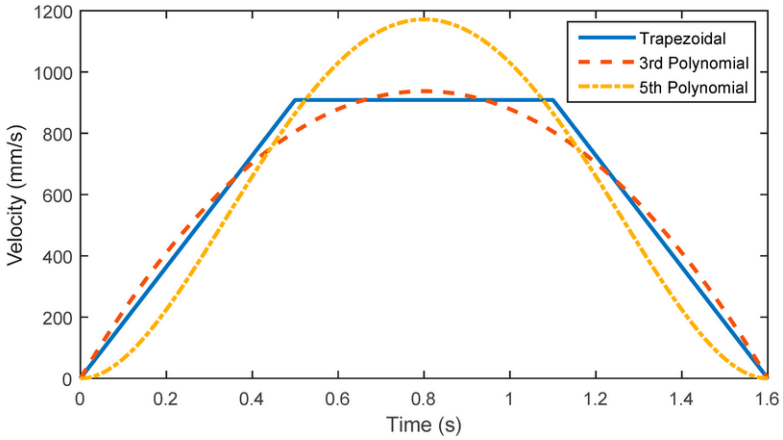


Figura 63 - Perfil de velocidade trapezoidal, polinomial de 3º ordem e polinomial de 5º ordem, para o mesmo intervalo de tempo

Na Figura 63, comparando o perfil trapezoidal com o polinomial, facilmente se percebe que o uso de um perfil polinomial projeta um movimento mais suave, sem mudanças radicais no declive, no entanto essa suavidade de movimento é contrastada com valores maiores de velocidade e aceleração máxima.

O estudo da trajetória polinomial que permite ao manipulador evoluir de uma posição para outra será, ao longo desta secção, explorada, primeiro em termos matemáticos e logo depois com a respetiva implementação em *Modelica*.

A dedução matemática dos coeficientes de uma equação polinomial cúbica será brevemente abordada. A escolha de uma função polinomial cúbica foi admitida pelas seguintes motivações:

- Uma função polinomial cúbica é suficiente para a quantidade de restrições a serem usadas(4);
- Uma função polinomial assegura um movimento suave, pois é uma função contínua e possui a primeira derivada também contínua;
- Caso o primeiro e último ponto estejam dentro do espaço das juntas, então os pontos intermédios também estarão.

As deduções expressas foram projetadas para o caso em que a velocidade inicial e final são nulas, no entanto as mesmas são também válidas caso se optasse pela utilização de *via points*, situação esta em que por norma são projetados pontos específicos de trajetória em que o manipulador passará com velocidade não nula.

Na equação 25 encontra-se presente a expressão que permite descrever a evolução do ângulo de cada junta ao longo do tempo. Nesta expressão pretende-se determinar a_0, a_1, a_2 e a_3.

$$\theta_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{25}$$

Derivando-se a equação 25 obtém-se a equação 26, que permite obter a velocidade angular em cada junta ao longo do tempo.

$$\dot{\theta}_i(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{26}$$

Restrições

As restrições (Erlhagen, Trajectory Generation in robotic manipulators, 2020) para a trajetória que se pretende realizar encontram-se nas equações 27-30 . Estas são o ângulo inicial ($\theta_{i,0}$) e final ($\theta_{i,f}$) da junta e a velocidade inicial ($\dot{\theta}_{i,0}$) e final ($\dot{\theta}_{i,f}$) da mesma. Nas equações 28 e 30, T é o tempo de duração do movimento, que deve ser igual para todas as juntas.

$$\theta_i(0) = \theta_{i,0} \tag{27}$$

$$\theta_i(T) = \theta_{i,f} \tag{28}$$

$$\dot{\theta}_i(0) = \dot{\theta}_{i,0} \tag{29}$$

$$\dot{\theta}_i(T) = \dot{\theta}_{i,f} \tag{30}$$

Determinação de a_0, a_1, a_2 e a_3

A partir das restrições presentes nas equações 25-26 e das equações 27-30 é possível escrever o sistema de equações (Erlhagen, Trajectory Generation in robotic manipulators, 2020) presente embaixo. Neste encontra-se assinalada a restrição que deu origem a cada equação.

$$\begin{cases} a_0 = \theta_{i,0} \\ a_1 = \dot{\theta}_{i,0} \\ \theta_{i,f} = \theta_{i,0} + \dot{\theta}_{i,0} T + T^2 a_2 + T^3 a_3 \\ \dot{\theta}_{i,f} = \dot{\theta}_{i,0} + 2a_2 T + 3a_3 T^2 \end{cases} \tag{31}$$

Resolvendo-se este sistema é possível obter as expressões que permitem calcular os coeficientes da equação polinomial cúbica que fornece a evolução do ângulo de uma junta, ao longo do tempo.

$$\begin{cases} a_0 = \theta_{i,0} \\ a_1 = \dot{\theta}_{i,0} \\ a_2 = -\frac{1}{T}\dot{\theta}_{i,f} - \frac{2}{T}\dot{\theta}_{i,0} + \frac{3}{T^2}(\theta_{i,f} - \theta_{i,0}) \\ a_3 = -\frac{2}{T}(\theta_{i,f} - \theta_{i,0}) + \frac{1}{T^2}(\dot{\theta}_{i,f} + \dot{\theta}_{i,0}) \end{cases} \quad (32)$$

8.1.2.3.2. Perfil Polinomial 3ª Ordem - Implementação

Apresentados os princípios teóricos do cálculo polinomial de 3ª ordem, será exibido agora a sua implementação em *Modelica*. Numa primeira instância foram calculados os coeficientes polinomiais de trajetória para cada um dos eixos. O coeficiente implicado na constrição da posição inicial adota o valor presente no *array q_begin*, enquanto que o de velocidade inicial adota o valor de 0. Uma velocidade inicial de 0, implica assim que $a_2 = \frac{3}{T^2}(\theta_{i,f} - \theta_{i,0})$ e ainda que $a_3 = -\frac{2}{T}(\theta_{i,f} - \theta_{i,0})$, cuja variação angular, $(\theta_{i,f} - \theta_{i,0})$, é determinada pela diferença entre os dois vetores de posição inicial e final definidos no modelo.

Coeficientes

(Eixo)

1
```
Real a0_1 = q_begin[1];
Real a1_1 = 0;
Real a2_1 = if startTime <> 0 then 3 / T ^ 2 * (q_end[1] - q_begin[1]) else 0;
Real a3_1 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[1] - q_begin[1]) else 0;
```

2
```
Real a0_2 = q_begin[2];
Real a1_2 = 0;
Real a2_2 = if startTime <> 0 then 3 / T ^ 2 * (q_end[2] - q_begin[2]) else 0;
Real a3_2 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[2] - q_begin[2]) else 0;
```

3
```
Real a0_3 = q_begin[3];
Real a1_3 = 0;
Real a2_3 = if startTime <> 0 then 3 / T ^ 2 * (q_end[3] - q_begin[3]) else 0;
Real a3_3 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[3] - q_begin[3]) else 0;
```

4
```
Real a0_4 = q_begin[4];
Real a1_4 = 0;
Real a2_4 = if startTime <> 0 then 3 / T ^ 2 * (q_end[4] - q_begin[4]) else 0;
Real a3_4 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[4] - q_begin[4]) else 0;
```

5
```
Real a0_5 = q_begin[5];
Real a1_5 = 0;
Real a2_5 = if startTime <> 0 then 3 / T ^ 2 * (q_end[5] - q_begin[5]) else 0;
Real a3_5 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[5] - q_begin[5]) else 0;
```

6
```
Real a0_6 = q_begin[6];
Real a1_6 = 0;
Real a2_6 = if startTime <> 0 then 3 / T ^ 2 * (q_end[6] - q_begin[6]) else 0;
Real a3_6 = if startTime <> 0 then (-2 / T ^ 3) * (q_end[6] - q_begin[6]) else 0;
```

Figura 64 – Excerto da implementação de um perfil cinemático polinomial de 3ªordem, contido no código envolvido no modelo de geração do perfil cinemático

Deve-se exaltar o facto, de este cálculo acabar por descrever o cálculo de coeficientes para modelos de perfil de movimento em que uma trajetória direta é verificada. Ou seja, sem o modo *Via Point* ativado. Nas situações em que se recorre a *Via Points,* estes definem pontos por onde a trajetória da ponta do manipulador deve passar, mas não necessariamente parar neles de forma absoluta. O objetivo é a observação de um abrandamento, de forma a certificar que a passagem pelo ponto de referência ocorre, sem mudanças bruscas na direção de trajetória e também sem ocorrência de uma paragem absoluta. Assim, o valor de velocidade final até ao *via point* e o valor de velocidade inicial a partir do *via point* deverão apresentar valores não nulos. Essa alteração nas restrições de velocidade, alteram desde logo a forma de cálculo dos restantes coeficientes (a_2 *e* a_3).

```
  Real junta1 = if startTime <> 0 and time < endTime then a0_1 + a1_1 * (time - startTime) + a2_1 * (time -
startTime) ^ 2 + a3_1 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[1] else q_end[1];
  Real junta2 = if startTime <> 0 and time < endTime then a0_2 + a1_2 * (time - startTime) + a2_2 * (time -
startTime) ^ 2 + a3_2 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[2] else q_end[2];
  Real junta3 = if startTime <> 0 and time < endTime then a0_3 + a1_3 * (time - startTime) + a2_3 * (time -
startTime) ^ 2 + a3_3 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[3] else q_end[3];
  Real junta4 = if startTime <> 0 and time < endTime then a0_4 + a1_4 * (time - startTime) + a2_4 * (time -
startTime) ^ 2 + a3_4 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[4] else q_end[4];
  Real junta5 = if startTime <> 0 and time < endTime then a0_5 + a1_5 * (time - startTime) + a2_5 * (time -
startTime) ^ 2 + a3_5 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[5] else q_end[5];
  Real junta6 = if startTime <> 0 and time < endTime then a0_6 + a1_6 * (time - startTime) + a2_6 * (time -
startTime) ^ 2 + a3_6 * (time - startTime) ^ 3 elseif startTime == 0 then q_begin[6] else q_end[6];
  Real juntas[6] = {junta1, junta2, junta3, junta4, junta5, junta6};
  Real vel_juntas[6] = der(juntas);
```

Figura 65 - Excerto da implementação de um perfil cinemático polinomial de 3ªordem, contido no código envolvido no modelo de geração do perfil cinemático - Continuação

Calculados os coeficientes, o passo seguinte foi a escrita da equação do movimento (equação 30), ou seja, a evolução do ângulo ao longo do tempo. A passagem de código teve em consideração que a evolução de cada um dos ângulos, descritos pela equação, deve ocorrer no instante devido, pelo que a sua execução apenas ocorre dentro da condição estabelecida. Esta condição, define a variação do valor angular apenas quando o intervalo de tempo de simulação do sistema global *Pick&Place* se encontra entre o instante em que o modelo é ativado, sendo este instante definido pela modelação paralela a este modelo, e o instante de tempo definido pelo tempo inicial em que o modelo é "ativado" + tempo final do perfil de movimento. A função está definida para t com $0 < t < T$, pelo que t se define como o tempo total de simulação menos o instante de tempo em que este modelo inicializa o cálculo do perfil de movimento e T se define como o tempo total de deslocamento dos 6 eixos.

8.1.2.4. Perfil Cinemático Polinomial 5ª Ordem

8.1.2.4.1. Perfil Polinomial 5ª Ordem – Teoria

Na eventualidade de se preferir um movimento ainda mais suave comparativamente à função polinomial de 3ª ordem, uma função polinomial de 5ª ordem foi desenvolvida, em que restrições de aceleração podem ser configuradas, no início e final do movimento, com o valor de 0. Desta forma, somos confrontados com mais 2 coeficientes, ao longo da função, que são utilizados para satisfazer as restrições de aceleração mencionadas.

Na equação 33 encontra-se presente a expressão que permite descrever a evolução do ângulo de cada junta ao longo do tempo. Nesta expressão pretende-se determinar a_0, a_1, a_2, a_3, a_4 e a_5.

$$\theta_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (33)$$

Derivando-se a equação 33 obtém-se a equação 34, que permite obter a velocidade angular em cada junta ao longo do tempo.

$$\dot{\theta}_i(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \quad (34)$$

Restrições

As restrições (Erlhagen, Trajectory Generation in robotic manipulators, 2020) para a trajetória que se pretende realizar encontram-se na equações 35-40. Estas são o ângulo inicial ($\theta_{i,0}$) e final ($\theta_{i,f}$) da junta, a velocidade inicial ($\dot{\theta}_{i,0}$) e final ($\dot{\theta}_{i,f}$) e por fim a aceleração inicial ($\ddot{\theta}_{i,0}$) e final ($\ddot{\theta}_{i,f}$) da junta. Na equação, T é o tempo de duração do movimento, que deve ser igual para todas as juntas.

$$\theta_i(0) = \theta_{i,0} \quad (35)$$

$$\theta_i(T) = \theta_{i,f} \quad (36)$$

$$\dot{\theta}_i(0) = \dot{\theta}_{i,0} \quad (37)$$

$$\dot{\theta}_i(T) = \dot{\theta}_{i,f} \quad (38)$$

$$\ddot{\theta}_i(0) = \ddot{\theta}_{i,0} \quad (39)$$

$$\ddot{\theta}_i(T) = \ddot{\theta}_{i,f} \quad (40)$$

Determinação de a_0, a_1, a_2, a_3, a_4 e a_5

A partir das restrições presentes nas equações 33-34 e nas equações 35-40 é possível escrever o sistema de equações (Erlhagen, Trajectory Generation in robotic manipulators, 2020). Neste encontra-se assinalada a restrição que deu origem a cada equação.

$$\begin{cases} a_0 = \theta_{i,0} \\ a_1 = \dot{\theta}_{i,0} \\ a_2 = \dfrac{\ddot{\theta}_i}{2} \\ a_3 = \dfrac{20\theta_f - 20\theta_0 - (8\dot{\theta}_f + 12\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f){t^2}_f}{2{t^3}_f} \\ a_4 = \dfrac{30\theta_0 - 30\theta_f + (14\dot{\theta}_f + 16\dot{\theta}_0)t_f + (3\ddot{\theta}_0 - 2\ddot{\theta}_f){t^2}_f}{2{t^4}_f} \\ a_5 = \dfrac{12\theta_f - 12\theta_0 - (6\dot{\theta}_f + 6\dot{\theta}_0)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f){t^2}_f}{2{t^5}_f} \end{cases} \quad (41)$$

8.1.2.4.2. Perfil Polinomial 5ª Ordem - Implementação

À semelhança dos coeficientes calculados para o cálculo polinomial de 3ª ordem, o coeficiente implicado na constrição da posição inicial adota o valor presente no *array q_begin*, enquanto que o da velocidade inicial adota o valor de 0 e, da mesma forma, o coeficiente a_2 da aceleração inicial, também. Tendo em perspetiva os valores nulos ditados para os coeficientes destinados às restrições iniciais, fica-se com $a_3 = \frac{20(\theta_{i,f} - \theta_{i,0})}{2{t^3}_f}$, $a_4 = \frac{30(\theta_{i,0} - \theta_{i,f})}{2{t^4}_f}$ e $a_5 = \frac{12(\theta_{i,f} - \theta_{i,0})}{2{t^5}_f}$.

```
Real a0_1_5th = q_begin[1];
Real a1_1_5th = 0;
Real a2_1_5th = 0;
Real a3_1_5th = if Tempo_Inicial then (20 * (q_end[1] - q_begin[1])/(2*T^3)) else 0;
Real a4_1_5th = if Tempo_Inicial then (30 * (q_begin[1] - q_end[1])/(2*T^4)) else 0;
Real a5_1_5th = if Tempo_Inicial then (12 * (q_end[1] - q_begin[1])/(2*T^5)) else 0;

Real a0_2_5th = q_begin[2];
Real a1_2_5th = 0;
Real a2_2_5th = 0;
Real a3_2_5th = if Tempo_Inicial then (20 * (q_end[2] - q_begin[2])/(2*T^3)) else 0;
Real a4_2_5th = if Tempo_Inicial then (30 * (q_begin[2] - q_end[2])/(2*T^4)) else 0;
Real a5_2_5th = if Tempo_Inicial then (12 * (q_end[2] - q_begin[2])/(2*T^5)) else 0;

Real a0_3_5th = q_begin[3];
Real a1_3_5th = 0;
Real a2_3_5th = 0;
Real a3_3_5th = if Tempo_Inicial then (20 * (q_end[3] - q_begin[3])/(2*T^3)) else 0;
Real a4_3_5th = if Tempo_Inicial then (30 * (q_begin[3] - q_end[3])/(2*T^4)) else 0;
Real a5_3_5th = if Tempo_Inicial then (12 * (q_end[3] - q_begin[3])/(2*T^5)) else 0;

Real a0_4_5th = q_begin[4];
Real a1_4_5th = 0;
Real a2_4_5th = 0;
Real a3_4_5th = if Tempo_Inicial then (20 * (q_end[4] - q_begin[4])/(2*T^3)) else 0;
Real a4_4_5th = if Tempo_Inicial then (30 * (q_begin[4] - q_end[4])/(2*T^4)) else 0;
Real a5_4_5th = if Tempo_Inicial then (12 * (q_end[4] - q_begin[4])/(2*T^5)) else 0;

Real a0_5_5th = q_begin[5];
Real a1_5_5th = 0;
Real a2_5_5th = 0;
Real a3_5_5th = if Tempo_Inicial then (20 * (q_end[5] - q_begin[5])/(2*T^3)) else 0;
Real a4_5_5th = if Tempo_Inicial then (30 * (q_begin[5] - q_end[5])/(2*T^4)) else 0;
Real a5_5_5th = if Tempo_Inicial then (12 * (q_end[5] - q_begin[5])/(2*T^5)) else 0;

Real a0_6_5th = q_begin[6];
Real a1_6_5th = 0;
Real a2_6_5th = 0;
Real a3_6_5th = if Tempo_Inicial then (20 * (q_end[6] - q_begin[6])/(2*T^3)) else 0;
Real a4_6_5th = if Tempo_Inicial then (30 * (q_begin[6] - q_end[6])/(2*T^4)) else 0;
Real a5_6_5th = if Tempo_Inicial then (12 * (q_end[6] - q_begin[6])/(2*T^5)) else 0;
```

Figura 66 - Excerto da implementação de um perfil cinemático polinomial de 5ªordem, contido no código envolvido no modelo de geração do perfil cinemático

Calculados os coeficientes, o passo seguinte foi a escrita da equação do movimento (equação 44), à semelhança do verificado para o perfil polinomial de 3ª ordem.

```
  Real junta1_5th = if Tempo_Inicial and time < endTime then a0_1_5th + a1_1_5th * (time - startTime) + a2_1_5th * (time - startTime) ^ 2 + a3_1_5th *
(time - startTime) ^ 3 + a4_1_5th * (time - startTime) ^ 4 + a5_1_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[1] else
q_end[1];
  Real junta2_5th = if Tempo_Inicial and time < endTime then a0_2_5th + a1_2_5th * (time - startTime) + a2_2_5th * (time - startTime) ^ 2 + a3_2_5th *
(time - startTime) ^ 3 + a4_2_5th * (time - startTime) ^ 4 + a5_2_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[2] else
q_end[2];
  Real junta3_5th = if Tempo_Inicial and time < endTime then a0_3_5th + a1_3_5th * (time - startTime) + a2_3_5th * (time - startTime) ^ 2 + a3_3_5th *
(time - startTime) ^ 3 + a4_3_5th * (time - startTime) ^ 4 + a5_3_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[3] else
q_end[3];
  Real junta4_5th = if Tempo_Inicial and time < endTime then a0_4_5th + a1_4_5th * (time - startTime) + a2_4_5th * (time - startTime) ^ 2 + a3_4_5th *
(time - startTime) ^ 3 + a4_4_5th * (time - startTime) ^ 4 + a5_4_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[4] else
q_end[4];
  Real junta5_5th = if Tempo_Inicial and time < endTime then a0_5_5th + a1_5_5th * (time - startTime) + a2_5_5th * (time - startTime) ^ 2 + a3_5_5th *
(time - startTime) ^ 3 + a4_5_5th * (time - startTime) ^ 4 + a5_5_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[5] else
q_end[5];
  Real junta6_5th = if Tempo_Inicial and time < endTime then a0_6_5th + a1_6_5th * (time - startTime) + a2_6_5th * (time - startTime) ^ 2 + a3_6_5th *
(time - startTime) ^ 3 + a4_6_5th * (time - startTime) ^ 4 + a5_6_5th * (time - startTime) ^ 5 elseif Tempo_Inicial == false then q_begin[6] else
q_end[6];
```

Figura 67 - Excerto da implementação de um perfil cinemático polinomial de 5ªordem, contido no código envolvido no modelo de geração do perfil cinemático - Continuação

8.1.2.5. Trajetória Direta – Trajetória com Via Points

O desenvolvimento do *Pick* e do *Place* visou numa fase inicial a execução de cada um dos dois percursos recorrendo a uma trajetória direta, ou seja, o perfil de movimento é efetuado tendo por base o ponto inicial em que o robot se encontra e o ponto final em que o movimento do *gripper* é inicializado, e dessa forma fazer a interação com a embalagem a ser agarrada/largada.

Esta primeira abordagem é sem dúvida a mais simplista e a que reduz ao máximo, quer o tamanho dos cálculos de trajetória envolvidos, quer o tempo de funcionamento de robot para que a operação seja devidamente realizada. Surge no entanto, a relevância de sublinhar o quão inadequada esta forma de execução poderá ser, quer por motivos de segurança, tendo em consideração a trajetória menos sequenciada e sem pontos de abrandamento, quer do ponto de vista de logística, com a possibilidade de o *gripper* embater nas embalagens antes de se posicionar para amarrá-las por cima. Tendo por base estes princípios, e no facto da maioria dos processos industriais implementarem uma trajetória do movimento recorrendo a *Via Points*, existiu a tentativa de replicar este procedimento através de lógica programada na modelação orientada a objetos e dessa forma tentar analisar uma performance mais realista do processo em causa.

Os *Via Points* acabam por ser pontos de referência no espaço, sobre o qual o braço robótico passa, antes de atingir o ponto cartesiano pretendido. Os *Via Points* adotados acabam por ser pontos por cima de cada uma das caixas a serem transportadas, num ponto do espaço ligeiramente mais acima da posição em que o movimento do *gripper* é inicializado. Ocorre assim uma primeira aproximação à embalagem cuja posição cartesiana xy já obedece à posição final desejada, variando apenas no eixo de cotas. A utilização dos *Via Points* poderia ser alargada a um maior número de pontos cartesianos, ou até mesmo à passagem repetida pelo ponto de aproximação depois da embalagem ser agarrada, no entanto para efeitos de testagem do conceito, a seguinte sequência de movimentos foi entendida como suficiente.

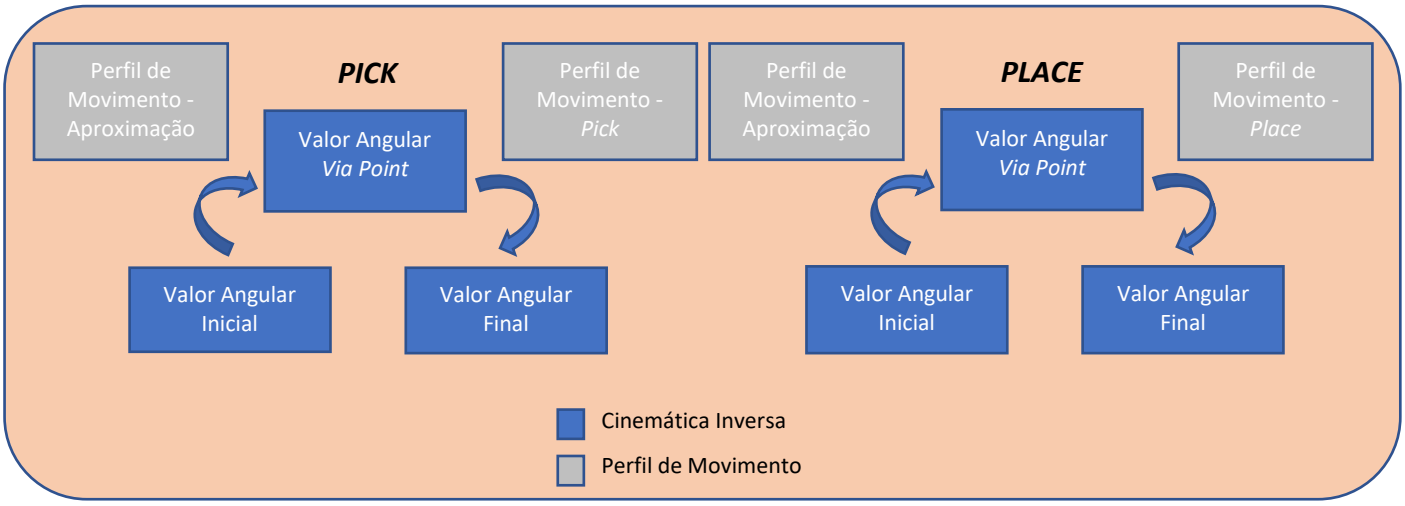


Figura 68 - Sequência de movimentos com Modo *Via Point*

A implementação do modo de operação com *via points* implicou uma reestruturação na modelação da cinemática inversa assim como no perfil de movimento. No lugar de dois cálculos de cinemática inversa e dois de perfil de movimento, passa-se a ter quatro para cada uma das diferentes fases. As mudanças que este novo conceito introduz na modelação em *Modelica* serão prontamente explicadas.

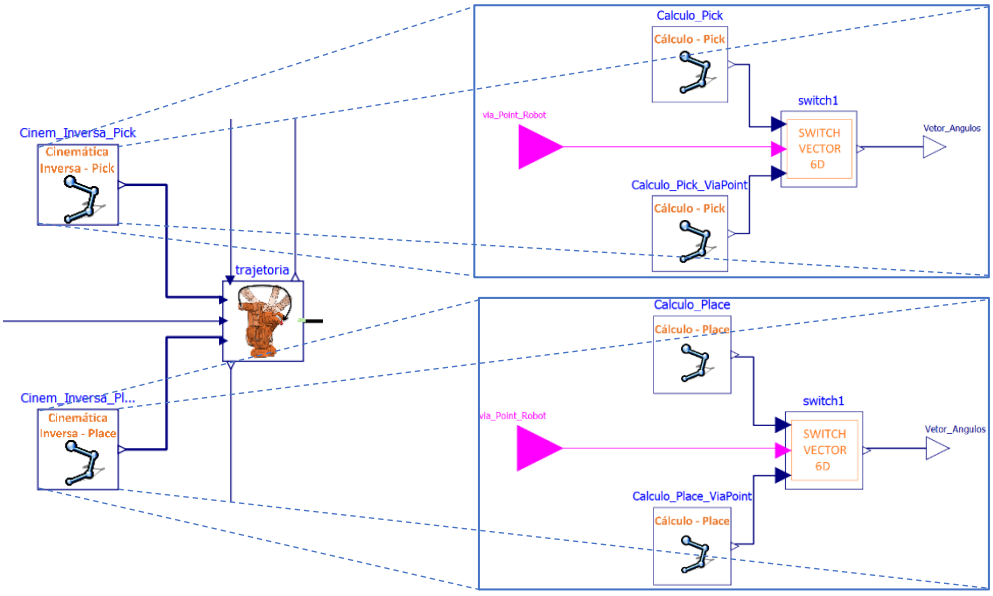


Figura 69 - Reestruturação na modelação da cinemática inversa. Modelo de *Pick* e do *Place* à esquerda, sendo que cada um é constituído, por sua vez, por 2 modelos de cinemática inversa (um de aproximação e outro de finalização do movimento).

No lugar de dois modelos de cinemática inversa, em que ocorre de forma direta o cálculo dos valores angulares finais do *Pick* e do *Place* e posteriormente associados ao modelo em que o perfil de movimento da trajetória é calculada, tem-se um modelo em que apenas no seu interior um modelo de cálculo é apresentado. Assim, dentro do modelo de cinemática inversa, destinado ao *Pick,* é apresentado um diagrama constituído por dois modelos de cálculo, um para a cinemática inversa do ponto cartesiano final do *Pick*, caso o modo *Via Point* não esteja ativado, e outro para a cinemática inversa do ponto cartesiano de aproximação do *Pick* caso o modo *Via Point* esteja ativado. A seleção de um dos dois, surge na sequência de um booleano, cujo parâmetro se encontra diretamente igualado ao parâmetro *Via Point* definido no sistema global pelo utilizador. Caso o booleano se encontre verdadeiro, tem-se a transmissão do vetor dos ângulos dos seis eixos por parte do modelo que calcula a cinemática inversa do *Via Point*, através de um modelo *switch* (Figura 69). O modelo *switch* foi elaborado

de raiz com a particularidade de, face ao *switch* da *Modelica Standard Library*, se destinar a um *array* de 6 dimensões.

Estabelecido este rearranjo, de forma a reunir uma modelação de acordo com as pretensões de trajetória, declaradas pelo utilizador, resta um reajuste no perfil de movimento, desta feita um pouco mais complexo, e ainda um modelo de cálculo adicional para a cinemática inversa relativa ao *Pick* e *Place,* caso o modo *Via Point* seja ativado. O modelo do perfil de movimento elaborado será de seguida explicado com atribuição numerada, por ordem de funcionamento, a cada uma das secções do diagrama presente no modelo.

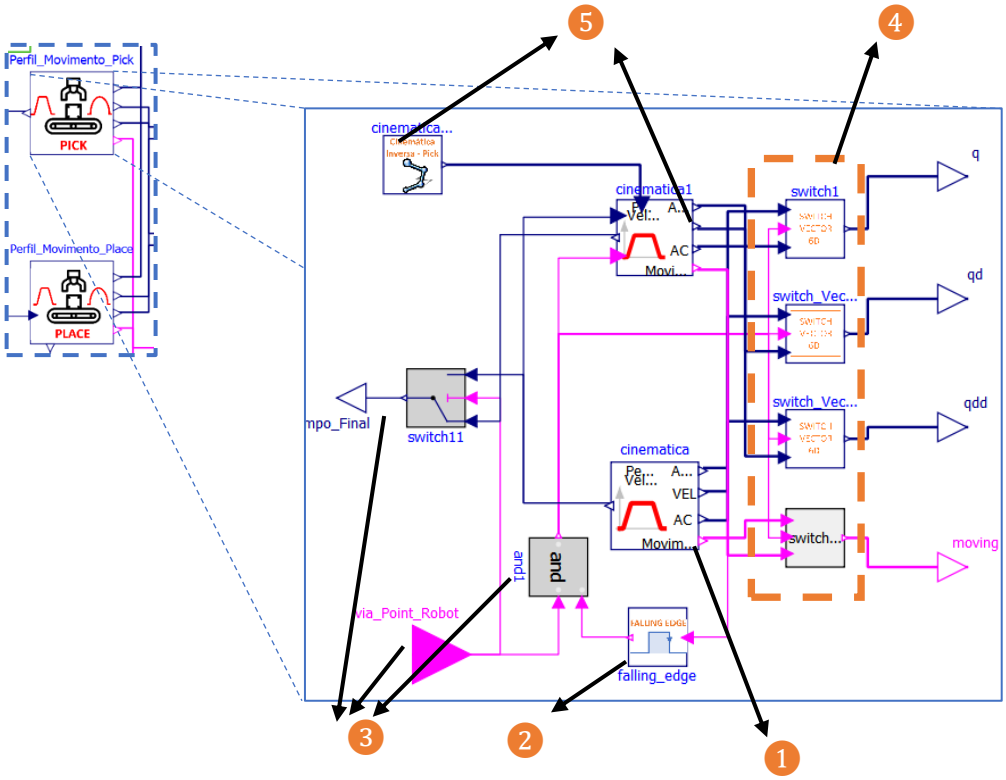


Figura 70 - Reestruturação do modelo de perfil de movimento

1 - Primeiro modelo relativo ao cálculo do perfil de movimento Posição Inicial – Posição de *Pick*. Caso se esteja perante o modo *Via Point* ativo, os valores programados que dão entrada neste modelo são os valores angulares das juntas da posição inicial, assim como da posição de aproximação do *Pick*. Se por outro lado este modo não estiver ativo, os valores das juntas finais correspondem às variações angulares para a posição cartesiana final do *Pick*.

2 - Modelo elaborado de raiz, que admite como saída um booleano positivo, de tempo indeterminado, após um *falling edge* ser verificado. No diagrama, este *falling edge* acaba por

dizer respeito ao cessamento de movimento por parte do comando dos eixos. No modo *via point*, o cessamento de movimento diz respeito à aproximação à embalagem, enquanto que na ausência deste modo tem-se o movimento de *Pick* (do braço) completado na sua plenitude.

3 - Conjunto constituído por um *input* real que determina a existência ou não do modo *via point*, de uma condição *and* que estabelece a saída de um booleano em função da existência conjunta do modo e do *falling edge* (ou seja, cálculo do primeiro perfil de movimento finalizado) e ainda de um *switch* cujo estado de saída é determinado pela condição *and.* Desta forma, na ausência do modo via point, quando o modelo do perfil de movimento termina o cálculo da variação de posição, velocidade e aceleração das juntas, o tempo de término sai do modelo do perfil e entra no switch no estado de falso, visto que o booleano *and* não se encontra verdadeiro. Se por outro lado o modo *via point* estiver ativo, o booleano *and* adota o estado verdadeiro, transmitindo esse estado para o 2º modelo do perfil de movimento, para que o cálculo no mesmo seja inicializado. Ao longo deste modo, a saída do tempo final apenas tomará lugar após o cálculo no do 2º perfil de movimento.

4 - Da mesma forma que existe um *switch* responsável pela transmissão do tempo de término de cálculo de todo o movimento *Pick*, estabelecido quer para um movimento direto ou para um movimento com aproximação pelo meio do percurso, também existe apresentado na secção mais à direita do modelo, um conjunto de *switch's,* desta feita para vetores a 6 dimensões, responsáveis por transmitir às saídas do modelo cada uma das posições, velocidades, acelerações e estados de movimento, definidos ao longo dos modelos de cálculo do perfil ao longo do tempo. O booleano *and* é, novamente, a condição que define qual dos comandos serão posteriormente enviados através dos *switch's*.

5 - Na eventualidade de se encontrar ativo o modo via point, um segundo perfil terá de ser calculado, representativo do movimento entre o ponto de referência de aproximação e o ponto cartesiano final. Para o efeito, dão entrada no segundo modelo de perfil o valor do instante de tempo em que este se inicia, e corresponde ao instante de término do 1ºmodelo de perfil, e ainda o booleano de inicialização, permitindo não só a ativação dos *switch's* como também desencadear o cálculo necessário dentro do modelo do perfil. Para o 2º modelo do perfil de movimento, tem-se ainda os valores angulares iniciais que correspondem aos valores angulares finais do 1º modelo, sendo esta equivalência programada no *texto view* do programa, e os valores angulares finais são calculados com um modelo de cálculo de cinemática inversa, também este devidamente programado, em que os valores angulares são

calculados para o ponto cartesiano em questão, variando este em função dos tapete de entrada escolhido pelo utilizador.

8.1.2.6. Soma de Trajetórias

Elaborado o cálculo individual do perfil cinemático *Pick* e o perfil cinemático *Place*, procede-se a soma dos dois perfis ao longo do tempo, perfazendo um perfil cinemático único que possa ser enviado para o controlo dos diferentes eixos.

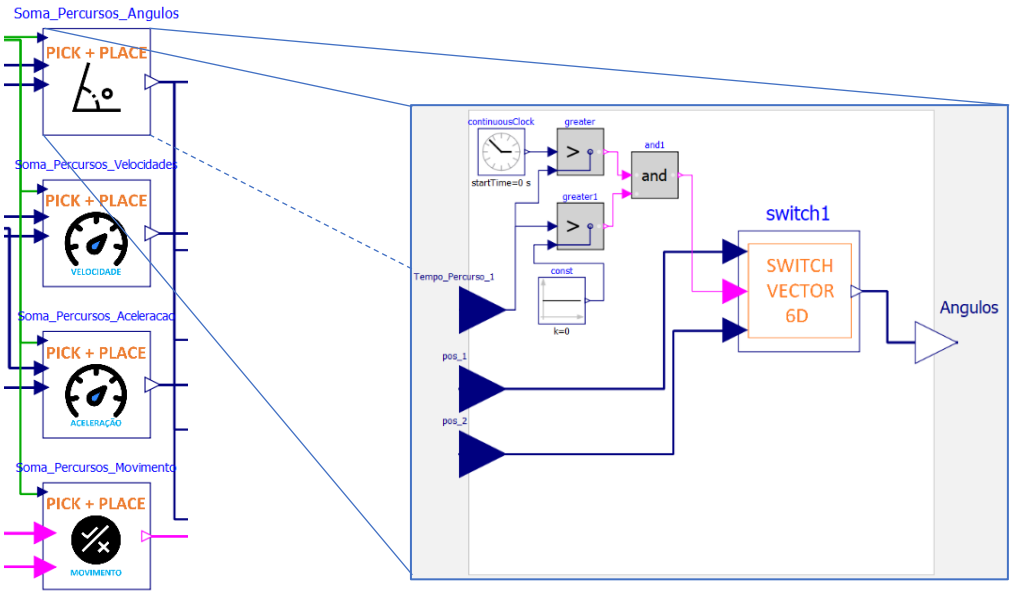


Figura 71 - Modelos destinados à soma dos perfis de movimento ao longo do tempo (*Pick* + *Place*)

A conceção desse perfil único é feita com o auxílio de um modelo de cálculo do vetor final, cuja saída do mesmo é o vetor do perfil. Esse modelo pode ser observado na Figura 72.

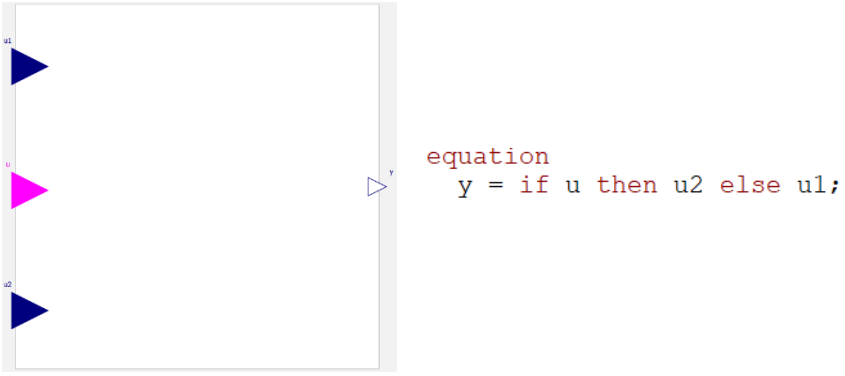


Figura 72 - Modelo que define o vetor final

O vetor de saída equivale ao vetor de entrada proveniente do perfil *Pick,* até uma condição booleana ser verificada, sendo esta variável booleana ativada quando o instante de tempo no cronómetro (cujo valor é iniciado aquando do início da simulação) for superior ao instante de tempo que dá entrada no modelo. O valor do instante de tempo que dá entrada no modelo é equivalente ao tempo de término do perfil cinemático *Pick*. Depois da condição booleana ser verificada o vetor de saída passa a equivaler ao vetor proveniente do perfil cinemático *Place.* Há assim uma "troca" de perfis ao longo do tempo, de modo que à saída do bloco o vetor seja constituído pela soma de ambos.

Os modelos destinados ao cálculo dos vetores finais de posição, velocidade, aceleração e movimento, apresentam todos eles a mesma estrutura. Há a exceção, no entanto, para o modelo que retrata se o robot está em movimento em cada um dos eixos ou não, em que uma pequena alteração é efetuada com o uso de vetores booleanos e não vetores reais à entrada do modelo auxiliar de cálculo.

8.1.2.7. Distribuição dos perfis finais pelos eixos

As variáveis de saída dos modelos de soma dos 2 perfis, a nível de posição, velocidade, aceleração e movimento apresenta a seguinte forma:

$$q = \begin{bmatrix} q_1 \\ \vdots \\ q_7 \end{bmatrix};\ qd = \begin{bmatrix} qd_1 \\ \vdots \\ qd_7 \end{bmatrix};\ qdd = \begin{bmatrix} qdd_1 \\ \vdots \\ qdd_7 \end{bmatrix}; moving = \begin{bmatrix} True/False \\ \vdots \\ True/False \end{bmatrix} \qquad (42)$$

Cada um destes vetores são, por fim, distribuídos em função do eixo em causa, ou seja, cada uma das variáveis $q[i]$, $qd[i]$, $qdd[i]$ e $moving[i]$ são conectadas ao bloco do eixo i respetivo.

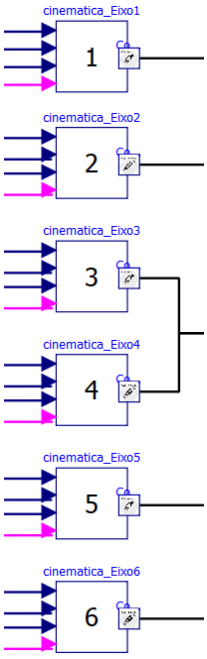


Figura 73 - Modelos responsáveis por recolherem as diferentes variáveis destinadas ao comando de cada eixo

8.1.3. Braço Robótico – Controlo

Ao longo deste capítulo irá fazer-se uma introdução aos princípios de funcionamento de um braço robótico, dando maior destaque ao método de controlo dos diferentes eixos. A tecnologia IRC5, presente por exemplo nos robots da ABB, servirá assim como uma referência, permitindo uma compreensão generalizada do controlo dos braços robóticos industriais. A pertinência do estudo desta tecnologia advém do facto de ser um modelo base de controlo cujo modo de operação se aproxima com a maioria das técnicas de controlo empregadas em manipuladores industriais.

Assimilados esses conceitos, uma transposição para o sistema de modelação a ser efetuado, será elaborada, com os devidos ajustes ou modificações que se acharem adequadas. O objetivo, assim, não se baseia numa replicação digital exata de um robot industrial específico, mas sim obter um entendimento sólido da forma como estes operam, genericamente, e a partir daí partir para meios de modelação que possam ser implementados, auxiliando assim no estudo do sistema.

- **Alimentação dos motores**

Em simultaneidade com o controlo dos motores, através de um processo de *feedback*, os motores, que no caso da ABB são maioritariamente motores trifásicos síncronos, devem ser devidamente alimentados.

Para o efeito, uma das topologias mais utilizadas em sistemas industriais é o conversor AC/DC/AC de frequência, também denominado de *back-to-back* (Babu & V., 2020). Este conversor é empregado no sistema de driver, pois não só admite energia elétrica a ser fornecida ao motor, como também a controla por comandos, numa fase final. O conversor poderá ser dividido em diferentes fases.

Figura 74 - Diferentes fases do conversor

Este apresenta a particularidade de transformar voltagem de corrente alternada, fornecida pela rede, em voltagem de corrente contínua, com recurso a um retificador. O retificador é assim a principal unidade de alimentação do sistema de driver.

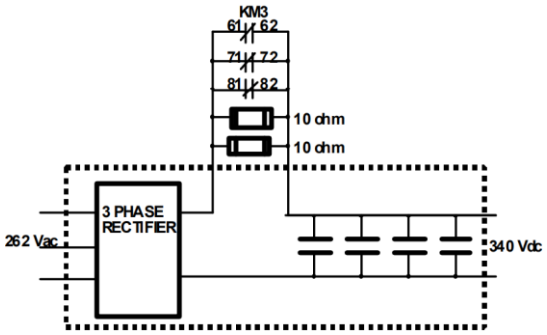


Figura 75 - *DC-Link* do manipulador industrial ABB IRB 140 (ABB, 1995)

O *DC-Link* é a secção de ligação entre o retificador e o inversor, providenciando a este último, voltagem de corrente contínua. Este apresenta um conjunto de condensadores, capazes de providenciar uma voltagem *DC* mais estável, limitando as flutuações quando picos de corrente são atingidos.

A secção inversora, devido ao facto de se possuir seis motores, um para cada eixo, é dividida em seis placas/unidades de driver.

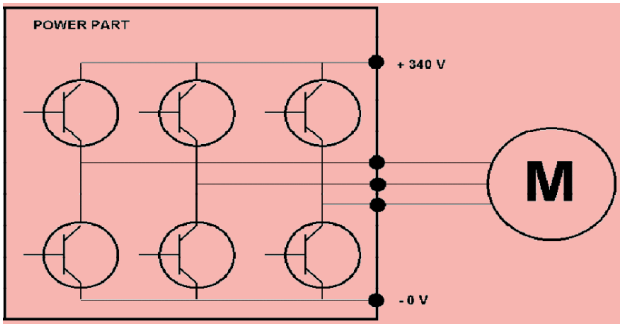


Figura 76 - Secção inversora connectada a um motor trifásico

Cada uma das unidades driver recolhe a corrente contínua proveniente do *DC-Link*, fornecendo essa corrente aos transístores. Os transístores são controlados por *PWM*, atuando como *switch*, através da comunicação direta que estabelecem com o computador do robot. Dependendo da posição dos ímans no motor, os transístores receberão uma configuração *on-off* que determinará o movimento do motor. Três fios elétricos ligam o motor à unidade *driver*, ou seja, um fio para cada fase (ABB, 1995).

- ***Resolvers***

Os motores associados ao braço usam *resolvers* (Yu, et al., 2022) de maneira a determinar a posição e velocidade de cada motor. Um *resolver* é um aparelho analógico que pode ser extremamente preciso na informação recolhida de *feedback* quanto às respetivas posições dos motores. Esta informação é transmitida sob a forma analógica ou com uma onda senoidal. Um enrolamento no rotor é aplicado ao veio do motor. Este enrolamento é energizado, ou excitado, com uma onda senoidal de 10KHz. Outros dois enrolamentos, de seno e cosseno, são aplicados na carcaça do *resolver* a uma distância angular exata de 90 graus entre cada um. Uma corrente é induzida nos 2 enrolamentos estáticos pelo enrolamento acoplado no rotor. Esta indução provoca uma forma de onda em cada um dos enrolamentos (ABB, 1995).

Relativamente a um *Encoder,* um *Resolver* apresenta as vantagens de não necessitar de calibração, ser mais preciso e é mais fácil de trabalhar.

O ângulo ou posição do enrolamento do rotor relativamente aos restantes dois enrolamentos, determinará a amplitude e polaridade dos sinais induzidos. Estes sinais por sua vez serão posteriormente transmitidos e processados pelo SMB, permitindo assim a sua interpretação. O SMB calcula, com base nos sinais, a posição exata do veio do motor, o que por sua vez indicará a posição exata do próprio robot através do cálculo por cinemática direta.

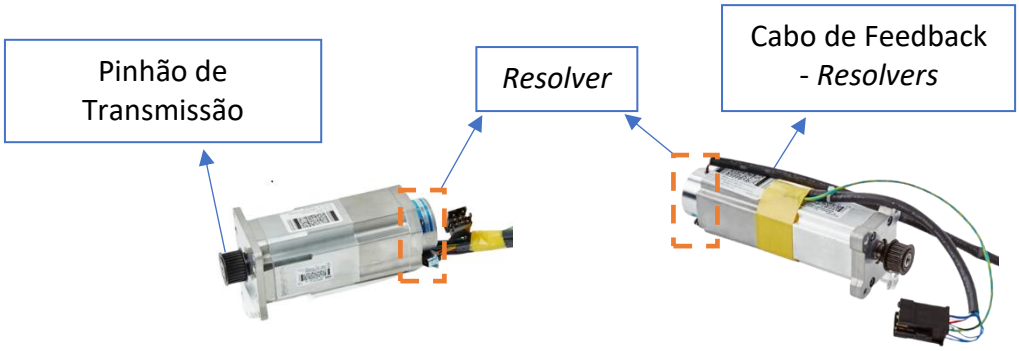


Figura 77 - Ilustração de motores pertencentes ao manipulador IRB 140

Cada um dos motores do braço, que por sua vez se encontram associados a cada um dos eixos, irão ter um *resolver* acoplado. Dentro do *resolver,* a bobina excitada pelo rotor irá criar um campo eletromagnético, que por sua vez excitará as dois enrolamentos restantes, gerando uma função seno e cosseno, respetivamente.

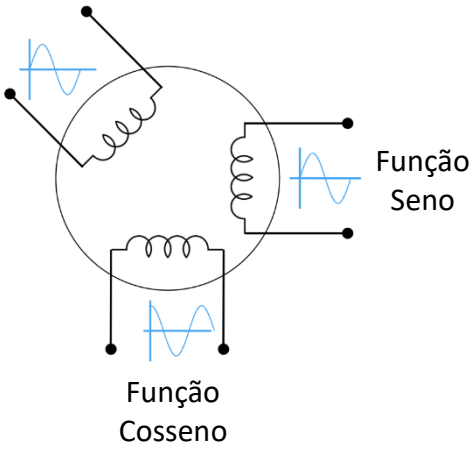


Figura 78 - Constituição de um Resolver

Os valores de seno e cosseno obtidos determinarão a respetiva posição angular instantânea do rotor.

SMB

O SMB lê os feedbacks provenientes dos *resolvers* e converte a voltagem analógica em sinais digitais. Depois o SMB converte os sinais digitais numa comunicação serial. A comunicação da posição em cada eixo é recolhida através dos *resolvers* e enviada para o computador do robot a cada 5 milissegundos, sensivelmente.

Os sinais analógicos provenientes dos *resolvers* poderão ser verificados através de um osciloscópio ligado ao *SMB* (ABB, 1995),

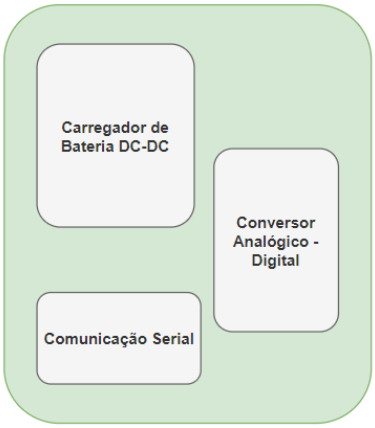


Figura 79 - Esquema da estrutura do *SMB*

MAIN COMPUTER

A "*pose*" é gerada por este componente, ou seja, o movimento planeado que o robot irá executar. No entanto, no computador outras informações são inseridas, tais como o peso, comprimento e orientação da ferramenta, os efeitos da gravidade e inércia, etc. Estas informações permitem que o sistema drive admita uma *performance* sólida.

A "*pose*" é gerada pelo programa presente no computador principal ou por um *joystick*, por exemplo, que se vem acompanhar pelo robot. O *software* e os parâmetros do utilizador determinarão os cálculos exatos a serem executados de forma a criar a trajetória do movimento. Esta trajetória é enviada do computador principal para o computador do robot, através de comandos de posição.

ROBOT COMPUTER

O computador do robot compara os comandos de posição provenientes do computador principal com os valores instantâneos calculados através do conjunto *resolvers*/*SMB* (ABB, 1995). Se uma diferença entre esses dois valores for admitida, um sinal será enviado para as unidades driver ao longo de cada eixo, segundo a técnica *PWM* (K. & A. , 2018). Este sistema de controlo é, portanto, um sistema de controlo de ciclo fechado.

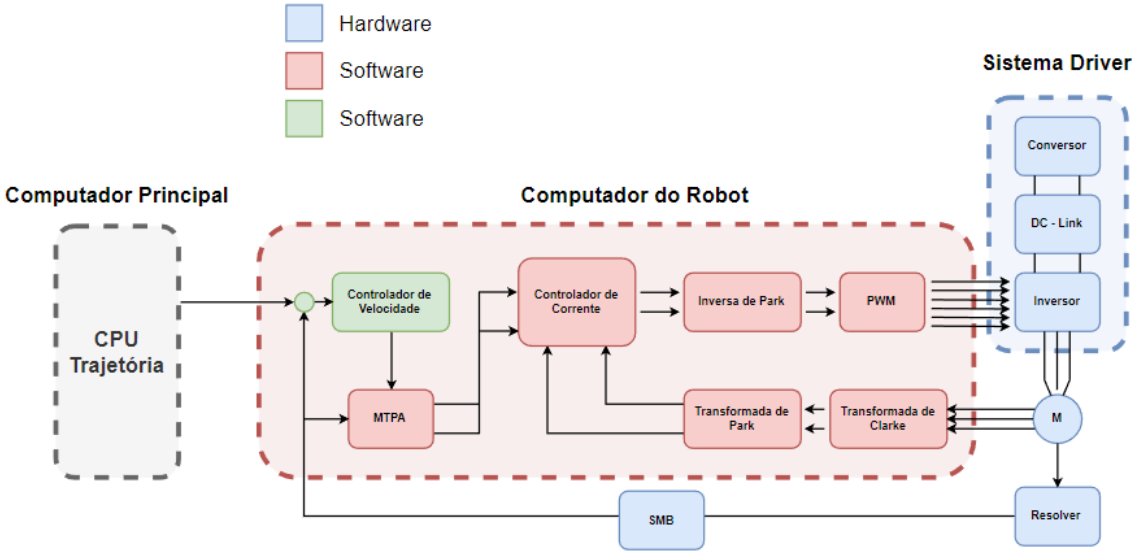


Figura 80 - Esquematização do sistema de controlo de um motor, baseado no manual do IRB -140

8.1.4. Modelação do Funcionamento do Robot

Uma vez analisada uma tecnologia de controlo como o IRC5, é importante incorporar parte dos seus princípios de controlo inerentes, na modelação do sistema de controlo do braço robótico a ser concebido. Facilmente se percebe que a modelação de um sistema com esta tecnologia, obedecendo a cada uma das secções e componentes abordados, exige um modelo global muito detalhado e uma certa complexidade multi-domínio.

No sentido de preservar a propriedade multi-domínio que uma tecnologia de controlo de um braço exige, mas procurando explorar um conjunto de modelos mais simplificados, a modelação em *Modelica* do braço robótico visou uma série de modificações e simplificações, das quais se destacam:

- O motor *AC* será substituído por um motor *DC* simplificado e, com isso, um menosprezo da secção de alimentação, constituída pelo retificador, *DC – Link* e ainda inversor *DC-AC*.
- Controlo por corrente usando a técnica de modulação *SVPWM (Space Vector Pulse Width Modulation)* é substituído por um controlo de corrente usando amplificadores operacionais.
- Um controlo de posição é acrescentado, admitindo dessa forma um controlo mais simples, mas compacto.
- O *resolver*, assim como todo o sistema de cálculo de posição presente no *SMB* é substituído por um sensor de posição e velocidade presente na *Modelica Standard Library*.
- Introdução de um sensor de corrente.

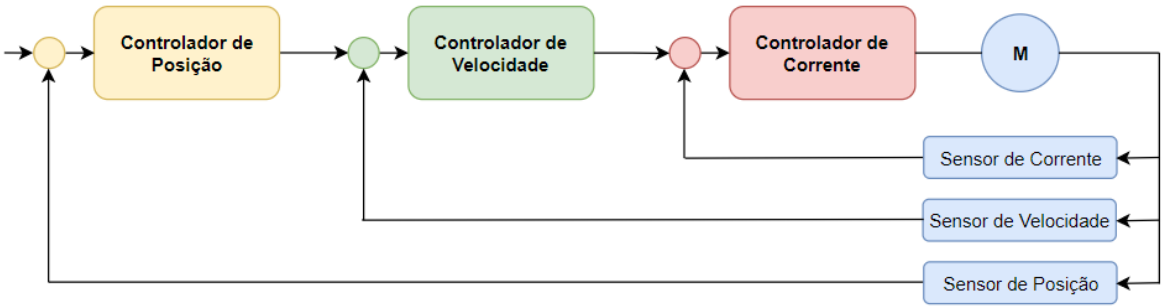


Figura 81 - Esquematização do sistema de controlo simplificado e de mais fácil adaptação à linguagem orientada a objetos

8.1.5. Controlo axial e Atuador

Para o controlo do motor em função das instruções providenciadas pelo computador do robot, foi modelado um driver de um servomotor. O driver aqui modelado também se poderá apelidar de amplificador na indústria de controlo de movimento. Este admite um sinal de comando da posição, velocidade ou corrente e ajusta a voltagem e corrente aplicada no servomotor. A lógica do driver inclui três tipos de ciclos ou *loops*: corrente, velocidade e posição. Cada um dos ciclos utilizam sinais de *feedback* para ajustar os valores à saída do ciclo e produzir dessa forma os resultados desejados.

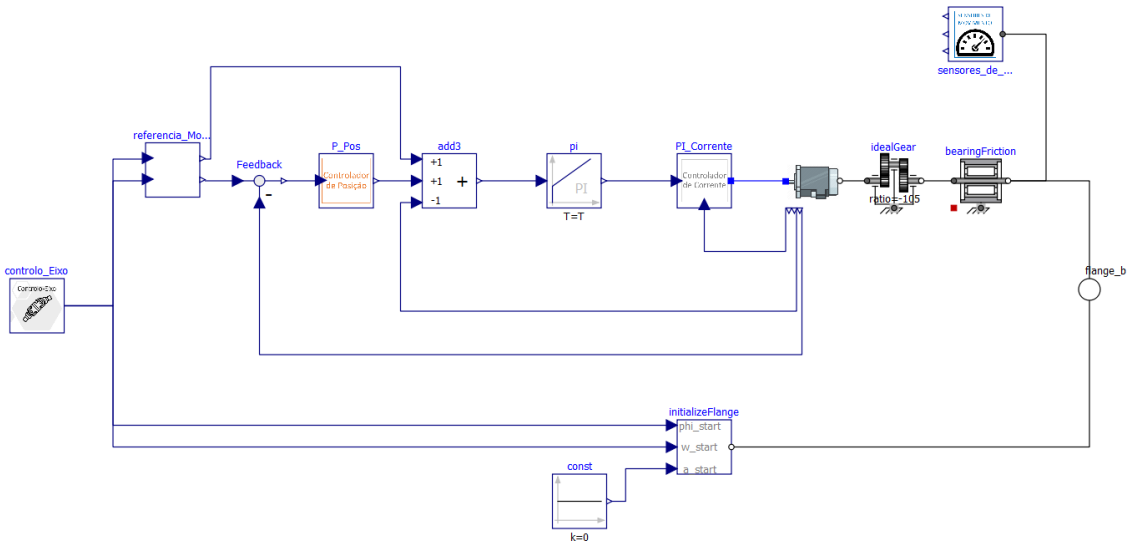


Figura 82 - Modelação do controlo axial, servomotor e transmissão

O controlo *PI/PID* (Wang, 2020) é um aspeto basilar no mundo automatizado. Desde a configuração de uma temperatura pretendida até ao controlo da pressão de uma ferramenta, o controlo *PI/PID* é um dos métodos de controlo mais eficazes. Ao longo deste capítulo será

examinado com algum detalhe os elementos que compõem o controlo P - PI inserido no controlo de cada eixo do braço. Numa fase inicial, a teoria geral de controlo será abordada, sendo procedida pela aplicação dos mesmos princípios em circuitos de amplificadores operacionais, que atendem ao mesmo propósito, mas por meio de controlo analógico.

Um diagrama de blocos para um sistema de controlo geral poderá ser observado na Figura 83, em que um sinal de entrada, $e(t)$, é enviado para o bloco do controlador. Depois, o sinal de controlo obtido, $u(t)$, é transmitido ao bloco da planta ou sistema físico a ser controlado. O ciclo de *feedback* compara o sinal de saída do bloco da planta, $h(t)$, com o sinal de entrada no sistema de controlo, $d(t)$. A diferença entre o sinal de saída e o sinal de entrada dá o termo do erro, $e(t)$, que alimenta novamente o controlador. Matematicamente, o termo do erro poderá ser calculado como:

$$e(t) = h(t) - d(t) \tag{43}$$

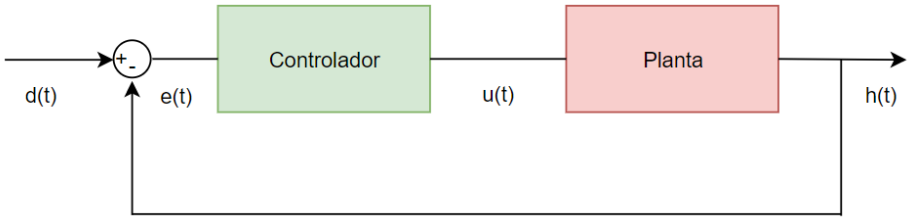


Figura 83 - Diagrama de blocos - Controlo Geral

A definição do controlo foi feita com base em sucessivas simulações do sistema, permitindo analisar a resposta e comportamento do mesmo.

8.1.5.1. Controlo de Posição

Para o controlo de posição ao longo de cada eixo, um *loop* externo de posição é inserido junto ao *loop* de velocidade, no que é também conhecido como um ciclo em cascata posição-velocidade. O ciclo de posição determina um erro que constitui o desvio entre a posição instantânea e a posição de referência, emitindo um comando de velocidade que visa reduzir ou eliminar esse mesmo erro. Num sistema em cascata, o ciclo de posição, por norma, apenas usa um ganho proporcional como controlo, tendo sido esse o controlo de posição adotado.

No controlo proporcional utilizado, o erro é multiplicado por uma constante de proporcionalidade, por norma apelidada de K_p, servindo o propósito de fazer com que o sistema reaja mais rapidamente, uma vez que o termo do erro é ampliado. O sinal de controlo pode ser escrito como:

$$u(t) = K_p . e(t) \tag{44}$$

Em que $u(t)$ é o sinal de controlo ao longo do tempo e, da mesma forma, $e(t)$ assume-se como o erro de posição obtido ao longo do tempo. A implementação do controlador de posição recorreu ao uso de um bloco proporcional, cujo ganho atribuído no bloco se assume como a constante de proporcionalidade K_p.

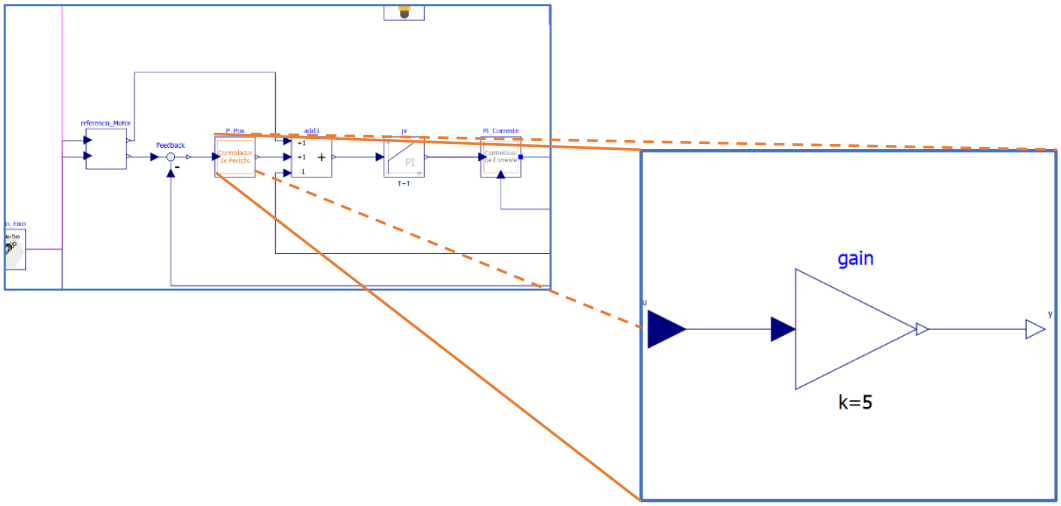


Figura 84 - Modelo do controlo de posição com um bloco proporcional

8.1.5.2. Controlo de Velocidade

O *loop* de velocidade é o mais comum para controlo de servomotores. Há semelhança do *loop* de posição, os valores instantâneos obtidos são recolhidos pelo *resolver*, através da derivação dos valores da posição angular. O ciclo de velocidade assume-se como um controlador PI (Ibrahim, Hamoodi, & Salih , 2020), uma vez que utiliza o ganho proporcional e o ganho integrador para determinar o correto comando a ser enviado ao controlo de corrente.

No controlo PI, o controlo proporcional é expandido com a adição de um termo de controlo. A constante integradora K_i, é multiplicada pelo intergral do termo do erro. Com a adição do termo integral, o sinal de controlo passa a ser:

$$u(t) = K_p.e(t) + K_i.\int e(t)\ dt \tag{45}$$

O controlo PI de velocidade foi implementado recorrendo diretamente a um bloco controlador providenciado pela biblioteca principal, em que os termos são definidos de forma direta.

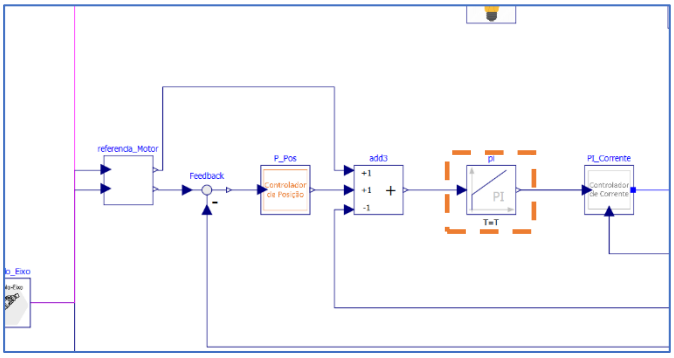


Figura 85 - Modelo do controlo de velocidade com um bloco controlador PI

8.1.5.3. Controlo de Corrente

Para criar um esquema de controlo da corrente, foi elaborado um circuito de amplificadores operacionais, que são análogos aos blocos de controlo *PI – PID*. Esta abordagem para o controlo da corrente permite desde logo mostrar a relação entre a teoria de controlo e circuitos de controlo analógicos.

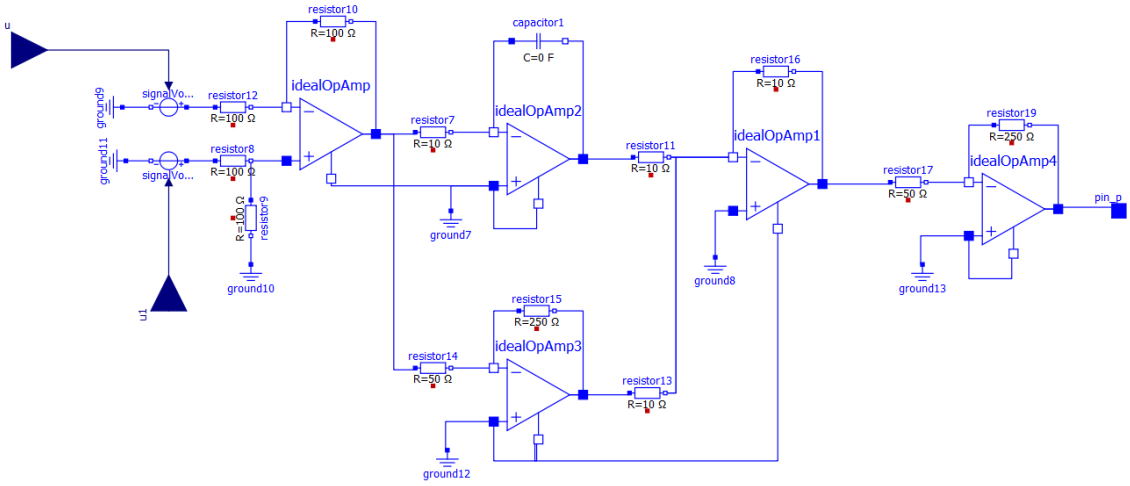


Figura 86 - Circuito com amplificadores operacionais usados no controlo de corrente

O primeiro amplificador operacional admite uma subtração entre o valor proveniente do sensor de corrente (que obtém a amperagem instantânea extraída do motor) e o valor de comando de corrente, proveniente do ciclo exterior de velocidade no sistema de controlo.

Ambos os comandos de corrente que dão entrada no modelo, são um sinal de referência que toma a forma de uma voltagem (dada a proporcionalidade exibida entre corrente e voltagem). É importante referir que, do ponto de vista real e não de modelo, os valores comando transmitidos entre os respetivos controlos de posição, velocidade e corrente, são valores analógicos que se traduzem numa voltagem após serem processados por um conversor digital-analógico (DAC) (Pickering, 2014).

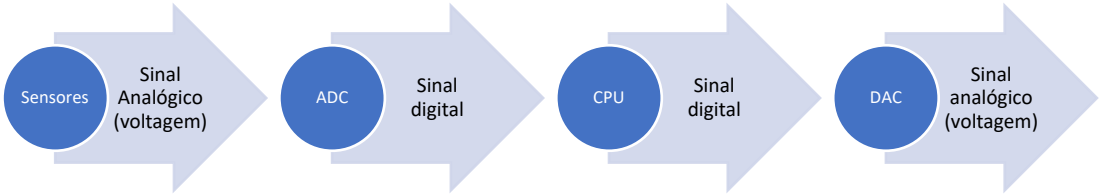


Figura 87 - Processo de obtenção de um comando de referência sob a forma de voltagem

O processo pelo qual um valor de referência sob a forma de voltagem é obtido, vai de encontro como a aplicação de aquisição de dados que os conversores conseguem providenciar a determinados sistemas. Para o caso em estudo, os valores das propriedades a serem medidas são digitalizados por um ADC (conversor analógico-digital) e enviadas posteriormente para um processador. No processador os cálculos de controlo são elaborados e transmitidos depois para um DAC (conversor digital-analógico) permitindo assim que uma referência analógica seja recolhida pelo driver.

Recebida a voltagem de referência, assim como a voltagem recolhida no motor, um processo de feedback poderá ser iniciado e desse modo, também, o controlo da corrente. O circuito de controlo analógico poderá ser dividido em diferentes etapas. Sendo que para uma dessas etapas, existe um amplificador operacional (ROBERGE, 1975) a executar uma tarefa específica.

8.1.5.3.1. Amplificador Operacional Diferencial

Numa primeira instância, é utilizado um amplificador operacional diferencial (ROBERGE, 1975), que estabelece à saída um sinal de voltagem que caracteriza o erro, inerente ao *delay*, entre a voltagem de referência e a voltagem instantânea. À semelhança de um bloco *feedback* num diagrama de controlo por ciclos, este amplificador tem a função de apresentar o erro entre o valor desejado e o valor obtido.

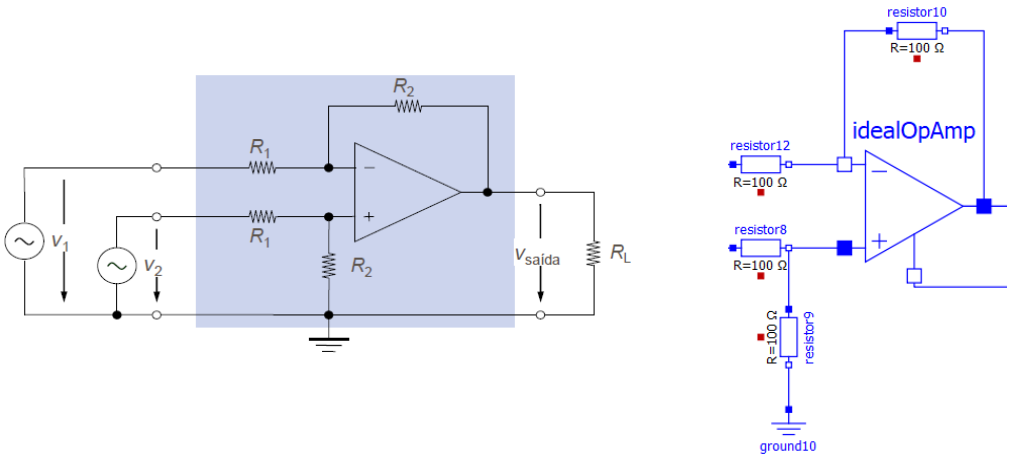


Figura 88 - Amplificador diferencial implementado e nomenclatura associada

Tendo por base as resistências apresentadas na Figura 88, a voltagem de saída define-se da seguinte forma:

$$V_{saída} = V'_{saída} + V''_{saída} = -\frac{R_2}{R_1}V_1 + V_2\frac{R_2}{R_1} = \frac{R_2}{R_1}(V_2 - V_1) \quad (46)$$

Tendo em conta a equação apresentada e os valores atribuídos às resistências, tem-se que a voltagem de saída é igual à diferença entre os dois inputs (o ganho aqui é inexistente uma vez que o valor atribuído às resistências é igual).

8.1.5.3.2. Amplificador Operacional Inversor (proporcional)

É possível recorrer ao uso de um amplificador operacional para o desenvolvimento de um controlador proporcional. O amplificador operacional, também denominado de *op-amp*, segue um conjunto de regras simples: as entradas do amplificador apresentam corrente elétrica praticamente inexistente e a saída é uma voltagem positiva se $V_+ > V_-$, ou uma voltagem negativa caso $V_+ < V_-$. Na Figura 89 é apresentado um *op-amp* em que uma voltagem de entrada se encontra associada à entrada negativa do amplificador, enquanto que a entrada positiva do amplificador se encontra conectada à terra. O amplificador não permite a admissão de corrente nas entradas, pelo que:

$$I_{R1} = I_{R2} \quad (47)$$

E uma vez que à saída do amplificador irá ser apresentada uma qualquer voltagem necessária para permitir que a corrente das duas voltagens flua através de R_2, as quedas de tensão podem ser escritas como:

$$\frac{0 - V_{ent}}{R_2} = \frac{V_{saída} - 0}{R_1} \tag{48}$$

Que com um pequeno rearranjo, obtem-se:

$$V_{saída} = (V_{ent})\frac{-R_2}{R_1} \tag{49}$$

$$\frac{V_{saída}}{V_{ent}} = \frac{-V_2}{V_1} = \frac{-R_2 I}{R_1 I} = -\frac{R_2}{R_1} \tag{50}$$

Visualizando a equação 50, é percetível a capacidade de desenvolvimento de um controlador proporcional com um *op-amp* integrado num circuito elétrico, em que V_{ent} poderá ser considerado o erro $e(t)$, e ainda $R_2/\ R_1 = K_p$.

Este amplificador permite assim que o erro, ou diferença entre valor desejado e obtido, seja submetido a um ganho, correspondendo desta forma ao elemento proporcional de um controlador PI.

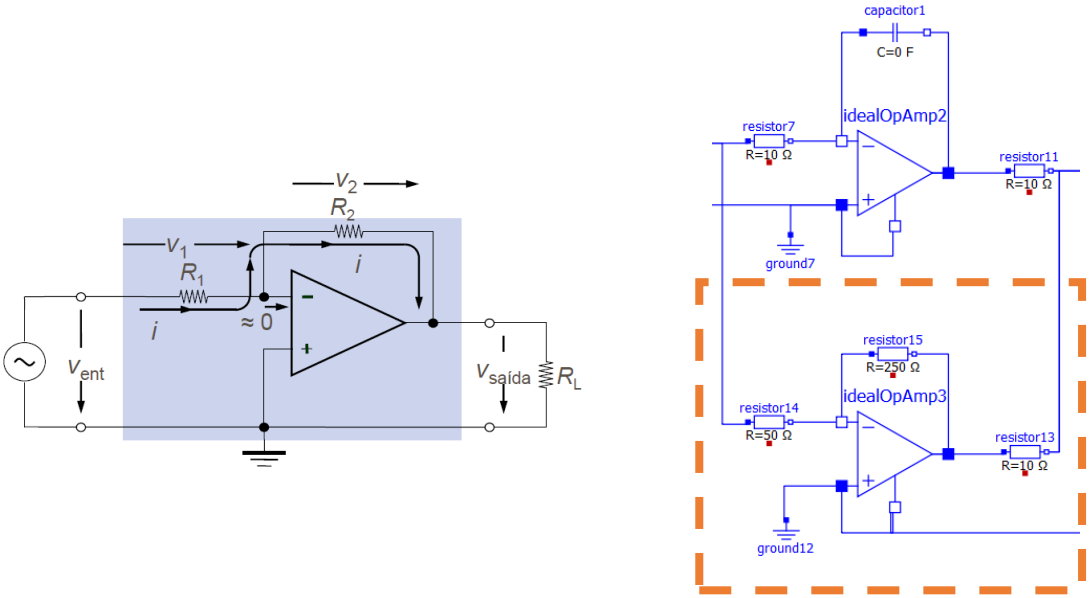


Figura 89- Amplificador inversor proporcional implementado e nomenclatura associada

8.1.5.3.3. Amplificador Operacional Integrador

Amplificador que constitui o elemento integrador do controlador PI, aqui definido sob a forma de amplificadores operacionais. A maneira como o controlo integrador é implementado é através de um *op-amp* como o esquematizado na figura 90. A expressão será agora projetada tendo em mente que a corrente em R é igual à corrente em C :

$$\frac{V_{ent}}{R} = -C.\frac{dV_{saída}}{dt} \tag{51}$$

Isolando $V_{saída}$ passa-se a ter

$$dV_{saída} = \frac{-1}{RC}.V_{ent}\,dt \tag{52}$$

E por fim

$$V_{saída} = \frac{-1}{RC}\int V_{ent}(t)\,dt \tag{53}$$

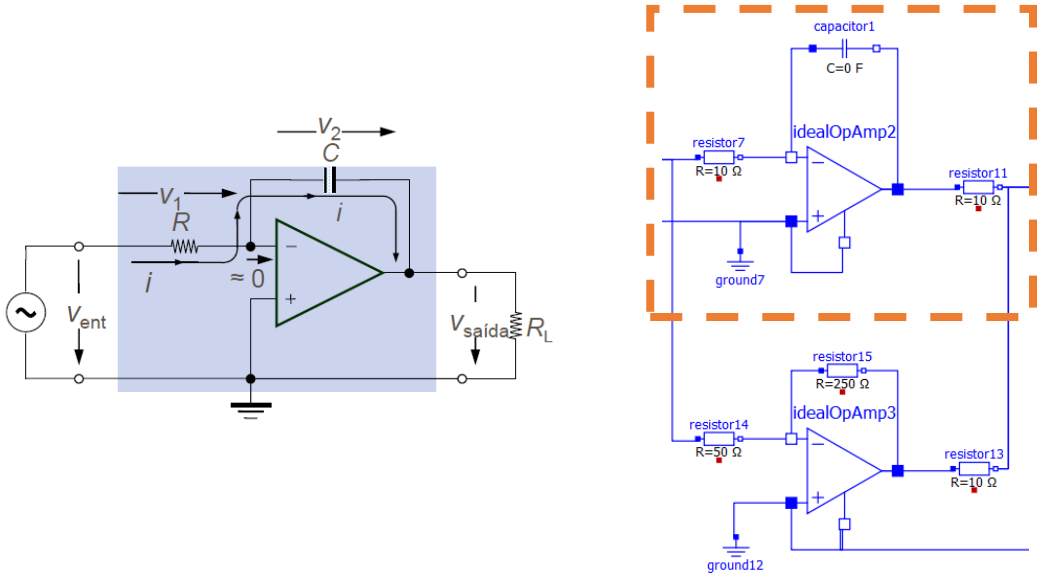


Figura 90 - Amplificador integrador implementado e nomenclatura associada

$$I = \frac{V_1}{R} = \frac{V_{ent}}{R} \tag{54}$$

8.1.5.3.4. Amplificador Operacional Somador

Amplificador que tem o objetivo de "reunir" ambos os elementos do controlador através da sua soma. A soma entre o elemento proporcional e o elemento integrador é assim realizada, sendo que para o efeito, nenhum coeficiente foi adicionado a ambos os termos, ou seja, o valor das resistências associadas é assim igual.

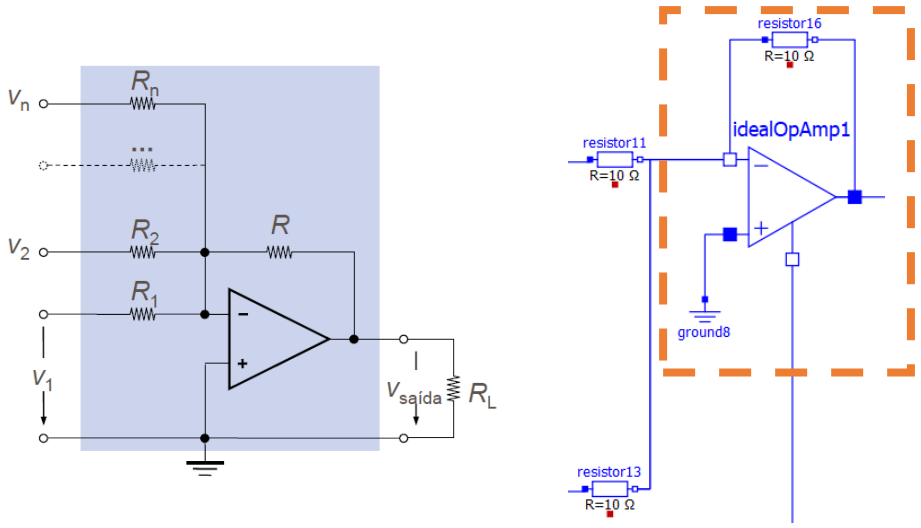


Figura 91 - Amplificador somador implementado e nomenclatura associada

$$V_{saída} = -\left(\frac{R}{R_1}V_1 + \frac{R}{R_2}V_2 + \cdots + \frac{R}{R_n}V_n\right) \quad (55)$$

8.1.5.3.5. Amplificador Operacional Inversor

À saída do driver, surge ainda a preocupação de inverter o sinal, de negativo para positivo, de maneira a ajustar a ordem na diferença de valores realizada à entrada do driver. O presente amplificador, além do ganho que realiza, permite que essa inversão de sinal seja feita.

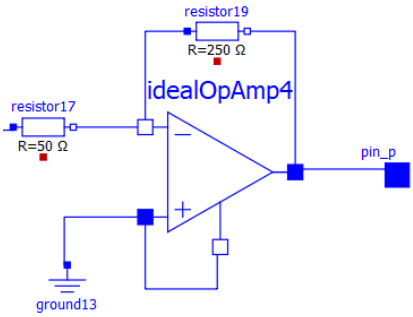


Figura 92 - Amplificador inversor implementado no fim do circuito

8.1.5.4. Atuadores – Servomotores

Um motor elétrico pode ser modelado com um circuito elétrico equivalente, como representado esquematicamente na Figura 93. Neste, uma voltagem de alimentação que se mantém constante é inserida no circuito acompanhada por uma resistência, representativa da resistência interna dos fios de cobre utilizados para criar as espiras. Por outro lado, o indutor é representativo da indutância, descrevendo a tendência dos fios em oporem-se a uma mudança na corrente elétrica que flui sobre eles. Por fim, um componente onde a força eletromotriz é descrita, é inserido no circuito.

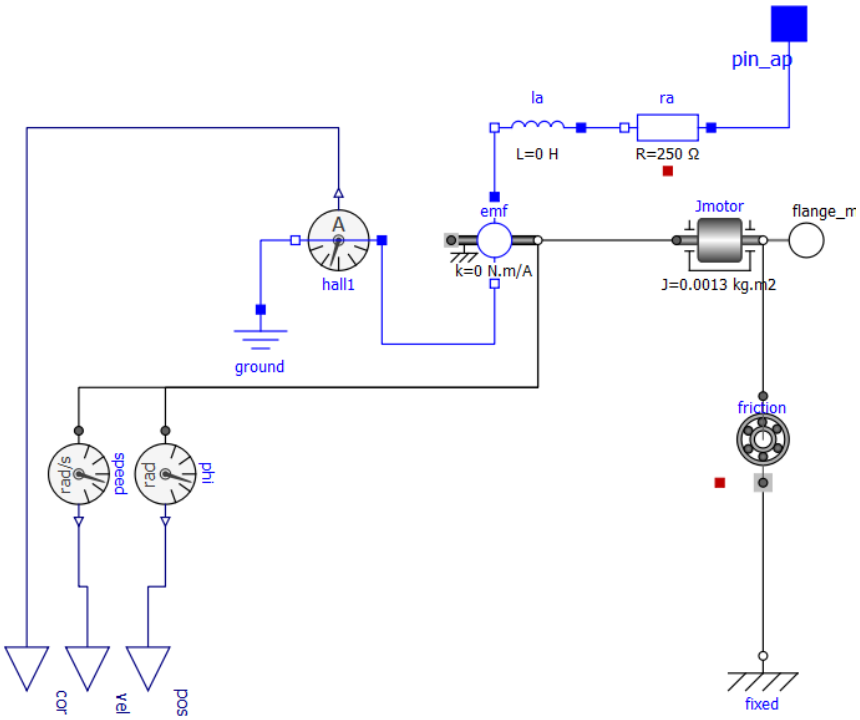


Figura 93 - Modelo do Servomotor

Elaborada a componente elétrica, resta a componente mecânica do sistema em questão. Para o efeito, um amortecedor rotativo foi acrescentado, permitindo absorver e abrandar o movimento rotativo gerado pela força eletromotriz do motor, alterando a vibração, ruído e desgaste dos componentes, alicerçando assim um movimento mecânico mais suave. Além disso, um momento de inércia foi acrescentado permitindo representar a necessidade energética a ser fornecida pela força eletromotriz induzida, de maneira que a rotação do eixo seja constatada. Este parâmetro encontra-se diretamente relacionado com o rotor do motor elétrico.

8.1.5.5. Função Transferência – Servomotor

Analisada a modelação subjacente ao motor, e percebendo a sua importância no processo envolvido nos ciclos de controlo, foi elaborada a função de transferência representativa do servomotor. Esta permite desde logo abordar a forma como o desenvolvimento do modelo é encarado, pois permite a simplificação do processo de controlo. Assim, o modelo do servomotor poderá ser substituído por um bloco que designa a função de transferência, em que todas as equações e variáveis envolvidas nos componentes do circuito elétrico e mecânico se encontram subjacentes. Para o modelo em questão, a função transferência não foi implementada, uma vez que para esta situação em específico não providenciava qualquer benefício, além de que se incompatibilizaria com o controlador de corrente, uma vez que é analógico. No entanto, o estudo foi explorado, permitindo perceber as principais equações envolvidas na conceção do servomotor e do seu funcionamento.

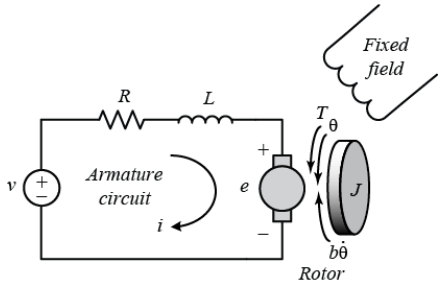


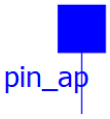
Figura 94 - Circuito elétrico incorporado no servomotor

A função transferência obtém-se através da razão entre a variável de saída a ser estudada e a variável manipulada de entrada no sistema. Deste modo, a função poderia ser definida como

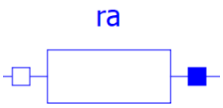
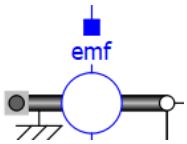
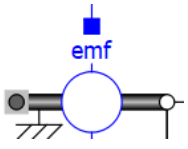
a razão entre a velocidade à saída do eixo do rotor e a voltagem fornecida ao circuito elétrico incorporado na armadura do servomotor, passível de ser observado na Figura 94.

O conjunto de equações e parâmetros, envolvidos em cada um dos subsistemas do motor poderão ser encontrados na tabela.

Tabela 9 - Equações envolvidas nos diferentes domínios do servomotor

<table>
<tr><th>Componente</th><th>Variável/ Parâmetro</th><th>Equilíbrio/Equação</th></tr>
<tr><td> 
</td><td>E_a</td><td rowspan="4">Sistema Elétrico

$E_a - R_a I_a - L_a S I_a - V_b = 0$ (56)

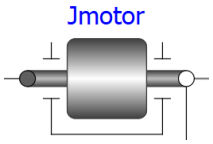
E_a – Voltagem fornecida
R_a – Resistência
L_a – Indutância
V_b – Voltagem no motor em si mesmo</td></tr>
<tr><td> 
</td><td>R_a</td></tr>
<tr><td> 
</td><td>L_a</td></tr>
<tr><td> 
</td><td>V_b</td></tr>
<tr><td> 
</td><td>V_b</td><td>Constantes do Motor

$T_m = K_t I_a \rightarrow I_a = \frac{T_m}{K_t}$ (57)
$V_b = K_b \dot{\theta}_m \rightarrow V_b = K_b S \theta_m$
$E_a - R_a \left(\frac{T_m}{K_t}\right) - L_a S \left(\frac{T_m}{K_t}\right) - K_b S \theta_m = 0$

T_m – Torque do Motor

K_t – Constante de Torque

K_b – Constante da força eletromotriz do Motor</td></tr>
</table>

	J	**Sistema Rotativo** $T_m - D\dot{\theta}_m = J\ddot{\theta}_m$ $T_m = J\ddot{\theta}_m + D\dot{\theta}_m$ $T_m = JS^2\theta_m + DS\theta_m$ $T_m = \theta_m(JS^2 + DS)$ (58)

Com base nas equações presentes na tabela, mais especificamente a substituição em (56) de (57) e (58), a equação 59 poderá ser escrita.

$$E_a - \theta_m(JS^2 + DS)\left(\frac{R_a + L_aS}{K_t}\right) - K_bS\theta_m = 0 \quad (59)$$

Sabendo que $S\theta_m = w_m$ e que $K_t = K_b$, então a formulação final da função transferência dá-se do seguinte modo

$$\frac{w_m}{E_a} = \frac{KS}{(JL_aS^3 + (JR_a + DL_a)S^2 + (DR_a + K^2)S)} \quad (60)$$

O estudo da função transferência obriga assim a um entendimento e análise minuciosa do sistema em causa, despontando uma maior clarividência para as opções de controlo e modelação envolvidas. Quando a "planta" ou objeto de controlo é conhecido com todos os seus parâmetros já bem ressalvados, e o estudo se foca mais nas variáveis de saída e não necessariamente na manipulação das mesmas, a conceção de uma modelação com uma função transferência apresenta-se como vantajosa.

8.1.5.6. Transmissão

Compreendido entre o eixo à saída do motor e o eixo de cada uma das juntas, foram modelados os elementos de transmissão. O primeiro elemento caracteriza uma caixa de velocidades que possui um eixo à entrada que aciona e um eixo acionado à saída. As engrenagens envolvidas são ideais, em que a inércia, elasticidade, amortecimento e *backlash* são desconsiderados.

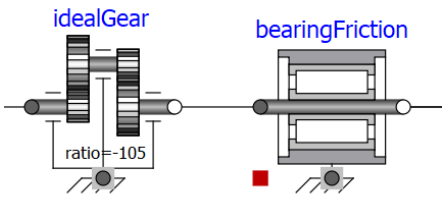


Figura 95 - Componentes de transmissão utilizados

8.1.5.7. Modelos Auxiliares

De maneira a manter um correto funcionamento do controlo das diferentes variáveis medidas nos atuadores, é relevante que a discrepância entre a variação angular ao longo do eixo do rotor dos servomotores e a real variação angular verificada ao longo de cada eixo no braço, seja combatida. Falando especificamente da posição e velocidade angular, estas duas propriedades têm que ser devidamente ajustadas com o índice de transmissão presente nas engrenagens de transmissão, e dessa forma gerar uma diferença nos diferentes feedbacks com valores inseridos na mesma escala operacional.

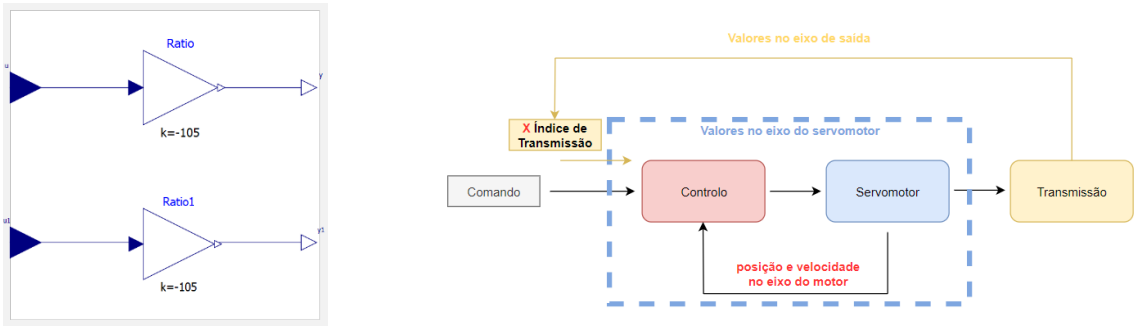


Figura 96 - Ajuste no índice de transmissão, proporcionando um correto funcionamento do controlo

Na transmissão, os valores das propriedades em estudo, sofrem uma redução e adotam valor negativo, ou seja, o sentido de rotação é invertido à saída das engrenagens. Essa redução e inversão de sentido é equilibrada, com a multiplicação dos valores de referência de posição e velocidade pelo índice de transmissão à entrada do controlador e dos respetivos ciclos de controlo, tendo sido desenvolvido um modelo para o efeito.

Associada à flange de saída do modelo de controlo axial encontra-se um componente responsável por atribuir uma posição angular inicial à flange.

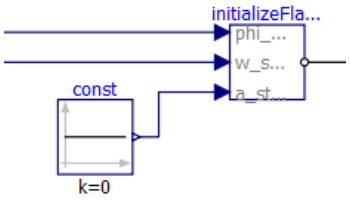


Figura 97 - Componente *initializeFlange* responsável por atribuir valores de posição, velocidade e aceleração à flange de saída do eixo

Este componente permite assim que a posição inicial (definida) do robô seja garantida.

Associado ao eixo, existe ainda um modelo com uma resistência, simulando dessa forma uma pequena lâmpada de sinalização de movimento, que se liga quando o comando de movimento é transmitido. Desta forma, a deteção de uma avaria física do processo sai facilitada, a ausência de movimento por parte do eixo quando a luz se encontra ligada permite assim identificar uma possível anomalia no sistema físico.

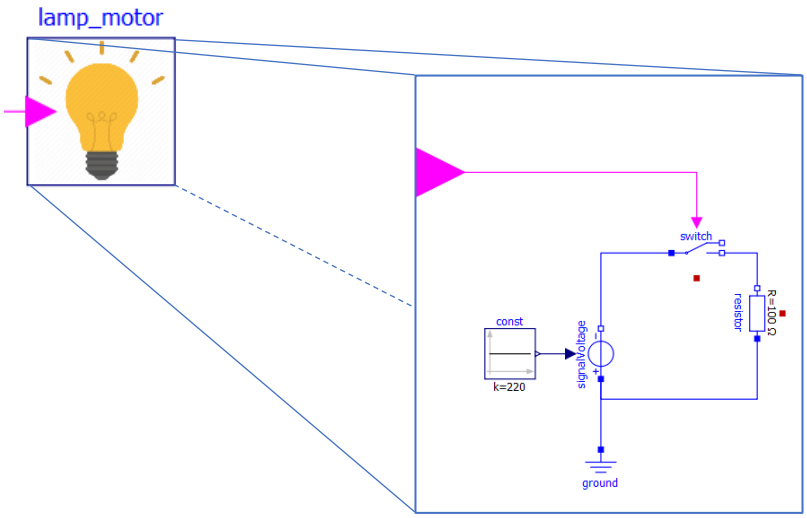


Figura 98 - Modelo que replica uma lâmpada de sinalização de movimento no eixo

8.1.6. Estrutura Mecânica

De forma a simular apropriadamente o sistema, uma estrutura mecânica do braço robótico a executar a tarefa *Pick and Place* foi elaborada. A nível estrutural, o objetivo inicial seria a modelação tendo por base o perfil físico do robot *ABB IRB 140* (IRB 140 - ABB, s.d.), sendo que em ambiente de simulação cada um dos elos modelados em *Modelica* adotaria a forma real dos diferentes componentes constituintes do *IRB 140*, em compatibilidade com os respetivos componentes modelados num *software* CAD. No entanto, o processo de compatibilidade que permitisse a transcrição dos diferentes componentes fracassou, muito por força das

limitações oferecidas pelo programa *open-source OpenModelica*. Nesse sentido, houve um esforço em modelar cada um dos elos do robot, de raiz, em *OpenModelica*, com base, de forma exclusiva, nas formas geométricas providenciadas pelo programa (cilindros e paralelepípedos). O resultado encontra-se na Figura 99.

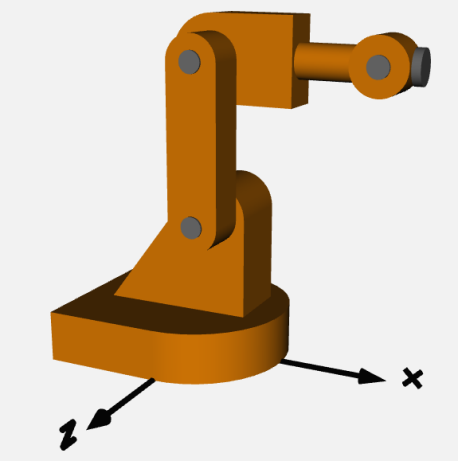
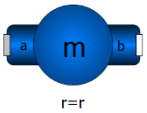
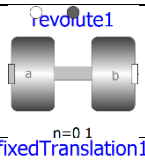
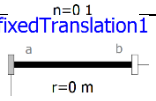
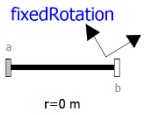
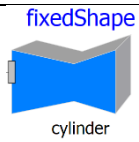


Figura 99 - Representação em simulação da estrutura mecânica modelada

A obtenção da estrutura total foi conseguida através de um modelo constituído essencialmente por "bodyshape's" que retratam cada um dos elos de ligação ou componentes estruturais. Nestes, a massa, o centro de massa e a forma (altura, largura e comprimento) são definidos, com recurso a, fundamentalmente, vetores. Entre cada um dos elos, juntas rotativas ou "*revolute's*" foram modelados, constituindo as juntas cujos eixos serão respetivamente acionados pelos servomotores. A fixação das juntas num local específico do elo é feito recorrendo aos componentes "*fixedTranslation*" ou "fixedRotation", que acolhem o vetor entre a extremidade final do elo e o início da junta, e vice-versa. Para complementaridade da forma física do modelo, foram ainda adicionadas formas físicas através de "*fixedShape's*".

Tabela 10 - Componentes utilizados para a conceção da estrutura mecânica

Componente	Ilustração
BodyShape	

Revolute	
FixedTranslation	
FixedRotation	
FixedShape	

Cada um dos eixos, provenientes da secção da modelação destinada ao controlo axial, dá entrada no modelo da estrutura e conecta-se aos *revolutes* acionando desse modo as respetivas juntas, sempre que assim for requerido.

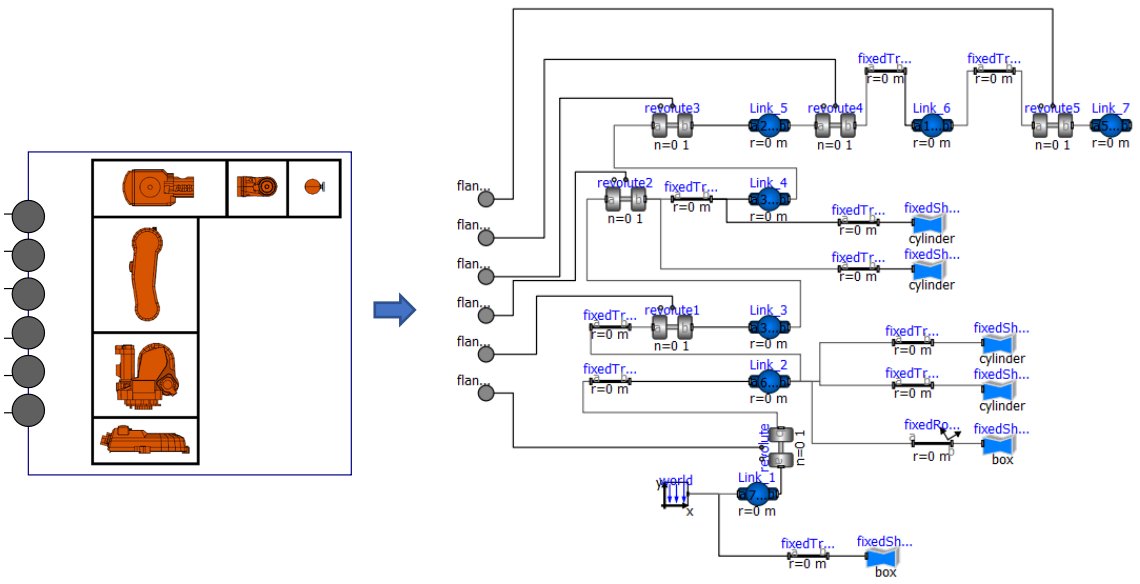


Figura 100 - Modelo da estrutura mecânica do manipulador

8.1.6.1. Conexão Robot – Gripper

Uma tentativa de conexão robot – Gripper foi explorada com alguma exaustão mas sucessivamente fracassada, em que uma ligação por *frames* entre a estrutura mecânica do robot e a estrutura mecânica dos 3 dedos constituintes do *gripper* foi modelada. A modelação apresentou erros sucessivos devido a um sobre carregamento no sistema. Desta forma, o *gripper* foi simulado de forma independente, do ponto de vista estrutural, do manipulador. No entanto, para não comprometer o estudo da performance do manipulador, um ponto de massa foi acrescentado na extremidade da estrutura do manipulador, tentando assim replicar a massa da presença do *gripper* ao longo da estrutura.

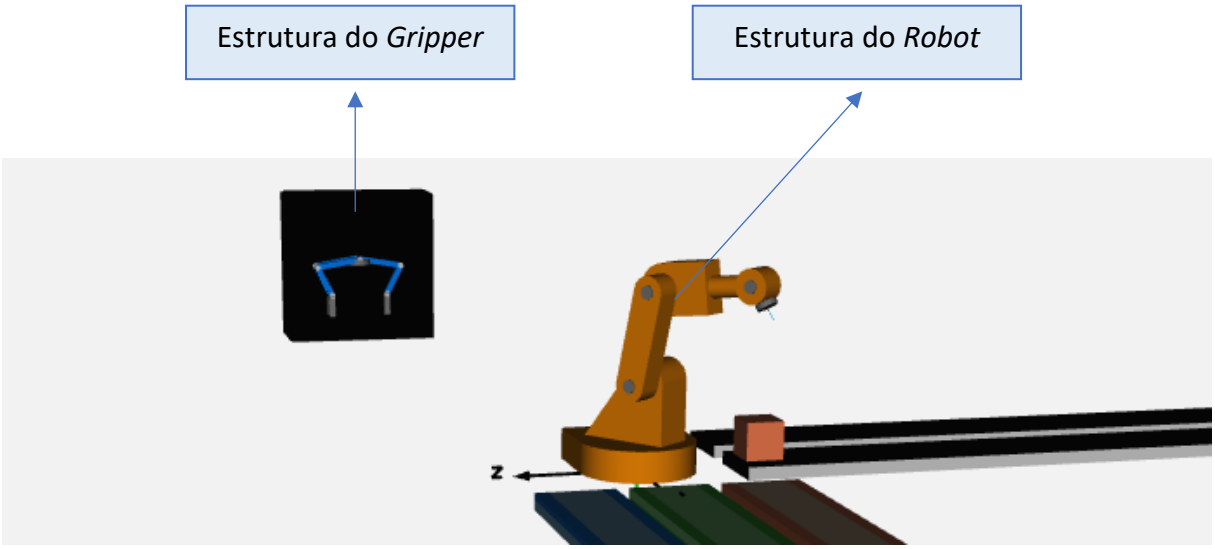


Figura 101 - Estrutura do Gripper modelada independente do manipulador, do ponto de vista estrutural

Na Figura 102 é possível observar o bloco correspondente à massa adicional que visa estudar, como já foi mencionado, a replicação da presença do *gripper* na extremidade da estrutura do *robot*.

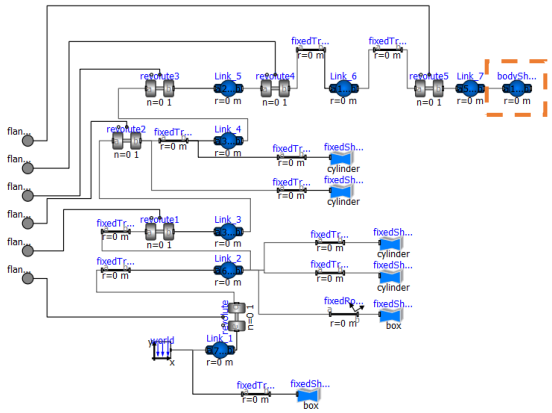


Figura 102 - Anexação da massa do *gripper* na estrutura do manipulador

Pode-se concluir que, e como se torna claro através de uma constatação gráfica, um aumento de massa aplicada na extremidade da estrutura implica um aumento de torque aplicado ao longo de cada um dos eixos do robot.

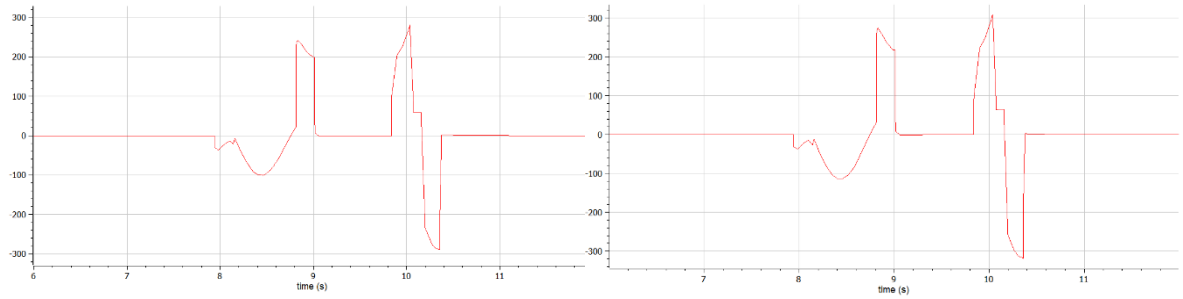


Figura 103 - Amplitude do torque ao longo do tempo é aumentada em cada eixo. Gráfico da esquerda apresenta um torque para 5kg de massa do *gripper*. Gráfico da direita apresenta um torque para 10kg de massa do *gripper*

8.1.7. Fluxo Térmico -Eixos do Motor

No sentido de tentar replicar eventuais perdas térmicas ao longo do funcionamento do robot, um modelo de transferência de calor entre um *port* de entrada e o exterior foi desenvolvido. A transferência de calor para o exterior ocorre sob efeito convectivo, e poderá ser representado através da seguinte fórmula.

$$\dot{Q} = h.A.dT$$

Em que $\dot{Q}$ corresponde à transferência de calor $[W]$, h e A correspondem respetivamente ao coeficiente de transferência de calor $[W/m^2°\mathrm{C}]$ e a área total $[m^2]$ sobre a qual a transferência ocorre, e por fim a diferença de temperatura verificada entre os 2 meios $[°\mathrm{C}]$.

O modelo desenvolvido visa o estudo da transferência de calor por convecção, da superfície em estudo, com uma capacidade calorífica própria, para o exterior que admite uma temperatura constante ao longo do tempo. No presente projeto, a utilização deste modelo permitiu o estudo das perdas térmicas no motor, assim como nas perdas por fricção a longo do componente destinado para o efeito, que se encontra na secção de transmissão.

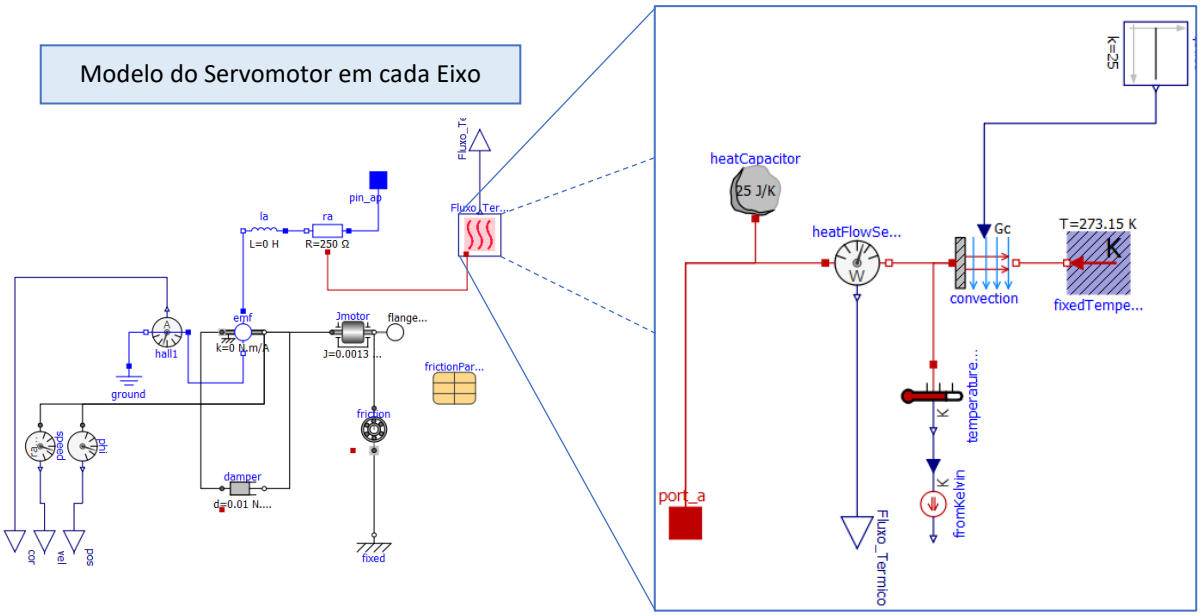


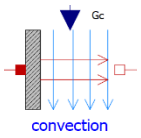
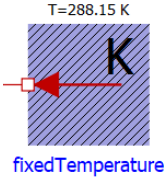
Figura 104 - Modelo de fluxo térmico associado a cada eixo

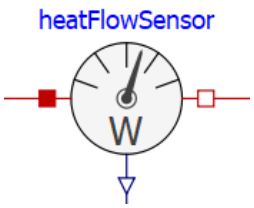
A possibilidade de utilização de um componente representativo de transferência sob a forma de condução, antes da modelação da convecção, replicaria possivelmente uma situação mais realística do processo a ser aqui modelado, no entanto, o facto de o componente de transferência por condução apresentar uma temperatura padrão (20 graus) diferente da temperatura inicial ambiente a ser considerada para o rolamento e motor (15 graus), implica uma transferência de calor inicial para colmatar essa diferença de temperaturas ao longo da secção de transferência. A transferência de calor em função do funcionamento do sistema seria pouco percetível, uma vez que se estaria perante um outro fator para que a mesma ocorresse.

Na seguinte tabela 11 será possível observar de que forma os conceitos abordados, de transferência por convecção, assim como da capacidade calorífica, se incorporam na modelação apresentada na Figura 104. Para o efeito um conjunto de componentes foram utilizados e cuja respetiva descrição será prontamente apresentada.

Tabela 11 - Componentes presentes no modelo de fluxo térmico

Componente	Descrição
	Componente que evidencia a capacidade calorífica C, ou seja, a quantidade de calor fornecida para que a temperatura

	suba uma unidade, considerando uma temperatura constante do corpo ao longo de todo o seu volume. A capacidade calorífica acaba por ser obtida através da multiplicação do calor específico c_p pela massa do corpo m. Na modelação recorrendo a este componente, o estabelecimento de C é um parâmetro definido de forma direta, pelo que um conhecimento inerente do corpo a ser estudado deve existir, tais como o material do qual o corpo é constituído assim como a massa e/ou volume do mesmo. $$C = m.\ c_p$$
	Componente que define o estado convectivo da transferência, mais concretamente o coeficiente de transferência de calor e a área total de transferência, como evidenciado na equação. Para o efeito o parâmetro Gc é definido em que o mesmo se determina com base na multiplicação da área pelo coeficiente. $$Gc = h.A$$
	Componente que define a temperatura exterior e cuja diferença para com o corpo gera o potencial térmico necessário para que a transferência ocorra.

	Componente responsável pela medição do calor transferido entre o objeto de estudo e o exterior. Este encontra-se ligado à saída do modelo.

Para a situação do rolamento, pode-se visualizar em baixo a diferença de temperatura dT entre este e a temperatura ambiente (que adota o valor de 15 graus). Quando ao longo do processo essa diferença de temperatura aumenta, é possível observar que esse aumento é acompanhado também pelo aumento da transferência de energia sob a forma de calor, cujos valores são obtidos pela multiplicação da capacidade calorífica do rolamento.

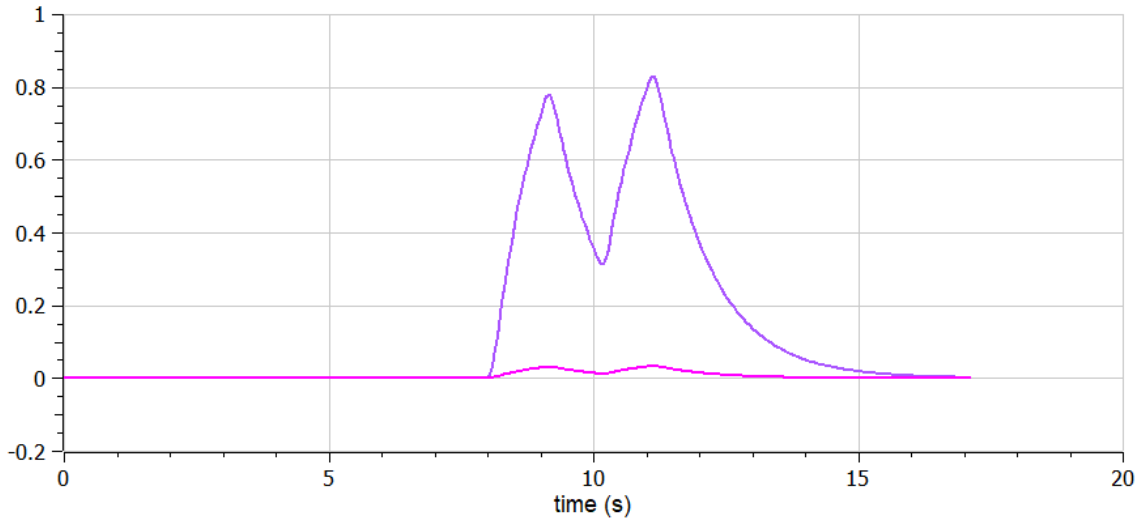


Figura 105 - Variação da temperatura e consequente aumento da transferência de energia

Numa fase inicial, quando o rolamento se encontra em repouso, a transferência de calor é nula, uma vez que o rolamento, antes de ser acionado, tem a sua temperatura igualada à temperatura ambiente. Quando o movimento de *Pick* é iniciado, e dessa forma o movimento do rolamento, o aumento de calor devido à fricção eleva a transferência de calor até o movimento do braço cessar e o estágio de funcionamento do *gripper* ser inicializado, resultando numa queda na transferência, dada a estabilização nas diferenças de temperatura. Quando o movimento *Place* é iniciado, tem-se um aumento da transferência novamente e o processo repete-se. De salientar o facto, que para esta situação em específico, embora o movimento de *Place* tenha sido de menor duração relativamente ao *Pick*, é neste que é atingido o pico de transferência de calor. Isto explica-se pelo facto, de que o aumento da

transferência aquando do *Place* é precedido por um estado do rolamento em que, embora parado, ainda apresenta um desvio de temperatura devido à *performance* no *Pick.* Há assim a necessidade de uma pausa considerável no funcionamento do manipulador, caso se pretenda que os valores da temperatura retomem os valores ambiente.

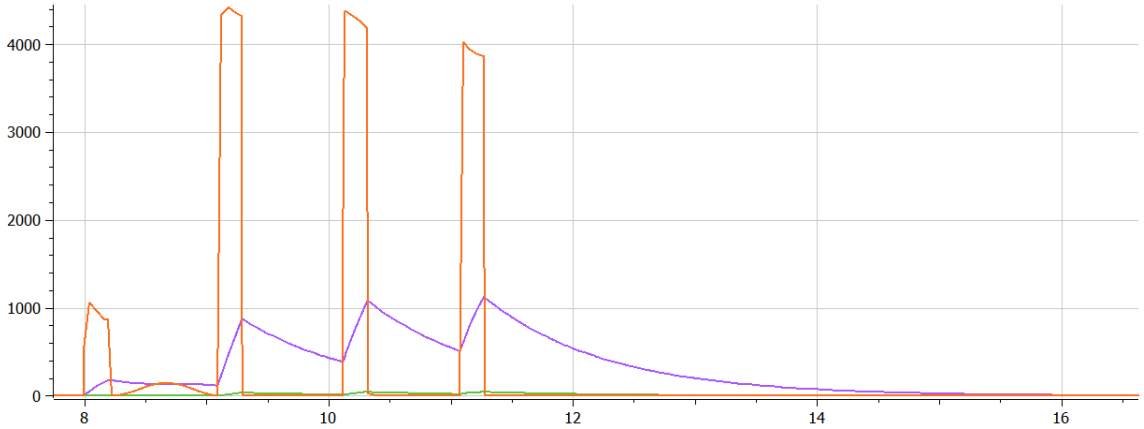


Figura 106 - Transferência de calor (roxo) proveniente do motor (eixo 1)

Relativamente à transferência térmica que advém do motor, é possível observar que a mesma se regista nos momentos de aceleração e desaceleração do motor (a laranja). Nos momentos de aceleração o influxo de corrente e voltagem aumenta. Da mesma forma, perante os instantes de desaceleração, o valor de corrente aumente novamente, desta feita em sentido contrário. Aquando dos picos de corrente, observa-se naturalmente variações nos valores da temperatura, que por sua vez desencadeiam uma transferência de calor por convecção.

As perdas pelo fenómeno de transferência de calor em cada eixo poderão ser observadas no seguinte gráfico.

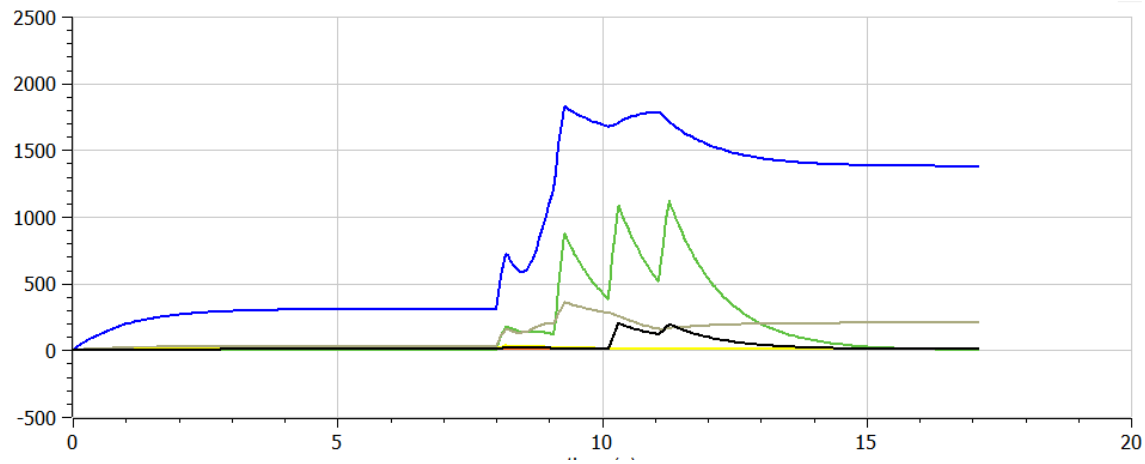


Figura 107 - Transferêmcias térmicas envolvidas nos 6 eixos

As perdas térmicas são calculadas individualmente em cada um dos eixos, no entanto de maneira a facilitar a consulta das mesmas, um modelo foi concebido com o propósito de canalizar os diferentes valores e apresentá-los num só modelo com indicadores dinâmicos.

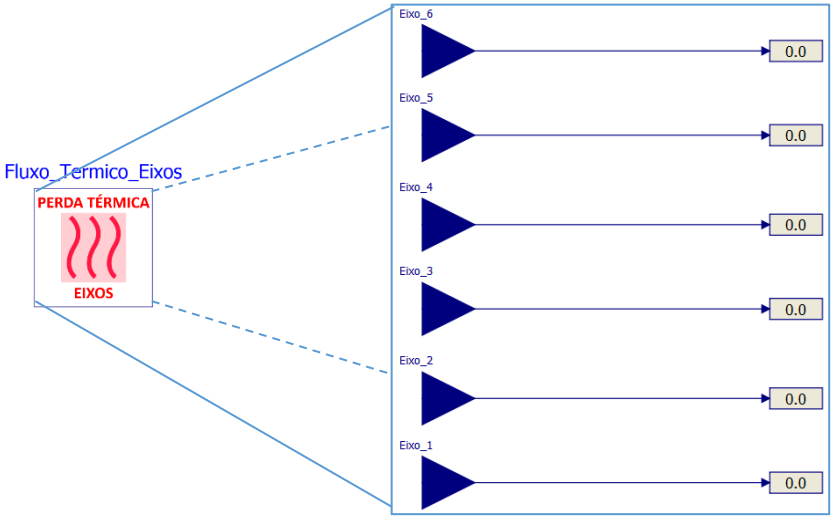


Figura 108 - Modelo que apresenta as perdas térmicas em todos os eixos

8.1.8. Continuação do Estudo de Simulações – Robot

Alguns resultados derivados das simulações e decorrentes do manipulador são aqui apresentados.

Relativamente à variação dos ângulos associados a cada um dos eixos do manipulador, é observado no gráfico que os eixos 1, 3 e 5 têm uma variação angular em *Pick* e uma variação em sentido oposto ao longo do *Place*. O eixo 4 não chega a ser acionado ao longo do transporte *Pick&Place* uma vez que as configurações requeridas para o percurso total assim não o exigem. O eixo 6 apresenta uma variação exclusiva no movimento *Place* e o eixo 2 possui a particularidade de ser acionado duas vezes no mesmo sentido.

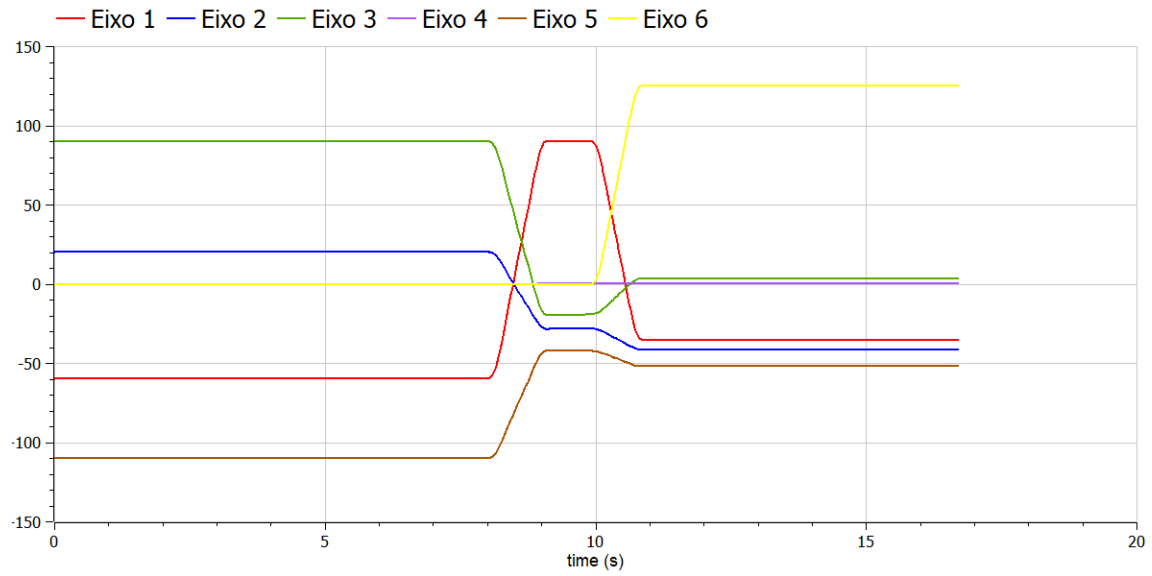


Figura 109 - Variação angular nos 6 eixos, com perfil de velocidades trapezoidal e sem modo *Via Point* – Tapete de Entrada 2 e Embalagem de Transporte 1

O perfil trapezoidal de velocidades pode ser observado (110). O tempo requerido para cada perfil trapezoidal efetuado é igual, como já anteriormente referido, permitindo que cada um dos perfis se iniciem e acabem nos mesmos instantes. Este tempo de percurso requerido equivale ao tempo proveniente do eixo que demora mais tempo a completar o perfil trapezoidal tendo em consideração o deslocamento angular para esse eixo e ainda as restrições cinemáticas axiais implicadas. Apenas deste modo se garante que todos os eixos sejam capazes de completar o perfil trapezoidal no mesmo intervalo de tempo. A consequência direta desta imposição é que apenas um eixo atinge a velocidade máxima (o eixo cuja razão entre o deslocamento angular e velocidade máxima definida seja maior), sendo que os restantes eixos apresentam um pico de velocidade menor do que a velocidade máxima definida de forma indivual. Para a situação apresentada no gráfico (111) , apenas o eixo 1 adota a sua velocidade máxima definida cujo valor é 3 rad/s.

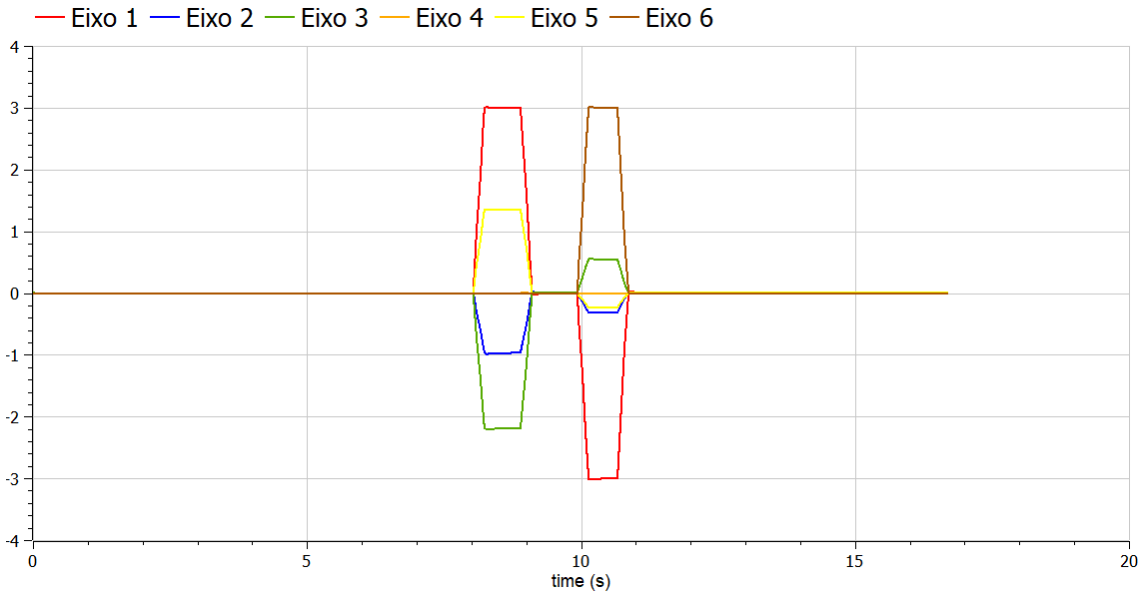


Figura 110 – Perfil de velocidades trapezoidais nos 6 eixos, sem modo Via Point – Tapete de Entrada 2 e Embalagem de Transporte 1

Na Figura 111 é possível observar os 3 perfis de velocidade calculados para o eixo 1 do manipulador. O perfil de velocidade trapezoidal apresentado no gráfico foi extraído dos sensores de posição presentes no eixo (ou seja, foi o perfil implementado) enquanto que os restantes 2 perfis foram calculados sem implementação através do modelo de geração de trajetória.

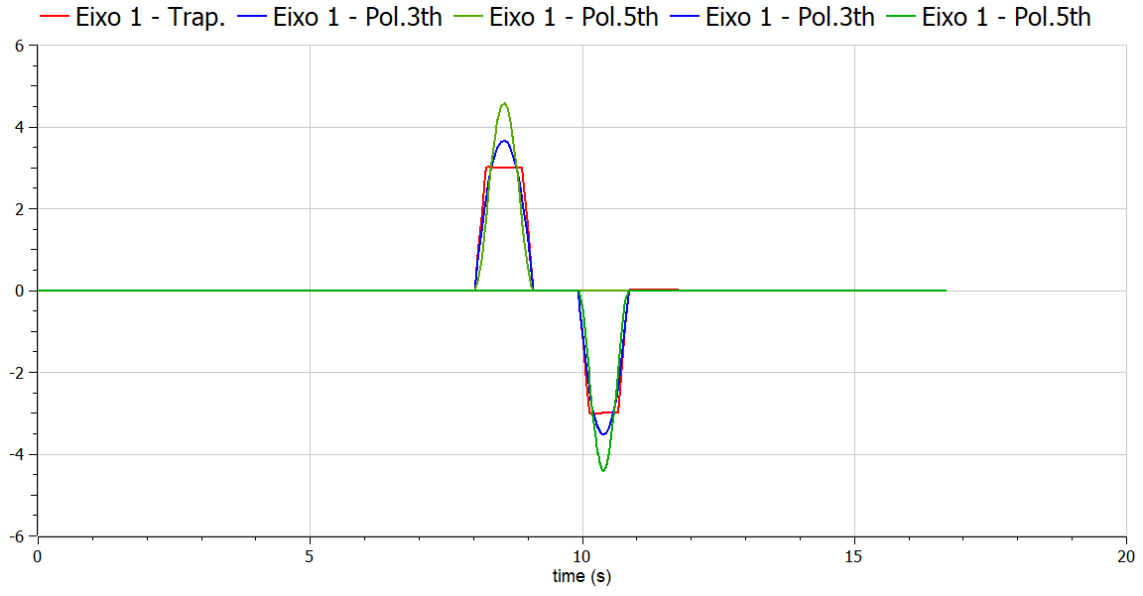


Figura 111 - Perfil de velocidade trapezoidal, polinomial de 3ª ordem e polinomial de 5ª ordem, no eixo 1

Relativamente ao modo *Via Point,* é possível concluir no gráfico presente da Figura 112, que apenas 3 eixos (eixo 2, 3 e 5) estão envolvidos num movimento angular de aproximação e posteriormente um movimento angular de recolha/ deposição, garantindo um perfil

trapezoidal “duplo” tanto no *Pick* como no *Place*. Por outro lado, o eixo 1 e 6 têm o seu movimento angular exclusivamente envolvido na aproximação, tanto no *Pick* como no *Place*, sem a apresentação de um perfil trapezoidal “duplo”.

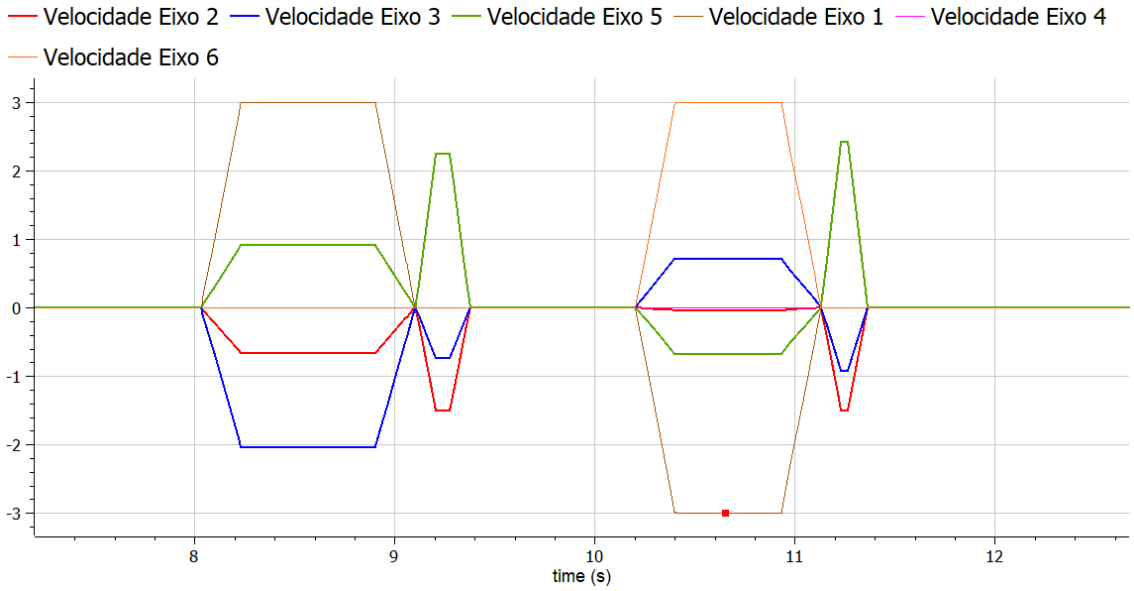


Figura 112 - Perfil de velocidade trapezoidal nos 6 eixos, com modo *Via Point*

Como na fase entre o ponto de aproximação e recolha/deposição apenas 3 eixos iniciam movimento angular, o eixo que atinge a sua velocidade máxima definida muda (passa-se a ter o eixo 2 em detrimento do eixo 1, atingindo uma velocidade máxima de 1.5 rad/s).

Quanto ao impacto do perfil de movimento na *performance,* a ordem obtida do menor para o maior gasto térmico ao longo do funcionamento é Polinomial de 3ª Ordem -> Trapezoidal -> Polinomial de 5ª Ordem. O facto de o polinomial de 3ª ordem apresentar menos gasto que o trapezoidal explica-se pelo movimento mais suave do motor. Por outro lado, um maior gasto que o trapezoidal, por parte do polinomial de 5ª ordem, advém do facto de apresentar acelerações mais elevadas, o que obrigatoriamente se traduz num maior gasto elétrico e térmico. Assim, aquando da escolha de um perfil de movimento do ponto de vista de gasto térmico, a escolha deve recair sobre um perfil polinomial em que um equilíbrio entre a suavização de movimento e as acelerações seja constatado.

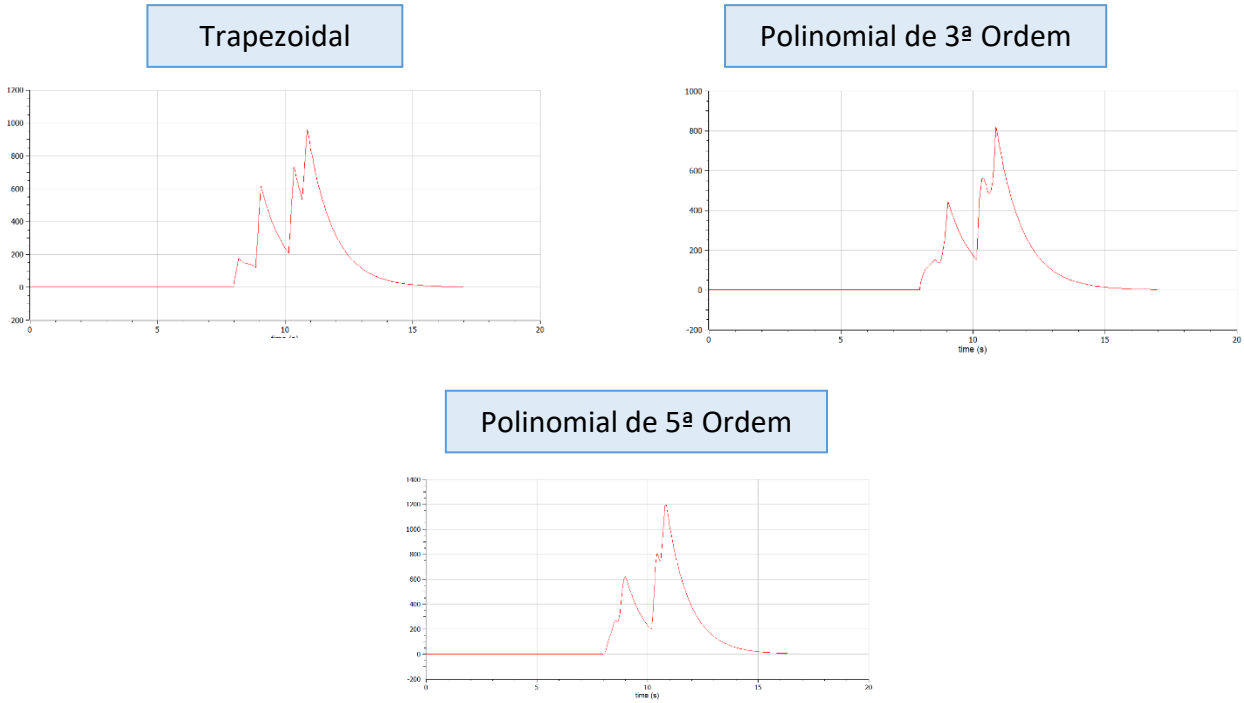


Figura 113 - Perdas térmicas (watts) ao longo do 1ºeixo, em função do perfil de movimento escolhido

Na Figura 114 é possível observar o resultado obtido através do controlador de corrente, em que a vermelho é possível observar a oscilação da diferença de voltagem, entre o valor de referência e obtido no motor, que dá entrada no controlador. Por outro lado a azul é possível observar a resposta dada, ou seja, o valor de corrente enviado ao motor para que as oscilações observadas sejam colmatadas. Existem duas instâncias com 2 picos de corrente em sentido contrário, isto é explicado pelo aumento (fase de aceleração) e diminuição (fase de desaceleração) de velocidade ao longo do *Pick* e do *Place* respetivamente.

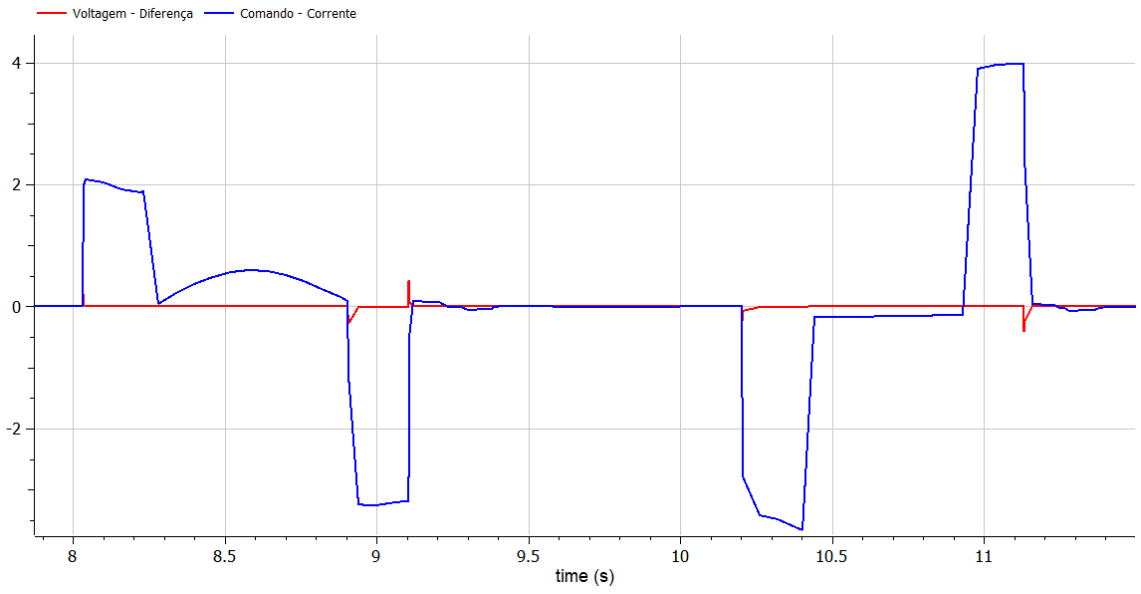


Figura 114 - Controlo de corrente. Diferença de voltagem à entrada e comando de corrente à saída

Caso se foque na diferença de valores entre os valores de corrente desejados (comando) e os medidos diretamente no motor, ao longo do controlador PI de corrente, rapidamente se apercebe que a diferença apenas se torna significativa quando a aceleração de movimento requerida varia.

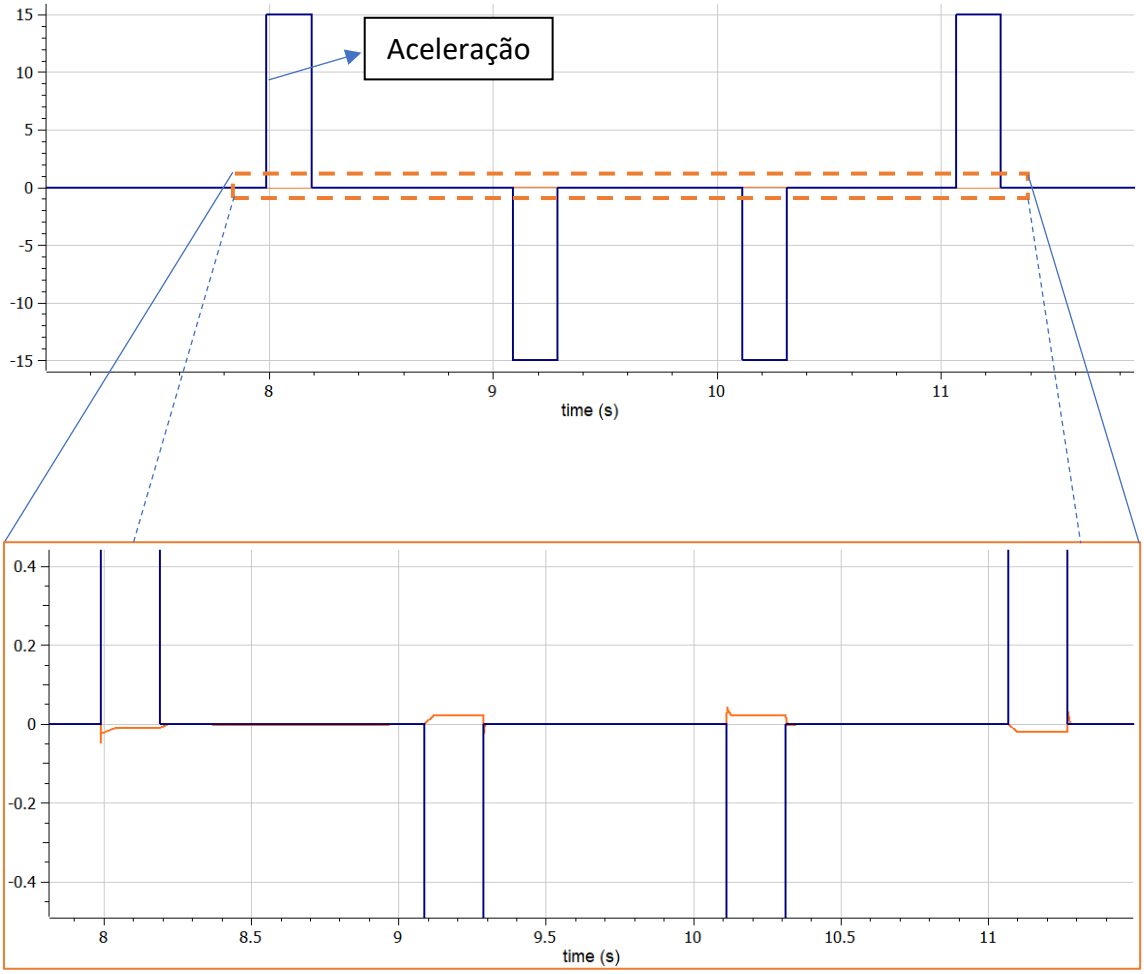


Figura 115 - Diferença de voltagem desejada e medida, no controlador de corrente, diretamente relacionada com a aceleração

Pode-se assim concluir que, este bloco de controlo de corrente, atua fundamentalmente para controlar o movimento ao longo de estágios de aceleração não nulos. Rapidamente se percebe que isto acontece porque precedido deste bloco existem 2 blocos, cada um deles para controlo de posição e velocidade, mas nenhum deles se destina ao controlo da variação da velocidade ao longo do tempo. As diferenças de voltagem calculadas assim à entrada deste bloco surgem na sequência dessa diferença estabelecida, e um controlo por amplificadores

operacionais permite que uma resposta, a nível de corrente a ser fornecida ao motor, seja executada.

8.2.Tapetes Transportadores

Para os diferentes tapetes transportadores utilizados ao longo do processo, o tipo de transportador escolhido foi o transportador de cintos.

Os diferentes tapetes presentes no sistema apresentam, na sua globalidade, uma estrutura de modelação bastante próxima entre si.

Figura 116 - Estrutura de modelação base, em cada um dos tapetes transportadores

A pertinência do desenvolvimento de modelos relativos a tapetes com tarefas divergentes, permitiu, sem perder foco na estrutura base de modelação, refinar cada uma das fases, de forma individual, sempre que necessário.

As diferentes fases integradas na estrutura de modelação dos tapetes serão descritas.

8.2.1. Definição do Percurso

É importante salientar que o percurso do movimento dos tapetes, ao longo do tempo, foi projetado de maneira a transportar a caixa apenas ao longo de um ciclo de funcionamento. De uma outra forma, a interpretação e modelação do funcionamento dos tapetes seria muito provavelmente diferente. É ao longo desta fase que a posição/velocidade de referência, referente ao percurso da caixa ao longo do tapete, são definidas, sendo que estas diferenciam-se consoante a tarefa específica desempenhada pelo transporte. Assim, duas formas foram consideradas para a geração de trajetória: geração de uma trajetória “trapezoidal”, onde se encontram incorporados os tempos de arranque, de movimento linear uniforme e ainda paragem, condicionados pelas constrições cinemáticas desejadas; e geração de trajetória com arranque e movimento linear uniforme concebida por pulsos ou sinais *step.* Sendo esta última um manifesto da tentativa de replicar tapetes de deposição à saída do sistema, em que a paragem no movimento de transporte não chega a ser verificada.

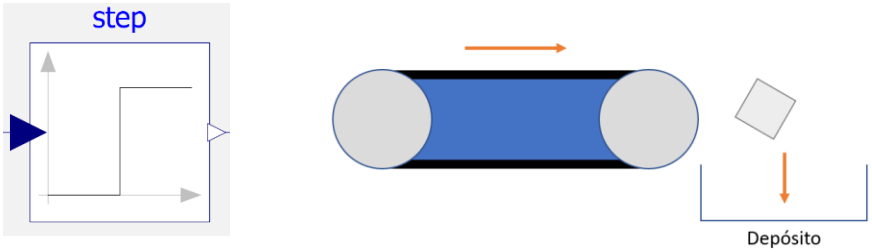


Figura 117 - Tapete transportador que visa o depósito da embalagem

O facto de se debruçar apenas sobre um ciclo de operação, e a necessidade de colocar a embalagem num local específico do transportador não existir, faz com que um estágio de desaceleração ou paragem seja desprezada na definição do percurso, desencadeando assim um movimento linear uniforme do tapete que se extingue, exclusivamente, aquando da interrupção da simulação.

Por outro lado, na exploração dos percursos trapezoidais, dois tipos de percursos foram considerados: percurso com perfil trapezoidal único; e ainda percurso com perfil trapezoidal cíclico.

Os tapetes transportadores à entrada, recolhem uma embalagem e transportam-na até ao ponto de recolha efetuado pelo braço robótico, existindo dessa forma um ponto inicial e final em que a velocidade absoluta é zero e um perfil trapezoidal de velocidade definido entre esses 2 pontos, constituído assim um perfil trapezoidal único. Em contrapartida, num dos tapetes de saída, constata-se a existência de diferentes células de operação ao longo do transporte da embalagem, resultando assim num percurso de velocidade trapezoidal cíclico, em que sucessivas paragens da embalagem são verificadas.

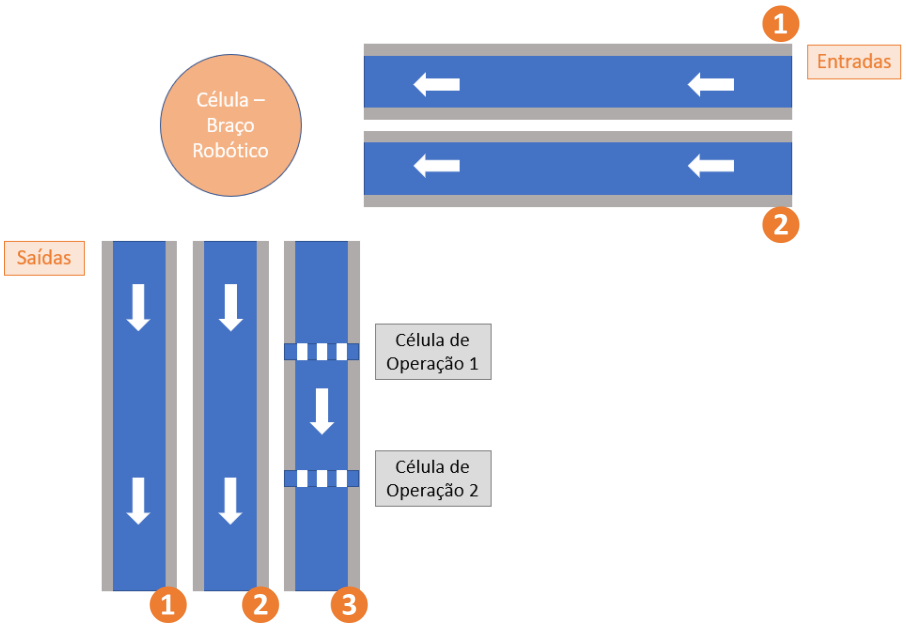
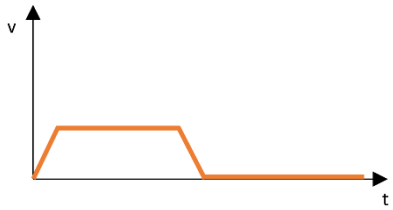
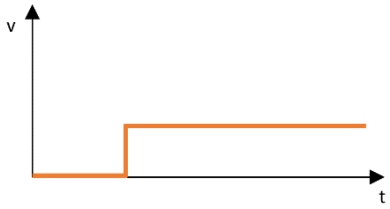
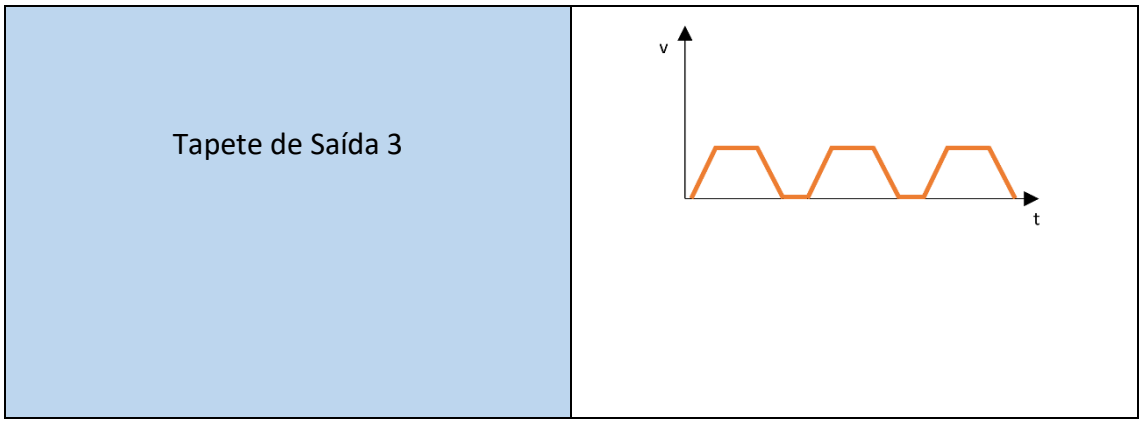


Figura 118 - Disposição dos tapetes transportadores, com um último tapete de saída que apresenta paragens ao longo de um ciclo de operação

Para os diferentes transportadores do sistema visualizados na Figura 118, os perfis de velocidade encontram-se demonstrados na tabela.

Tabela 12 - Perfis de velocidade nos diferentes tapetes

Tapete	Perfil de Velocidade
Tapete de entrada 1 e 2	
Tapete de saída 1 e 2	



No caso do perfil trapezoidal único, o modelo elaborado recorre ao algoritmo utilizado na elaboração de trajetória no braço robótico. Definidas as constrições cinemáticas inerentes ao perfil trapezoidal, como posição inicial, posição final, velocidade máxima e aceleração máxima, o perfil de posição e velocidade é gerado.

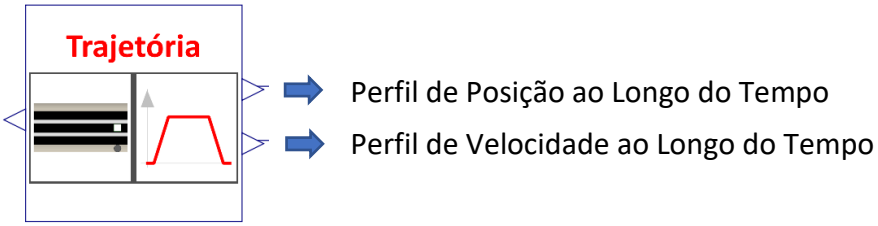


Figura 119 - Modelo utilizado para a geração do perfil trapezoidal único

Para o perfil trapezoidal cíclico, o modelo utilizado de trajetória foi desenvolvido recorrendo ao modelo anterior.

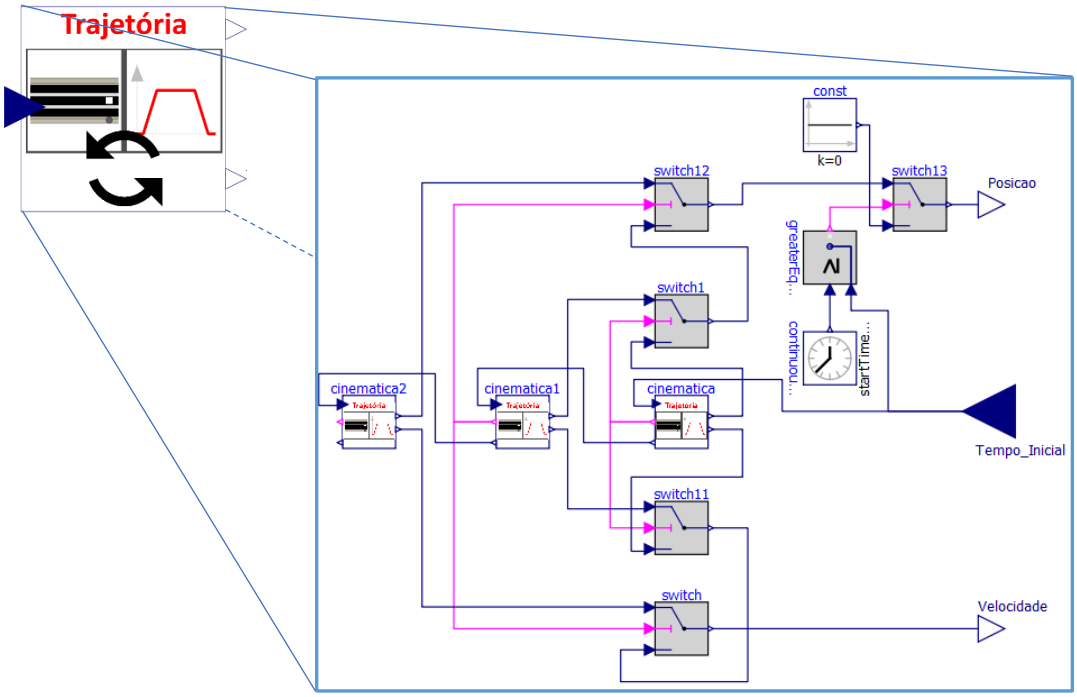


Figura 120 - Modelo utilizado para a geração do perfil trapezoidal cíclico

Antes de se partir para a explicação do diagrama que se encontra no interior do modelo, é importante clarificar que o perfil de velocidade final é constituído por 3 pulsos trapezoidais, um entre o ponto de início, em que a caixa é largada, e o ponto em que é realizada a primeira operação na mesma. Um segundo pulso entre os dois pontos de operação e ainda um terceiro entre o segundo ponto de operação e o ponto final do transportador, onde a embalagem deve admitir a velocidade 0, para que seja processada num processo posterior.

No diagrama apresentado, a modelação do perfil de velocidade e posição, que poderá ser dividido em 3 como referido, admite um aproveitamento do modelo do perfil trapezoidal já elaborado, ficando sujeito apenas a pequenas alterações, mais concretamente às variáveis de entrada e saída que tomam lugar em cada um dos modelos de trajetória. Assim, foi desenvolvido um sequenciamento dos mesmos através da transmissão de sinais de iniciação através de booleanos, e um envio dos valores de temporização, ou seja, os instantes de tempo em que os respetivos perfis devam ser calculados.

Na Figura 121 encontra-se presente o ciclo de controlo do movimento do tapete, constituindo desta forma a arquitetura final do modelo dos diferentes tapetes. Relativamente à geração de trajetória, e como anteriormente mencionado, os tapetes de entrada e o tapete de saída 3 geram perfis de posição e velocidade (a presença de um perfil de velocidades trapezoidal no

transporte é mais facilmente obtida segundo um controlo em cascata posição-velocidade) e nos tapetes de saída 1 e 2 são gerados perfis de velocidade, de forma exclusiva, devido a uma necessidade de controlo menos criteriosa, como consequência da ausência de um abrandamento no final do movimento.

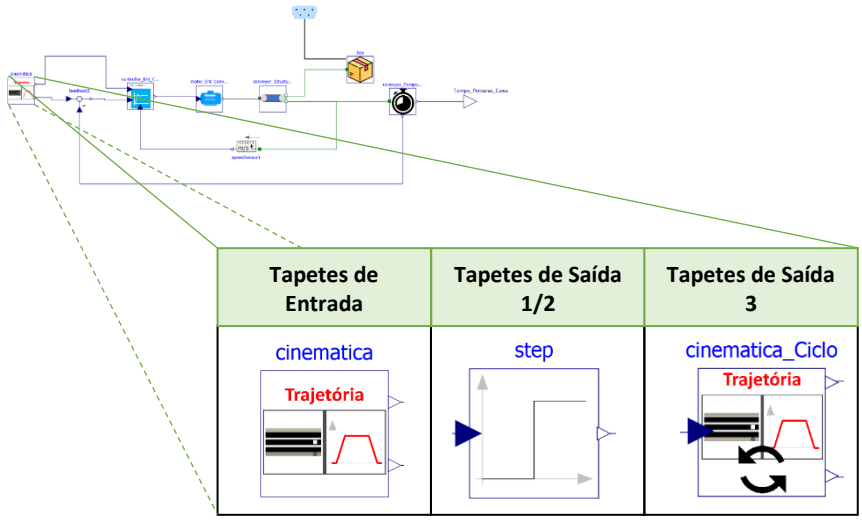


Figura 121 - Ciclo de controlo presente nos modelos dos tapetes e as opções presentes para a geração do perfil do movimento

8.2.2. Controlo e Motor

À semelhança do servomotor desenvolvido no modelo do manipulador, a modelação do motor dos tapetes transportadores apresenta o mesmo formato, com a única distinção na forma como a corrente é transmitida ao circuito elétrico. Nesta, um bloco *SignalVoltage* é utilizado dando entrada de forma direta no circuito. A voltagem ao longo do circuito fechado desenvolvido, é mediada assim por este bloco que transmite uma função contínua de diferença de potencial ao longo do tempo. Esta voltagem, por sua vez, acaba por ser o comando disponibilizado pelo controlo P-PI.

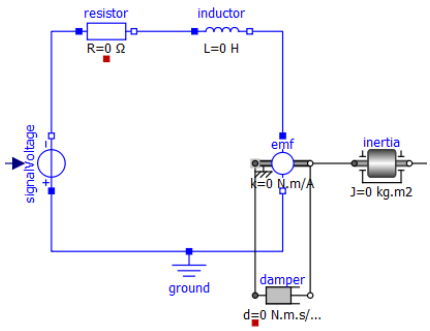


Figura 122 - Modelação de motor utilizado nos tapetes transportadores

O controlo P-PI, encontra-se diretamente associado à geração da trajetória, com um controlo proporcional da posição e um controlo PI da velocidade (tapetes de entrada).

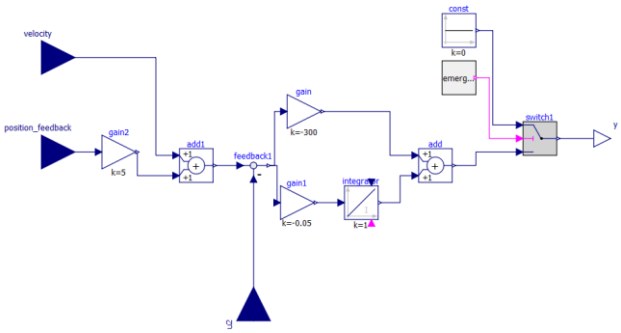


Figura 123 - Controlo P-PI associado ao motor nos tapetes de entrada

Nos tapetes de saída, a modelação do controlo foi modificada. Nos 2 primeiros tapetes de saída, destaca-se a ausência de um termo de controlo proporcional para a posição. A ausência de um comando de posição nestes 2 tapetes determina um controlo único de velocidade PI. Por último, para o controlo do tapete de saída 3, um controlo em cascata que atende aos comandos de posição, velocidade e corrente (à semelhança do perfil de controlo inserido nos motores do manipulador) foi implementado.

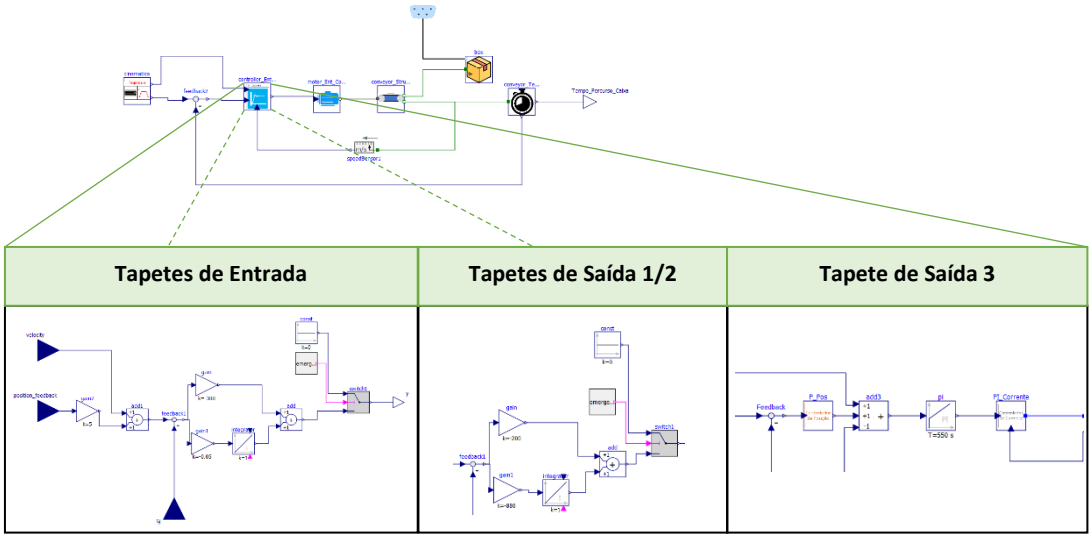


Figura 124 - Ciclo de controlo presente nos modelos dos tapetes e as opções presentes para o controlador

Há, assim, um aumento das imposições inerentes aos ciclos de controlo, indo de encontro às necessidades exigidas por cada tarefa desempenhada por cada tapete.

8.2.3. Estrutura do Tapete Transportador

Começando por analisar o modelo propriamente dito do *conveyor*, ou seja, toda a conjuntura inerente à operacionalidade da carga mecânica. A carga mecânica representa a força requerida para mover as embalagens sucessivas ao longo do tapete em ambiente industrial. A modelação do transporte da carga mecânica implica a divisão do sistema do tapete em diferentes competências, tendo sido a abordagem inicial constituída pelas engrenagens, a inércia do sistema, amortecimento, a transmissão que traduz a linearidade do movimento e por fim a massa a ser transportada, tendo sido esta abordagem inicial baseada na esquematização presente em (M.Tiller, Multi-Domain Modeling - Conveyor System , 2001).

Para a modelação da estrutura e carga mecânica foram, ainda, desenvolvidos dois modelos distintos, um mais simplificado e outro com maior complexidade, ou seja, com um número de parâmetros e variáveis envolvidas maior, determinando assim um estudo do sistema mais rigoroso. O conjunto dos diferentes componentes utilizados serão abordados em maior detalhe de seguida.

É importante perceber que para o motor acelerar ou desacelerar o tapete, o motor tem que vencer não só a inércia da massa transportada, como também a inércia conjuntural de todo o sistema.

A simplificação do sistema poderá conduzir a um sistema constituído por um tapete engrenado em dois cilindros/polias. Estas polias por sua vez podem ser classificadas em função de sua atuação no sistema. São classificadas como polias motoras ou condutoras aquelas responsáveis por transmitir a força. Já a polia que recebe o movimento é chamada de polia movida ou conduzida.

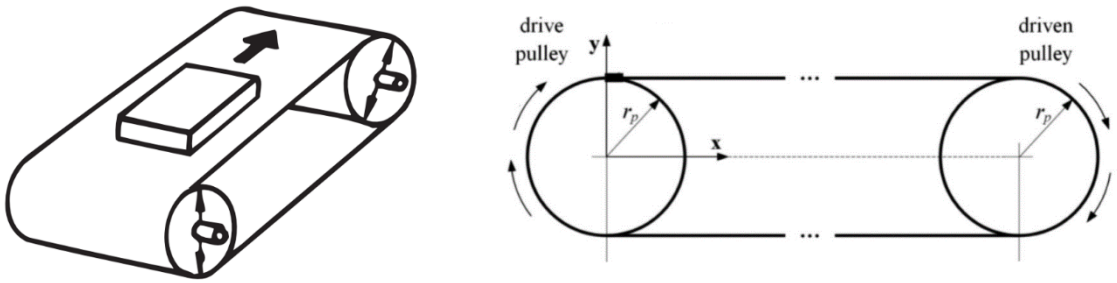


Figura 125 - Tapete com cilindros/polias associadas

A polia motora, acionada pelo eixo do motor, desencadeia o movimento do tapete transportador e, anexada entre esta e o eixo do motor, um conjunto de engrenagens de

grande eficiência é introduzida, aumentando o torque de saída e diminuindo a velocidade, de saída do motor, a ser transmitida.

Relativamente à inércia/momento de inércia, esta constitui-se como a medida de distribuição de massa de um corpo em torno de um eixo de rotação, que para o sistema modelado, se trata do eixo de transmissão. O momento de inércia avalia a dificuldade do tambor/cilindro inerente à transmissão rodar sobre o seu eixo, sendo que esta mesma dificuldade se relaciona de forma direta com os momentos de inércia da carga (tapete e embalagens) assim como com os momentos de inércia dos tambores de transporte. A inércia total depende assim de vários parâmetros implícitos ao tipo de tapete presente no sistema.

Os principais pontos de interesse já referidos a serem tidos em consideração ao longo da modelação de um tapete transportador: engrenagens, inércia do sistema, amortecimento, transmissão que traduz a linearidade do movimento e a massa a ser transportada, servirão de base para o desenvolvimento mais simplista do modelo da estrutura do sistema. Destes, o ponto que requer uma análise mais cuidada é a inércia total do sistema, envolvida no torque requerido para que o movimento ocorra como desejado. Desta forma, as devidas formulações serão apresentadas.

8.2.3.1. Cálculo do Torque e da Inércia total do sistema

Partindo por exemplo de um perfil de velocidade trapezoidal (verificado no tapete de saída 3), os diferentes torques requeridos ao longo do processo são o torque para a aceleração, para a velocidade constante e o torque de desaceleração, sendo que na maioria das aplicações, o torque máximo ocorre durante a aceleração. Devido ao facto do valor máximo se verificar maioritariamente no início, nos processos envolvidos com tapetes, um torque constante, em muitas aplicações, é fornecido ao sistema pelo motor, sendo o seu cálculo efetuado através da raiz quadrada de todos os diferentes torques requeridos ao longo do processo (a utilização do torque constante será desconsiderada na modelação). Os diferentes torques envolvidos nas diferentes fases serão apresentados.

8.2.3.1.1. Torque – Velocidade Contínua

Para um sistema como um tapete transportador, o torque do motor requerido durante a velocidade constante é simplesmente a força axial (F_a) total no tapete multiplicado pelo raio da polia (r_p).

$$T_c = \frac{F_a.\ r_p}{\eta} \tag{61}$$

Em que a eficiência η do sistema é incluída na equação do torque. Esta eficiência relaciona-se com as perdas de atrito entre o tapete e as polias/cilindros. É importante ainda notificar que foi assumido que ambas as polias, tanto a de acionamento como a acionada apresentam o mesmo diâmetro, o que acaba por ser o caso para a generalidade dos tapetes transportadores usados em sistemas de movimento linear.

Os tapetes transportadores por norma não são projetados para suportar forças axiais externas, pelo que a força axial total consiste de forma exclusiva na força necessária para mover a carga, que é o peso ($m.g$) da embalagem a ser transportada assim como do tapete, multiplicada pelo coeficiente de atrito da guia que suporta a carga.

$$F_a = m.g.\mu \tag{62}$$

8.2.3.1.2. Torque – Aceleração

A fase de aceleração do perfil de movimento é tipicamente o período em que o torque máximo requerido pelo motor é atingido, sendo este torque (T_a) frequentemente considerado também o torque intermitente. O torque necessário durante a aceleração do tapete inclui o torque necessário a velocidade constante (T_c) mais o torque necessário para a aceleração da carga (T_{ac}).

$$T_a = T_c + T_{ac} \tag{63}$$

O torque de aceleração da carga é obtido pela multiplicação da inércia total do sistema (J_t) pela aceleração angular (α).

$$T_{ac} = J_t\ .\alpha \tag{64}$$

A inércia total do sistema inclui a inércia do motor, acoplamento, polias/cilindros e ainda a carga (tapete e embalagem).

$$J_S = J_{P1} + J_{P2} + J_T + J_C + J_M + J_{Acop} \quad (65)$$

Tabela 13 - Cálculo das diferentes inércias constituintes na inércia do sistema global

Componente	Cálculo da Inércia
Inércia Cilindros	$J_{P1} = \frac{1}{8} . m_{P1} . D^2 \qquad J_{P2} = \frac{1}{8} . m_{P2} . D^2$ ou $J_{P1} = \frac{1}{2} . m_{P1} . r^2 \qquad J_{P2} = \frac{1}{2} . m_{P2} . r^2$
Inércia Tapete	$J_T = \frac{1}{4} . m_T . D^2$ ou $J_T = m_T . r^2$
Inércia Caixa	$J_C = \frac{1}{4} . m_c . D^2$ ou $J_C = m_c . r^2$
Inércia Motor	J_M
Inércia Acoplamento	J_{Acop}

8.2.3.1.3. Torque – Desaceleração

$$T_d = T_c - T_{acc} \quad (66)$$

O torque necessário durante a desaceleração do tapete inclui o torque necessário a velocidade constante menos o torque necessário para a aceleração da carga.

Entre a inércia, que representa a resistência da carga total a ser transportada e a transmissão que converte o movimento rotativo em linear, foi modelado um amortecimento presente nos

rolamentos. A força ou torque de amortecimento (T_a) é igual à multiplicação da constante de amortecimento (c_a) e da velocidade angular relativa (w_r). A perda de energia (P) nos rolamentos é igual ao torque de amortecimento multiplicado pela velocidade relativa.

$$T_a = c_a \,.\, w_r \tag{67}$$

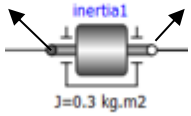
$$P \ = T_a \,.\, w_r \tag{68}$$

8.2.3.2. Implementação

O torque do motor é responsável pelo movimento de toda a carga mecânica envolvida no tapete. Na eventualidade de se querer manter uma velocidade constante, e desprezando as perdas de atrito entre o tapete e as polias/cilindros, um aumento de carga impulsiona um "contra-torque" (torque em sentido oposto) no motor, que desencadeia a diminuição da velocidade do tapete, pelo que um aumento do torque do motor é necessário para que um contrabalanço, entre o torque que desencadeia o movimento e o torque que oferece resistência ao movimento, exista, e dessa forma impeça a diminuição de velocidade. Assim, quando o torque e o contra-torque se igualam, uma estabilização da velocidade é gerada, tornando-a constante. Quando o torque é superior ao contra-torque tem-se um aumento de velocidade e uma consequente aceleração. Por outro lado, para um contra-torque superior, uma desaceleração é verificada.

Tabela 14 - Componente de transmissão e inercial usado na modelação

Componente	Descrição
	Componente envolvido no cálculo do torque a velocidade constante, com recurso à força axial requerida para o transporte da carga. O índice de transmissão sujeito à transformação de movimento rotacional em linear é tido em consideração. $T_c = F_a . r_p$

Flange A  Flange B	Componente que define a inércia total do sistema, calculando ainda o torque de aceleração da carga com base na aceleração admitida nas flanges. $J_S = J_{P1} + J_{P2} + J_T + J_C + J_M + J_{Acop}$ $T_{ac} = J_t \,.\, \alpha$

Torque de aceleração T_{ac}, sendo o mesmo responsável, de forma exclusiva, pelo aumento de velocidade. Corresponde ao torque do motor ao longo da aceleração T_a menos o torque de velocidade constante T_c (correspondendo ao contra-torque T_{ct}) exercido pela carga mecânica.

$$\begin{aligned} & T_a = T_{ac} - T_c \\ \leftrightarrow\ & T_a = T_{ac} - T_{ct} \\ \leftrightarrow\ & T_{ac} = T_a + T_{ct} \text{ , com } \|T_a\| > \|T_{ct}\| \end{aligned} \tag{69}$$

O torque de aceleração é 0 na fase de velocidade constante sendo, nessa mesma fase, o torque do motor ao longo da aceleração definido por um valor igual mas simétrico ao contra-torque.

$$T_{ct} = -T_a \text{ , para } T_{ac} = 0 \tag{70}$$

Por outro lado, para o torque de desaceleração T_{des}, o mesmo é responsável, de forma exclusiva, pela diminuição de velocidade. Sendo o cálculo formulado da mesma forma que o torque de aceleração, com a particularidade de:

$$T_{des} = T_a - T_{ct} \text{ , com } \|T_a\| < \|T_{ct}\| \tag{71}$$

Pode-se considerar a flange b como a flange em que o contra-torque se encontra definido, a flange a como a flange em que o torque de velocidade constante se define e por fim a inércia total do sistema, introduzida no bloco, que define o torque de aceleração, em conjunto com a aceleração.

```
J*a = flange_a.tau + flange_b.tau;
```

Figura 126 - Excerto de código no componente da inércia, no qual a equação que estabelece o torque é definida

8.2.3.2.1. Modelo Simplificado

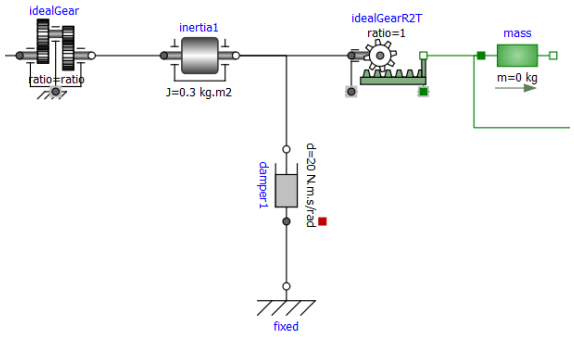


Figura 127 - Modelação simplificada da estrutura do tapete

A modelação a 1 dimensão da transformação da rotação em translação foi conseguida através de um componente/objeto caracterizado como uma caixa de engrenagens ideal (sem massa e inércia considerada), em que o índice de transmissão é devidamente definido, juntamente com os blocos definidos na tabela 14, um bloco destinado a um amortecimento rotativo, permitindo assegurar um movimento rotativo suave e controlado, e por fim o componente destinado à massa da embalagem a ser transportada.

8.2.3.2.2. Modelo com perdas consideradas

Relativamente ao 2º modelo elaborado, que visou um estudo do desenvolvimento da carga mecânica mais aprofundado, apoiou-se no conjunto de objetos já desenvolvidos, acrescentando mais alguns, expandindo assim o modelo total.

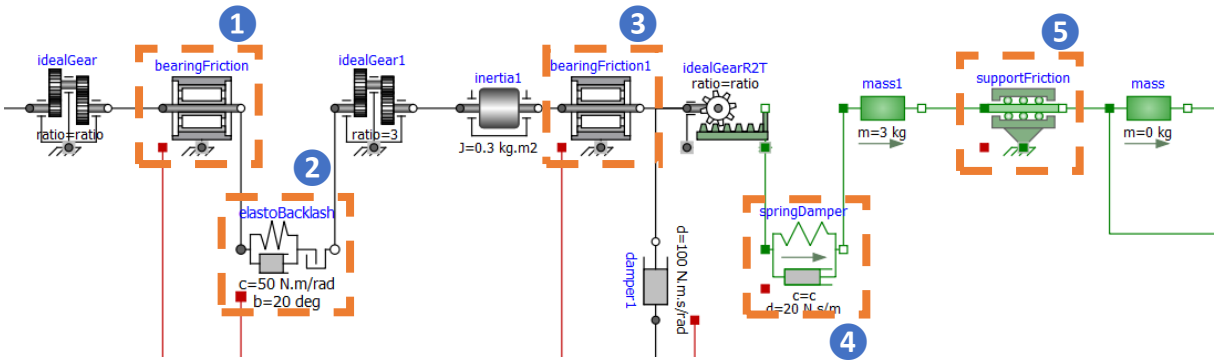
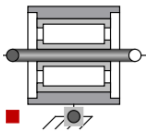
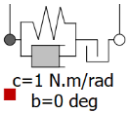
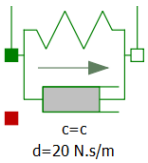
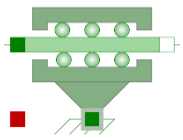


Figura 128 - Modelação com perdas associadas da estrutura do tapete

Este modelo não só admite uma, mas sim duas reduções por engrenagens. Os componentes adicionais focam-se em perdas de energia verificadas sob a forma de atrito, amortecimento, elasticidade e ainda *backlash*.

Tabela 15 - Descrição dos componentes associados a perdas de energia no sistema do tapete

Componente	Descrição
	Componente que caracteriza a fricção de Colombo entre a flange ao longo de um suporte presente (por exemplo uma chumaceira). Nesta, a fricção existente resulta do contacto entre a flange e uma superfície do rolamento.
	Componente que estabelece uma ligação em série entre um elemento de *backlash* e uma ligação paralela, composta respetivamente por uma mola e um amortecedor. Desta forma, os efeitos por elasticidade, amortecimento e backlash conseguem ser extraídos relativamente à segunda redução por engrenagens presente no modelo, conseguindo desta forma modelar com uma exatidão maior a respetiva caixa de velocidades.

	O bloco destinado à transmissão rotativa-linear inicialmente presente no 1º modelo do tapete, é ideal, pelo que despreza desde logo as eventuais perdas. Este bloco, no entanto, modela elasticidade e amortecimento presente nessa transmissão, garantindo assim uma definição de transmissão mais próxima da real.
	À semelhança do modelo/componente do atrito no rolamento, também este componente se foca no atrito de Colombo mas desta feita, não dependente do torque de atrito e da velocidade angular absoluta, visto tratar-se de um movimento translacional, mas sim entre a força de atrito, atuada entre a superfície e a carga, e a velocidade translacional absoluta.

Tem-se assim no 1º modelo, desprezando as perdas, 2 blocos que definem o torque requerido pelo motor. Estes, no entanto, poderão ser complementados com os blocos reproduzidos na tabela 15 e que retratam devidamente as perdas envolvidas. Estes blocos/modelos complementares irão assim definir as equações já abordadas.

1 - Blocos representativos da eficiência final relacionada com as perdas entre as engrenagens e o tapete e as polias/cilindros.

2 – Bloco representativo do coeficiente de atrito da guia que suporta a carga.

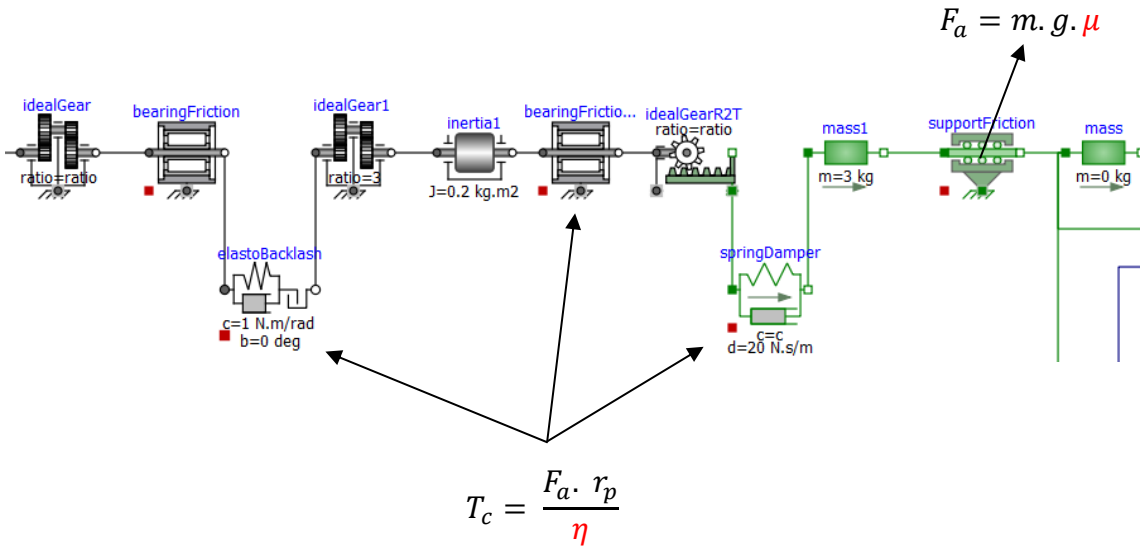


Figura 129 – Variáveis referentes à eficiência do sistema (vermelho) presentes nas equações envolvidas no torque e força axial requeridas

Atendendo às diferentes modelações da estrutura dos tapetes de entrada, as mesmas encontram-se inseridas no ciclo final do modelo do tapete, precedidas pelo modelo do motor e do controlo associado.

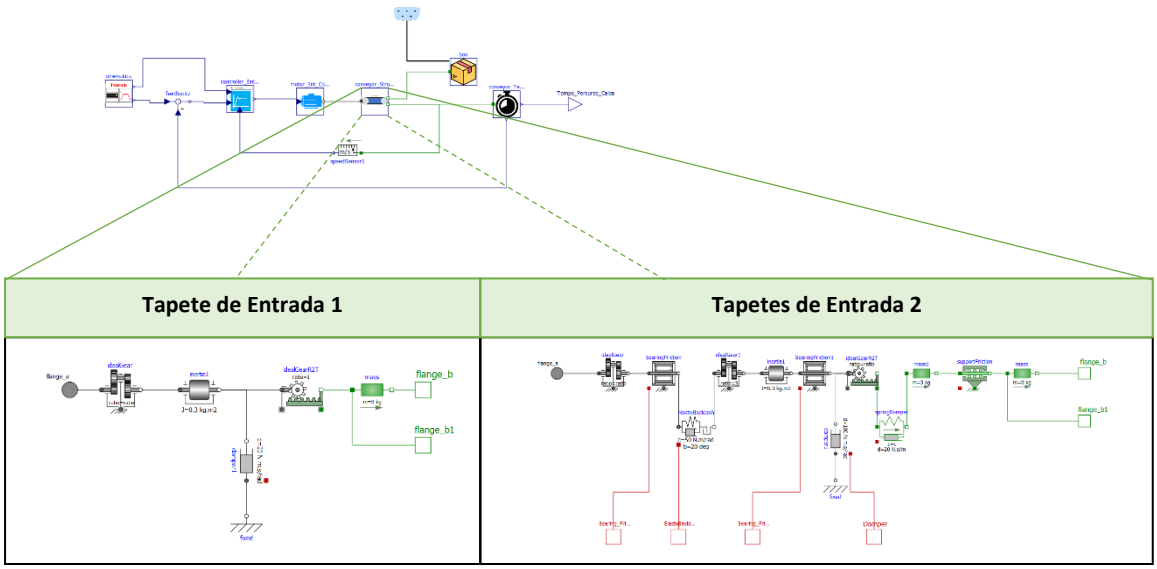


Figura 130 - Ciclo de controlo presente nos modelos dos tapetes e as opções para a estrutura do tapete transportador

8.2.4. Temporização

De maneira a estabelecer o tempo de percurso da caixa ao longo do tapete, uma secção constituída por blocos lógicos é elaborada.

Um sensor de posição é conectado à caixa transportada pelo *conveyor* e um feedback é responsável pela subtração entre o valor do comprimento do percurso a ser realizado, pela caixa, e a posição da mesma ao longo do percurso, medida pelo sensor. Quando essa diferença for igual a 0, com uma tolerância de 0.01 metros, um *timer* é ativado.

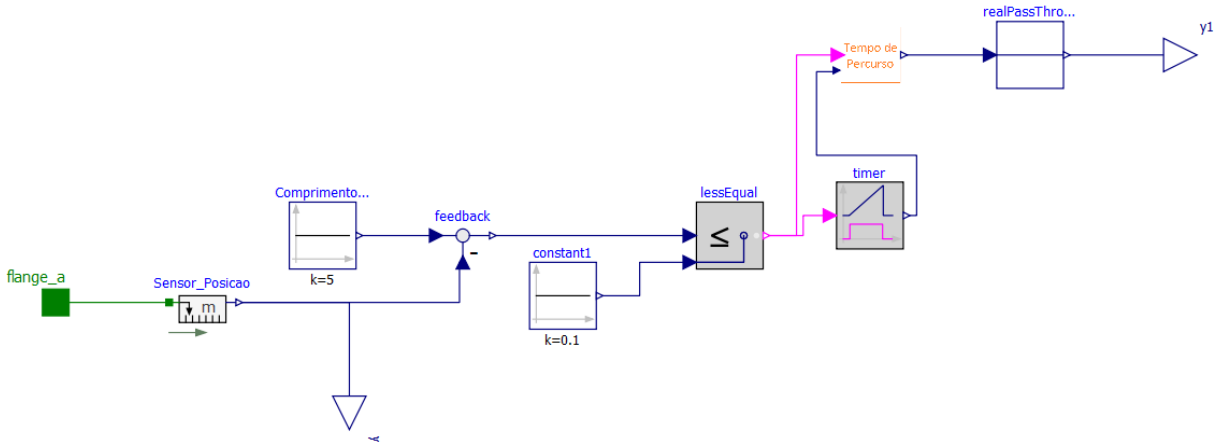


Figura 131 - Modelação da temporização do percurso no tapete

O bloco tempo de percurso, recolhe o sinal booleano, que determina se o percurso foi finalizado ou não, e o sinal proveniente do *timer,* que funciona como um cronómetro que é inicializado quando o booleano adota o valor de 1. Recolhidos os sinais, estes são tratados pela equação presente no bloco, em que caso o booleano seja 1, o valor de saída corresponde à diferença entre o tempo de simulação menos o tempo marcado na saída do *timer.* O tempo de simulação e o tempo no *timer* são valores que crescem de forma contínua, marcando a passagem do tempo, pelo que a subtração entre ambos será um valor real, valor esse que corresponde ao instante de tempo em que a caixa atinge a posição onde será posteriormente recolhida.

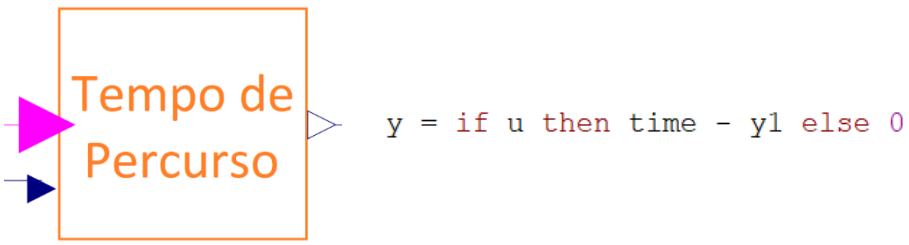


Figura 132 - Modelo de cálculo do tempo de percurso

8.2.5. Comparação – Modelação de Tapetes

Um aspeto relevante de observar, são as diferenças obtidas nos resultados de simulação obtidos para cada um dos tapetes de entrada. Aqui pode-se concluir que a complexidade da modelação, envolvida nos diferentes fatores passíveis de serem observados na vida real, marca a diferença. É de salientar, assim, a importância de uma aproximação, o mais exata possível, do processo real.

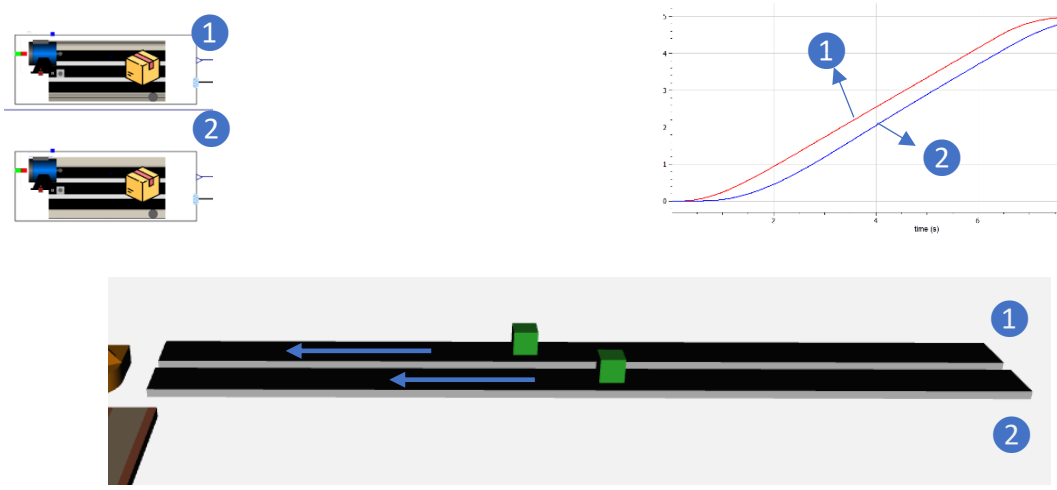


Figura 133 - Simulação dos 2 tipos de modelações realizadas para o tapete de entrada e os respetivos resultados gráficos

Além de distinguir essa importância, do quão próxima a modelação poderá estar na atribuição de fatores envolvidos no processo real, o desenvolvimento de modelos mais simplificados em paralelo com modelos mais complexos poderá ter a sua importância, pois permite perceber a significância desses mesmos fatores e a forma como se refletem na *performance* final do sistema.

8.2.6. Sistema de Arrefecimento

8.2.6.1. Fluxo Térmico para o Exterior

À semelhança do conjunto de modelos de fluxo térmico elaborados para o motor, uma exploração de dissipação térmica foi também realizada para um dos modelos dos tapetes transportadores, mais concretamente o 2º tapete de entrada modelado. A escolha deste tapete em específico prende-se com a particularidade de que, na modelação da estrutura do *conveyor*, maiores perdas mecânicas e térmicas são consideradas. Até agora, as perdas térmicas foram analisadas de um ponto de vista passivo, ou seja, os fluxos térmicos envolvidos

nos eixos das juntas do robot foram calculados, mas nunca na perspetiva de alterar esses mesmos fluxos e com isso evitar sobreaquecimento. Quer o funcionamento de um manipulador industrial, quer o funcionamento de um sistema *pick&place* complementado com diversos tapetes transportadores, por norma não requere um sistema de arrefecimento, a não ser que o fluxo de produção seja muito avultado. Claro que essa necessidade torna-se ainda menos relevante quando se foca apenas num ciclo de funcionamento, como analisado ao longo das simulações do modelo deste projeto. No entanto, no sentido de examinar essa eventual necessidade (por exemplo se for necessário que a variação da temperatura das embalagens a serem transportadas mantenham uma tolerância bastante rigorosa) assim como investigar e introduzir uma nova área de interesse de modelação no sistema, foi desenvolvido um sistema de arrefecimento, associado ao tapete transportador.

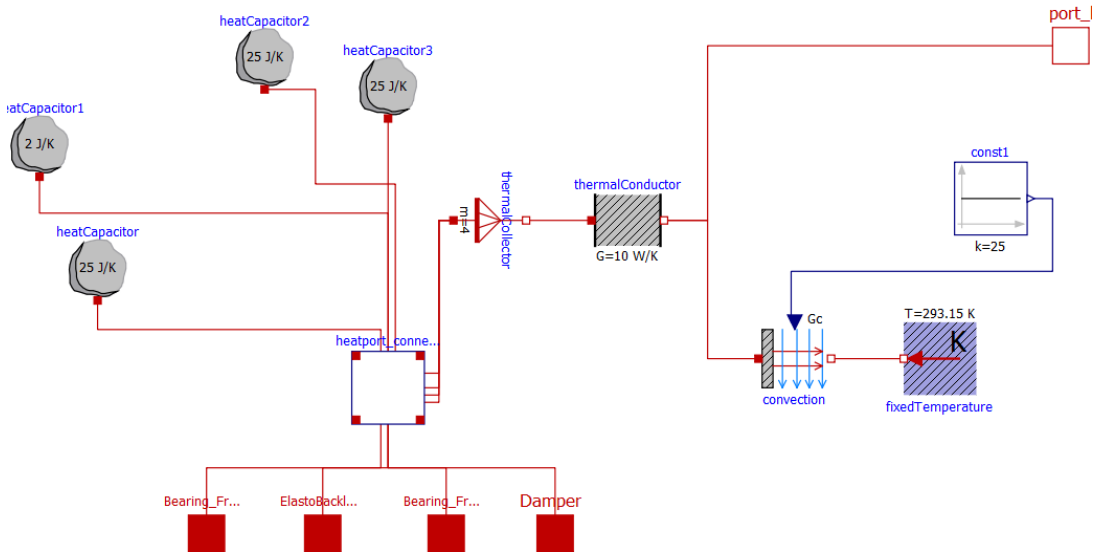
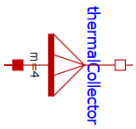
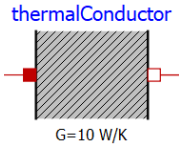


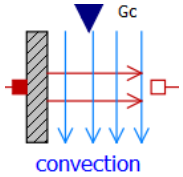
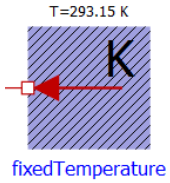
Figura 134 - Modelação do fluxo térmico para o exterior

Associadas a cada um dos componentes do tapete que se procura estudar as perdas térmicas (componentes de fricção, amortecimento e *backlash*), pode-se observar que foram introduzidos vários componentes destinados às diferentes capacidades caloríficas, uma para cada componente, representando assim a resistência à variação de temperatura de cada componente. Estabelecidas as capacidades caloríficas, um componente foi utilizado que permite somar os fluxos térmicos de cada um dos componentes e dessa forma associar um fluxo térmico único a um elemento condutivo. Esse elemento condutivo poderá ser definido como a fronteira física entre o tapete e o exterior, em que além de um contacto direto com

as embalagens, tem-se a partilha de um fluxo térmico convectivo com a temperatura ambiente, no seu exterior.

Tabela 16 - Componentes de Modelação utilizados no fluxo térmico para o exterior no tapete

Componente	Descrição
	Capacidade calorífica, ou seja, a quantidade de calor fornecida para que a temperatura suba uma unidade, considerando uma temperatura constante do corpo ao longo de todo o seu volume. A capacidade calorífica acaba por ser obtida através da multiplicação do coeficiente de transferência de calor k pela área da superfície do corpo A.
	Componente que reúne à sua entrada diferentes portes térmicos (m). A soma dos fluxos térmicos apresentados em cada um dos portes à entrada do componente são somados e o resultado dessa soma é apresentada sob a forma de fluxo térmico no porte térmico à saída. A temperatura estabelecida, tanto à entrada como à saída do componente é equivalente ao resultado do equilíbrio térmico do sistema em que o componente se encontra.
	Componente que retrata a condutividade térmica de um material em estudo. Do ponto de vista matemático pode ser definido como a quantidade de calor transmitida por unidade de tempo, através de um objeto com espessura L unitária, numa direção normal à área da

	superfície da sua secção reta *A*, também unitária devido a uma variação de temperatura ΔT unitária entre as extremidades longitudinais.
	Componente que define o estado convectivo da transferência, mais concretamente o coeficiente de transferência de calor e a área total de transferência, como evidenciado na equação. Para o efeito o parâmetro Gc é definido em que o mesmo se determina com base na multiplicação da área pelo coeficiente. $$Gc = h.A$$
	Condição fronteira em que uma temperatura fixa é estipulada.

O processo de convecção introduzido neste modelo obedece à equação de transferência já abordada nos eixos do robô, pelo que a modelação aqui introduzida em tudo se iguala à já previamente modelada no fluxo térmico ao longo dos eixos, com o componente de convecção diretamente conectado a um componente que caracteriza a temperatura exterior. Há no entanto, uma componente de condução térmica introduzida, cuja transferência se define através da seguinte fórmula.

$$\dot{Q} = \frac{Q}{\Delta t} = \frac{k\,.\,A\,.\,\Delta T}{L} \qquad (72)$$

Em que a condutância térmica $\frac{k\,.\,A}{L}$ se encontra parametrizada no componente destinado à condutividade térmica do corpo.

Como é possível constatar através do processo de dissipação térmica analisado, ocorre uma contínua transferência de calor, entre o tapete transportador e o ambiente exterior, sempre que este entra em funcionamento e a sua temperatura geral aumenta. Assim, é fácil perceber que um arrefecimento é sempre inevitável, quando o funcionamento do tapete é interrompido e a taxa de transferência térmica por convecção, implicada por uma transferência de calor para o ambiente circundante, é superior à geração de calor por parte dos componentes do tapete.

O desenvolvimento de um sistema de arrefecimento surge da necessidade de aumentar essa taxa de transferência de calor sempre que desejado e, assim, controlar de forma mais consistente a temperatura média da superfície condutora do tapete.

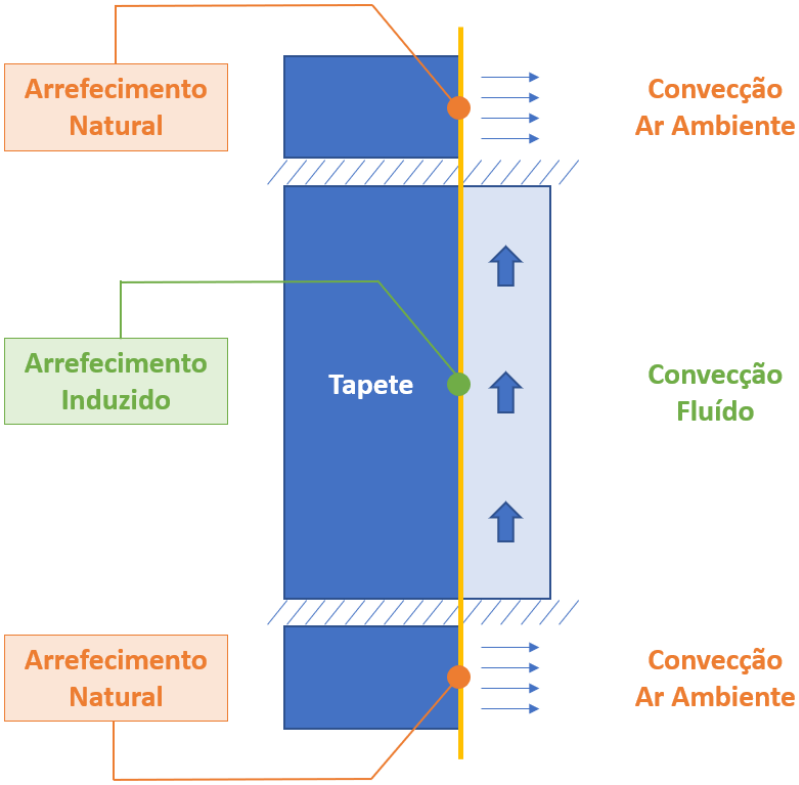


Figura 135 - Arrefecimento natural e arrefecimento induzido

De maneira a anexar a secção de fluído, e provocar um arrefecimento suplementar sempre que necessário, uma nova porção terá que ser modelada e associada ao componente que retrata a condutividade térmica da superfície do tapete, servindo assim de componente intermediário entre o fluxo total que advém do interior da estrutura do tapete e ainda do fluxo que advém do tubo por onde o fluído circula. O sistema de circulação do fluído de

arrefecimento foi inspirado num exemplo presente na biblioteca Modelica (.Modelica.Electrical.Machines.Examples.DCMachines.DCPM_Cooling, s.d.).

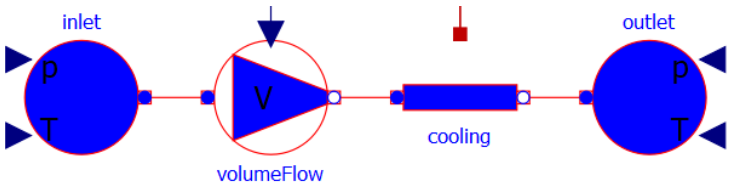
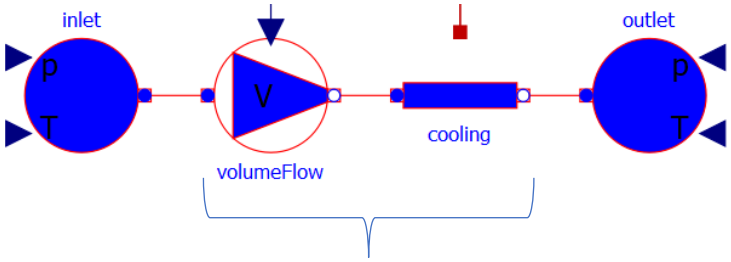


Figura 136 - Modelação do sistema de arrefecimento anexado

8.2.6.2. Aspetos Termodinâmicos

O modelo referente ao sistema de arrefecimento apresenta um conjunto de aspetos termodinâmicos que serão prontamente referidos. Os diferentes parâmetros e variáveis que apresentam relevância nos 2 principais modelos do sistema de arrefecimento poderão ser visualizados na Figura 137.



TwoPort	• *Dp* - Diferença de pressões entre os dois ports • *V_flow* (vazão volumétrico) = Flow Vazão Mássica (do porte a) (kg/s) / Densidade (kg/m3) • *T_a* = H - Entalpia Específica (porte a) / Cp - Capacidade calorífica específica(J/(kg.K)) – a pressão constante • *T_b* = H - Entalpia Específica (porte b) / Cp - Capacidade calorífica específica(J/(kg.K)) – a pressão constante
flowPort	• *P* - Pressão (Pa) • *M_flow* - Flow Vazão Mássica (kg/s) • *H* - Entalpia Específica • *H_flow* - Flow Fluxo Entalpia (watts)
medium	• *Rho* - Densidade (kg/m3) • *Cp* - Capacidade calorífica específica(J/(kg.K)) – a pressão constante • *Cv* - Capacidade calorífica específica(J/(kg.K)) – a volume constante • *Lambda* - Condutividade térmica (W/(m.K)) • *Nu* - Viscosidade cinemática (m2/s)

Figura 137 - Diferentes parâmetros e variáveis que apresentam relevância nos 2 principais modelos do sistema de arrefecimento

Modelo parcial *TwoPort* em que propriedades do fluído se encontram definidas. Estas propriedades servem de base por sua vez para a elaboração dos modelos *volumeFlow* e *cooling* . Este modelo parcial é constituído por 2 *flowports* em que pressão *p* (Pa), vazão mássico *m_flow* (kg/s), entalpia específica *h* e fluxo entálpico *h_flow* (watts) se encontram definidos. Nestes 2 *flowports* um parâmetro *medium*, com propriedades relativas ao fluído, é ainda estabelecido, nos quais densidade *Rho* (kg/m3), capacidade calorífica específica *cp* (J/(Kg.K)) a pressão constante, capacidade calorífica específica *cv* (J/(Kg.K)) a volume constante, condutividade térmica *lambda* (W/(m.K)) e viscosidade cinemática *nu* (m2/s) são declaradas.

O modelo parcial *TwoPort* calcula a variação entre as pressões nos 2 *flowports*, o vazão volumétrico através da divisão do vazão mássico no primeiro *flowport* pela densidade, as temperaturas nos 2 *flowports* através da razão entre a entalpia específica e a capacidade calorífica específica a pressão constante e por fim a temperatura relevante para a troca de calor com o ambiente. Ao longo deste modelo parcial é definido ainda um equilíbrio mássico entre os 2 *ports* e um equilíbrio energético em que o fluxo entálpico nos 2 *flowports* mais o calor trocado com o exterior/ambiente se iguala a $m \,.\, c_v \,.\, dT$ sendo m a massa do fluído, c_v a capacidade calorífica específica a volume constante e por fim dT a derivada da temperatura no segundo *flowport* ao longo do tempo.

No modelo *simpleFriction* tem-se uma definição do tipo de escoamento presente ao longo do tubo, com base na variação de pressão verificada. Para uma variação de pressão menor que *dpLaminar* , o escoamento é considerado laminar, pelo que para uma variação de pressão maior que o valor *dpLaminar* admite-se um escoamento turbulento. Perante um escoamento laminar, a dependência da variação de pressão no vazão volumétrico é linear, enquanto que perante um escoamento turbulento esta dependência assume-se como quadrática.

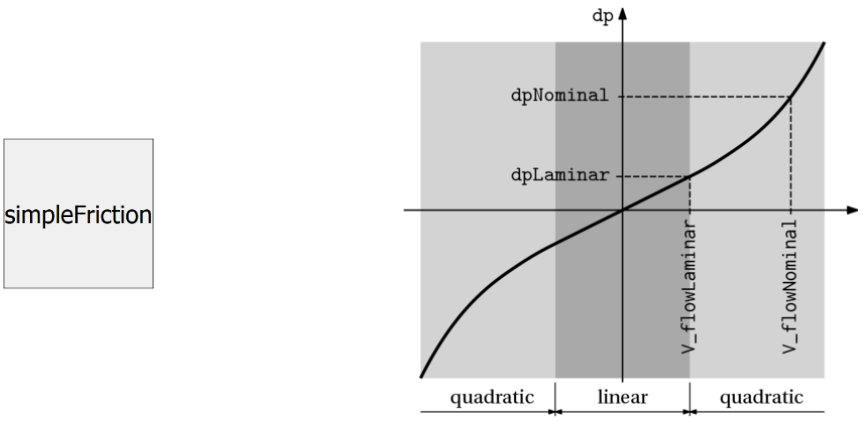
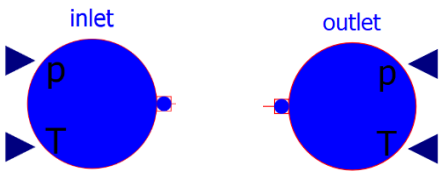
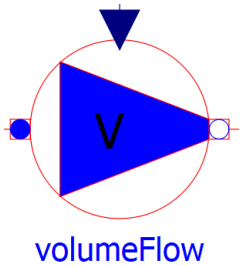
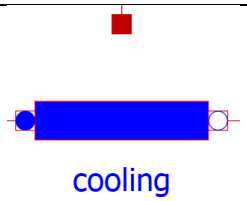


Figura 138 - Componente *simpleFriction*

Há no entanto, por motivos de facilitamento de simulação, uma desconsideração das perdas por atrito, decorrentes de um fator de atrito atribuído, que em conjunto com a queda de pressão e vazão volumétrico permite o seu calculo. Deste modo, o calor trocado com o meio exterior, é exclusivamente determinado pelo balanço energético estabelecido no sistema.

Tabela 17 - Descrição dos componentes presentes no sistema de arrefecimento

Componente	Descrição
	*Inlet* e *outlet* definem-se como dois componentes que determinam as condições fronteira do sistema, em que um ambiente com as propriedades de pressão e temperatura se encontram estabelecidas.
	Componente que tem a capacidade de definir diretamente o vazão volumétrico (m3/s). Na ausência dessa definição por parte do utilizador, o vazão volumétrico tido em consideração advém das equações termodinâmicas patentes no modelo parcial dos dois portes.

	Admite uma troca de calor ideal com o exterior.

8.2.6.3. Controlo do Sistema de Arrefecimento

Na modelação do sistema de arrefecimento, houve uma preocupação em otimizar o arrefecimento com recurso a um controlo do vazão volumétrico. Um maior valor de vazão volumétrico resulta numa maior troca de calor decorrente entre a tubagem e a superfície de condução térmica no tapete transportador, diminuindo assim a temperatura.

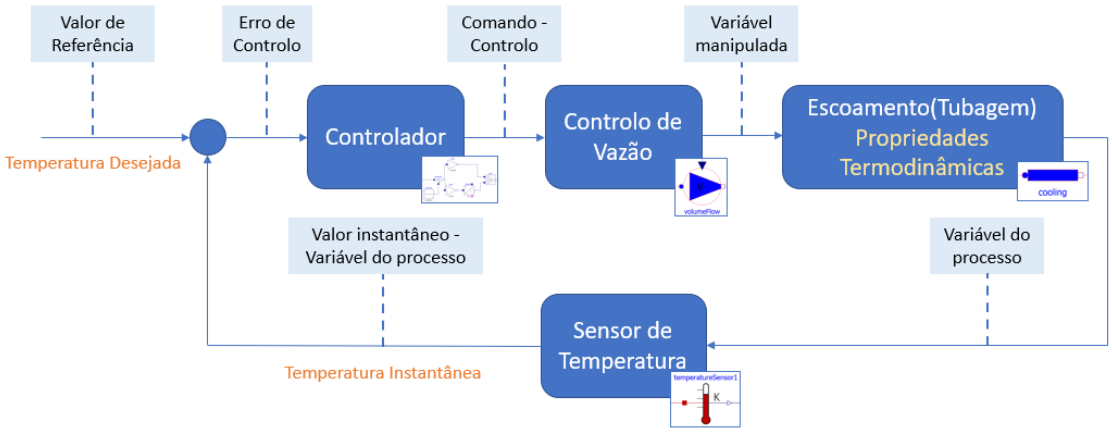


Figura 139 - Esquematização do sistema de controlo

De modo a garantir este controlo do vazão, a temperatura à saída do *port* térmico (variável do processo) no bloco *cooling* é continuamente medida, sendo o respetivo valor um valor que dá entrada num controlador PI. Este controlador por sua vez manipula ao longo do tempo o vazão volumétrico no sistema, de modo a garantir que a diferença entre a temperatura desejada na superfície de condução térmica do tapete e a temperatura instantânea obtida se aproxime do 0 absoluto.

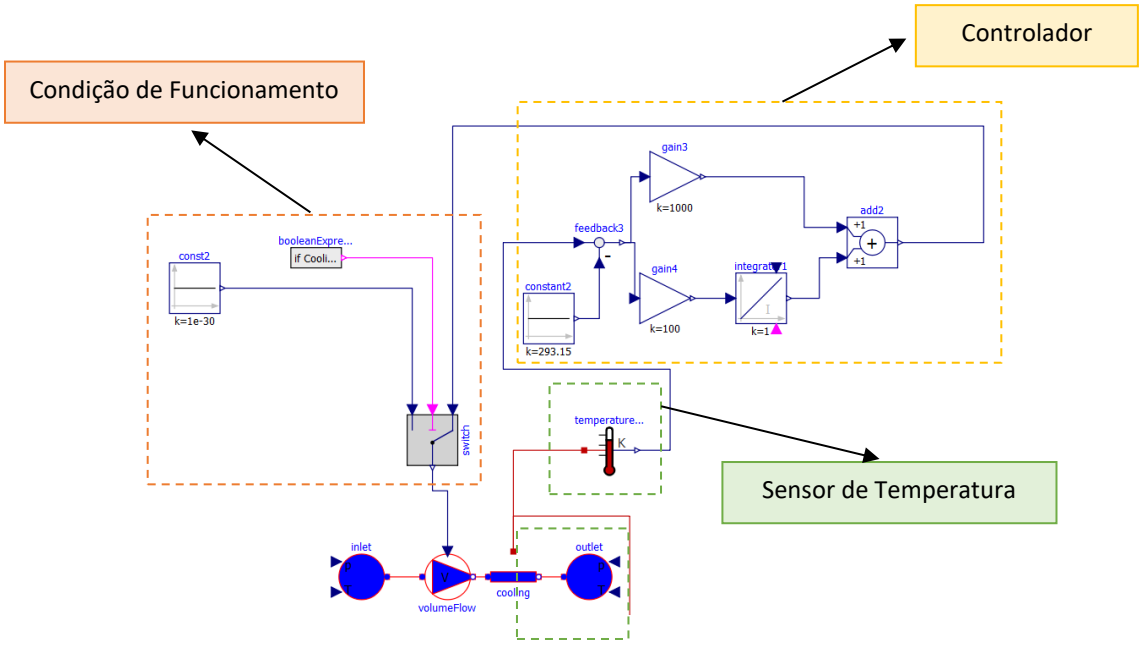


Figura 140 - Modelação do sistema de controlo

Uma condição adicional é inserida ao longo do sistema, que se prende com ativação do sistema de arrefecimento, na eventualidade de o respetivo booleano de ativação se encontrar verdadeiro o sistema de controlo opera conforme supramencionado, de outra forma, o vazão volumétrico admitido no sistema é 0 e a ocorrência de um arrefecimento suplementar não é verificado.

O modelo final do tapete 2, com as propriedades de dissipação térmica e arrefecimento do sistema, podem ser observadas na ilustração embaixo.

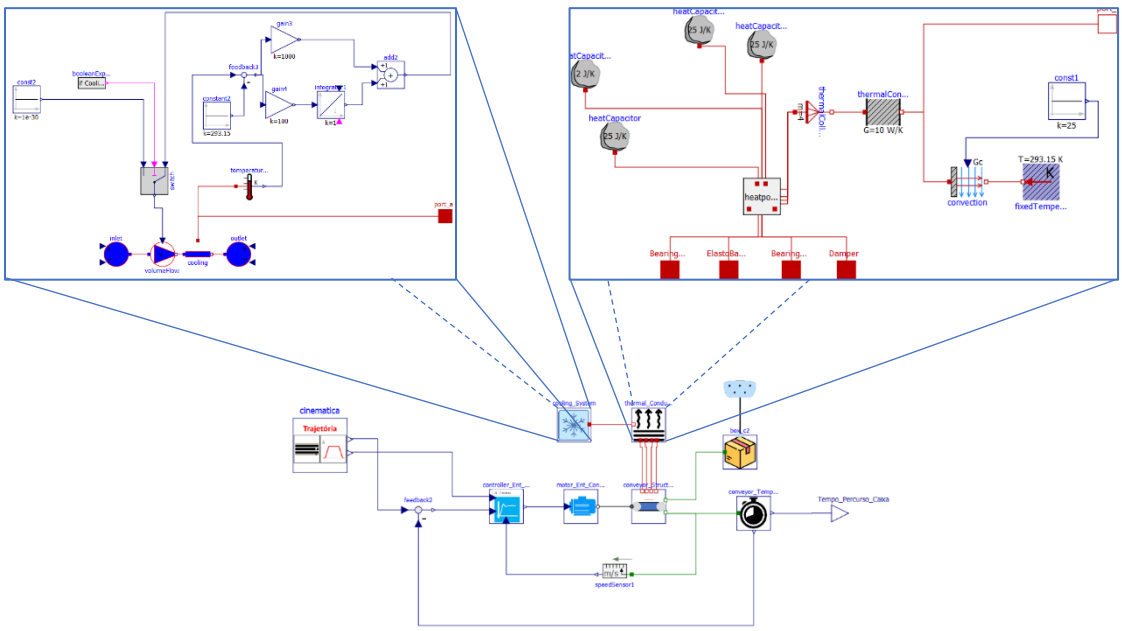


Figura 141 - Modelo final do tapete 2 com sistema de condução, convecção e arrefecimento térmico

8.2.6.4. Arrefecimento - Resultados

- Sem arrefecimento induzido

É possível verificar que um aumento de temperatura ao longo do tapete se verifica, em simultaneidade com o movimento de transporte do tapete. Quando o movimento do tapete é finalizado, o pico de temperatura é atingido, sendo que a temperatura entra posteriormente em declínio, resultado da transferência de calor para o exterior.

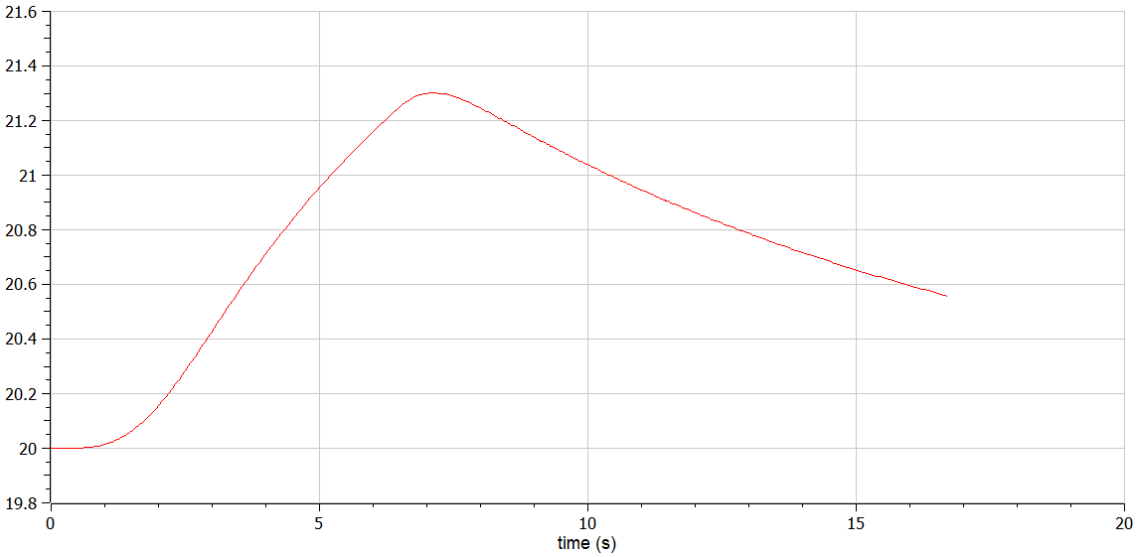


Figura 142 - Variação de temperatura no tapete sem sistema de arrefecimento ativado

- Com arrefecimento induzido

Com o sistema de arrefecimento ativo, a diferença da variação de temperatura é notória. Registando-se um pico de aumento, de variação da temperatura, de 0.116 em detrimento de 1.299 quando não sujeito ao arrefecimento suplementar.

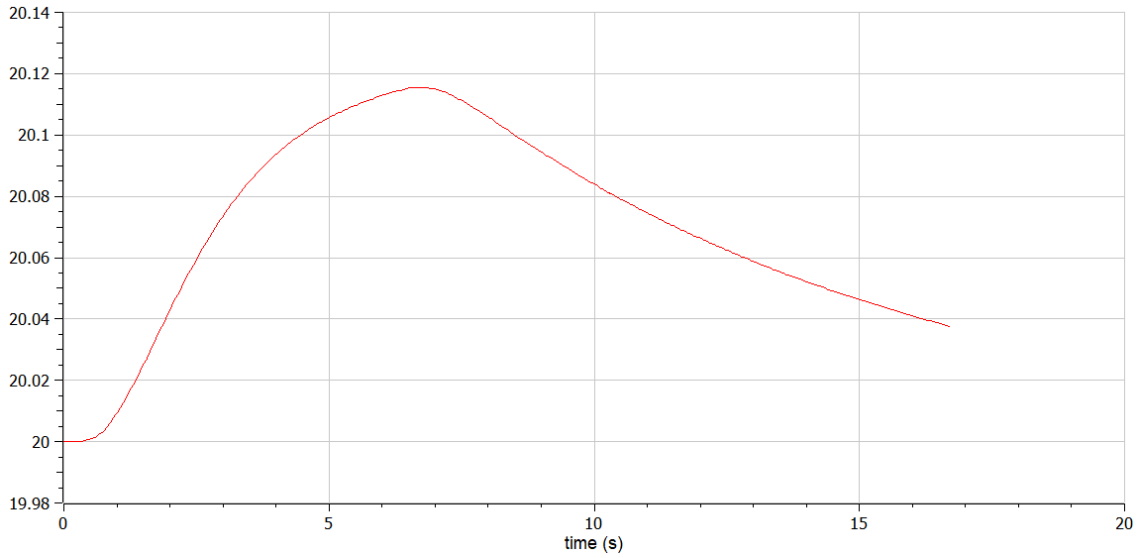


Figura 143 - Variação da temperatura no tapete com o sistema de arrefecimento ativado

8.2.7. Continuação do Estudo de Simulações – Tapetes

Alguns resultados derivados das simulações e decorrentes dos tapetes são aqui apresentados.

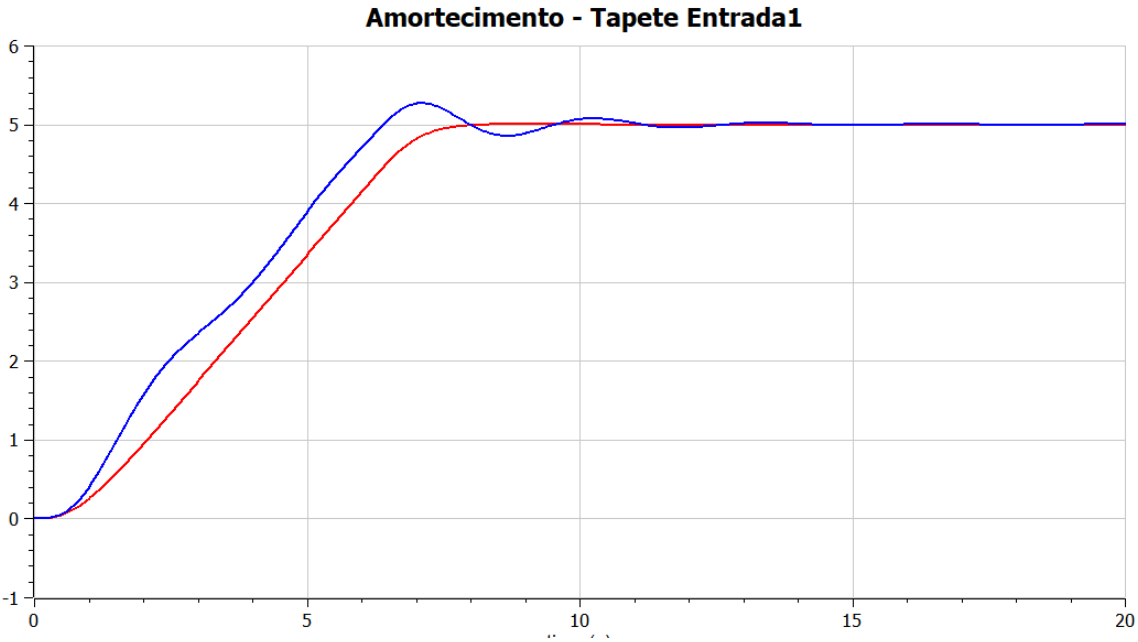


Figura 144 - Tapete de entrada 1 com e sem amortecedor

Neste gráfico é possível observar o tapete de entrada 1 que foi simulado à parte do sistema global com 2 configurações distintas, de forma a permitir observar o efeito que a inserção de um amortecedor tem na *performance* dos diferentes tapetes. É notório que a sua inserção (função a vermelho) permite um movimento muito mais suave, sem perturbações e/ou *overshooting*.

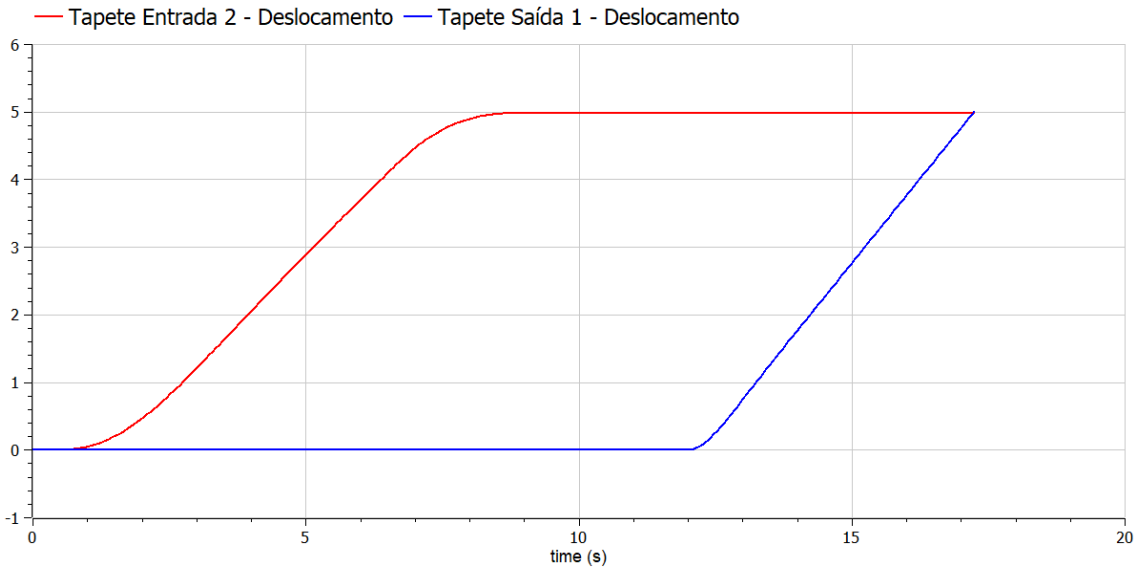


Figura 145 – Movimento linear de tapete de entrada 2 e tapete de saída 1

Aqui é possível observar o movimento da embalagem no tapete de entrada e no tapete de saída (embalagem 1). Com a utilização do tapete de entrada considerando as perdas, as acelerações e desacelerações são mais suaves em contraste com o movimento de saída, em que as perdas não são consideradas.

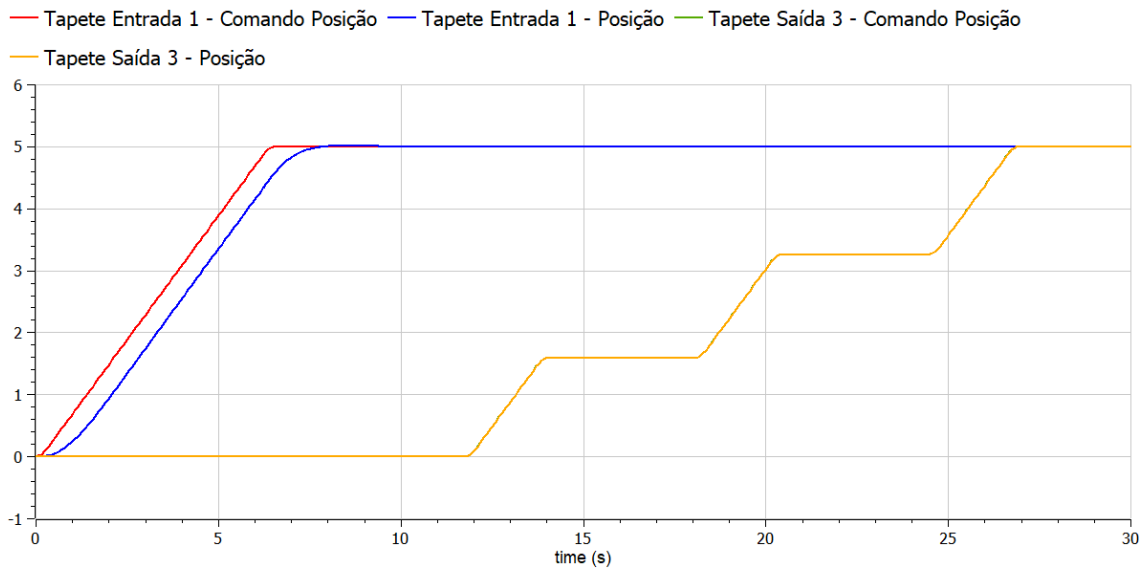


Figura 146 - Comando e Posição/Movimento do tapete de entrada 1, e comando e Posição/Movimento do tapete de saída 3

As diferenças estabelecidas entre os comandos de posição e as posições verdadeiramente adotadas poderão ser observadas no gráfico, para o tapete de entrada 1 e o tapete de saída 3. Enquanto que o tapete de entrada apresenta um desfasamento, o tapete de saída apresenta um comportamento quase perfeito, com a sobreposição da posição adquirida à desejada. O desfasamento verificado no tapete entrada pode ser explicado por um *tunning* pouco exato e pela ausência do controlo de corrente, que se encontra presente no controlo do tapete de saída. O facto de o tapete de saída obrigar a paragens sucessivas em postos de operação específicos e cujo tempo de repouso de 4 segundos deve ser obedecido com muito pouca tolerância, o desenvolvimento do controlo neste tapete foi mais rigoroso.

Relativamente aos torques, é importante referir que a aceleração exigirá um torque de aceleração ($T_{ac} = J_t \,.\, \alpha$). Quando a aceleração cessa e uma velocidade constante é atingida, tem-se torque igual ao *contra-torque* (diferença entre ambos torna-se nula como constatado graficamente), gerando um equilíbrio de movimento.

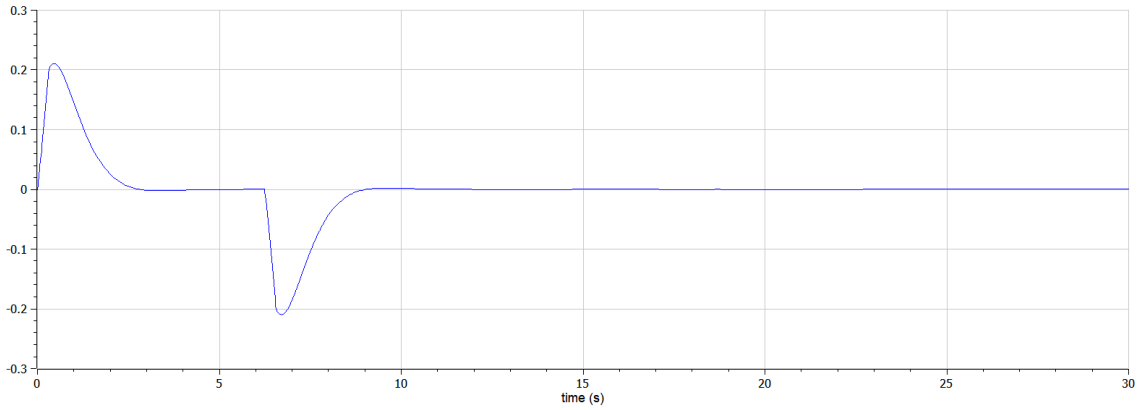


Figura 147 - Diferença entre o torque e o contra-torque

Quando a aceleração ocorre, o torque resultante do motor é maior que o contra-torque (resultando numa diferença positiva em que aceleração é verificada) e numa fase final tem-se um torque proveniente do motor menor que o contra-torque resultante da inércia do sistema (resultando numa diferença negativa em que desaceleração é verificada).

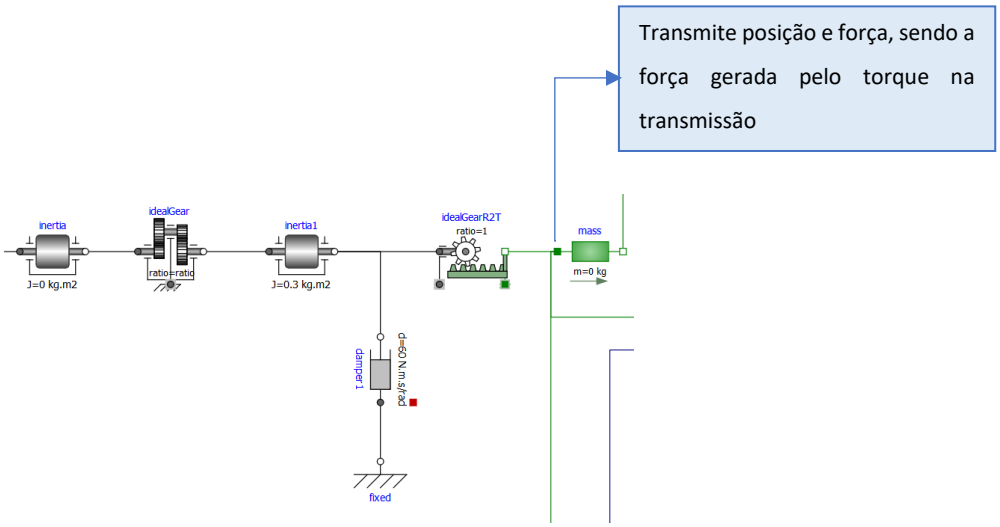


Figura 148 - Torque envolvido na transmissão diretamente proporcional à força axial requerida para o transporte

A força axial, e consequentemente o torque no bloco de transmissão rotação - translação, adotará sempre um valor próximo de 0 entre a aceleração e desaceleração, uma vez que , desconsiderando as perdas, a carga manterá um movimento retilíneo uniforme sem a aplicação de qualquer força axial atuada. Portanto o torque tenderá sempre para um valor nulo, nesta fase, independentemente da inércia do sistema total ou da constante de amortecimento considerada.

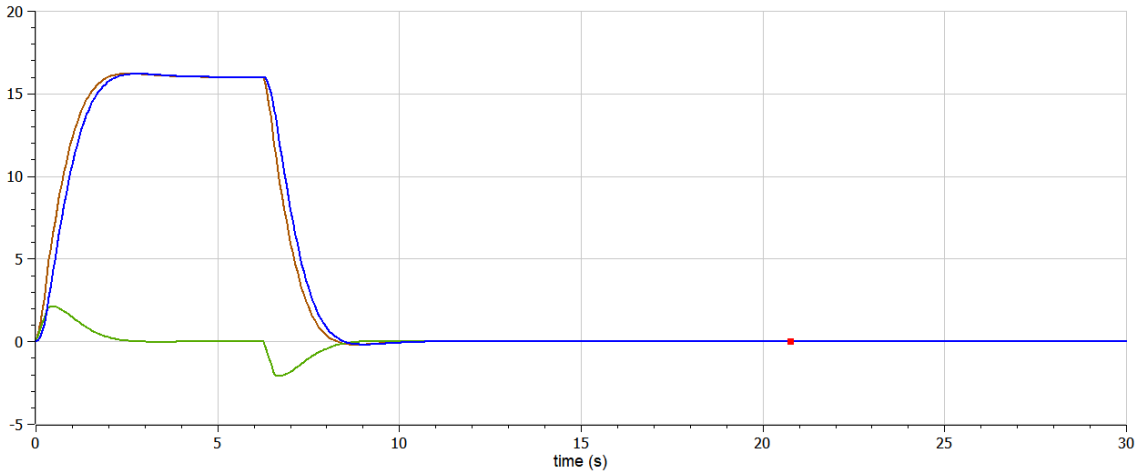


Figura 149 - A castanho o torque requerido pelo motor. A verde o contra-torque de aceleração e desaceleração em função da força axial exigida e por fim a azul o contra-torque fruto do amortecimento

No gráfico, é apresentado a castanho o torque proveniente da transmissão associada ao motor. Analisando a fórmula decorrente do torque com velocidade constante, seria expectável que este torque exigido ao motor adota-se um valor próximo de 0, uma vez que não existindo aceleração nesta fase, a força decorrente da variação de velocidade também não existe, resultando num torque próximo de 0. A força axial de movimento existe, no entanto, não pela variação de velocidade mas pelo contra-torque, consequente da utilização de um amortecedor (para evitar *overshooting*). O contra-torque tem o seu torque diretamente proporcional à velocidade, pelo que assume, à semelhança da velocidade, um perfil próximo do trapezoidal. O torque do motor (linha castanha) adota assim um valor ao longo do tempo, decorrente da soma entre o torque para a aceleração e desaceleração da carga (linha verde) e o contra-torque (linha azul) resultante da força de atrito no amortecedor. Quando o contra-torque do amortecedor supera o torque proveniente do motor, a desaceleração é verificada.

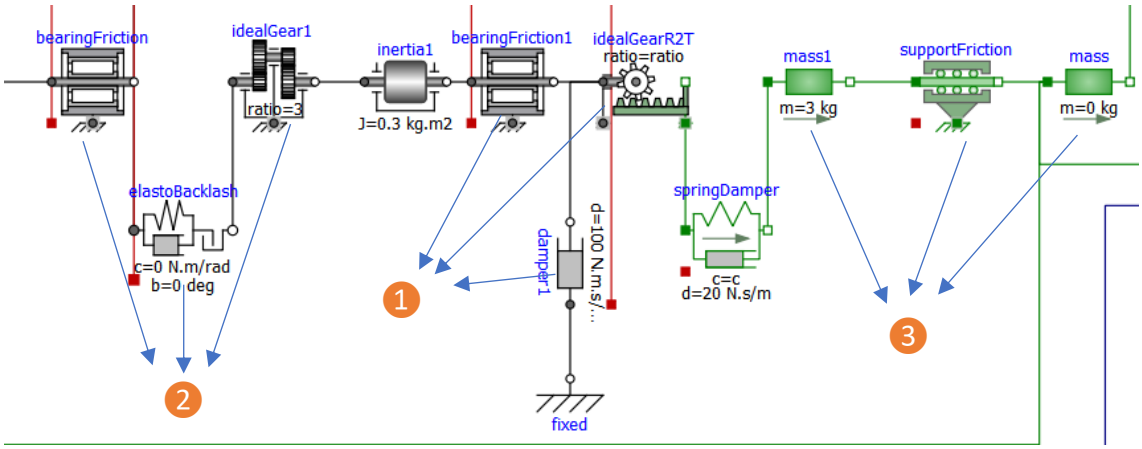


Figura 150 - Modelação da estrutura do tapete no tapete de entrada 2

Relativamente à simulação do modelo com uma estrutura em que as perdas estão compreendidas, à semelhança do modelo do tapete de entrada 1, o torque presente no bloco da inércia é equivalente (1) ao contra-torque do atrito no amortecedor mais torque de aceleração e desaceleração da carga, mas neste caso o contra-torque presente no atrito dos rolamentos também é considerado. O torque presente no bloco inercial não é, no entanto, o torque requerido pelo motor, uma vez que este bloco é precedido por uma transmissão por engrenagens com *backlash* (2). O efeito *backlash* é responsável por eliminar qualquer tipo de contacto entre as engrenagens para um intervalo angular estabelecido, desencadeando um torque superior, quando esse contacto se introduz novamente, de modo a compensar a ausência de transmissão prévia, gerando deste modo uma variação de velocidade mais abrupta e menos suave. Uma resistência suplementar ao movimento é ainda introduzida com um 2ª bloco de fricção nos rolamentos.

Neste exemplo, foi realizada a modelação de uma estrutura guia (3) que serve de base para a carga mecânica (tapete mais a embalagem, tendo sido a massa do tapete desconsiderada). A força axial exercida na estrutura guia (linha vermelha) é assim superior à força axial exercida na carga mecânica (linha azul), uma vez que a modelação de força de atrito axial foi realizada. É de salientar ainda a presença de uma força axial e consequentemente um torque ao longo da aceleração mais acentuado do que aquele verificado ao longo da desaceleração. Este pico de força e torque requerido é fundamentalmente explicado pelo *backlash*, que atua numa primeira fase em que a carga passa do estado de repouso para o estado de movimento.

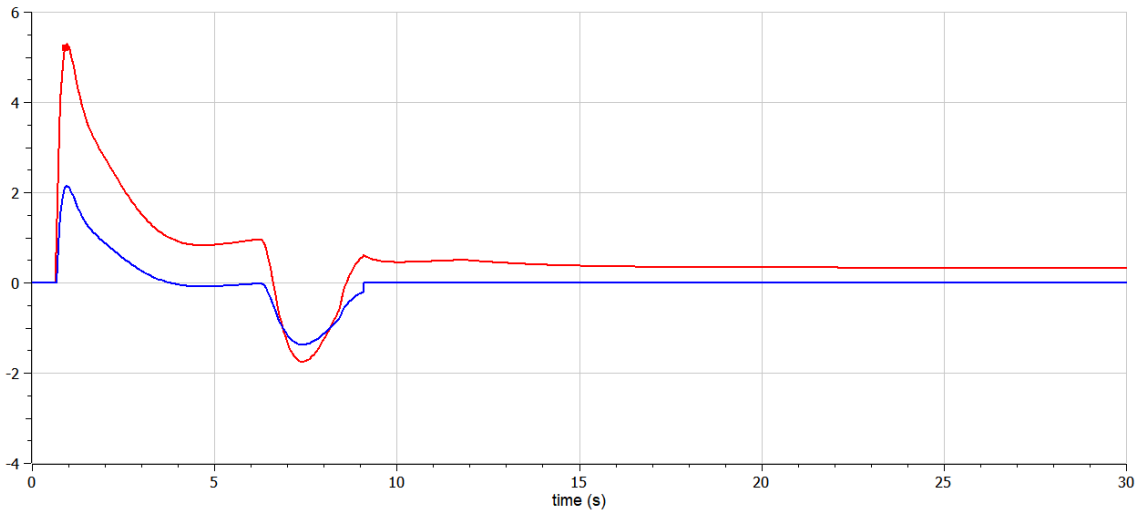


Figura 151 - Força axial atuada na estrutura guia (vermelho) e força axial atuada na carga mecânica (azul)

Por fim, o torque final proveniente do motor (linha verde) corresponde à soma do torque de aceleração e desaceleração (linha laranja, cujo perfil de variação é igual ao da força axial requerida na estrutura guia), mais os torques requeridos para suprimir as diferenças resistências ao movimento envolvidas no amortecimento, fricção e *backlash*, que conjugadas admitem um perfil próximo do trapezoidal.

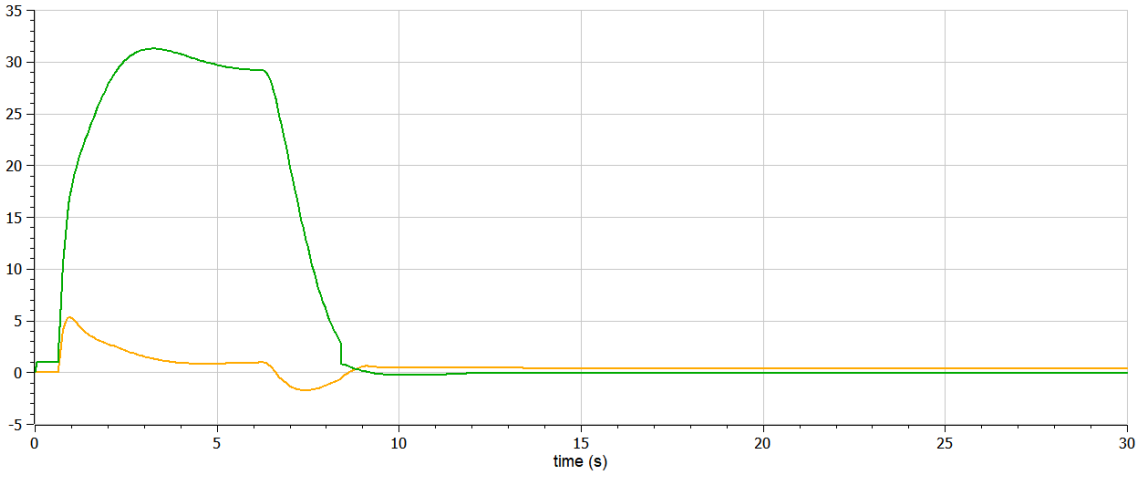


Figura 152 - Torque proveniente do motor (verde) e torque de aceleração e desaceleração (amarelo)

De ressalvar que o torque envolvido na transmissão rotação – translação, em contraste com a modelação simplificada da estrutura, deixa assim de ser próximo de 0, uma vez que para que uma velocidade constante seja mantida um torque tem que ser fornecido de forma a vencer as forças de atrito envolvidas entre a estrutura guia e a carga mecânica.

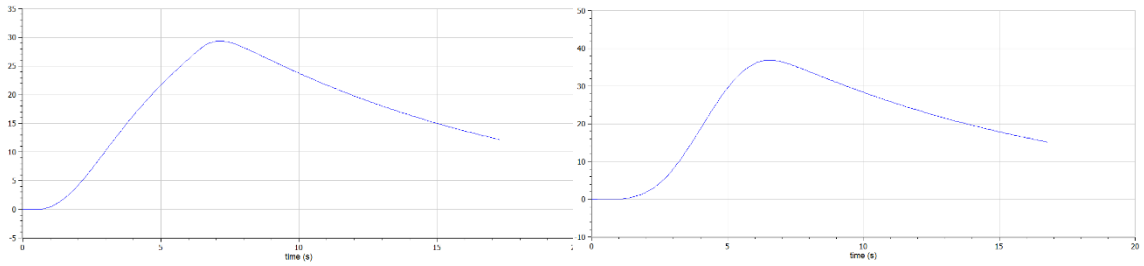


Figura 153 - Transferência de calor por convecção no tapete de entrada 2, sem sistema de arrefecimento, com perfil de velocidade trapezoidal (esquerda) e polinomial de 3ªordem (direita)

No robot, o perfil trapezoidal de velocidade apresenta valores mais elevados de transferência de calor relativamente ao polinomial de 3ªordem. No tapete, o inverso é verificado. Isto poderá ser explicado pela amplitude de movimento e pelo tempo implicado em cada um dos movimentos. Os picos de aceleração são sempre verificados nos perfis trapezoidais, mas enquanto que estes picos, no manipulador, se transcrevem numa maior transferência de calor, nos tapetes essa maior transferência não é garantida porque o perfil polinomial embora não atinja valores elevados de aceleração, apresenta valores de aceleração maioritariamente distantes de valor nulo, ao contrário do trapezoidal que admite uma velocidade constante entre a aceleração e desaceleração.

Assim, embora o perfil polinomial seja mais suave sem aumentos ou diminuições bruscas de velocidade, este poderá apresentar uma variação de velocidade o suficientemente constante ao longo de todo o tempo do percurso, para contraste do perfil trapezoidal, ditando assim uma maior transferência de calor. A discrepâncias entre os torques e transferências de calor implicados nos 2 perfis, definem-se assim pela aceleração máxima delimitada no perfil trapezoidal e ainda na amplitude de movimento.

Uma das particularidades do estudo de diferentes tipos de embalagens tem a ver com o facto de diferentes momentos de inércia poderem ser analisados. Uma maior massa implicará obrigatoriamente uma maior inércia exercida nos motores. Este aspeto tem assim uma maior significância quando o tapete de modelação número 2 é considerado, uma vez que as perdas consideradas são muito maiores.

Tabela 18 - Tempos de percurso para as diferentes embalagens

Embalagem	Temporização(s)

	Tapete 1	Tapete 2
Caixa 1 - 2 kg	7.98969	8.0321
Caixa 2 - 2.5 kg	7.86907	7.98287
Caixa 3 - 3 kg	7.77209	7.94024

Quanto maior a massa da embalagem, menor o tempo de percurso. Estes resultados podem à primeira vista parecer contraditórios, uma vez que seria expectável, por ventura, um tempo de percurso mais demorado em proporcionalidade com o peso da embalagem a ser transportada. O inverso é verificado e isto poderá ser explicado pela resposta do sistema tendo por base os mesmo valores de atribuição aos parâmetros do controlador. Assim, se se mantiver os parâmetros, uma maior massa a ser transportada exigirá uma resposta mais robusta e de forma consequente mais rápida. Claro que, em contrapartida, haverá um maior *overshooting* e possivelmente um maior desequilíbrio na resposta dada. Perante estas condições, o ideal, de forma a manter-se uma resposta continuamente equilibrada independentemente do peso transportado, seria um *tuning* automático dos valores dos parâmetros do controlador, e aí sim, para uma maior massa, maiores tempos de percurso seriam verificados. O uso de valores iguais de parâmetros para diferentes massas acaba no entanto, por não comprometer a integralidade do sistema, uma vez que as diferentes massas utilizadas apresentam valores muito próximos entre si, garantido dessa forma, respostas adequadas independentemente do tipo de embalagem.

8.3.Gripper

O *gripper* que possa ser conectado ao robot e ser assim utilizado para recolher a embalagem explorou a mímica de uma garra com três dedos, um dedo central disposto lateralmente e outros dois dispostos de forma simétrica no lado oposto. A modelação do mesmo foi elaborada tendo em mente que no sistema físico se encontram incorporados sensores de posição, de força e ainda atuadores elétricos, mais concretamente motores DC.

8.3.1. *Design* do *Gripper*

A estrutura do *gripper*, como anteriormente referido, apresenta 3 dedos. Uma categorização em similitude com a mão humana poderá ser traduzida em o dedo médio e o dedo indicador de um lado do *gripper* e ainda o dedo polegar do outro. Atuadores elétricos encontram-se

instalados adjacentes à primeira junta da base dos dedos, conectados ao pulso. As juntas de cada um dos dedos apresentam 1 grau de liberdade cada uma, providenciando um movimento rotacional. Cada um dos dedos apresenta assim, na sua totalidade, 3 graus de liberdade.
Assim, as seguintes considerações são tomadas:

- Os 3 contactos disponíveis devem ser aplicados simultaneamente no objeto, uma aplicação síncrona das constrições pode ajudar a reduzir movimentos descontrolados.

- Os elementos de contacto devem alojar um conjunto de sensores de proximidade facilitando o controlo dos procedimentos de aproximação, assim como sensores de força auxiliando também no controlo das forças de contacto envolvidas.

- A existência de apenas 3 atuadores, de maneira a limitar o tamanho e peso, e limitar a complexidade do sistema na extremidade do braço. Um atuador para cada dedo sendo cada um dos dedos atuado de forma independente.

Cada dedo é composto por três falanges, sendo cada uma das falanges conectadas por tendões/correias dentadas e por polias.

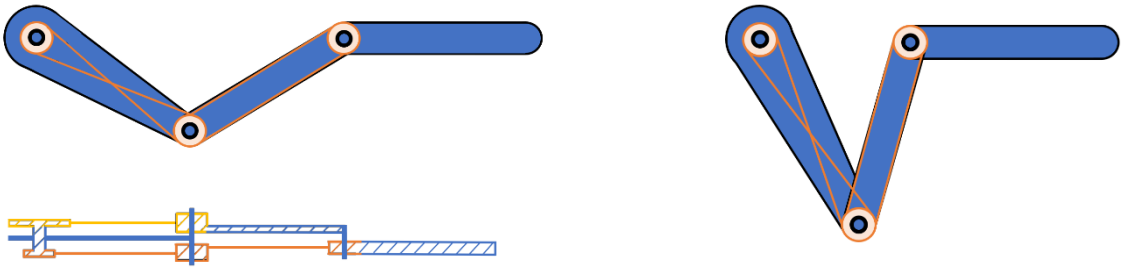


Figura 154 - Estrutura do dedo com vista transversal

A matriz do primeiro eixo encontra-se associada ao pulso, sendo o único eixo da estrutura acionado diretamente pelo servomotor. Os eixos das restantes polias são acionados por transmissão mecânica. A transmissão mecânica foi elaborada por forma a que a terceira, e última falange, se disponha de forma continuamente paralela à posição inicialmente assumida, com uma disposição paralela à superfície da embalagem a ser transportada. Dessa forma, entende-se que a área da superfície de contacto para com a embalagem a ser agarrada seja maior, permitindo que a ação seja executada de forma mais compacta e segura.

Relativamente à distribuição das polias ao longo das 3 falanges interligadas, 2 divisões na estrutura podem ser consideradas. Numa primeira, que se encontra na zona superior da imagem da vista transversal, tem-se apenas 2 polias, que ditam o posicionamento e orientação respetivo da 2ª falange. Enquanto na secção de baixo da imagem em corte, 3 polias são apresentadas, sendo responsáveis por ditar a orientação da terceira falange. Pode-se assim considerar que numa primeira fase, a posição e orientação das duas primeiras falanges é conseguida, sendo a orientação da terceira falange posteriormente ajustada, em função dos perfis das falanges precedentes.

Tabela 19 - Polias pertencentes a cada um dos dedos e respetiva variação angular

Secção	**Deslocamento Angular**			
	1ª Polia	**2ª Polia**	**3ª Polia**	**Ângulo de Saída**
Superior	α	2α	----	**3α**
Descrição	Polias que impulsionam o movimento cinemático da primeira e segunda falange, sendo o deslocamento angular absoluto total, equivalente à **soma** do movimento da primeira com a segunda polia			
Inferior	α	-2α	-3α	**-3α**
Descrição	Polias que determinam a orientação final da terceira falange. O deslocamento angular absoluto da terceira falange é ditado pela última polia.			

De forma a manter a mesma orientação no espaço, o deslocamento angular da junta no início da terceira falange tem que ser o valor inverso observado para o deslocamento angular absoluto da segunda falange. Com isso em mente, a proporção dos deslocamentos angulares foi estabelecida, em conformidade com os valores inseridos na tabela em cima, e a partir daí as proporções dimensionais das diferentes polias foram retiradas.

Tabela 20 - Polias pertencentes a cada um dos dedos e respetiva proporção dimensional

Secção	Diâmetro		
	1ª Polia	2ª Polia	3ª Polia
Superior	2d	d	----
Inferior	2d	d	2/3d

8.3.2. Modelação

De forma a modelar o *gripper* que se encontra acoplado ao pulso do braço robótico, foi necessário desenvolver, não só um sistema de controlo que inclua o movimento das juntas das falanges, visando a abertura e fecho do *gripper* aquando da tarefa de *Pick and Place,* como também o controlo de força a ser exercido aquando do fecho do mesmo. Tudo isto implica um sequenciamento do controlo por diferentes fases:

- Num estágio inicial é elaborado um controlo por *feedback* com um *setpoint* angular como referência, procurando definir desta forma o fecho do *gripper* e a aproximação dos dedos à embalagem.
- Numa segunda fase, tem-se um controlo por feedback com um *setpoint* de força como referência. Aqui, o contacto é estabelecido assim como uma força de compressão, permitindo pegar na embalagem. De forma simultânea a esta fase, tem-se o transporte da embalagem por parte do braço, entre o tapete de admissão e o de saída.
- Por fim, um *setpoint* de referência angular é novamente definido, tentando replicar a abertura do *gripper* e a colocação da embalagem.

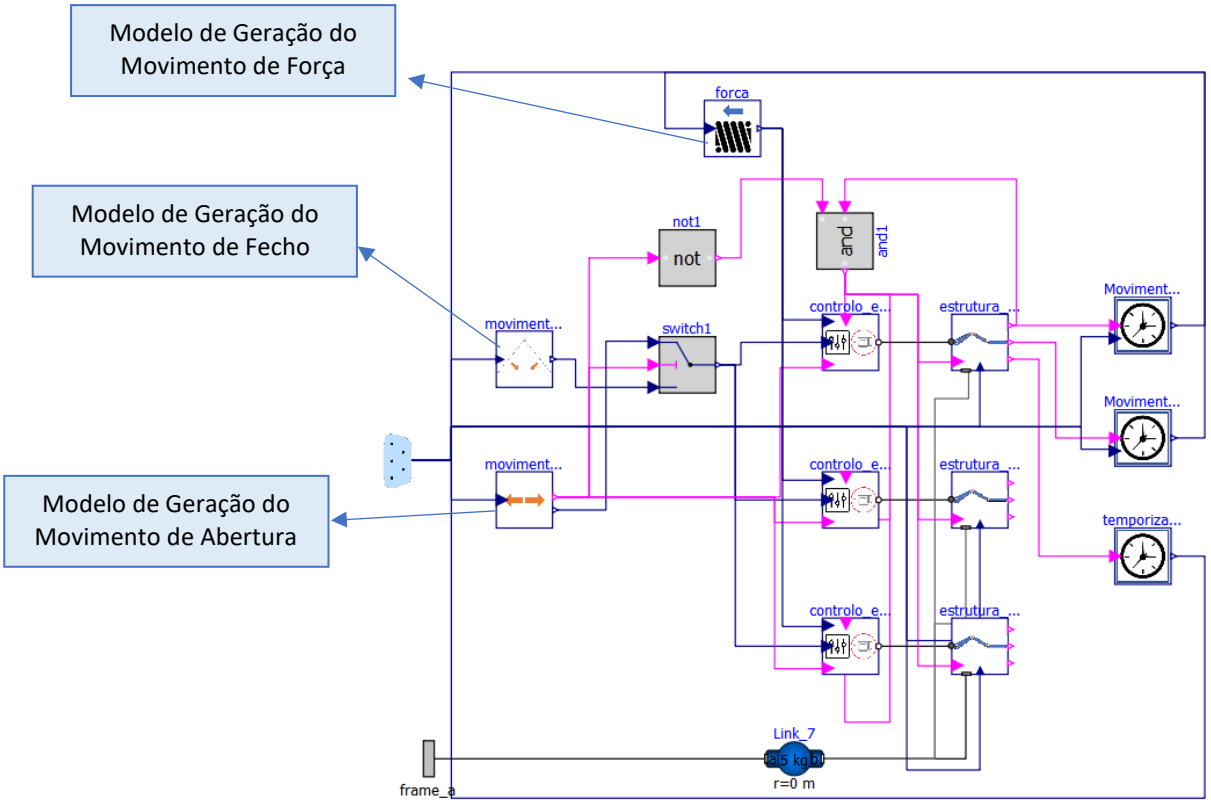


Figura 155 - Modelo do *gripper*

Na Figura 155 é possível observar o modelo do *gripper* onde se pode desde logo distinguir várias instâncias de modelação, como por exemplo os modelos do movimento de fecho, de força e de abertura, que expressam os sinais enviados para o sistema de controlo e que ditam o movimento da estrutura.

Relativamente à modelação de cada dedo, associou-se à 3ª falange uma junta prismática, com uma mola e amortecedor em paralelo, com o intuito de simular o contacto decorrente do *gripper* com a embalagem a ser transportada.

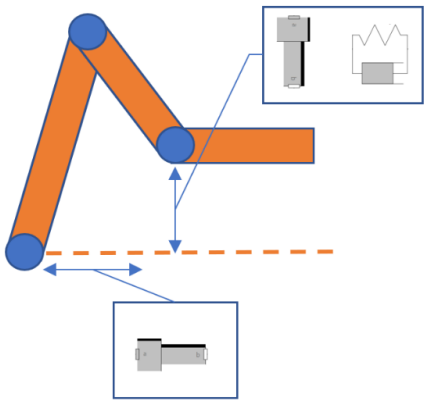


Figura 156 - Componentes específicos utilizados na modelação da estrutura do dedo

A junta prismática, onde o eixo da mola se encontra acoplado, permite que o movimento da mola seja perpendicular à superfície da terceira falange, cuja orientação se mantém constante ao longo do tempo. Desta forma, a força a ser calculada com recurso à mola e amortecedor, será também ela perpendicular à superfície de contacto do dedo com a embalagem. A mola e o amortecedor são atuados ao longo da 3ª falange e encontram-se conectados ao eixo do *gripper*, em que no mesmo apenas movimento no sentido do eixo é permitido, como expresso através da utilização de uma 2ª junta prismática. Pela 3ª lei de Newton, a mola aplica uma força igual e oposta à força aplicada pela falange. Na junta prismática, o deslocamento x ao longo do eixo constitui o deslocamento de qualquer ponto ao longo da 3ª falange e $v = dx/dt$ é a respetiva velocidade. A força atuada pela falange na mola amortecida ao longo da junta prismática pode ser expressa da seguinte forma:

$$F = -k(x - x_0) - bv \tag{73}$$

Em que os parâmetros admitidos são a constante elástica da mola k, o comprimento natural da mola x_0 e ainda a constante de amortecimento b.

Lógica

A lógica expressa na estrutura apresenta, ainda, a sua complexidade, decorrente da necessidade de modificar não só um dos três perfis de movimento, abertura, força ou fecho, como também a própria estrutura mecânica, ao logo do tempo, com a atribuição das propriedades da mola quando o perfil de força é iniciado.

Quando o deslocamento do braço até ao ponto de recolha é finalizado, um sinal do instante de tempo em que o mesmo ocorre, adota um valor diferente de 0 e dá entrada no modelo de hierarquia superior. Este por sua vez entra no modelo de geração do perfil de movimento para o fecho, com a ativação de um perfil trapezoidal e envio respetivo para o modelo de controlo.

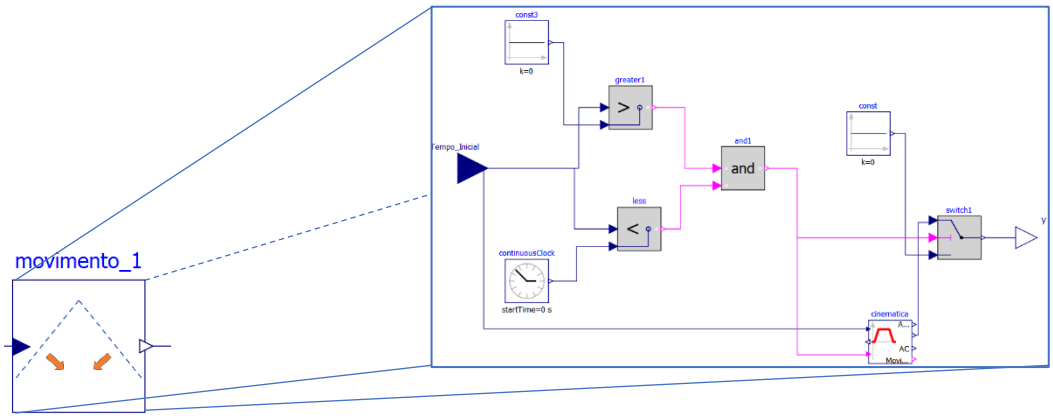


Figura 157 - Modelo de geração do perfil de movimento para o fecho

No modelo de controlo, são apresentadas 2 entradas à esquerda, a entrada cimeira define o perfil de força e a entrada inferior define os perfis de movimento (no modelo de hierarquia superior, os perfis de movimento de fecho ou abertura, têm a sua entrada neste modelo determinada por um *switch*).

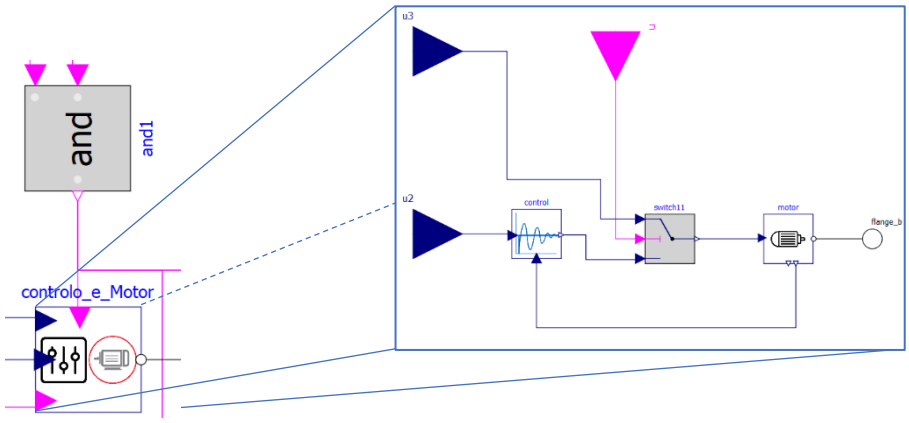


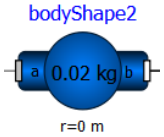
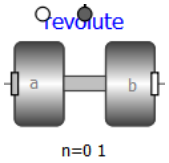
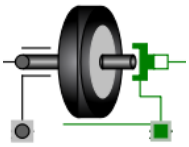
Figura 158 - Modelo central de controlo do *gripper*

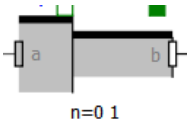
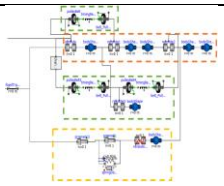
Os perfis são assim gerados e enviados como comandos para o motor, acionando o mesmo de acordo com a etapa de operação em que o *gripper* se encontra. Por fim, o movimento é transmitido às falanges através da flange de saída do modelo ilustrado em cima.

Na modelação da estrutura final do *gripper*, os respetivos componentes foram modelados da seguinte forma.

Tabela 21 - Componentes utilizados na modelação da estrutura final do *gripper*

Componentes	Ilustração	Descrição

Falange		Uso de *bodyshapes* para cada uma das falanges, admitindo os valores de massa, centro de massa, massa total e momentos de inércia. Alguns *bodyshapes* foram ainda adicionados também ao modelo, para efeitos de executabilidade do mesmo, mas sem interferência significativa na sua *performance*.
Junta Rotativa		Representa os eixos de rotação ao longo da estrutura. Sendo o primeiro acionado diretamente pelo servomotor e os restantes por transmissão através das polias e correias. Na segunda articulação, são elaborados 2 eixos, um para o deslocamento angular que constitui o movimento cinemático da segunda falange e um segundo eixo/junta rotativa que caracteriza o movimento angular a ser transmitido à polia inferior e, dessa forma, o índice de transmissão desejado é assegurado.
Polia		Constitui as diferentes polias distribuídas ao longo da estrutura. As polias modeladas são consideradas dentadas, onde se parametriza o diâmetro eficiente, o número de dentes e ainda o momento de inércia.
Correia		Correia que transmite momento entre as polias. A elasticidade da mesma poderá ser considerada com a definição de uma constante elástica.

Junta prismática		Juntas prismáticas que definem as constrições físicas do movimento da mola ao longo do movimento do gripper, com a intenção que a força exercida na mesma seja exclusivamente na vertical.
Mola c/ Amortecimento		Mola amortecedora, implicada na força necessária para que o *gripper* agarre na embalagem.

A modelação da estrutura do dedo pode ser visualizada em baixo. Para uma melhor perceção da modelação, esta encontra-se dividida em diferentes partes. Na componente realçada a laranja encontram-se as juntas e as falanges, que poderão ser posteriormente visualizadas nas imagens do modelo simulado. As secções realçadas a verde dizem respeito às secções superiores e inferiores das polias e correias responsáveis pelo acionamento juntas e movimento das falanges. A secção amarela diz respeito aos componentes que sujeitam a estrutura a uma série de constrições cinemáticas, nomeadamente juntas prismáticas e uma mola amortecedora que é ativada, exclusivamente, quando o *setpoint* de força é admitido no controlo, tentando replicar o contacto com a embalagem. O desenvolvimento da modelação que visa a estrutura foi parcialmente inspirada num modelo desenvolvido em (Ferretti, Magnani, Rocco, & Viganò, 2006).

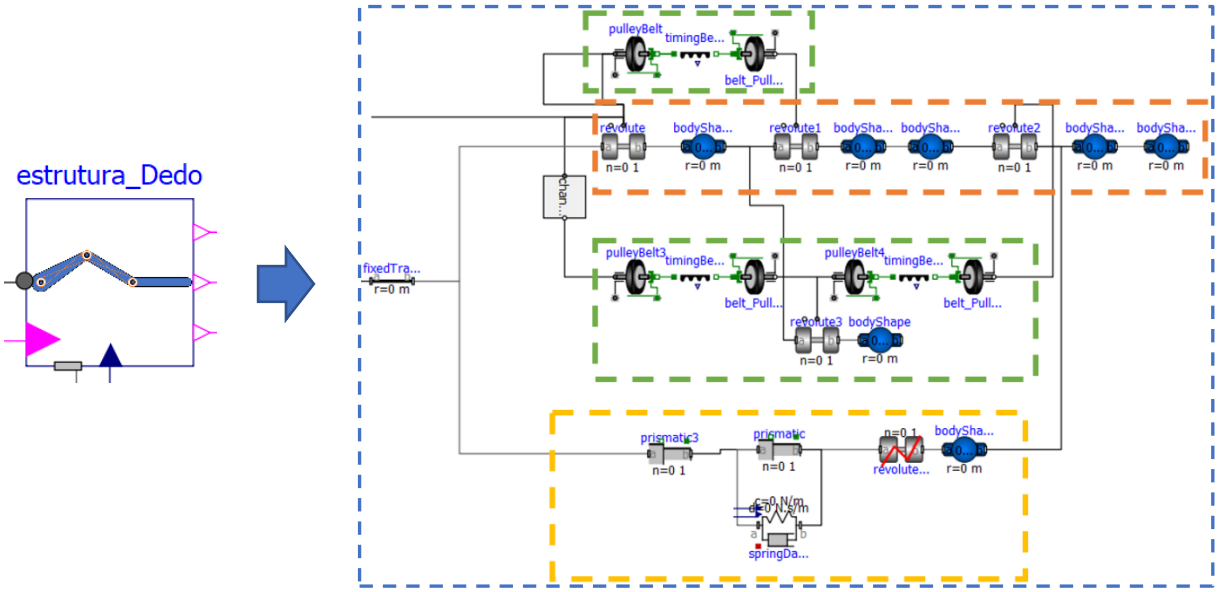


Figura 159 - Modelação da estrutura de um dedo

A mola amortecedora foi modelada de maneira que os parâmetros k (constante elástica) e b (constante de amortecimento) variem ao longo do tempo, em conformidade com os *inputs* recebidos no modelo da mesma.

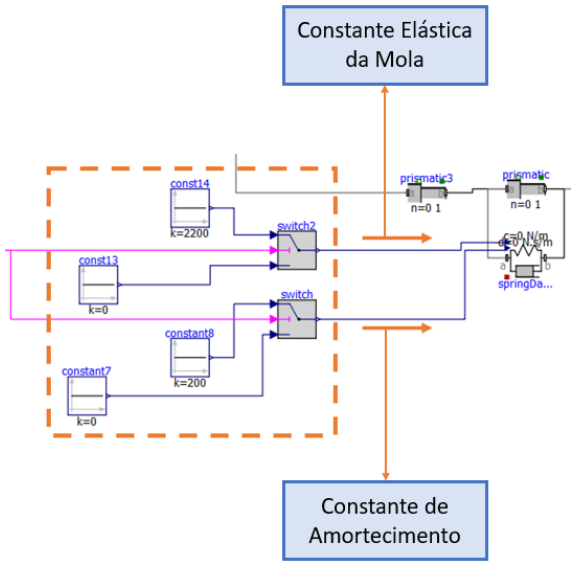


Figura 160 - Modelação de entradas no componente da mola amortecedora

Existem assim 2 entradas no componente, uma para cada parâmetro, cujos valores são determinados por um par de *switchs*. Cada um desses *switchs* é ativado por um booleano, sendo que um valor diferente de 0 é admitido quando a força de compressão é realizada para pegar na embalagem, em contrapartida com um valor de 0 quando o movimento de *gripper* se quer livre, quer para aproximar os dedos antes da força de compressão, quer para abertura dos dedos, e permitir que a embalagem seja largada.

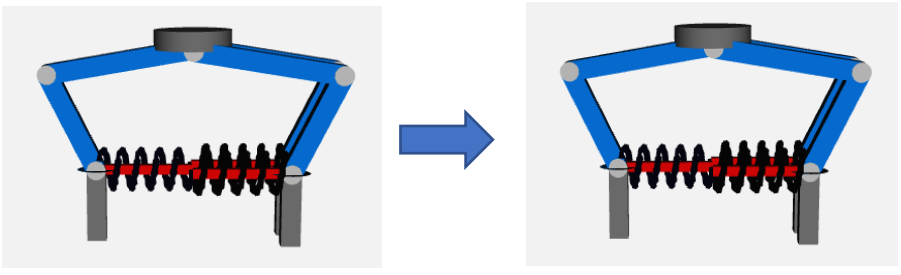


Figura 161 - Ilustração do gripper em simulação no momento de abertura e no momento de fecho

8.3.3. *Gripper* – Controlo

Foi abordado o facto de existir um modelo de controlo com 3 entradas que definem os comandos para o motor, dependendo da etapa de operação em que o *gripper* se encontra, sendo que apenas um deles se apropria, de cada vez, do sinal de controlo enviado ao motor. A definição dos instantes em que, ou o comando de movimento de fecho, ou o comando de força ou o de movimento de abertura são definidos, é garantido por um conjunto de feedbacks associados no modelo da estrutura do *gripper* e que acabam por atuar como sensores de variação angular da primeira junta, permitindo perceber quando os diferentes movimentos de fecho, força e abertura são completados.

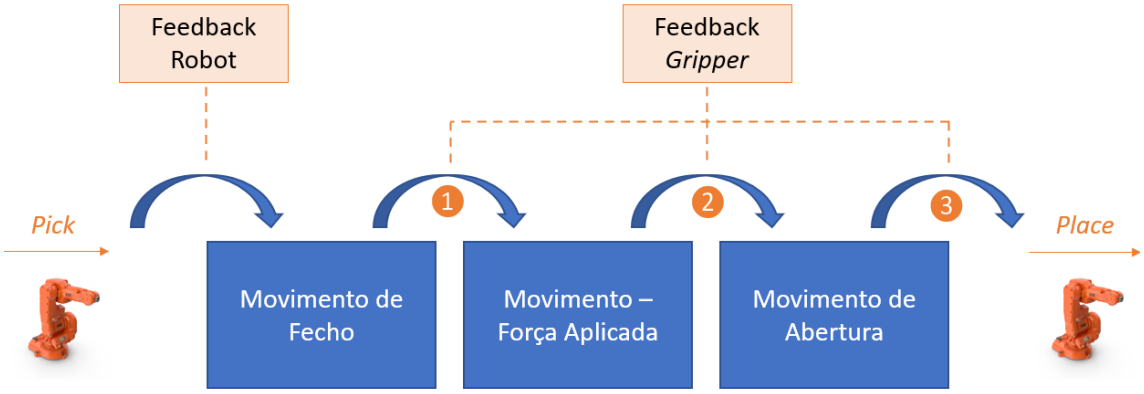


Figura 162 - Feedbacks requeridos na modelação de forma a permitir uma correta ordem nos movimentos

Os diferentes *feedbacks* presentes no modelo da estrutura do *gripper* assumem-se como sinais booleanos que admitem o valor de verdadeiro quando cada um dos movimentos é consumado. Estes sinais booleanos foram desenvolvidos com base num conjunto de modelos lógicos, cujo estado é definido pelos valores angulares das juntas, no caso dos movimentos de fecho e abertura, e ainda pelo valor de força no componente da mola amortecedora, no caso do movimento de força.

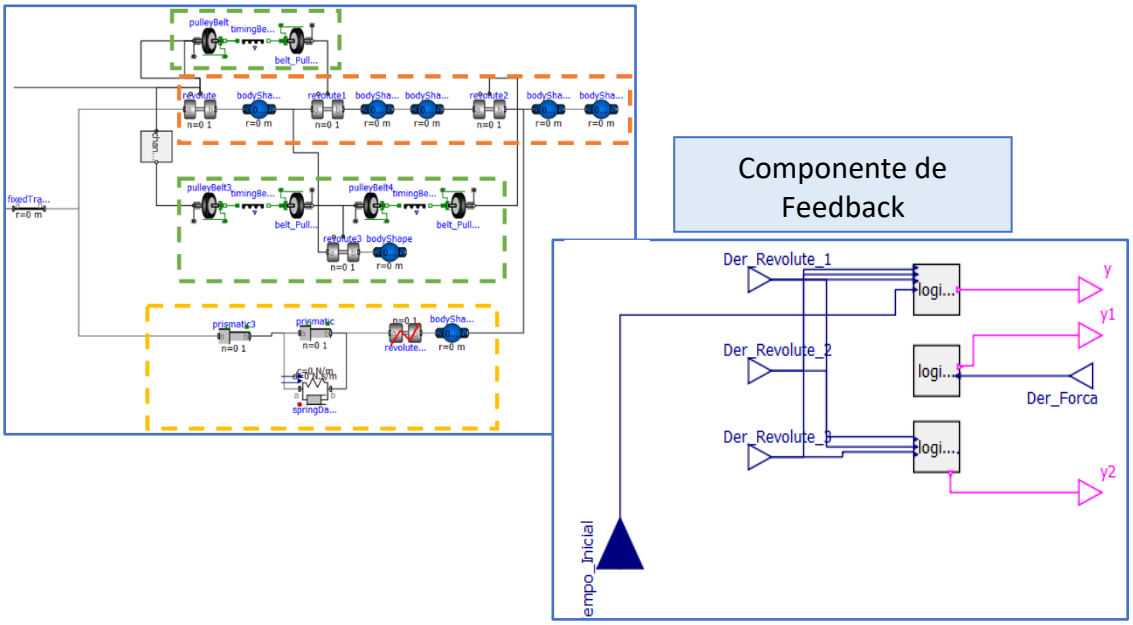


Figura 163 - Junto à modelação da estrutura de um dedo, uma secção de feedback foi desenvolvida

Na Figura 163, no que diz respeito ao componente de feedback, é possível observar as 4 entradas a azul, que se relacionam com as derivadas da posição angular das 3 juntas ao longo do dedo e ainda a derivada da força aplicada quando o movimento para agarrar a caixa é iniciado. Estes 4 valores funcionam como entradas nos modelos lógicos, que serão analisados de seguida, e à saída de cada um deles tem-se os *feedbacks* booleanos (a rosa) que retratam o estado do final de curso de cada movimento.

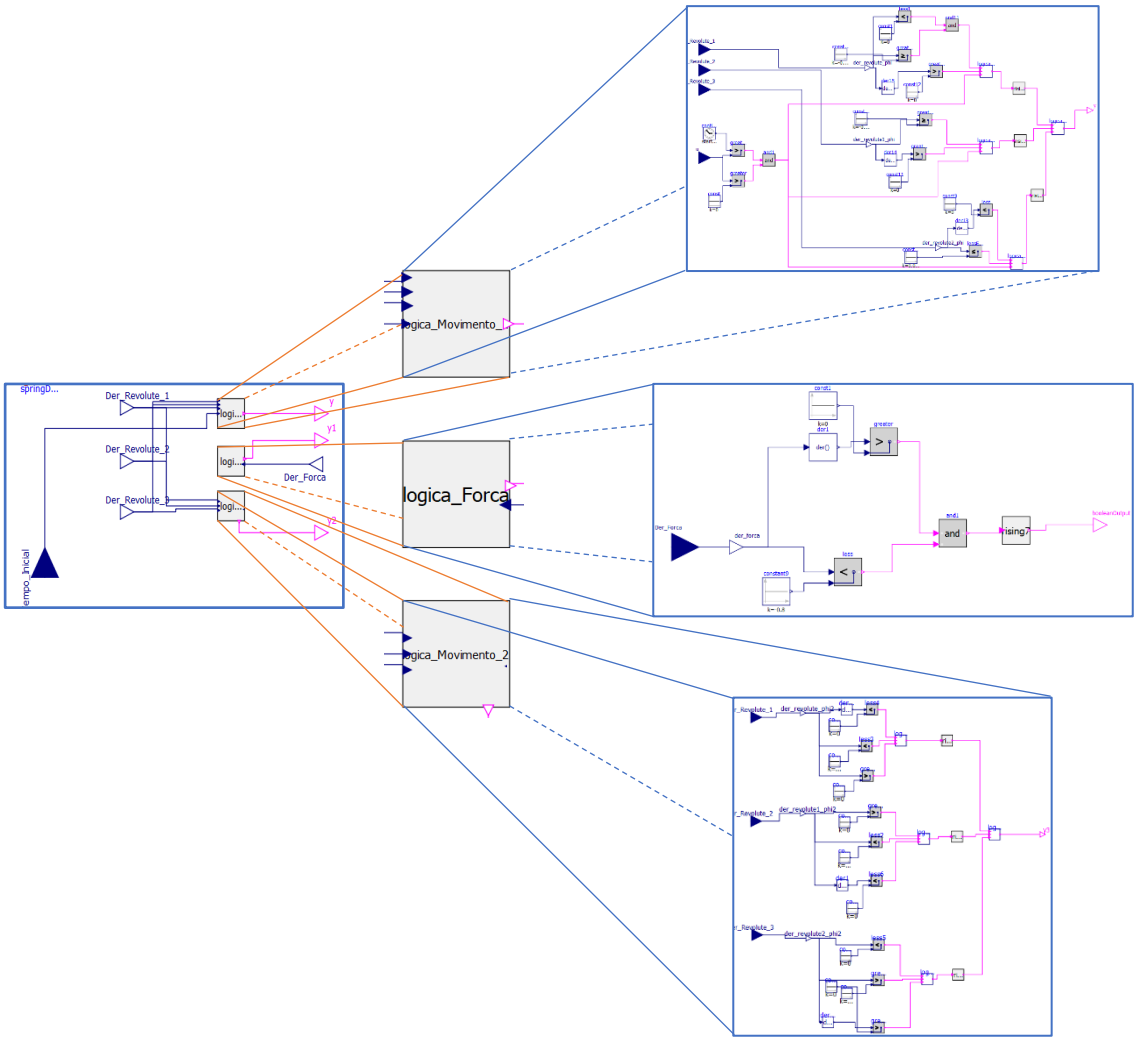


Figura 164 - As 3 modelações associadas aos blocos lógicos de cada movimento executado pelo *gripper*

Na Figura 164, é possível observar as 3 modelações inerentes à lógica programada dos 3 feedbacks de saída. Para o efeito foram utilizados blocos referentes às expressões matemáticas de maior e menor, blocos da condição lógica "e" quer para 2 como para 3 expressões (bloco "e" para 3 expressões foi elaborado de raiz, dada a sua ausência na biblioteca) e ainda um bloco realizado de raiz, que expressa um comportamento de *rising edge* em um valor de verdadeiro à saída do bloco é adotado quando uma transição falso → verdadeiro é verificada. O estado da saída apresenta ainda a particularidade de se tornar verdadeiro por tempo indefinido, após um *rising edge* se concretizar. A significância deste bloco é expressa assim por um interesse num único *rising edge* a ser verificado, apresentando assim uma posição neutra para com *rising edge* posteriores.

Tabela 22 - Estados lógicos associados à modelação do bloco lógico do movimento de fecho

Modelo	Variável	Lógica				
		Condição 1	Condição 2	Condição 3	Condição 4	Condição 5
Movimento Fecho	Derivada Junta 1 (x1)	(x1) < 0	*and*	*and*	*Rising Edge*	*and*
		(x1) >= -0.0002				
		Der(x1) > 0				
	Clock (c)	(c) > (ti)	*and*			
	Tempo Inicial (ti)	(ti) > 0				
	Derivada Junta 2(x2)	(x2) >= -0.001	*and*	*and*	*Rising Edge*	
		Der(x2) > 0				
	Clock (c)	(c) > (ti)	*and*			
	Tempo Inicial (ti)	(ti) > 0				
	Derivada Junta 3(x3)	(x3) <= 0.0004	*and*	*and*	*Rising Edge*	
		Der(x3) < 0				
	Clock (c)	(c) > (ti)	*and*			
	Tempo Inicial (ti)	(ti) > 0				

Tabela 23 - Estados lógicos associados à modelação do bloco lógico do movimento de força

Modelo	Variável	Lógica		
		Condição 1	Condição 2	Condição 3
Movimento Força	Derivada Força (x)	Der(x) > 0	*and*	*Rising Edge*

		(x) < -0.8		

Tabela 24 - Estados lógicos associados à modelação do bloco lógico do movimento de abertura

Modelo	Variável	Lógica				
		Condição 1	Condição 2	Condição 3	Condição 4	Condição 5
Movimento Abertura	Derivada Junta 1 (x1)	(x1) < 0.001 (x1) > 0 Der(x1) < 0	*and*	*and*	*Rising Edge*	*and*
	Derivada Junta 2(x2)	(x2) > 0 (x2) < 0.001 Der(x2) < 0	*and*	*and*	*Rising Edge*	
	Derivada Junta 3(x3)	(x3) < 0 (x3) > -0.002 Der(x3) > 0	*and*	*and*	*Rising Edge*	

8.3.4. Estudo de Simulações – *Gripper*

Alguns resultados derivados das simulações decorrentes do *gripper* são aqui apresentados.

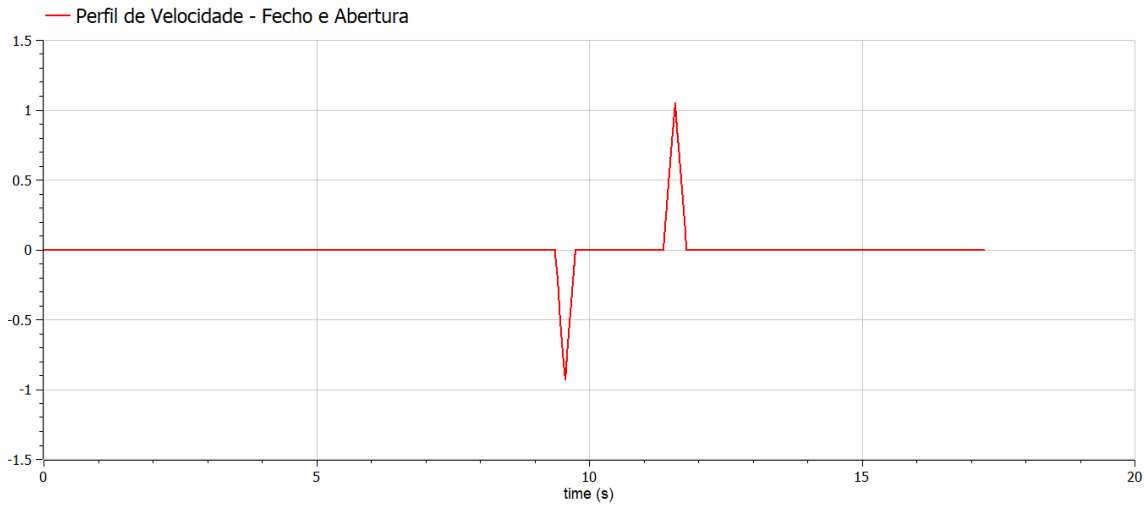


Figura 165 - Perfis de velocidade gerados para o fecho e abertura do gripper

O perfil de velocidades calculados nos modelos de geração de trajetória dentro do sistema de controlo do *gripper*, adotam um perfil de velocidade triangular. Pode-se considerar que o deslocamento realizado nas duas fases não é suficientemente grande, de forma a atingir a velocidade máxima definida (2.5 rad/s) e perfazer um perfil trapezoidal.

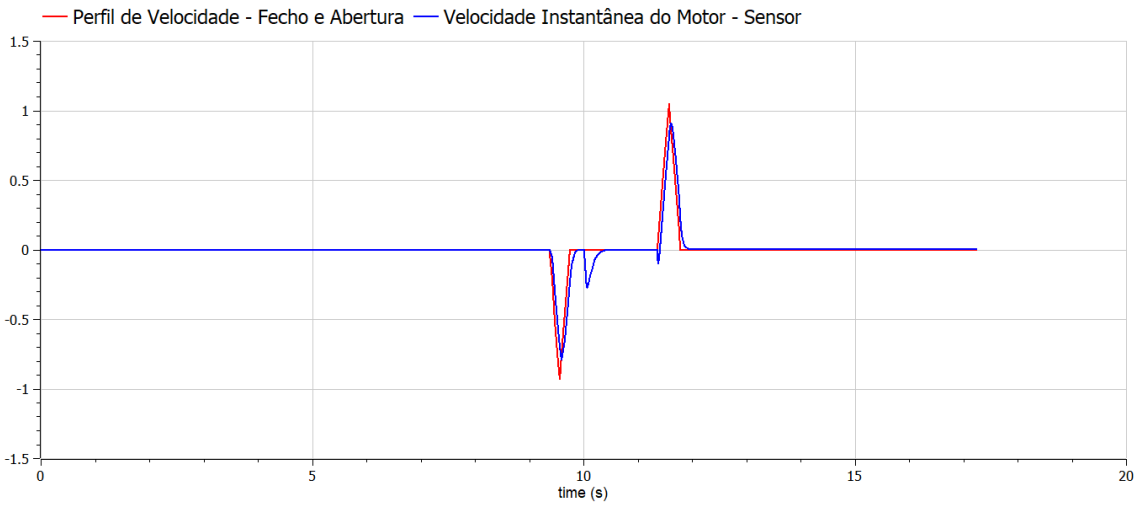


Figura 166 - Perfil de velocidade gerado e velocidade medida

No gráfico em cima é possível observar o perfil de velocidades de fecho e abertura e a velocidade medida no motor. Facilmente se verifica um certo desfasamento entre a velocidade desejada e a obtida, fruto de um controlo focado na velocidade de forma exclusiva e entendido como suficiente para o pretendido. Entre os dois perfis triangulares é possível observar ainda uma variação da velocidade, consequente da fase destinada ao comando de força, em que um pequeno deslocamento angular é garantido de modo a que a embalagem seja agarrada com firmeza.

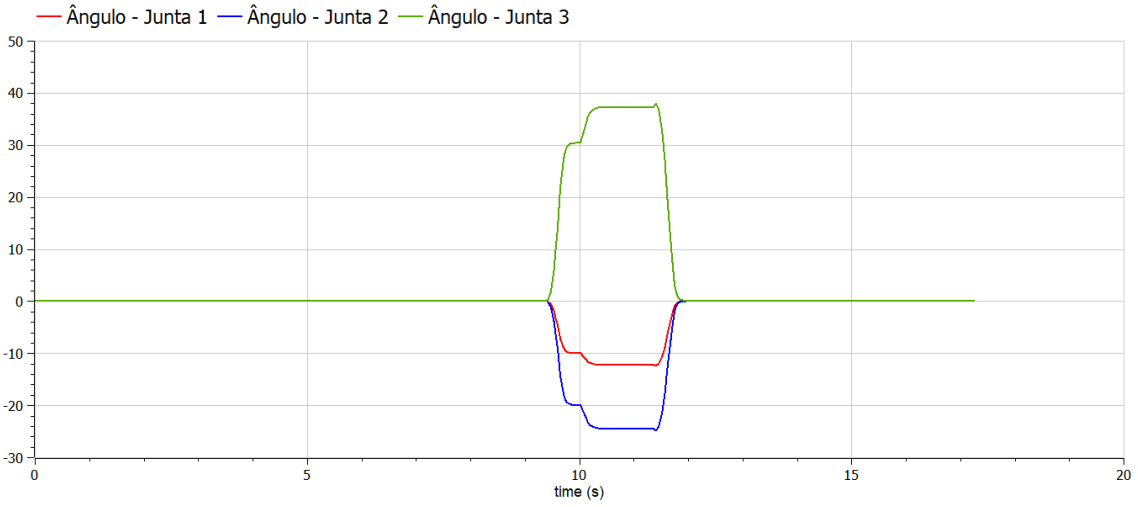


Figura 167 - Variação angular das 3 juntas presentes em cada dedo

Observando os valores de variação angular das diferentes juntas no *gripper* é possível observar uma variação numa fase inicial, coincidindo com o fecho do *gripper*. Quando a velocidade está próxima do 0 absoluto e a posição de referência é alcançada, então tem-se uma nova variação, desta vez de menor tamanho, representativa do perfil de força a ser aplicado. Atingida a força de atuação pré-definida, segue-se um período de variação nula, em que a embalagem é devidamente transportada pelo braço robótico sendo, por fim, colocada no tapete de saída através da abertura do *gripper*, em que uma variação angular, em sentido inverso às fases anteriores, é verificada.

Entre a fase de transporte/variação angular nula e a abertura do gripper, é possível observar uma pequena protuberância, facilmente explicada pela mudança brusca das propriedades da mola, modelada com o intuito de replicar a força a ser aplicada na embalagem a ser transportada.

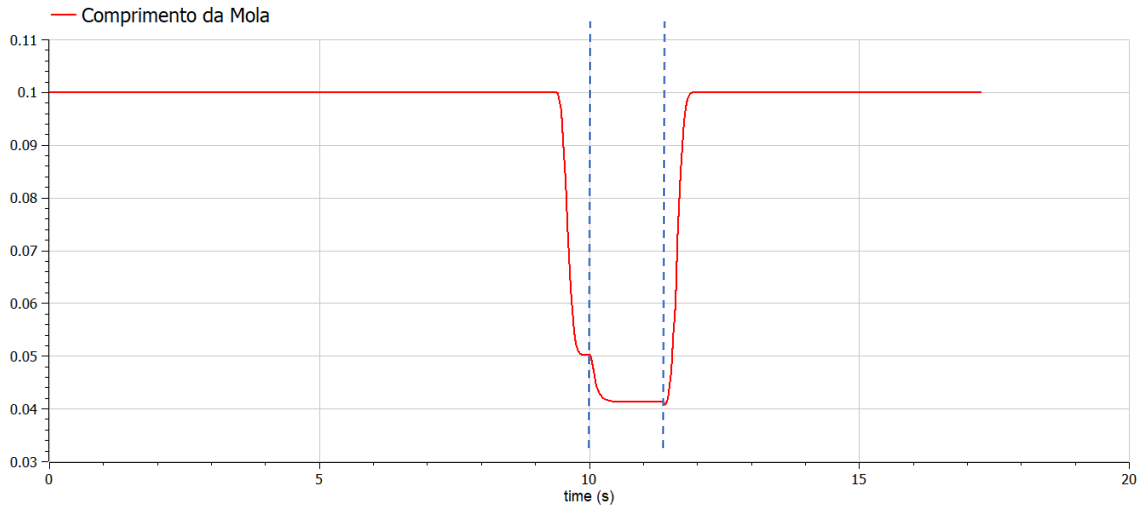


Figura 168 - Variação do comprimento da mola amortecedora

Variação do comprimento da mola, em que os momentos inicial e final de variação de comprimento poderão ser descartados, pois correspondem aos instantes em que as propriedades físicas da mola são igualadas a 0 e tentam dessa forma replicar um movimento livre das juntas e falanges, sem qualquer constrição elástica ou de amortecimento provocado pela mola.

8.4.Modelos Auxiliares

Além dos aspetos físicos a serem modelados, onde se destaca o braço robótico, os tapetes, resistências de sinalização e *gripper*, foi modelado um sistema paralelo de controlo, visando o registo da sequenciação de todo o processo, procurando sempre que necessário a partilha de variáveis entre os diferentes modelos subjacentes ao sistema global.

Relativamente à simulação do tapete de entrada, do braço robótico e ainda do tapete de saída estes não ocorrem simultaneamente, ou seja, o cálculo das variáveis ao longo do tempo no braço são posteriores ao cálculo das variáveis envolvidas no sistema do tapete de entrada, sendo o mesmo válido para o tapete saída relativamente ao braço robótico.

Numa fase inicial, o estado de sequenciação não foi considerado, pois não é de maior pertinência apresentar os valores de simulação do braço, por exemplo, no período exato ao longo da escala de tempo do ciclo de funcionamento, uma vez que o foco visa a análise de cada um dos subsistemas de forma separada.

O possível interesse nos tempos de execução envolvidos em cada um dos subsistemas quando operados de forma sequencial, desencadeou numa fase inicial o desenvolvimento de um conjunto de modelos, visando não só apresentar os tempos totais envolvidos no funcionamento de cada um dos subsistemas, mas também os tempos em que o funcionamento de cada um dos subsistemas é iniciado e finalizado ao longo da escala de tempo de um ciclo de funcionamento do sistema total.

Os mesmos foram organizados segundo o cálculo do tempo em que o subsistema é iniciado, o tempo de funcionamento e o tempo de término, sendo apresentados posteriormente, de forma dinâmica, num diagrama através da utilização de um bloco interativo.

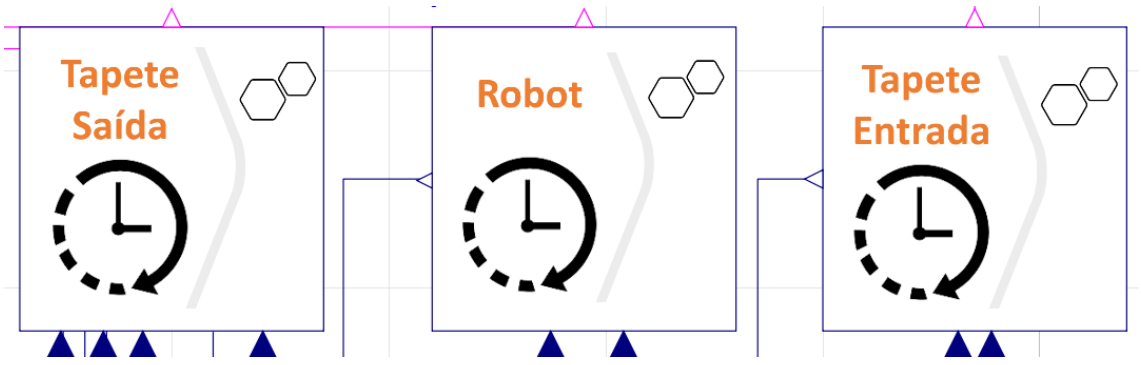


Figura 169 - Modelos destinados à temporização dos 3 estágios

- **Tapete de Entrada**

Começando pelo tapete de entrada, uma vez que o tempo inicial do mesmo corresponde ao tempo de início da simulação do sistema total, o tempo de funcionamento iguala-se ao tempo

em que o subsistema termina, pelo que apenas o tempo inicial (0) e o tempo de percurso da caixa ao longo do tapete, são apresentados.

O tempo de percurso apresentado é calculado através da soma dos tempos de saída em cada um dos tapetes de entrada (1 e 2), dado que apenas um deles se encontra em funcionamento e o valor de saída do tapete inativo é 0.

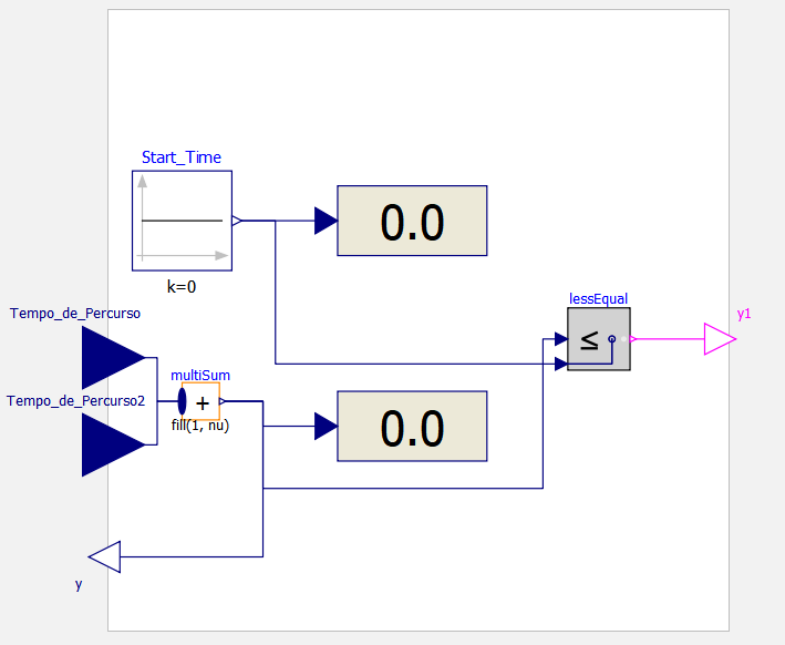


Figura 170 - Modelo de temporização do tapete de entrada

O valor obtido do tempo de funcionamento é ligado não só ao bloco dinâmico de visualização, mas também a um *output* real de saída do modelo e que dará entrada como valor inicial no modelo de temporização do robot. O mesmo ainda é ligado a um bloco lógico de "menor ou igual", ativando um booleano enquanto o valor de percurso é 0, ou seja, enquanto o tapete de entrada ainda se encontra em funcionamento e o valor de tempo do percurso total ao longo do mesmo ainda não deu entrada no modelo.

- **Robot**

O primeiro *Input* real a dar entrada neste modelo, é o valor de saída proveniente do modelo de temporização do tapete de entrada. Assim, o valor configurado no bloco dinâmico de visualização superior é o instante de tempo em que o braço robótico inicia o seu movimento ou, no caso do modo manual ativo, o transporte da embalagem manualmente. O segundo Input real a conectar-se com o modelo é o valor do tempo total de funcionamento do braço para um ciclo de funcionamento, ou seja, o tempo de execução do *Pick&Place*.

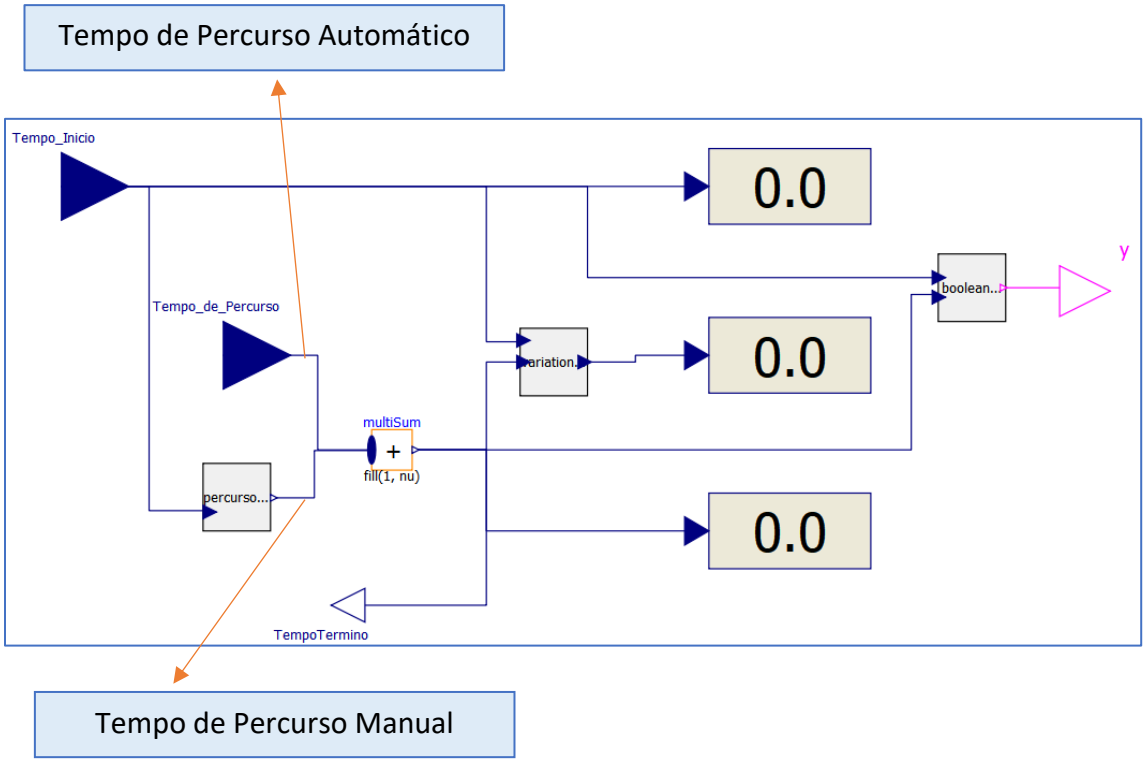


Figura 171 - Modelo de temporização do transporte *Pick&Place*

Além do tempo de execução *Pick&Place* recorrendo ao braço robótico. Existe um bloco que determina o tempo de transporte caso o modo manual seja adotado.

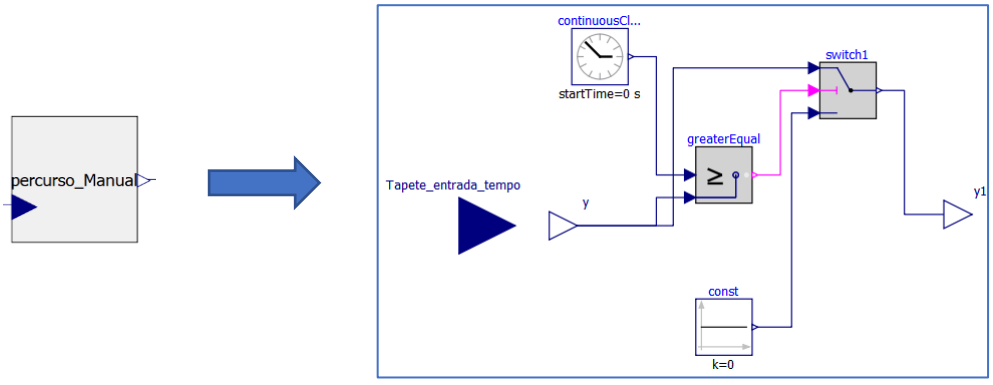


Figura 172 - Modelo que determina o tempo do percurso manual

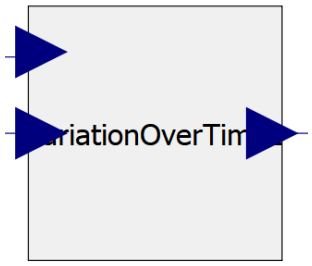
Neste bloco, um *input* dá entrada no *text view* assumindo o valor aleatório de transporte calculado no modelo do modo manual, acompanhado por um *input*, visível em diagrama, correspondente ao valor de término do tapete de entrada. Uma variável *y* é calculada com base nos 2 valores mencionados adotando, na existência de um booleano verdadeiro para o modo manual, a soma dos 2 valores referidos e adotando o valor de 0 caso o booleano seja falso.

```
Modelica.Blocks.Interfaces.RealOutput y = if Manual and Tapete_entrada_tempo<>0 then
(percurso_manual_tempo + Tapete_entrada_tempo) else 0   annotation( ...);
```

Figura 173 - Variável definida no bloco

Esta variável, correspondendo ao instante de tempo em o transporte manual finaliza, é admitida, por último, à saída do bloco, apenas quando o tempo de simulação atinge esse mesmo valor.

O valor proveniente do modelo de cálculo de término do transporte manual encontra-se ligado a um bloco de soma, dando entrada de forma paralela com o tempo de fim de percurso do movimento do braço (sabendo que, independentemente do modo de transporte, pelo menos um destes valores será sempre 0). O valor proveniente do bloco de soma dá entrada direta ao visualizador dinâmico de término do transporte *Pick&Place*. Da mesma forma, esse mesmo valor também se connecta a um bloco realizado de raiz, de forma a apresentar o valor de percurso na interface de visualização apenas quando o mesmo termina ao longo da escala de tempo, e dessa forma, tentar simular um cronómetro real, antes de dar entrada no visualizador de tempo de percurso. Neste bloco é realizada ainda a subtração do instante de término ao instante de inicialização, permitindo obter o valor do tempo de percurso.



```
model VariationOverTime2
  Modelica.Blocks.Interfaces.RealInput u annotation( .
  Modelica.Blocks.Interfaces.RealInput previous
annotation( ...);
  Modelica.Blocks.Interfaces.RealInput u_saida = if
time >= u and previous <> 0 and u <> 0 then u -
previous else 0 annotation( ...);
equation
end VariationOverTime2;
```

Figura 174 - Bloco de subtração programada e apresentação do valor no instante pretendido

- **Tapete de Saída**

Por fim, no mesmo molde estrutural que a temporização realizada para o modelo do manipulador, a temporização para o tapete de saída foi feita. Em que o primeiro valor real a dar entrada no modelo provém do modelo de temporização do robot, constituindo-se como o valor de término do mesmo e início do funcionamento do tapete de saída. A única diferença

observável neste modelo relativamente ao prévio, diz respeito ao tempo de percurso, que é calculado através da soma dos tempos de saída em cada um dos tapetes de saída (1, 2 e 3)), dado que apenas um deles se encontra em funcionamento e o valor de saída dos tapete inativos é 0, garantindo desta forma uma correta apresentação do valor do percurso final no tapete independentemente do tipo de embalagem.

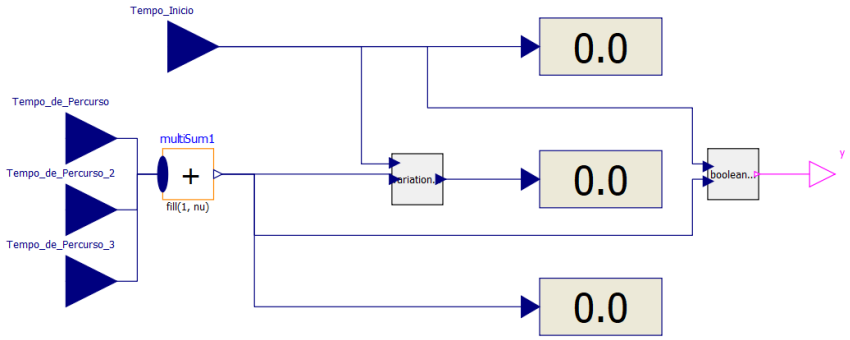


Figura 175 - Modelo de temporização do tapete de saída

Leds de Sinalização

Os diferentes booleanos que dão saída nos 3 modelos anteriormente abordados, respetivos à temporização de cada uma das etapas de funcionamento do sistema global, entram no presente bloco, e desta forma são as variáveis de condição que permitem que, de forma exclusiva, a corrente passe por uma das três resistências presentes no circuito apresentado. O circuito é constituído por um sinal de voltagem constante e uma resistência ligada à terra, sendo que, em função do booleano ativado ao longo do tempo, a resistência sob a qual passará corrente será diferente, simulando assim 3 leds presentes, que emitem luz em instâncias de tempo diferentes dependendo da etapa de funcionamento do sistema global.

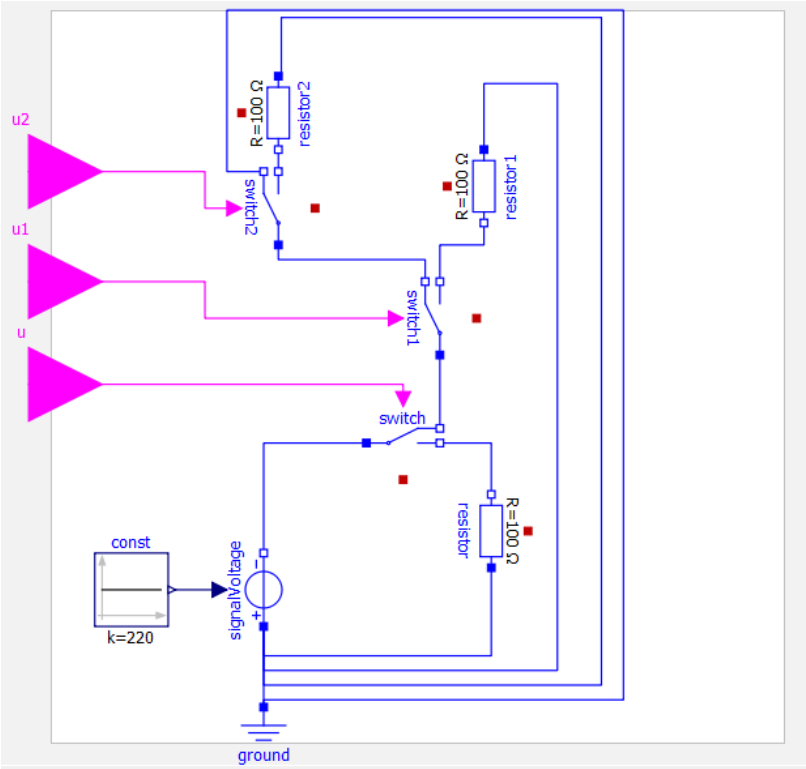


Figura 176 - Modelo com resistências de sinalização

8.5.Modelos – Tipos de Operação

O estudo de um sistema com base na modelação orientada a objetos procura, fundamentalmente, uma análise cuidada de um sistema físico e da sua *performance*. No entanto, a capacidade em complementar e enquadrar essa *performance,* nos devidos *timings* e segundo uma sequência ou tipo de operação específica é, de facto, um aspeto que demonstra a complexidade que poderá ser implementada com *Modelica.* A capacidade de programar e estabelecer determinados parâmetros que determinam o tipo de operação a ser efetuada, é um atributo que torna o estudo do funcionamento do sistema mais completo.

Com isso em mente, um conjunto de modelos foram desenvolvidos com o objetivo de dinamizar um pouco o sistema *Pick&Place*, introduzindo variáveis como o tipo de tapete de entrada, o tipo de tapete à saída dependente do tipo de embalagem a ser transportada, uma simulação de emergência e ainda uma simulação onde o transporte é efetuado de forma manual e, assim, procurar uma simulação com um estudo exclusivo dos tapetes.

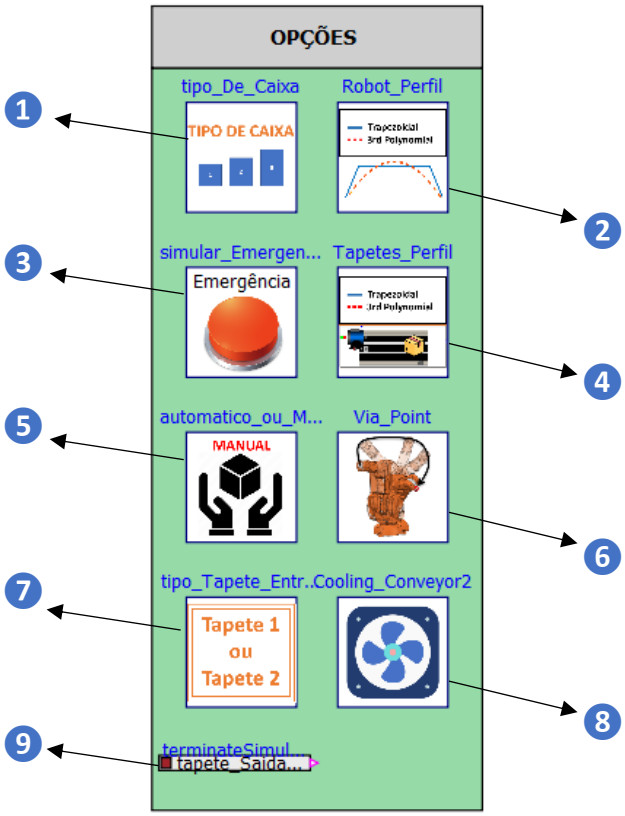


Figura 177 - Modos de operação (modelos)

Foram elaborados um conjunto de modelos cujos parâmetros são definidos pelo utilizador e determinam a operação a ser efetuada. Estes modelos constituem-se desta forma como modelos sobretudo de interação entre o utilizador e o sistema designado.

1 - Um modelo alusivo ao tipo de caixa permite escolher uma de três embalagens a ser transportada, sendo que cada uma apresenta uma massa distinta. Este modelo permite determinar ainda o tapete de saída a ser acionado, pois cada um dos tapetes de saída encontra-se destinado a uma embalagem em específico, e, de forma consequente, determinar o percurso a ser efetuado pelo braço robótico, mais especificamente o trajeto *Place* a ser efetuado.

2 - Neste modelo o tipo de perfil de movimento do manipulador é determinado. Cada um dos booleanos associados ao perfil requerido (trapezoidal, polinomial de 3ª ordem e polinomial de 5ª ordem) são definidos pelo utilizador.

3 - No modelo em que a emergência é simulada, um bloco gera um valor aleatório entre 0 e 1, sendo multiplicando esse valor de seguida por um ganho associado ao tempo mínimo de

término do sistema total, considerando todas as opções de operação e garantindo desta forma que o instante de tempo atribuído é válido ao longo de qualquer operação efetuada.

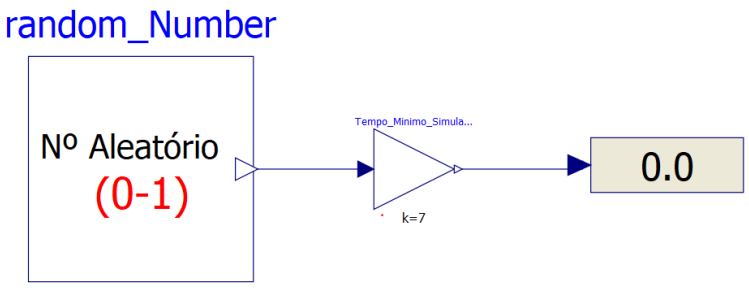


Figura 178 - Modelo de Emergência

4 - Modelo que, à semelhança do modelo 2 nas opções, determina o perfil de movimento, mas desta feita para os tapetes.

5 - Modelo que permite que o utilizador opte, exclusivamente, pela simulação dos tapetes, não ocorrendo movimento do manipulador e projetando um valor aleatório para o tempo de transporte *Pick&Place* do tapete de entrada para o de saída. Há assim uma tentativa de replicação de um transporte manual, em que o estudo exclusivo dos tapetes permite uma simulação final mais rápida.

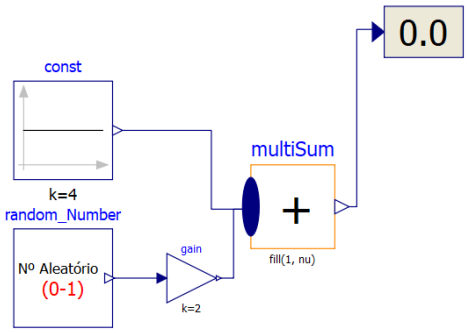


Figura 179 - Modelo de transporte manual

O tempo aleatório estabelecido, do transporte da embalagem do tapete de entrada para o tapete de saída, varia entre os 4 e os 6 segundos.

6 - Modelo que define o estado adjacente ao modo *Via Point,* em que um ponto de aproximação é estabelecido ao longo de todo o percurso *Pick* e do percurso *Place*, evitando

deste modo um percurso direto entre a configuração inicial do manipulador e a configuração destinada à recolha.

7 - Modelo que permite estabelecer qual dos dois modelos (se tapete de entrada 1 ou tapete de entrada 2) serão simulados, para o transporte da embalagem no sistema.

8 - Modelo que define se o modo de arrefecimento será ativado, com a existência de um sistema de arrefecimento suplementar, no segundo tapete de entrada.

9 - Na secção relativa à delineação das opções de operação, existe ainda uma opção que define, não um aspeto relativo à modelação, mas sim ao instante de término da simulação. O mesmo foi definido para o instante de tempo em que o tapete de saída termina o seu curso. Esta definição foi estabelecida tendo em conta um booleano à saída dos 3 tapetes, sendo que o booleano que determinará o fim da simulação, encontra-se diretamente dependente do tipo de caixa a ser transportada.

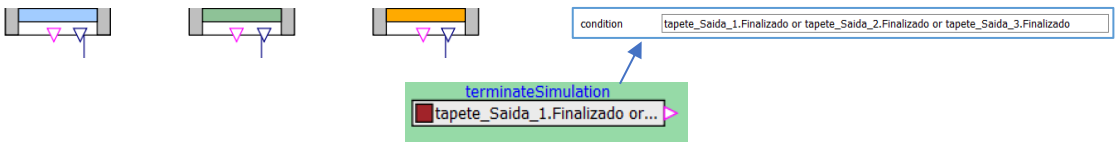


Figura 180 - Bloco que dita o fim da simulação através de uma condição

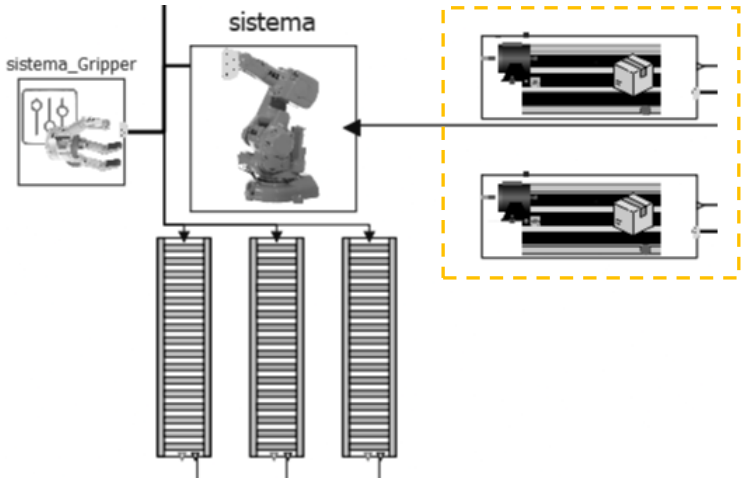
Na Figura 180 pode-se observar que a condição definida é assim a conversão de pelo menos um dos 3 booleanos para o estado de verdadeiro.

8.6. Parâmetros – Modos de Operação

Para que as informações provenientes dos parâmetros definidos pelo utilizador, relativos aos modos de operação/funcionamento, sejam transmitidos aos respetivos locais em que uma condição com o respetivo parâmetro seja programada, a utilização de conectores, maioritariamente booleanos, acarretaria uma complexidade elevada a nível de diagrama. Nesse sentido, a programação da transmissão do estado destes parâmetros foi elaborada recorrendo ao *Text View*.

Apresenta-se de seguida, o conjunto de parâmetros cuja transmissão entre modelos foi programada. A programação foi elaborada no modelo do sistema global, ou seja, no modelo de hierarquia superior assim como em cada um dos submodelos envolvidos. Desta forma, o bom funcionamento de cada um dos modelos, posteriormente apresentados, está assim dependente da entrada de cada um destes parâmetros.

Tabela 25 - Parâmetros definidos pelo utilizador nos modelos destinados ao Tapete de entrada

<table>
<tr><th>Modelo</th><th colspan="2">Parâmetros/Input's</th></tr>
<tr><td rowspan="3">Tapetes de Entrada</td><td>Tipo</td><td>Descrição</td></tr>
<tr><td>Real</td><td>• Valor da massa da embalagem (associado ao tipo de embalagem selecionada)</td></tr>
<tr><td>Booleano</td><td>• Ocorrência de Emergência
• Seleção de Tapete 1/2
• Seleção de embalagem tipo 1
• Seleção de embalagem tipo 2
• Seleção de embalagem tipo 3
• Seleção de perfil de movimento polinomial
• Seleção de arrefecimento suplementar</td></tr>
<tr><td colspan="3"> 
</td></tr>
</table>

Uma vez declarados cada um destes parâmetros no modelo do tapete de entrada 1, estes são por sua vez distribuídos pelos componentes/modelos presentes no respetivo modelo. Esta distribuição dos parâmetros será brevemente analisada.

Valor Massa da embalagem – Parâmetro que varia em conformidade com o tipo de embalagem selecionada e cujo valor real será igualado ao parâmetro do componente representativo da massa da embalagem, acoplado junto à estrutura do tapete transportador.

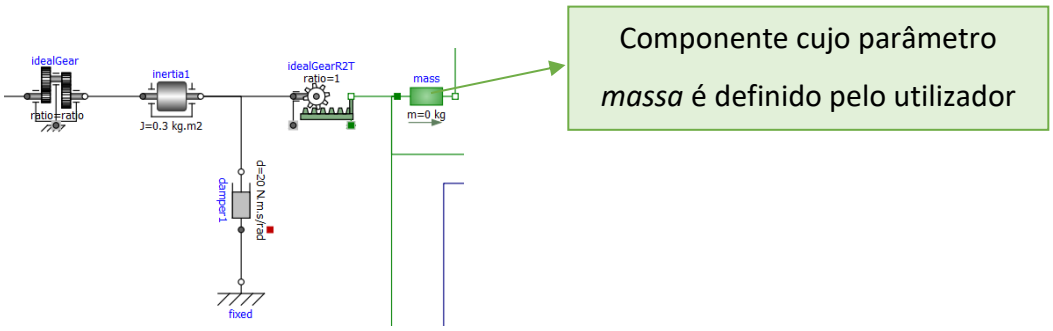


Figura 181 - Componente destinado ao parâmetro do valor de massa da embalagem

Ocorrência de Emergência – *Input* que procura simular uma paragem abrupta do tapete transportador numa situação de emergência. Caso o utilizador queira simular esta situação, o estado deste *Input* poderá sofrer uma alteração ao longo do tempo, mais concretamente de falso para verdadeiro. Se essa mudança se verificar, a saída no modelo de emergência, que se verifica na Figura 182, adota o valor verdadeiro e o valor de voltagem fornecida ao motor passa a ser zero.

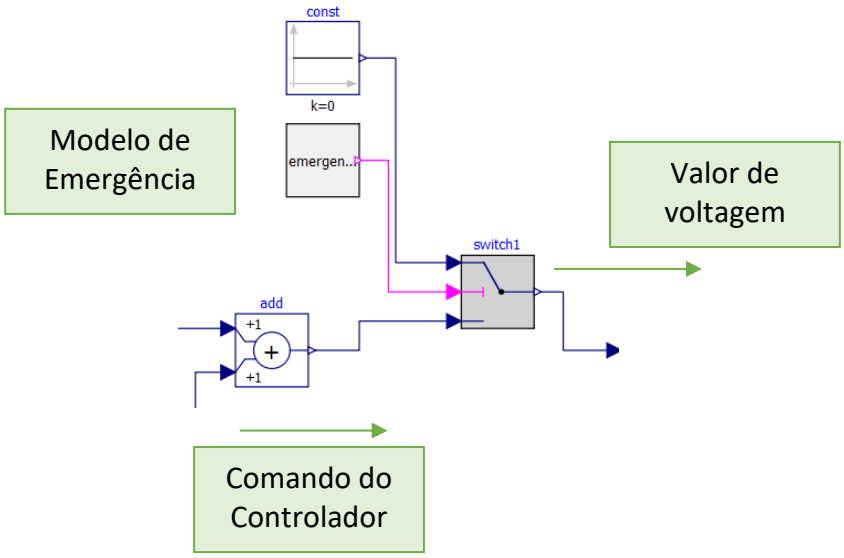
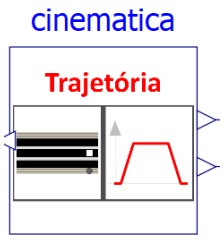


Figura 182 - Modelo de emergência em que a ocorrência de emergência é parametrizada

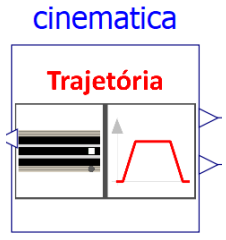
Seleção de Tapete 1 – Parâmetro que define se o tapete 1 é executado ou não. Caso não seja, o estado falso do parâmetro será lido no modelo de geração do perfil de movimento do tapete, com as variáveis de cálculo, envolvidas na obtenção da posição, velocidade e aceleração, a adotarem o valor de 0. Desta forma, as variáveis de posição, velocidade e aceleração, por sua vez, apresentarão também elas o valor de 0, responsáveis assim num comando ao motor "zerado" e a não ocorrência de movimento por parte do tapete.




```
if sd_max_inv <= eps or sdd_max_inv <= eps or Tapete_Tipo1 == false then
  sd_max = 0;
  sdd_max = 0;
  Ta1 = 0;
  Ta2 = 0;
  noWphase = false;
  Tv = 0;
  Te = 0;
  Ta1s = 0;
  Ta2s = 0;
  Tvs = 0;
  Tes = 0;
  sd_max2 = 0;
  s1 = 0;
  s2 = 0;
  s3 = 0;
  s = 0;
```

Figura 183 - Parâmetro de seleção do tapete 1 que permite estabelecer uma condição no modelo de geração do perfil de movimento do tapete

Seleção de perfil de movimento polinomial – Parâmetro que define o tipo de perfil de movimento a ser implementado no transporte da embalagem. Caso o tipo de perfil polinomial seja definido, o parâmetro adota o estado de verdadeiro e as duas variáveis de saída, correspondentes à posição e velocidade ao longo do tempo, adotarão os valores das mesmas calculadas sob o perfil polinomial. Se o parâmetro apresentar o estado falso, a posição e velocidade adotarão os valores calculados para o perfil trapezoidal.

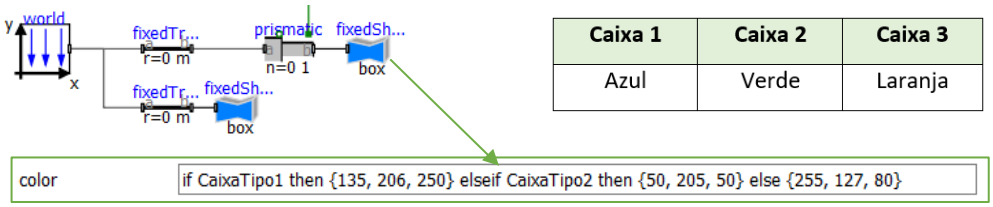



```
qd = if Polinomial then {vel_junta} else p_deltaq * sd;
q = if Polinomial then {junta1} else p_q_begin + p_deltaq * s;
```

Figura 184 - Parâmetro de seleção de perfil de movimento polinomial que permite estabelecer uma condição no modelo de geração do perfil de movimento do tapete

Seleção de embalagem tipo 1/2/3 – Além de acabar por definir a massa da embalagem a ser transportada, o tipo de caixa apresenta o detalhe de definir uma cor de visualização diferente

ao longo da simulação e desta forma tornar a sua compreensão mais intuitiva. A definição das cores feita com recurso a códigos *RGB.*



Caixa 1	Caixa 2	Caixa 3
Azul	Verde	Laranja

Figura 185 - Definição da cor da caixa com código *RGB*

Seleção de arrefecimento suplementar - Relativamente ao tapete 2, a definição dos parâmetros/input's é em tudo igual aos abordados no tapete 1, mas com a adição de um outro parâmetro destinado ao sistema de arrefecimento.

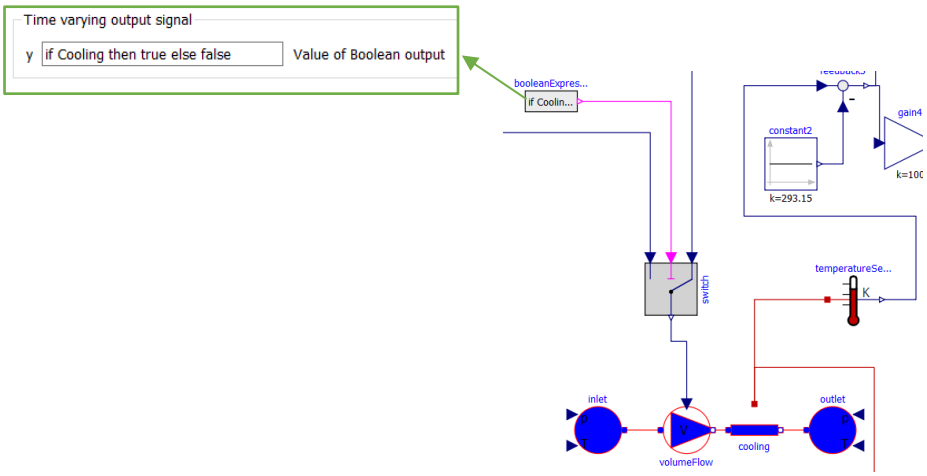
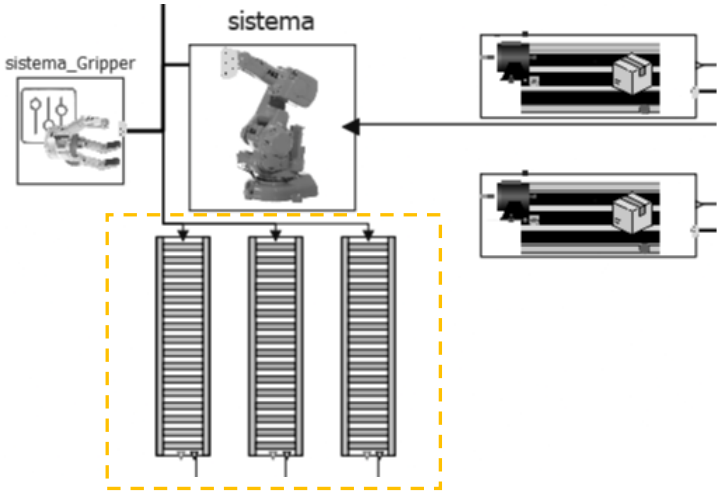


Figura 186 - Parâmetro que define se ocorre arrefecimento induzido ou não

Este parâmetro dá entrada no modelo do tapete de entrada 2, mais concretamente num bloco que determina o estado contínuo de um booleano. Este booleano, por sua vez, arbitra, através de um *switch*, qual o sinal que atribui um valor ao vazão volumétrico de fluído que entra no sistema de arrefecimento anexado junto ao tapete. Para um estado de falso, o valor 0 é atribuído ao vazão e dessa forma replica-se a inatividade do sistema de arrefecimento. Por outro lado, para um estado do booleano correspondente a verdadeiro, o sinal que atribui o valor do vazão volumétrico é o comando proveniente do sistema de controlo de temperatura.

Tabela 26 - Parâmetros definidos pelo utilizador nos modelos destinados ao tapete de saída

<table>
<tr><th>Modelo</th><th colspan="2">Parâmetros/Input's</th></tr>
<tr><td rowspan="3">Tapetes de Saída</td><td>Tipo</td><td>Descrição</td></tr>
<tr><td>Real</td><td>• Valor da massa da embalagem (associado ao tipo de embalagem selecionada)</td></tr>
<tr><td>Booleano</td><td>• Ocorrência de Emergência
• Seleção de embalagem tipo 1/2/3
• Seleção de transporte Pick&Place manual</td></tr>
<tr><td colspan="3"> 
</td></tr>
</table>

Nos tapetes de saída é percetível a diminuição de parâmetros associados aos diferentes estados definidos pelo utilizador. A diminuição dos parâmetros relativos ao tipo de embalagem selecionada facilmente se entende pelo simples facto de que em cada um dos tapetes apenas um tipo de caixa é admitida, pelo que em cada um dos tapetes apenas um parâmetro relativo ao tipo de embalagem é apresentado e não três, ou seja, há assim uma definição se o tipo de embalagem para o tapete específico é o escolhido ou não.

Há ainda um outro parâmetro não presente na tabela referente ao tipo de perfil polinomial. Este não se encontra presente nos 2 primeiros tapetes de saída, mas acena a sua presença no terceiro tapete, devido à trajetória de movimento mais complexa.

Seleção de transporte *Pick&Place* manual – Parâmetro que determina se o transporte *Pick&Place* se realiza de forma automática, auxiliada pelo movimento do braço robótico, ou

se de forma manual, com a projeção de um tempo de transporte aleatório antes de o tapete de saída receber ordem para ser inicializado.

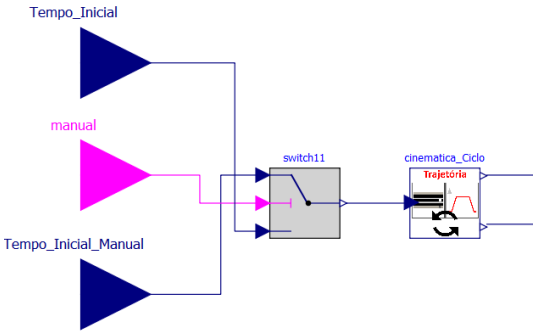


Figura 187 - Entradas presentes nos tapetes de saída. Entradas "Reais" são conectadas e entrada booleana programada

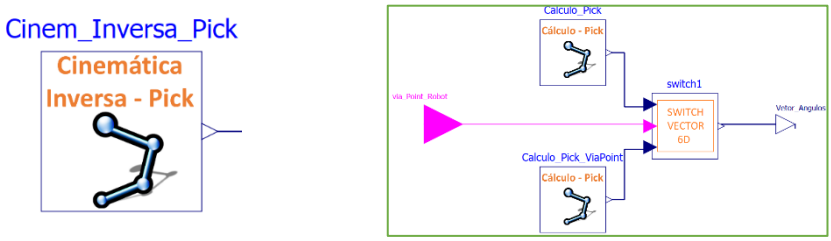
Na Figura 187 é possível observar 3 entradas (duas de valores reais e uma booleana). O estado booleano, que indica o estado do modo manual, dita qual dos dois valores serão admitidos no modelo responsável pela geração do perfil de movimento. Para o estado de falso, o instante, em que a geração de perfil é definida, é determinado pelo instante em que o robot finaliza o seu movimento enquanto que para o estado de verdadeiro é determinado pelo instante definido pela soma do tempo de transporte no tapete de entrada e do tempo aleatório gerado no bloco do modo manual.

Tabela 27 - Parâmetros definidos pelo utilizador no modelo do robot

Modelo	**Parâmetros/Input's**	
	Tipo	**Descrição**
Robot	Booleano	• Ocorrência de Emergência • Seleção de Tapete 1 • Seleção de embalagem tipo 1 • Seleção de embalagem tipo 2 • Seleção de embalagem tipo 3 • Seleção de perfil de movimento polinomial (3ª ordem) • Seleção de perfil de movimento polinomial (5ª ordem) • Seleção de trajetória com *via points*

Uma vez declarados cada um destes parâmetros no modelo do robot, estes são por sua vez distribuídos pelos componentes/submodelos do respetivo modelo. Esta distribuição dos parâmetros será brevemente analisada.

Seleção de Tapete 1 – A escolha do tapete 1 como tapete de entrada, implica que as coordenadas correspondentes à posição final do *gripper*, que dá entrada no modelo de cinemática inversa, sejam coincidentes com um ponto no espaço posicionado sobre o final do respetivo tapete. Assim, a definição das coordenadas de posição são definidas com recurso a uma condição e enviadas posteriormente para os modelos internos de cálculo.




```
Modelica.Blocks.Interfaces.RealOutput Vetor_Angulos[6] annotation( ...);
Robot.Cinematica_Inversa_3 Calculo_Pick_ViaPoint(Pose_tip = if Tapete_tipo1
then[-0.4; 0.4965; 0.45; 0; 0; 0] else [0; 0.4965; 0.45; 0; 0; 0]) annotation( ...);
Robot.Cinematica_inversa Calculo_Pick(Pose_tip = if Tapete_tipo1 then[-0.4; 0.4965;
0.30; 0; 0; 0] else [0; 0.4965; 0.30; 0; 0; 0]) annotation( ...);
```

Figura 188 - Parâmetro de seleção do tapete 1 de entrada que permite estabelecer uma condição no modelo de cinemática inversa do *Pick*

Este parâmetro apresenta ainda a sua utilidade numa fase posterior do cálculo da trajetória, caso o modo *via point* esteja ativado, em que um segundo modelo de cinemática inversa, correspondendo à variação angular dos eixos entre o ponto de abrandamento e o ponto final de recolha, é executado. Isto acontece porque no fundo, com o modo via point, as coordenadas cartesianas envolvidas na condição mostrada em cima relacionam-se com ponto de abrandamento e não o ponto final do movimento.

A condição em que este parâmetro é ressalvado neste segundo modelo de cinemática inversa, é em tudo semelhante ao mostrado em cima, com a diferença que as coordenadas envolvidas são, efetivamente, as finais de recolha e não as de abrandamento.

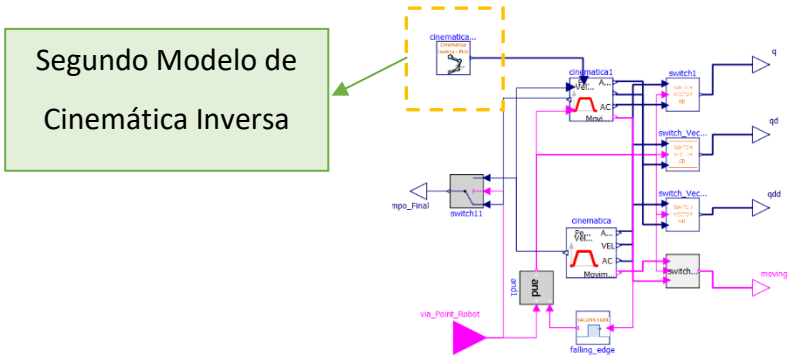


Figura 189 - Segundo modelo de cinemática inversa no modelo do perfil de movimento

A significância deste parâmetro é assim, de forma geral, em ditar os valores de entrada ao longo dos modelos de cinemática inversa, para que a mesma ocorra como previsto, dentro de uma das duas opções ditadas pelo utilizador, tapete 1 ou tapete 2.

A seleção do tapete 2 acaba por não se traduzir num parâmetro, uma vez que as condições encontradas ao longo dos modelos mostrados em cima, ditam diretamente o comportamento do sistema, caso o tapete 1 não seja selecionado, através do operador *else* presente na condição.

Seleção de embalagem tipo 1 / Seleção de embalagem tipo 2 / Seleção de embalagem tipo 3 – Cada um destes três parâmetros booleanos ditam um comportamento no sistema bastante semelhante ao definido pelo anterior associado à seleção do tapete 1.

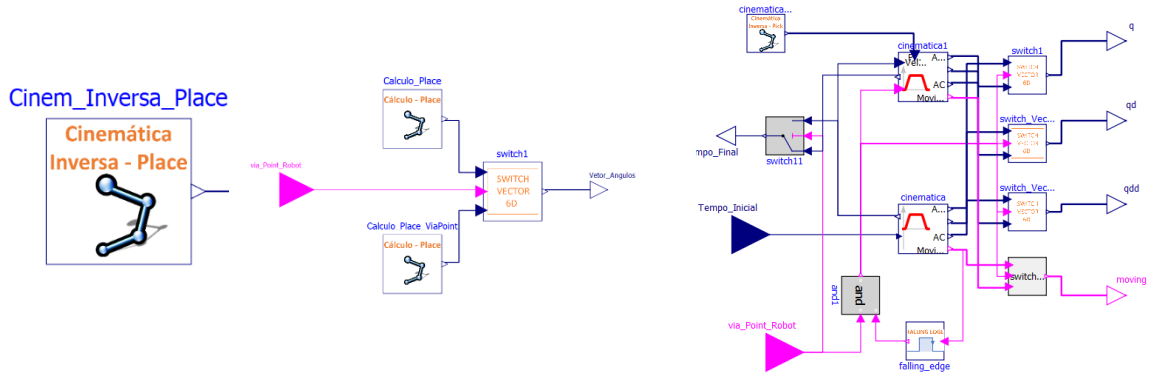


Figura 190 - Parâmetro de seleção do tipo de embalagem que permite estabelecer uma condição no modelo de cinemática inversa do *Place*

Seleção de perfil de movimento polinomial (3ª e 5ª ordem) – Os parâmetros polinomiais dão entrada no perfil de movimento de *Pick* e *Place.* Sendo que dentro destes 2 modelos, dão entrada nos modelos de geração do perfil de trajetória. Nestes modelos os cálculos para um

perfil trapezoidal e polinomiais de 3ª e 5ª ordem são calculados, independentemente da escolha realizada pelo utilizador, mas apenas um é selecionado no fim do *script*, permitindo determinar o comando. A escolha da variável final requerida é caracterizada em conformidade com o estado booleano do respetivo perfil designado. Na Figura 191 é possível observar a parametrização dos estados polinomiais, sendo que caso sejam ambos falsos o perfil trapezoidal é implementado.

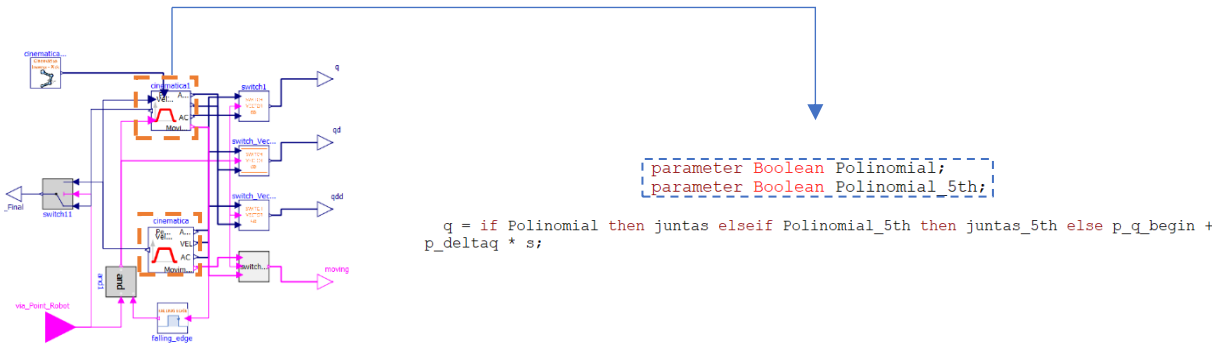


Figura 191 - Parametrização do perfil de velocidade e condição associada

Seleção de trajetória com *via points* – O parâmetro que determina a ocorrência das trajetórias *Pick* e *Place* com pontos de aproximação incorporados, dá entrada nos modelos de cinemática inversa e nos modelos de geração dos perfis de velocidade, de maneira a ativar os devidos modelos de cálculo. A esquematização presente na Figura 192 ilustra bem os booleanos que se fazem determinar pelo estado do modo (*Via Point*) assim como os respetivos modelos de cálculo associados.

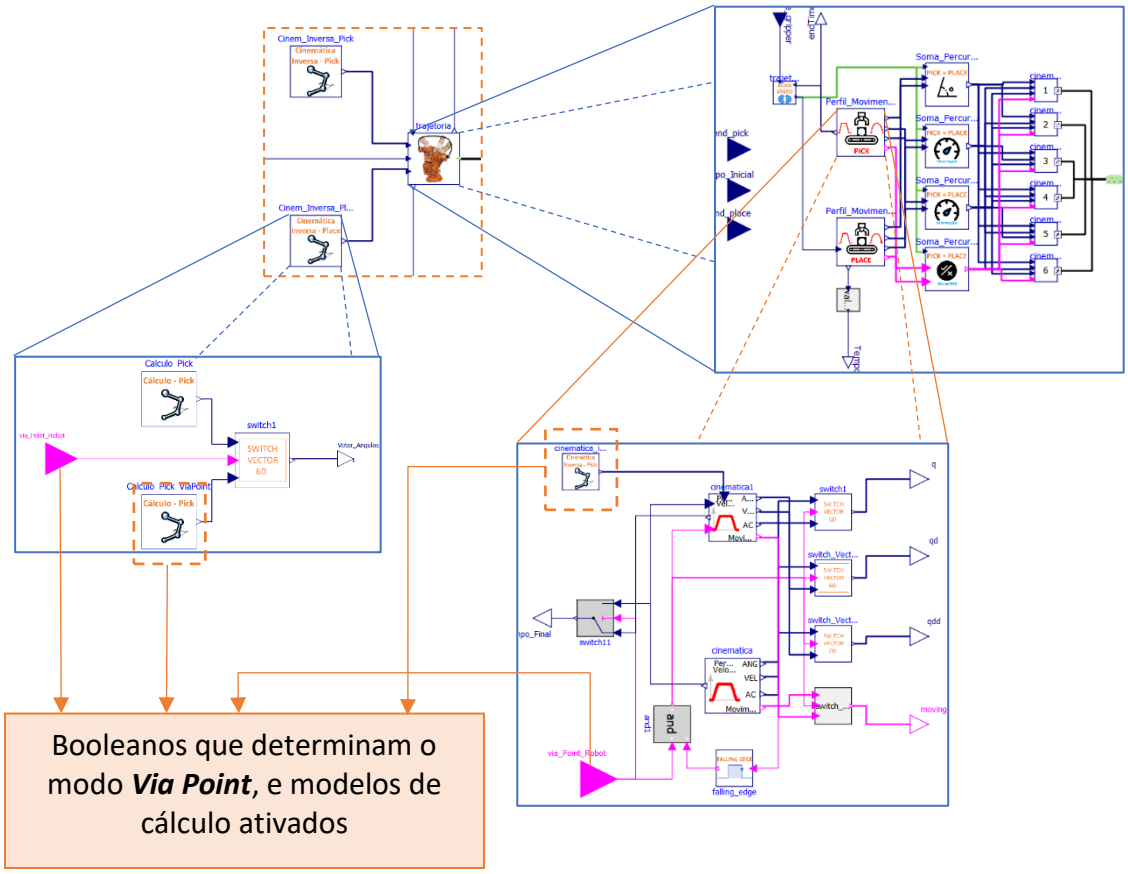


Figura 192 - Booleanos e modelos de cálculo associados ao Modo *Via Point*

8.7.Sequência – Transmissão de Variável Tempo ao longo da Simulação

Para que a simulação dos diferentes modelos ocorra sob a ordem pretendida, avançou-se para a elaboração de lógica programada envolvendo ativação de booleanos e transmissão de valores da variável associada ao tempo de simulação. A explicação da ordem sobre o qual essa mesma variável é transmitida, no modelo de hierarquia superior, será aqui analisada.

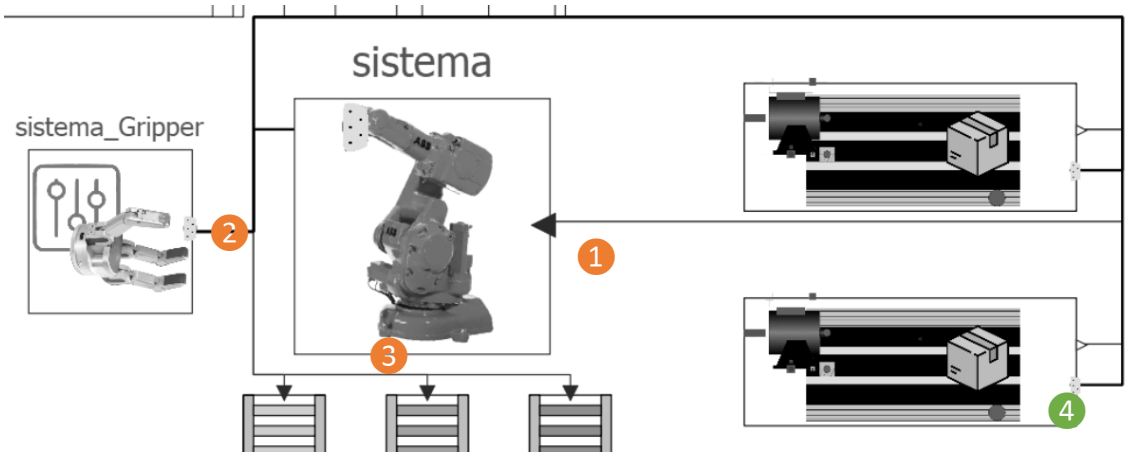


Figura 193 - Diferentes instâncias em que se verifica uma transmissão de variáveis (tempo) entre os diferentes modelos do sistema global

A transmissão da variável tempo, poderá ser desdobrada em 3 fases principais.

1 - Quando a embalagem finaliza o seu percurso, ao longo de um dos dois tapetes, o instante de tempo em que a situação se concretiza é registado sob a forma de valor real e transmitido, através da ligação 1, ao robot. Quando o *clock* de simulação atinge esse valor, tem-se assim a geração do comando que irá ser transferido para os controladores dos motores de cada um dos eixos, iniciando assim a reprodução do movimento do manipulador.

2 - Esta etapa relaciona-se com a partilha de valores que dizem respeito à variável tempo, ocorridas entre o início do movimento do robot para efetuar o *pick* e o final do movimento do robot para efetuar o *place*. Durante esta fase do processo, o *gripper* executa um movimento de fecho e abertura, pelo que um sincronismo, entre os movimentos do braço e os movimentos do *gripper,* tem que ser estabelecido. Dentro desta etapa pode-se dividir assim diferentes instâncias, que podem ser descritas do seguinte modo :

- 2.1 – O deslocamento do braço até à embalagem é completado, o registo do instante de tempo é efetuado e por sua vez transmitido ao modelo do *gripper* para que o mesmo possa iniciar o seu fecho e agarrar na embalagem.
- 2.2 – O *gripper* acaba de executar o seu movimento de fecho e o instante de tempo é registado e enviado ao modelo do manipulador, permitindo assim que o manipulador retome o seu movimento, desta feita para o *place*.
- 2.3 – O movimento do *place* é completado, o registo do instante de tempo é realizado e, dessa forma, tem-se o início da abertura do *gripper* e dessa forma a deposição da embalagem no tapete de saída.

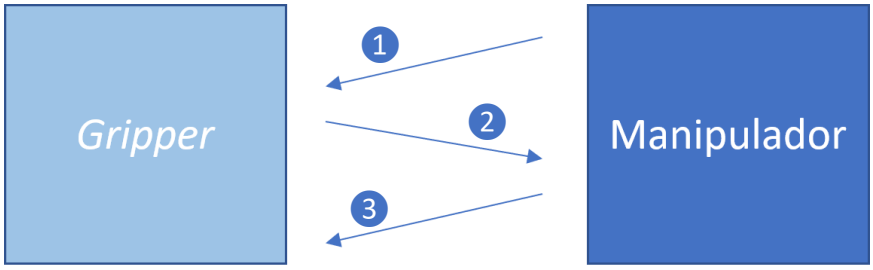


Figura 194 - Informações partilhadas entre o Manipulador e o *Gripper*

3 - Terminada a execução que envolve o manipulador e o *gripper* , resta a atuação do tapete de saída, que é iniciado no instante em que o *gripper* termina a sua abertura e a embalagem

já se encontra no ponto de recolha do tapete. O instante de tempo em que o *gripper* termina a sua abertura é assim transmitido a um dos três tapetes de saída, permitindo ,assim, o início da execução do mesmo, quer com sinais *step*, quer com a geração de um perfil no comando transmitido ao motor.

Os valores numéricos dos diferentes instantes de tempo analisados, foram transmitidos ao longo de um conector, contendo as variáveis associadas a cada um dos instantes temporais. O uso de um conector permite desde logo uma simplificação a nível de diagrama, evitando uma transmissão esquematizada individual. A descrição das variáveis presentes no conector que associa os diferentes modelos principais, poderão ser observadas na seguinte tabela.

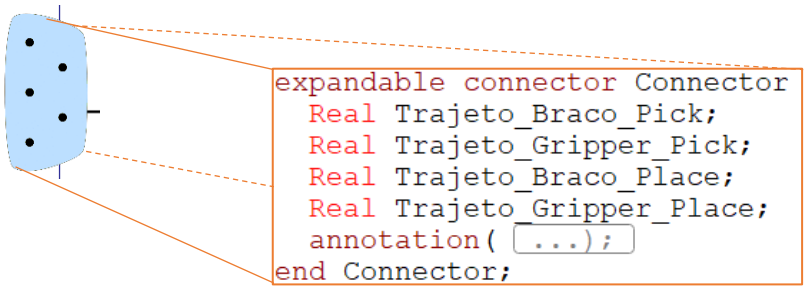


Figura 195 - *Connector*

Tabela 28 - Variáveis definidas no interior do Connector

Conector (variáveis)	
Manipulador	Instante de Tempo relativo ao fim do movimento *Pick*
	Instante de tempo relativo ao fim do movimento *Place*
Gripper	Instante de tempo relativo ao fim do fecho do *gripper*
	Instante de tempo relativo ao fim da abertura do *gripper*

4 - Uma última (dupla) ligação que poderá ser observada na Figura 193 relaciona-se não com a modelação no sentido físico, mas sim visual. Esta ligação permite que o instante de tempo em que o gripper finaliza o seu fecho seja associado a um componente visualizador dentro dos modelos dos tapetes e, dessa forma, interromper a visualização da embalagem no final da linha de um dos tapetes. A simulação e o transporte da embalagem tornam-se assim mais intuitivos, uma vez que a visualização das embalagens tanto no tapete de entrada como no tapete de saída são definidas de acordo uma tarefa *pick&place* real. A simulação em *Openmodelica* não apresenta qualquer tipo de engenharia física associada, pelo que a

visualização a nível de transporte tem que ser devidamente programada, incluindo a visualização dos componentes.

9. SIMULAÇÃO AMBIENTE FÍSICO

Finalizada a modelação do sistema global (cuja arquitetura poderá ser encontra no apêndice 3), o retrato da simulação física para um ciclo de operação será apresentado, fracionado em diferentes etapas. Antes de se partir para um conjunto de passos que permite a compreensão do ambiente de simulação, a descrição com o conjunto de componentes constituintes do sistema será ilustrada.

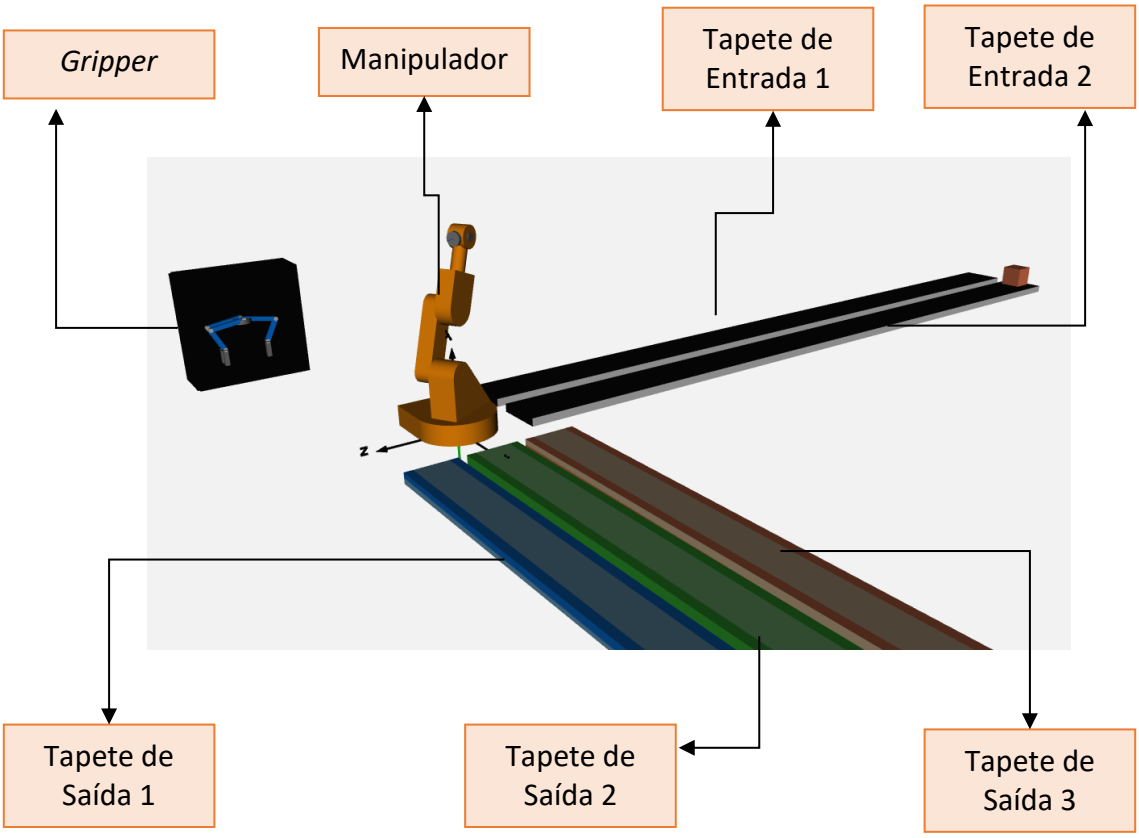


Figura 196 - Componentes modelados "animados". Representação física dos componentes na simulação.

9.1. Transporte da Embalagem - Tapete de Entrada

- Início do processo de transporte.

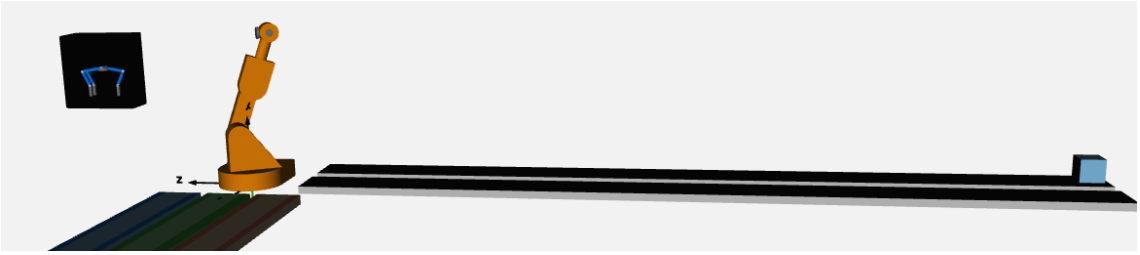


Figura 197 - Tapete de Entrada 1 - Caixa 1 - Repouso Inicial

- Transporte da Embalagem ao longo de toda a extensão do tapete.

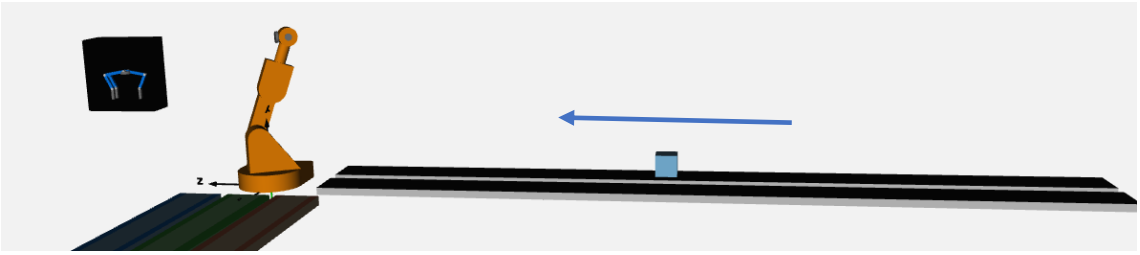


Figura 198 - Tapete de Entrada 1 - Caixa 1 - Estado de Movimento

- Término do transporte, com alocação da embalagem no ponto de recolha.

Figura 199 - Tapete de Entrada 1 - Caixa 1 - Repouso Final

9.2. Movimento do Manipulador – *Pick*

- Início do movimento *Pick* ainda com o manipulador no instante inicial, definido por uma configuração espacial aleatória.

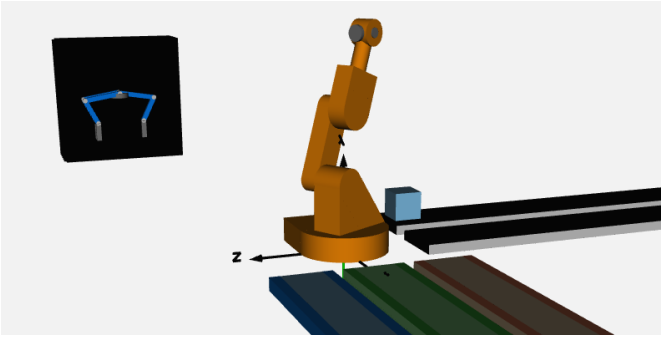


Figura 200 - *Pick* - Tapete 1 - Caixa 1 - Repouso Inicial

- Movimento *Pick* com variação da configuração espacial ao longo do tempo, através da variação angular dos diferentes eixos.

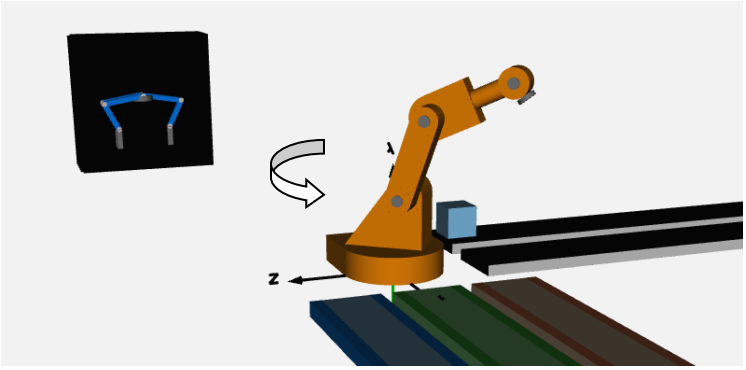


Figura 201 - *Pick* - Tapete 1 - Caixa 1 - Movimento

- Configuração final para recolha, atingida.

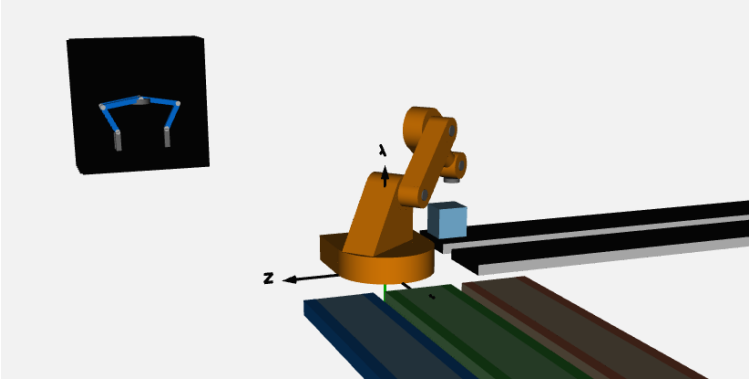


Figura 202 - *Pick* - Tapete 1 - Caixa 1 - Repouso Final

9.3.Movimento do *Gripper* - Fecho

- Gripper ainda no seu estado de repouso inicial, sem que o fecho tenha sido ainda iniciado.

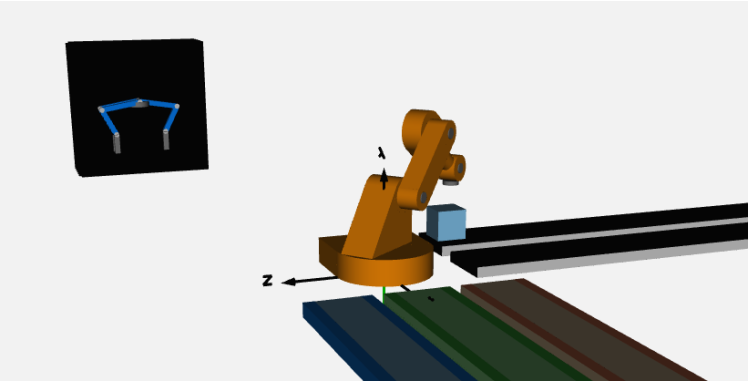


Figura 203 – Atuação do *Gripper* - Posição Inicial

- Movimento de fecho iniciado pelo *gripper*

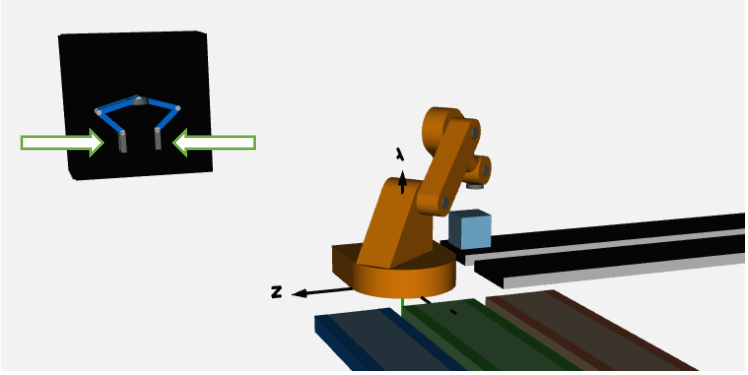


Figura 204 - Atuação do *Gripper* - Fecho

- Componente de força iniciada no *gripper* permitindo que a embalagem seja agarrada.

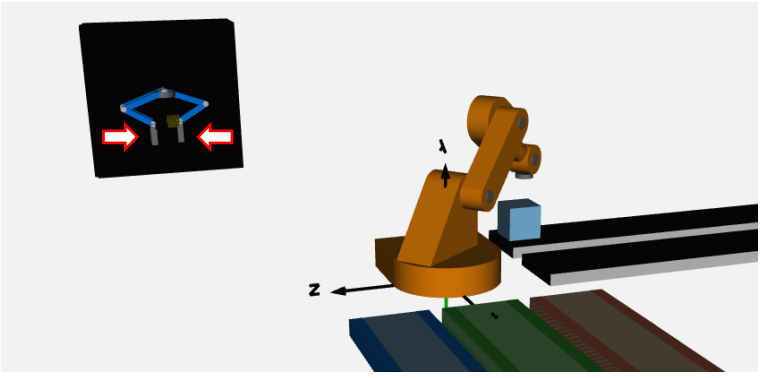


Figura 205 - Atuação do *Gripper* – Movimento de Força

9.4.Movimento do Manipulador - *Place*

- Início do movimento *Place* com a embalagem a ser agarrada e transportada. Nesta fase tem-se, visualmente, o desaparecimento da embalagem na simulação, evitando provocar qualquer tipo de incompatibilidade à compreensão da operação.

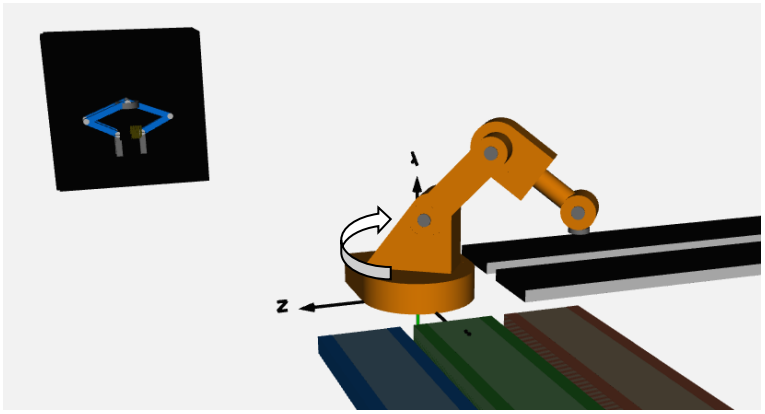


Figura 206 - *Place* - Tapete 1 - Caixa 1 - Movimento

- Movimento de *Place* finalizado.

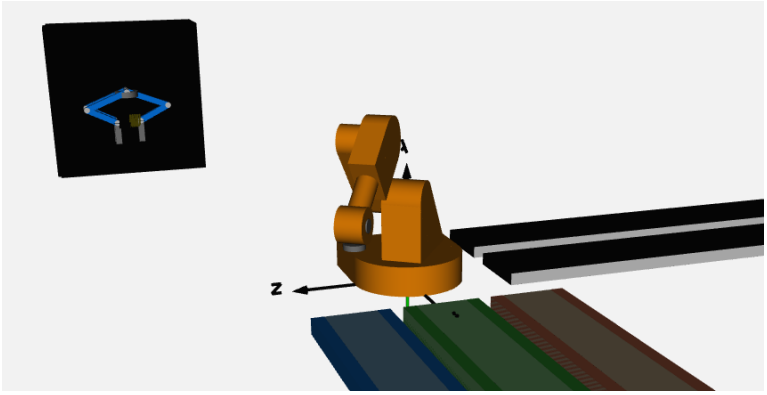


Figura 207 - *Place* - Tapete 1 - Caixa 1 – Repouso Final

9.5. Movimento do Gripper – Abertura

- Movimento de abertura inicializado.

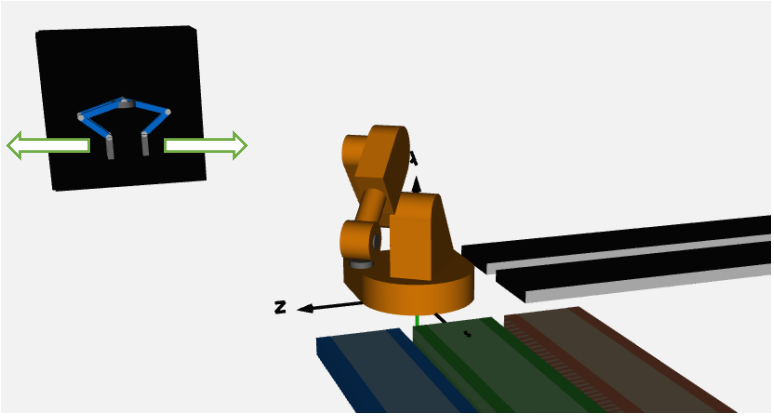


Figura 208 - Atuação do Gripper - Abertura

- Movimento de abertura finalizado com a deposição da embalagem no tapete de saída.

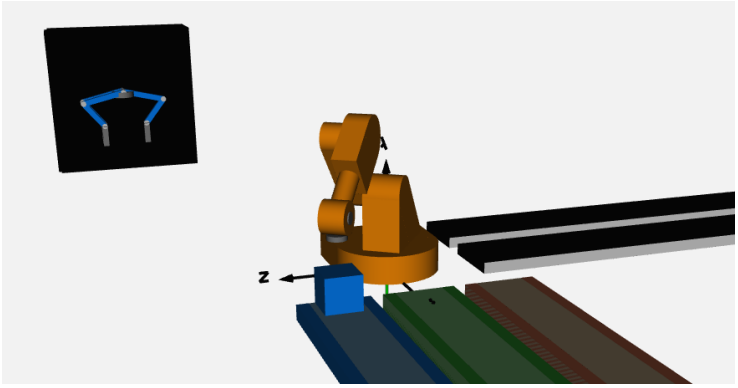


Figura 209 - Atuação do *Gripper* - Abertura finalizada e deposição da embalagem

9.6.Transporte da Embalagem - Tapete de Saída

- Início do processo de transporte de saída

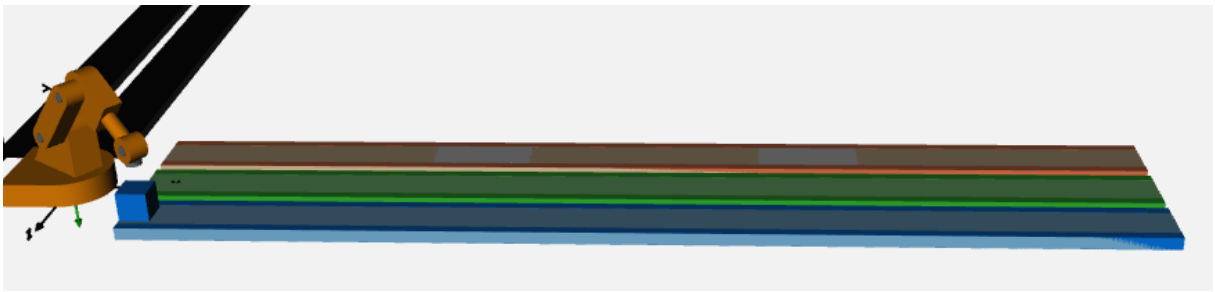


Figura 210 - Tapete de Saída 1 - Caixa 1 - Repouso Inicial

- Transporte da Embalagem ao longo de toda a extensão do tapete

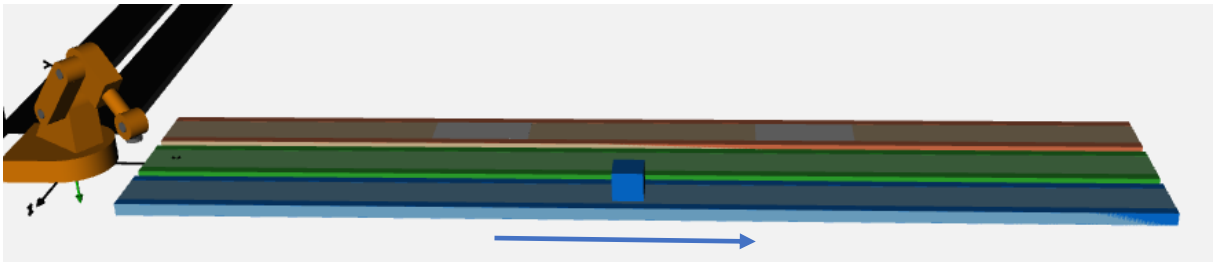


Figura 211 - Tapete de Saída 1 - Caixa 1 – Estado de Movimento

- Término do transporte, com alocação da embalagem fora do tapete, na sua totalidade, permitindo a sua deposição para processamento posterior. Nesta fase tem-se o fim da simulação.

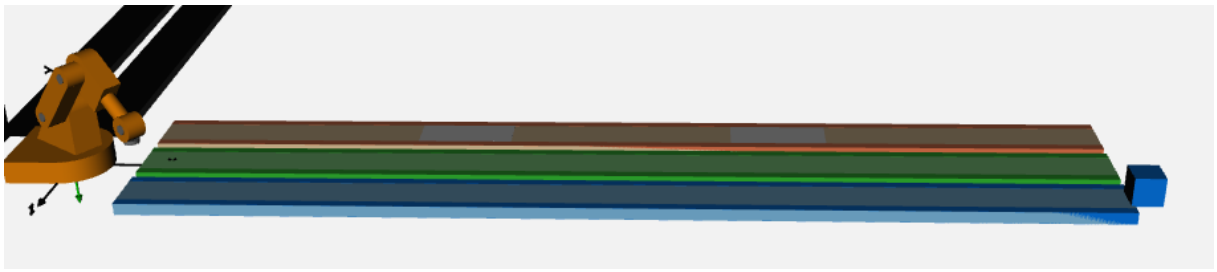
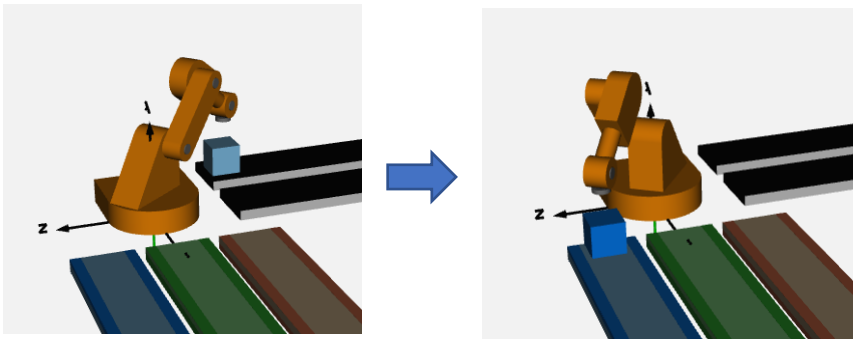
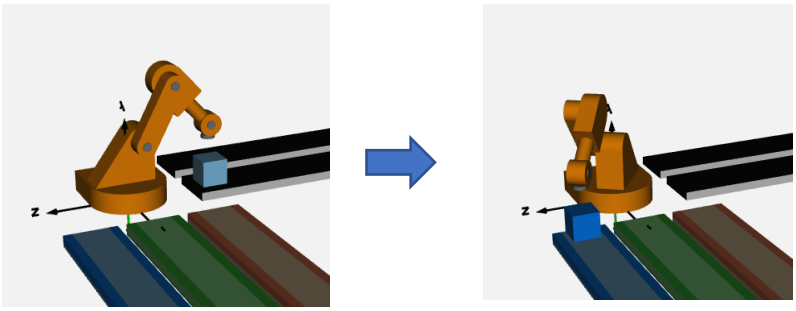
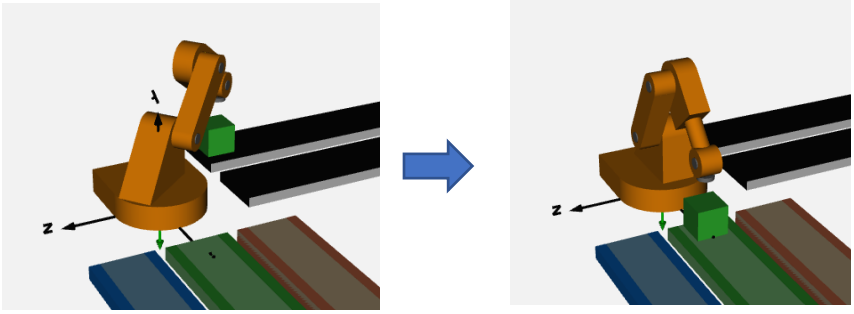
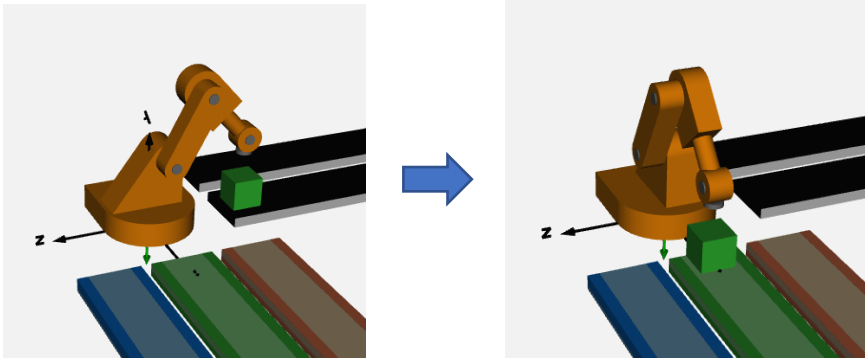
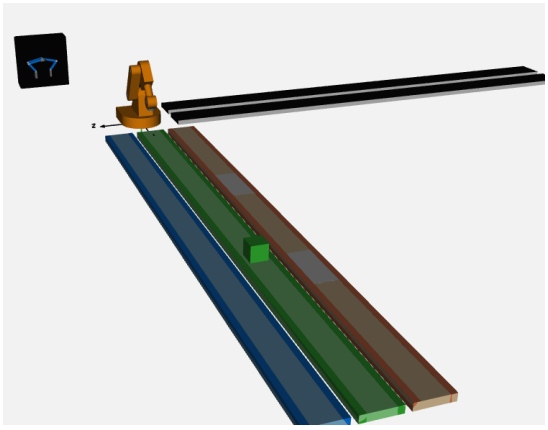
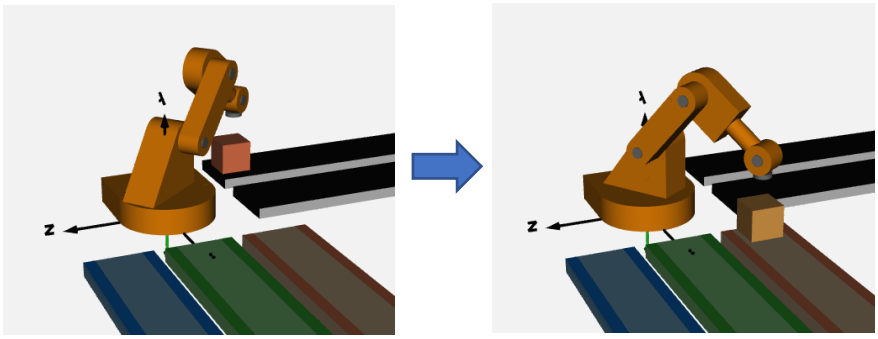


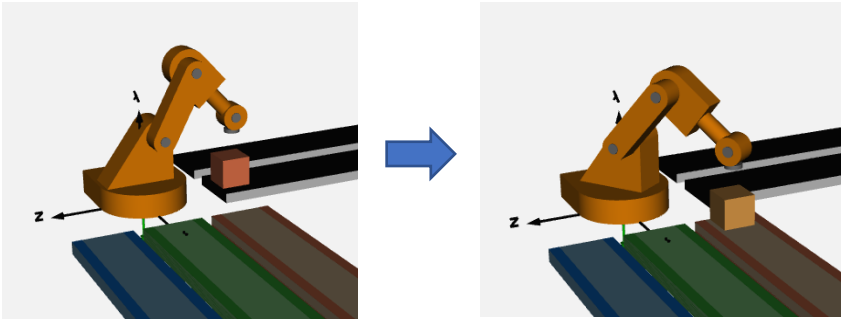
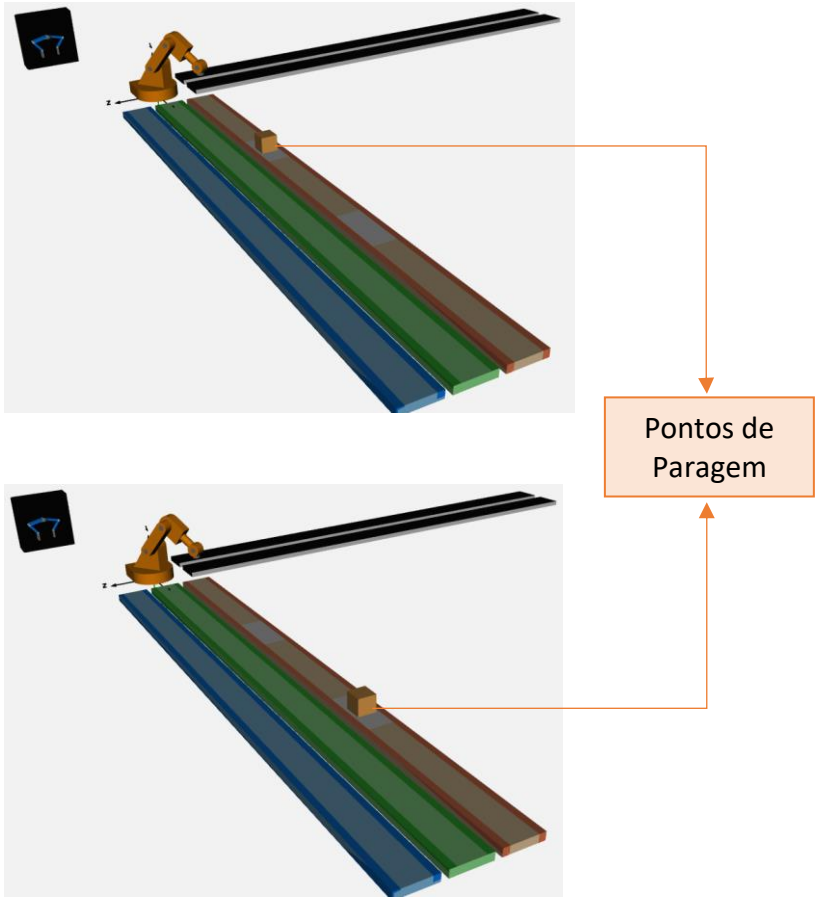
Figura 212 - Tapete de Saída 1 - Caixa 1 – Deposição Final

Com dois tapetes de entrada e três tapetes de saída, a combinação de trajetórias possíveis *Pick&Place* ascende a 6 trajetórias distintas (que se estende a mais se a opção com *Via Points* for considerada). As ilustrações inseridas na seguinte tabela procuram fazer um retrato sintetizado do percurso característico a cada combinação.

Tabela 29 - *Prints* ilustrativos dos percursos envolvidos nas restantes opções disponibilizadas

Processo	*Prints* de Simulação
Pick&Place **Tapete Entrada 1** **Caixa 1**	
Pick&Place **Tapete Entrada 2** **Caixa 1**	

Pick&Place **Tapete Entrada 1** **Caixa 2**	
Pick&Place **Tapete Entrada 2** **Caixa 2**	
Tapete Saída **Caixa 2**	
Pick&Place **Tapete Entrada 1** **Caixa 3**	

Pick&Place **Tapete Entrada 2** **Caixa 3**	
Tapete Saída **Caixa 3**	

É possível verificar nos gráficos embaixo a duração relativa para cada uma das fases de transporte da embalagem. A duração da fase do tapete de entrada apresenta uma maior duração no tapete de entrada 2, devido às perdas consideradas. A duração do transporte *Pick&Place* apresenta uma maior durabilidade de execução quando o tapete de entrada 1 e a embalagem 1 são utilizados, devido a uma maior distância implicada no percurso total. Já a duração do tapete de saída, exibe um tempo mais prolongado quando a embalagem 3 é transportada e o tapete de saída 3 é acionado, devido às paragens envolvidas.

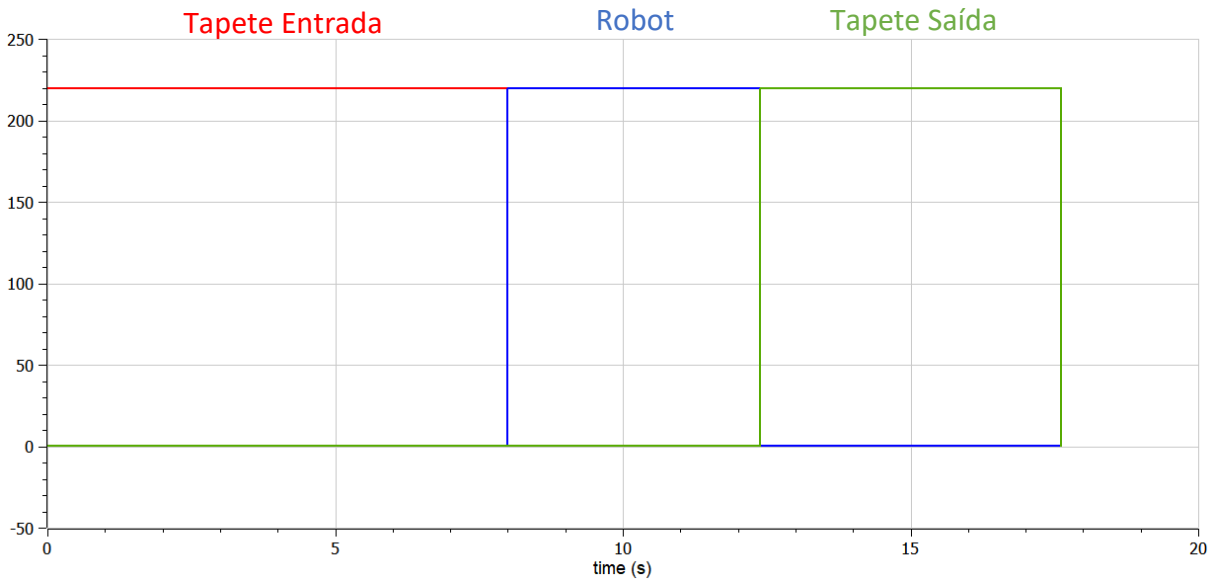


Figura 213 - Voltagens presentes nas lâmpadas de sinalização associadas a cada uma das fases de transporte. Tapete de Entrada 1 e Caixa 1

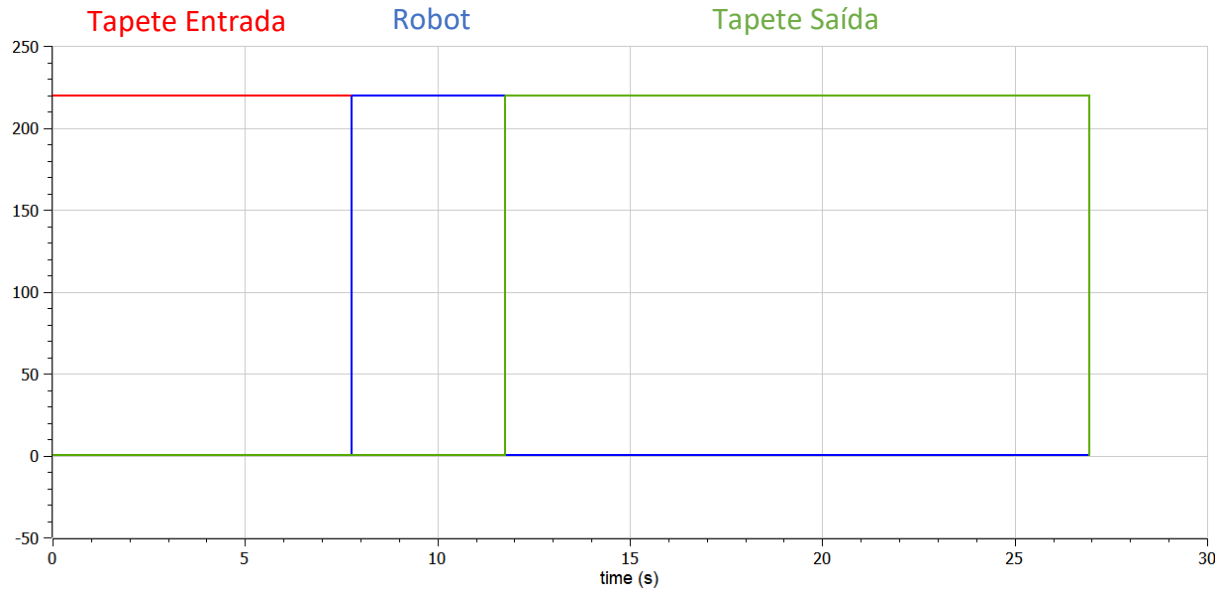


Figura 214 - Voltagens presentes nas lâmpadas de sinalização associadas a cada uma das fases de transporte. Tapete de Entrada 1 e Caixa 3

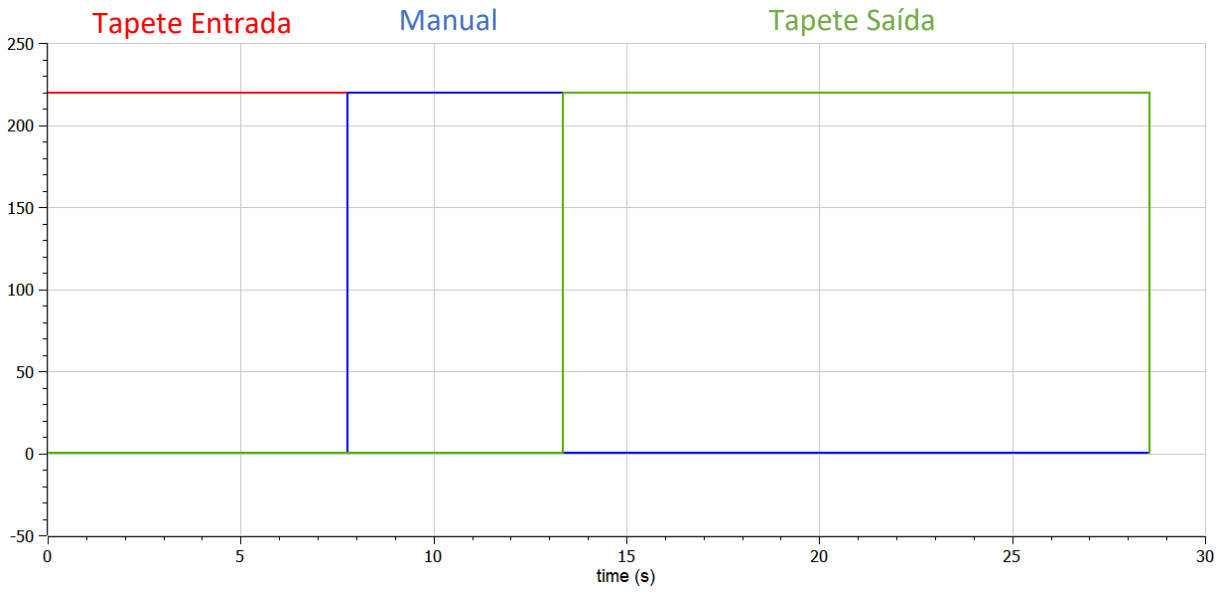


Figura 215 - Voltagens presentes nas lâmpadas de sinalização associadas a cada uma das fases de transporte. Tapete de Entrada 1 e Caixa 3 com transporte *Pick&Place* manual

CONCLUSÕES

Conclusões – Projeto Desenvolvido e Linguagem *Modelica*

O projeto desenvolvido serviu para enaltecer as potencialidades intrínsecas da modelação orientada a objetos. A comparação entre um sistema simulado em *Matlab/Coppelia* e em *OpenModelica,* permitiu perceber os diferentes tipos de simulação passíveis de serem reproduzidos que retratam um processo industrial, assim como trazer maior relevância aos aspetos que podem ser explorados com a linguagem *Modelica*.

Um dos principais aspetos a reter foi sem dúvida a versatilidade apresentada pela linguagem, em que ao longo da elaboração do sistema, segundo uma modelação orientada a objetos, o espetro de configurações subjacentes possíveis ficou bem patente, revelando assim um estudo bastante completo. O que tornou, sem grandes dúvidas, o desenvolvimento deste projeto bastante diferenciador, foi a capacidade em reunir:

- Um ramo de diferentes áreas bastante abrangente.
- Desenvolvimento de um design físico mas também de controlo numa só linguagem, projetando não só a capacidade de desenvolvimento de digramas, assim como código programado. Ao nível mais fundamental possível, a integridade da modelação acaba por ser toda ela programada em código, no entanto, a possibilidade de instanciar modelos permite desde logo simplificar e encurtar o uso direto desse mesmo código, uma vez que porções significativas do código encontram-se já reunidas sob a forma de blocos/modelos e a elaboração de diagramas através da associação desses modelos permite que se debruce apenas em equações que associam os modelos, através da transmissão de variáveis.
- Uma conjetura híbrida que admite estados ou discretizações, bem como também funções contínuas, permitindo uma amplitude de desenvolvimento muito mais ampla, como por exemplo, uma capacidade de controlo muito mais completa e sofisticada.
- Aliada à hierarquização e organização dos princípios teóricos envolvidos na modelação, assim como à apresentação dos valores das variáveis ao longo do tempo, é possível simular fisicamente o ambiente. Ainda que a simulação apresente algumas limitações comparada a outros *softwares* existentes, não deixa de ser um aspeto com a sua importância, uma vez que ressalva aspetos visuais envolvidos no processo físico.

- Apresentação de resultados gráficos, que determinam o comportamento das diferentes variáveis presentes no sistema modelado. Esta particularidade, não obstante, determinou, e muito, a aproximação tida no refinamento da modelação. A oportunidade de simular e analisar a variação do valor das variáveis, em simultâneo com a conceção dos modelos traz, desde logo, um maior discernimento no desenvolvimento do sistema. Erros de modelação ou de interpretação do processo são mais fáceis de serem detetados tendo por base um estudo da evolução gráfica das diferentes variáveis. A pressuposição teórica do comportamento do sistema permite a deteção de uma evolução desenquadrada, consequente de uma modelação com algum/alguns lapsos. Um entendimento prévio sobre o comportamento do sistema, permite assim, não só uma conceção mais realista, tendo por base de comparação o sistema real replicado, como também uma maior capacidade em detetar disfunções na simulação decorrentes de erros de modelação projetados.

Melhorias que podem ser implementadas:

- Redefinir alguns modelos, tentando conceber modelos instanciados, passíveis de utilização na modelação de outros sistemas. Grande parte dos modelos foram adaptados exclusivamente à modelação do presente sistema pelo que pecam por alguma versatilidade em termos de utilização posterior. A especificidade de grande parte dos modelos é assim elevada, mas modificações requeridas são, no entanto, não significativas, pendentes apenas por detalhes (fundamentalmente a nível de entradas, saídas e parametrização).

Conclusões – Sistema *Pick&Place* desenvolvido

O desenvolvimento da modelação de um sistema *Pick&Place* recorrendo à linguagem Modelica, revelou-se, por um lado, uma tarefa complexa pela multipolaridade de domínios implicados na modelação, cuja complexidade se revela cada vez maior em função da rigorosidade procurada no desenvolvimento dos diferentes modelos, ou de uma maior abrangência dos domínios de engenharia que se queiram quantificar. Mas por outro lado, o uso da linguagem permitiu contrabalançar essas exigências pela capacidade em suportar não só o *design* do sistema físico, sem impor qualquer tipo de limites artificiais que restringem o uso multi-domínio, como também o *design* do sistema de controlo.

Relativamente ao manipulador, as diferenças que saltam em evidência relativamente ao modelo de referência, retirado da biblioteca presente na linguagem, ditam fundamentalmente uma modelação mais refinada do comando do manipulador (replicando uma delineação de movimento própria do computador principal associado), permitindo um estudo mais aproximado dos percursos envolvidos em ambiente industrial, assim como o estudo de todas de variáveis implicadas no movimento. Os torques, e consequentemente o aquecimento dos motores que resultam numa dissipação térmica, apresentam picos para valores de massa da ferramenta (*gripper*) mais elevados, como seria expectável, e para um perfil de velocidades polinomial de 5ª ordem, explicado por variações de velocidade elevadas.

Na modelação dos tapetes, foi importante fazer uma exposição das diferentes opções de modelação para uma arquitetura igual, tendo cada uma das diferenças sido implementadas de acordo com a função específica de cada tapete. A modelação das perdas envolvidas é um fator importante a ter em consideração pelo seu impacto significativo na *performance* e, em paralelo com as conclusões retiradas com os perfis de movimento implementados no robot, foi possível perceber que a transferência de calor, como consequência do torque total abrangido, vê os seus valores a variarem dependendo do tipo de perfil de movimento implementado, e cujos fatores determinantes envolvidos são as constrições cinemáticas definidas nos perfis assim como a amplitude de movimento total.

Na modelação do *gripper*, os diferentes comandos a serem admitidos, em instâncias diferentes no circuito de controlo, foi sem dúvida o aspeto de maior relevância, comparativamente ao controlo do manipulador e tapetes. A modelação da estrutura mecânica de cada um dos dedos destacou-se ainda pela presença de um conjunto de restrições cinemáticas importantes para assegurar um funcionamento íntegro da recolha e deposição da embalagem.

Melhorias a serem implementadas:

- Maior foco nos valores envolvidos que são parametrizados ao longo do conjunto de modelos. O centro da atenção ao longo do projeto focou-se mais no desenvolvimento correto de modelos, colocando para plano secundário muitas das vezes os valores dos

parâmetros envolvidos. É evidente que nenhum modelo, por mais bem concebido que seja, traduzirá uma replicação aproximada de um processo real, se os parâmetros definidos e que caracterizam o sistema não estiverem bem projetados. Uma revisão assim destes valores, acompanhado por um estudo detalhado que se debruce nas limitações que estes possam oferecer assim como normalizações existentes, deverá ser efetuada.

- Conexão manipulador – *gripper* deverá ser solucionada, permitindo perfazer um estudo mais bem enquadrado entre os dois sistemas e as implicâncias que um apresenta no outro, principalmente o peso do *gripper* na estrutura do manipulador.
- Uma melhoria geral dos subsistemas, tendo por base a arquitetura global delineada.

REFERÊNCIAS BIBLIOGRÁFICAS

Hayat, A. A., Chittawadigi, R., Udai, A. D., & Saha, S. K. (2013). Identification of Denavit-Hartenberg Parameters of an Industrial Robot. *AIR '13: Proceedings of Conference on Advances In Robotics.* Pune, India.

.*Modelica.Blocks.Sources.KinematicPTP2*. (s.d.). Obtido de https://build.openmodelica.org/Documentation/Modelica.Blocks.Sources.KinematicPTP2.html

.*Modelica.Electrical.Machines.Examples.DCMachines.DCPM_Cooling*. (s.d.). Obtido de https://build.openmodelica.org/Documentation/Modelica.Electrical.Machines.Examples.DCMachines.DCPM_Cooling.html

.*Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.FullRobot*. (s.d.). Obtido de OpenModelica Build Server: https://build.openmodelica.org/Documentation/Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.FullRobot.html

.*Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Utilities.PathPlanning6*. (s.d.). Obtido de https://build.openmodelica.org/Documentation/Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Utilities.PathPlanning6.html

ABB. (1995). *Electrical Troubleshooting Manual - IRB6400 Industrial Robot & S4 Control System.*

ABB AB, R. (2004-2017). *Product specification - IRB 140 (Revision: H).* Västerås.

Babu, B. P., & V., I. (2020). Analysis of Back To Back (BTB) Converter Control Strategies in Different Power System Applications. *IOP Conference Series: Materials Science and Engineering.* IOP Publishing.

Banks, J. (1998). *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice.*

Craig, J. J. (2005). *Introduction to Robotics : Mechanics and Control.* Pearson Prentice Hall .

Erlhagen, E. B. (2020). Kinematic analysis of Industrial Serial Robots with 6 DoF. *Case study: Yaskawa Motoman MH5.* Guimarães.

Erlhagen, E. B. (2020). Trajectory Generation in robotic manipulators. Guimarães.

Ferretti, G., Magnani, G., Rocco, P., & Viganò, L. (2006). Modelling and simulation of a gripper with Dymola. *Mathematical and Computer Modelling of Dynamical Systems, 12*.

Fritzson, P. (2011). *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica.*

Ibrahim, M. A., Hamoodi, A. N., & Salih , B. M. (2020). PI controller for DC motor speed realized with simulink and practical measurements. *International Journal of Power Electronics and Drive System (IJPEDS)*, 119~126.

IRB 140 - ABB. (s.d.). Obtido de https://new.abb.com/products/3HAC020536-001/irb-140

K. , P., & A. , M. (2018). DC Motor Speed Control Using PWM. *International Journal of Innovative Science and Research Technology*, 584-587.

M.Tiller, M. (2001). Multi-Domain Modeling - Conveyor System . Em *Introduction to Physical Modeling with Modelica* (p. 232).

M.Tiller, M. (2019). *Modelica by Example.*

Maria, A. (1997). Introduction to modeling and simulation. *Proceedings of the 29th conference on Winter simulation*, 7-13.

ModelicaShortCourse. (s.d.). Obtido de https://openmodelica.org/doc/ModelicaShortCourse/ModelicaTutorial-slides-PeterFritzson-160202-BT.pdf

OpenModelica. (s.d.). Obtido de https://openmodelica.org/

Pickering, J. (2014). Analogue to Digital and Digital to Analogue Converters (ADCs and DACs): A Review Update. *Proceedings of the CAS-CERN Accelerator School: Power Converters* (pp. 363-377). Baden, Switzerland: CERN .

Polenghi, A., Fumagalli, L., & Roda, I. (2018). Role of simulation in industrial engineering: focus on manufacturing systems. *IFAC PapersOnLine*, 496-500.

R.K. Boel, B. D. (1999). *Approaches to modelling, analysis, and control of hybrid systems.*

ROBERGE, J. K. (1975). *OPERATIONAL AMPLIFIERS: Theory and Practice.* John Wiley & Sons (Wiley).

Siciliano, B., Sciavicco , L., Villani, L., & Oriolo , G. (2010). *Robotics : Modelling, Planning and Control.* Springer Science & Business Media.

Smith, J. S. (2003). Survey on the use of simulation for manufacturing system design and operation. *Journal of Manufacturing Systems*, 157-171.

Wang, L. (2020). Basics of PID Control. Em L. Wang, *PID Control System Design and Automatic Tuning using MATLAB/Simulink* (pp. 1-55). Wiley-IEEE Press.

Yoon, H. J., Chung, S. Y., Kang, H. S., & Hwang, M. J. (2019). Trapezoidal Motion Profile to Suppress Residual Vibration of Flexible Object Moved by Robot . *Electronics , 8*.

Yu, A., Wang, C., Guo, X., Li, Z., Zhang, C., & Guerrero, J. M. (2022). New Rotor Position Redundancy Decoding Method Based on Resolver Decoder. *Micromachines*.

APÊNDICE 1 - FORMALISMO GRAFCET – SISTEMA

O *Grafcet* é uma metodologia utilizada no controlo de processos sequenciais e que permite uma discretização de um sistema em estruturas de etapas, ações e transições correlacionadas de modo que a evolução do sistema depende sempre da verdade das transições e as ações estão associadas à etapa que as ativa. O *Grafcet* constitui deste modo um método claro e sintetizado de definir o controlo de um sistema que se pretende automatizar.

- **Entradas e Saídas do Sistema**

É importante em qualquer sistema que se pretenda conceber, disponibilizar ao autômato associado quais as entradas e saídas do sistema. As entradas referem-se às variáveis que serão transmitidas ao controlador e mediante as quais o programa do controlador manipulará o sinal das variáveis de saída.

Tabela 30 - Entradas do sistema e descrição

Entradas	Significado
START	Variável booleana, representa o botão monoestável que inicia o ciclo quando estão garantidas as condições para tal
STOP	Variável booleana, representa o botão monoestável que interrompe o ciclo de funcionamento quando não estão garantidas as condições para que o mesmo prossiga
tp1_0	Deteção de uma caixa no início do Tapete 1
tp1_1	Deteção de uma caixa no fim do Tapete 1
tp2_0	Deteção de uma caixa no início do Tapete 2
tp2_1	Deteção de uma caixa no fim do Tapete 2
pose_0	Configuração espacial do manipulador definida como inicial

Tabela 31 - Saídas do sistema e descrição

Saídas	Significado
MT1	Acionamento do Motor associado ao movimento do Tapete 1
MT2	Acionamento do Motor associado ao movimento do Tapete 2

Com base na tabela 30 e tabela 31 é possível esquematizar as entradas e saídas do sistema, do modo presente na Figura 216, que permite de uma forma sucinta ter uma ideia global do controlador do funcionamento do sistema.

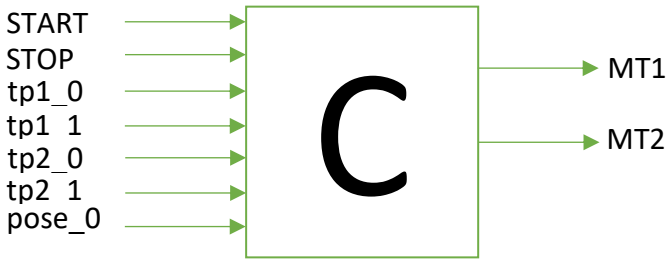


Figura 216 - Controlador com as entradas e saídas representadas

De salientar a simplicidade apresentada pelo *grafcet* elaborado. A sua extensão com a adição de condições adicionais foi desconsiderada, uma vez que os pressupostos em que este se baseou foram os mesmos tomados para a modelação orientada a objetos em que apenas um ciclo de operação é admitido, que ao longo do ciclo o manipulador já se encontra disposto com a uma configuração previamente estabelecida e que nenhuma caixa se encontra previamente num dos 2 tapetes (entrada e saída).

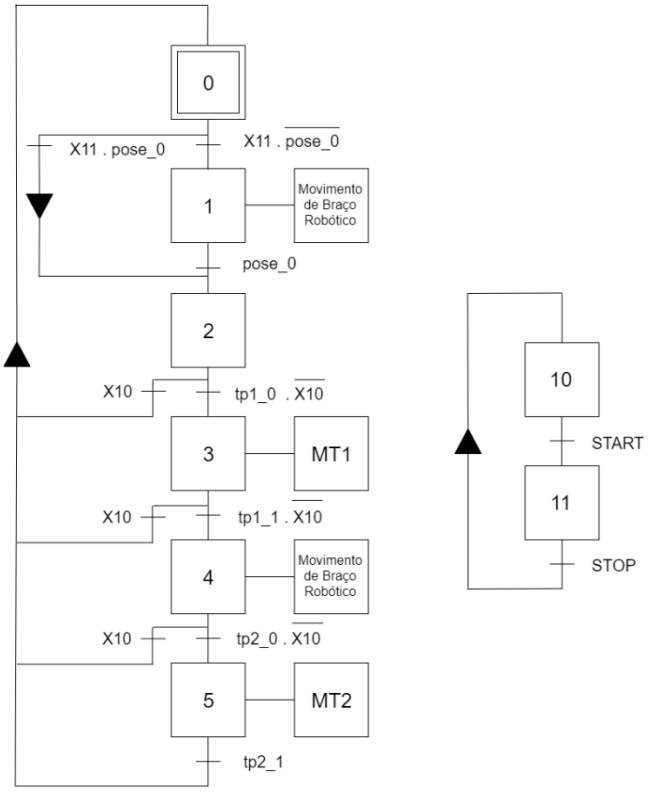


Figura 217 - *Grafcet* do sistema global *Pick&Place*

APÊNDICE 2 - SIMULAÇÃO DO AMBIENTE FÍSICO – *MATLAB* E *COPPELIA*

A modelação e simulação orientada a objetos é precedida por um estudo cuidado do processo físico, nomeadamente o braço robótico, pela complexidade de controlo superior requerida, em comparação com os tapetes transportadores. Com esse propósito, um estudo da cinemática inversa do braço foi realizado, acompanhado pelo desenvolvimento de um *script* em *Matlab*, não só da cinemática inversa mas sim de toda a logística responsável pela implementação de trajetória.

Cinemática Inversa - Matlab

- Definição do comprimento dos elos, com base no vetor L e onde se incluem os valores das dimensões $d1, a1, a2, d4$ e $d6$ implicados nos parâmetros de *Denavit – Hartenberg*.
- Declaração dos parâmetros de *Denavit – Hartenberg* a_i , α_i , d_i e θ_i para os diferentes referenciais ($DH\_alpha,\ DH\_a, DH\_theta, DH\_d$).

```
function [joints] = InvKin(x, y, z, yaw, pitch, roll)

%Link length
L = [0.352 0.07 0.177 0.36 0.177 0.38 0.065];
%L(1, 1) = 0.34 - 0.0787; %Coppelia referencial adjustment
%L(1, 4) = 0.126 + 0.0238; %Tool adjustment

d1 = L(1);
a1 = L(2);
a2 = L(4);
d4 = L(6);
d6 = L(7);

%Denavit-Hartenberg parameters 7 DoF
%DH: [a, alpha,    d, theta]

DH_alpha = pi/180*[0, 90, 0, 90, -90, 90]';
DH_a = [0, a1, a2, 0, 0, 0,]';
DH_theta = pi/180*[-90, 90, 0, 0, 0, 0]';
DH_d = [d1, 0, 0, d4, 0, d6]';
```

Figura 218 - Função de Cinemática Inversa em Matlab - Excerto 1

- Definição do vetor $Pose\_tip$, em que se incorpora nos 3 valores iniciais a posição final do *end-effector* (P_{06}) e nos 3 valores finais os ângulos *Roll*, *Pitch* e *Yaw* (RPY), sendo que os mesmos são arbitrados com valor de 0.
- Declaração da matriz rotação R_{06}, com base nos valores declarados para os 3 ângulos envolvidos na orientação do *end-effector* (RPY) e ainda a matriz associada ao *end-effector* RPY_{tip} que define a matriz transformada entre o referencial base e o referencial do *end-effector*
- Definição da matriz global T_{06}, com base na matriz de rotação R_{06} e no vetor de posição cartesiana do *end-effector* (P_{06}).

```
Pose_tip = [x, y, z, yaw, pitch, roll]';
P06 = Pose_tip(1:3);
RPY = Pose_tip(4:6)*pi/180;

yaw_x = RPY(1);
pitch_y = RPY(2);
roll_z = RPY(3);

R06 = GetRot(yaw_x, pitch_y,roll_z);

RPY_tip = [0 1 0;
           1 0 0;
           0 0 -1];
RPY_tip_z = RPY_tip(1:3,3);

T06 = eye(4,4);
T06(1:3,1:3) = R06;
T06(1,4) = P06(1);
T06(2,4) = P06(2);
T06(3,4) = P06(3);
```

Figura 219 - Função de Cinemática Inversa em Matlab - Excerto 2

- Cálculo da posição do pulso (P_{05}), equivalente à subtração entre a posição especificada para a ponta do *end-effector* (P_{06}) e a distância entre o *end-effector* e o pulso (junta 5) multiplicada pelo eizo z da matriz de rotação RPY (RPY_{tip_z}).
- Definição dos valores das 2 soluções apresentadas para θ_1, $theta1\_L$ (quando o "cotovelo" esquerdo é utilizado) e $theta1\_R$ (quando o "cotovelo" direito é utilizado).

```
%THETA1
z06 = T06(1:3,3);
P05 = P06 - (d6+0.0238)*(RPY_tip_z);
P05_x = P05(1);
P05_y = P05(2);
P05_z = P05(3);

if abs(P05_y) <= 0.01
    theta1_L = 0;
else
    theta1_L = atan2(P05_y, P05_x);
end

theta1_R = theta1_L + pi;

if theta1_L < -pi || theta1_L >pi
    theta1 = theta1_R;
elseif theta1_R < -pi || theta1_R > pi
    theta1 = theta1_L;
else
    theta1 = theta1_L;
end
theta1_deg = theta1*180/pi;
```

Figura 220 - Função de Cinemática Inversa em Matlab - Excerto 3

- Cálculo do comprimento do vetor entre o ombro e o pulso r' ($r3$ no *script*), segundo a seguinte formulação $r3 = \sqrt{{P_{25_x}}^2 + {P_{25_y}}^2 + {P_{25_z}}^2}$. Precedido deste cálculo é apresentado o cálculo de P_{25} através da subtração entre o vetor entre a base e o pulso P_{05} e o vetor P_{02} definido entre a base e o segundo eixo do manipulador, sendo $P_{25} = \{P_{25_x},\ P_{25_y},\ P_{25_z}\}$.
- Cálculo de β_3($beta3$) segundo a equação, precedido pelo cálculo do argumento ($arg3$) $\frac{L_B^2 + L_C^2 - r'^2}{2\ .\ L_B\ .\ L_C}$ em que L_B e L_C se igualam às dimensões $a2$ e $d4$, respetivamente.
- Cálculo das duas soluções de θ_3 ($theta3\_L$ e $theta3\_R$). A solução que perfila o cotovelo esquerdo dá-se pela equação, enquanto que a solução estabelecida pelo cotovelo direito é calculada através da seguinte fórmula.

$$\theta_3 = 270 - \beta_3$$

```
% THETA2 AND THETA 3

%position of nodule 2 (shoulder)
P02 = [a1*cos(theta1), a1*sin(theta1), d1]';
%distance between shoulder and wrist(r3)
P25 = P05 - P02;
P25_x = P25(1);
P25_y = P25(2);
P25_z = P25(3);
r3 = sqrt ((P25_x)^2 + (P25_y)^2 + (P25_z)^2);

% beta3 calculation
LB=a2;
LC=d4;
arg3 = ((LB^2 + LC^2 - r3^2)/(2*LB*LC));
if arg3>= -1 || arg3<=1
    beta3= acos(arg3);
else
    %return %not feasible
end

% theta3 calculation
theta3_L = (beta3 - pi/2);
theta3_R = (3*pi/2 - beta3);
theta3 = theta3_L;
theta3_deg = theta3*180/pi;
```

Figura 221 - Função de Cinemática Inversa em Matlab - Excerto 4

- Cálculo de r' projetado ao longo do plano xy (r)
- Cálculo de α_2 ($alfa2$) recorrendo à cota do vetor P_{25} ($P25\_Z$) e r
- Cálculo de β_2($beta2$) segundo a equação, precedido pelo cálculo do argumento ($arg2$)

  $\frac{L_B^2 + r'^2 - L_C^2}{2\,.\,L_B\,.r'}$.
- Cálculo das duas soluções de θ_2 ($theta2\_L$ e $theta2\_R$). A solução que perfila o cotovelo esquerdo dá-se pela equação, enquanto que a solução estabelecida pelo cotovelo direito é calculada através da seguinte fórmula.

$$\theta_2 = 90^o - \alpha_2 + \beta_2$$

- Cálculo das 3 primeiras matrizes de transformação individual ${}^0T_1$, ${}^1T_2$ e ${}^2T_3$ ($T01$, $T12$ e $T23$).
- Cálculo da matriz ${}^3R_6$ ($R36$), necessária para o cálculo dos 3 ângulos (θ_4 , θ_5 e θ_6).

```
% alfa2 e beta 2 calculation
r = sqrt((P25_x)^2 + (P25_y)^2);
alfa2 = atan2(P25_z,r);
arg2 = ((LB^2 + r3^2 - LC^2)/(2*LB*r3));
if arg2>=-1 || arg2 <=1
    beta2 = acos(arg2);
else
    % return %not feasible
end

% theta 2 calculation
theta2_L = - (pi/2 - alfa2 - beta2);
theta2_R = - (pi/2 - alfa2 + beta2);
theta2 = theta2_L;
theta2_deg = theta2*180/pi;

% We know theta1,theta2 and theta3, then we can compute T01,T12,T23
T01 = TransfMatrix(DH_alpha(1), DH_a(1), DH_d(1), (DH_theta(1)+theta1));
T12 = TransfMatrix(DH_alpha(2), DH_a(2), DH_d(2), (DH_theta(2)+theta2));
T23 = TransfMatrix(DH_alpha(3), DH_a(3), DH_d(3), (DH_theta(3)+theta3));

%COMPUTE R36
T03 = T01*T12*T23;
R03 = T03(1:3, 1:3);
R30 = R03';
R36 = R30*R06;
```

Figura 222 - Função de Cinemática Inversa em Matlab - Excerto 5

- Com base na definição da matriz ${}^{3}R_{6}$, cada um dos elementos constituintes é posteriormente determinado (r_{11}, r_{12}, r_{13}, etc).

```
%COMPUTE R36
T03 = T01*T12*T23;
R03 = T03(1:3, 1:3);
R30 = R03';
R36 = R30*R06;

r11 = R36(1,1);
r12 = R36(1,2);
r13 = R36(1,3);
r21 = R36(2,1);
r22 = R36(2,2);
r23 = R36(2,3);
r31 = R36(3,1);
r32 = R36(3,2);
r33 = R36(3,3);
```

Figura 223 - Função de Cinemática Inversa em Matlab - Excerto 6

- Cálculo das duas soluções para θ_5, recorrendo aos elementos presentes na equação.
- Cálculo das soluções genéricas para θ_4 e θ_6, assim como das soluções condicionais dependentes do valor de θ_5.
- Definição final do vetor $joints$ constituído pelos 6 ângulos.

```
%theta5 calculation

theta5_R = atan2(sqrt(1-(r23)^2), r23);
theta5_L = atan2(-sqrt(1-(r23)^2), r23);
theta5 = theta5_L;
theta5_deg = theta5*180/pi;
if abs(theta5) < 0.01
    theta4 = 0;
    theta6 = atan2(r12, r32);
elseif abs(theta5-pi)< 0.01
    theta4 = 0;
    theta6 = -atan2(-r12, -r32);
else
    theta4 = atan2(r33/sin(theta5), -r13/sin(theta5));
    theta6 = atan2(-r22/sin(theta5), r21/sin(theta5));
end

theta4_deg = theta4*180/pi;
theta6_deg = theta6*180/pi;

joints = [theta1 theta2 theta3 theta4 theta5 theta6];
disp((DH_theta(2)+theta2));

end
```

Figura 224 - Função de Cinemática Inversa em Matlab - Excerto 7

Implementação – *Pick and Place*

Ao longo deste subcapítulo, o código correspondente à componente de Integração de Trajetórias em *Matlab*, que permitem ao robot executar o modo de operação de *Pick and Place*, será apresentado e descrito respetivamente. Ao longo deste será apresentado apenas a logística de código associada à trajetória "Centrar", sendo que os restantes ciclos designados para as restantes trajetórias seguem sempre a mesma linha de raciocínio. De referir que, em relação à modelação orientada a objetos, um maior número de *via points* foi considerado, para todo o processo.

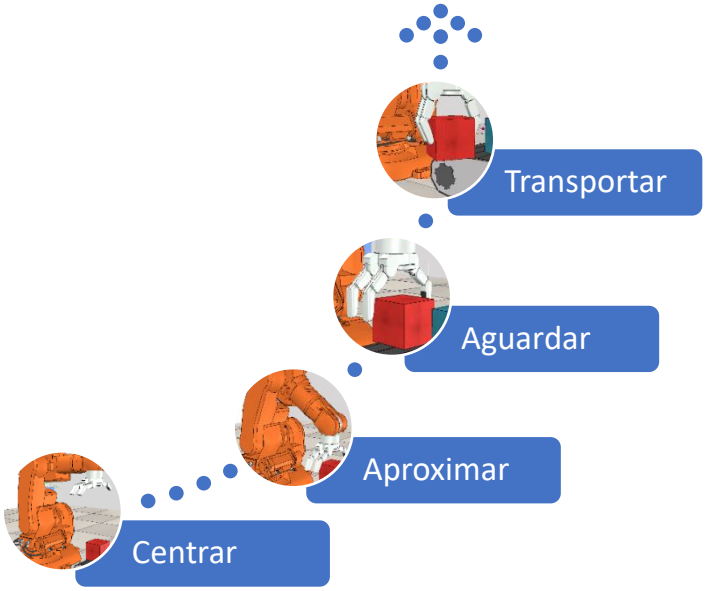


Figura 225 - Representação dos 3 tipos de trajetórias associadas aos diferentes modos de operação *Pick&Place*

Numa fase inicial definimos o nome do robot, de forma a que o simulador o reconheça e permita que este seja inicializado, procedido pela definição de um conjunto de funções que permitem a leitura das características do robot, o movimento do tapete transportador assim como a abertura e fecho da mão.

```
% Before running this script, open the scenario in CoppeliaSim, e.g
% Do not run simulation!
clear
% need to choose the arm to control
obj_name = 'IRB140';
%obj_name = 'UR10';
%obj_name = 'LBR_iiwa_7_R800';
%obj_name = 'AROS';
[robot_arm,error] = arm_robot(obj_name); %initialize robot class
if error == 1
    return;
end

timestep = robot_arm.get_simulation_timestep(); %get time step value (normally dt=50ms)

[error1,nJoints,Links,MinPositionJoint,MaxPositionJoint] = robot_arm.get_RobotCharacteristics();
%nJoints - number of arm joints.
%Links - dimensions of the links between the axes of rotation
%MinPositionJoint - array with minimum position for joint 1-6
%MaxPositionJoint - array with maximum position for joint 1-6

stop=0;
error = robot_arm.move_conveyorbelt();
%Function that allows you to put the conveyor belt in motion

error = robot_arm.open_hand();
%Function that allows open the hand
```

Figura 226 - Integração de trajetória em *Matlab* para simulação em *CoppeliaSim*

❖ Para que as diferentes trajetórias ao longo dos pontos iniciais e finais fossem estabelecidas, diferentes ciclos while foram criados para que estas corressem de forma independente e apenas quando desejado. As condições de início de cada ciclo foram associadas ao conjunto de variáveis criadas sendo que as mesmas poderão ser distinguidas da seguinte forma:

- **Trajetória *Test_pegar*:** *test_pegar*
- **Trajetória *test_agarrar*:** *test_agarrar*
- **Trajetória *test_aguardar*:** *test_transporte*
- **Trajetória *test_transporte***

Desta forma, o ciclo while associado apenas será iniciado sempre que cada uma das variáveis adota o valor lógico de 1.

```
i=1;
j=0;
test_pegar=0;
test_agarrar=0;
test_aguardar=0;
test_transporte=0;

test_pegar_2=0;
test_agarrar_2=0;
test_aguardar_2=0;
test_transporte_2=0;

test_pegar_3=0;
test_agarrar_3=0;
test_aguardar_3=0;
test_transporte_3=0;

final_position=0;

%Referencial adjust and consider toll height
Links_M = cell2mat(Links);
Links_M (1, 1) = 0.352 - 0.0764;
Links_M (1, 7) = 0.065 + 0.0238;

%Cell to Matrix
MinPositionJoint_M = cell2mat(MinPositionJoint);
MaxPositionJoint_M = cell2mat(MaxPositionJoint);
```

Figura 227 - Integração de trajetória em *Matlab* para simulação em *CoppeliaSim* - Definição das variáveis relativas ao tipo de trajetória a ser implementada

❖ Ainda na porção de código apresentado em cima, os valores dos elos e dos limites das juntas que são lidos através da função de leitura de características inicial, são guardados numa matriz para que possam posteriormente ser utilizados como entrada nas diferentes funções que sejam necessárias para a elaboração dos cálculos de trajetória como por exemplo a função da cinemática inversa. Este detalhe torna a configuração conjuntural do algoritmo final mais versátil, visto que caso se decida por exemplo pela mudança da classe do robot, essas características dimensionais seriam atualizadas de forma automática nas funções referidas.

```
while stop==0
    i=i+1
    %% Robot interface
    % set and get information to/from vrep
    % avoid do processing in between ensure_all_data and trigger_simulation
    robot_arm.ensure_all_data();

    % get joint value (rad) for arm
    [ReadArmJoints] = robot_arm.get_joints()

    % get robot position
    [armPosition] = robot_arm.get_robot_position()

    % get target position
    [targetPosition]=robot_arm.get_target_position()

    %get simulation time
    sim_time = robot_arm.get_simulation_time()
    %trigger simulation step
    robot_arm.trigger_simulation();

    % --- YOUR CODE --- %
    tol=1e-2;
```

Figura 228 - Integração de trajetória em *Matlab* para simulação em *CoppeliaSim* - Ciclo *While* Geral

Ciclo while é criado com a condição de que o stop associado à interrupção de simulação não se encontre ativada no simulador. Este ciclo engloba a trajetória total admitida com o conjunto de todas as trajetórias segmentáveis e integradas ao longo dos restantes ciclos while. Aqui tem-se ainda a inserção das diferentes funções que comunicam com o Simulador em tempo real, obtendo de forma atualizada e constante todos os valores necessários para a programação desejada posterior.

```
    %Inverse Kinematics calculation

            armJoints{1}= 0;
            armJoints{2}= 0;
            armJoints{3}= 0;
            armJoints{4}= 0;
            armJoints{5}= 0;
            armJoints{6}= 0;
            error = robot_arm.set_joints(armJoints)

            [ReadArmJoints] = robot_arm.get_joints()


            ReadArmJoints_M = cell2mat(ReadArmJoints);
            armJoints_M = cell2mat(armJoints);

            if (norm (ReadArmJoints_M-armJoints_M)<=tol)
                test_pegar=1;
            end
```

Figura 229 - Integração de trajetória em *Matlab* para simulação em *CoppeliaSim* - Definição da *home position* e da variável de saída

❖ Definição dos valores da matriz das juntas a um conjunto de 0's, correspondendo assim à posição inicial definida para o robot. Quando a norma da diferença entre valor da matriz das juntas lido pelo programa e a matriz de zeros é menor que a tolerância definida, então

admitimos que a posição do braço já se encontre na posição inicial e a primeira trajetória entre os dois primeiros via points poderá ser iniciada, com a variável *test_pegar* tomando o valor de 1.

```
while  test_pegar==1 && test_agarrar==0 && stop==0 % Advance to the target
    j=j+1
    %% Robot interface
   % set and get information to/from vrep
   % avoid do processing in between ensure_all_data and trigger_simulation
   robot_arm.ensure_all_data();

   % get joint value (rad) for arm
   [ReadArmJoints] = robot_arm.get_joints()

   % get robot position
   [armPosition] = robot_arm.get_robot_position()

   % get target position
   [targetPosition]=robot_arm.get_target_position()

   %get simulation time
   sim_time = robot_arm.get_simulation_time();



   %trigger simulation step
   robot_arm.trigger_simulation();
```

Figura 230 - Integração de trajetória em *Matlab* para simulação em *CoppeliaSim* - Ciclo associado à trajetória *test_pegar*

❖ Criação de um ciclo while, cuja condição de entrada no ciclo apresente o valor da variável da trajetória em causa de 1, mas garantido ainda que a variável de saída do ciclo admita um valor de 0 e o stop de simulação não tenha sido premido. As funções de leitura e comunicação são novamente inseridas para que a comunicação seja estabelecida e informação recolhida de todas as vezes que ocorre uma iteração no interior deste ciclo.

❖ No início do ciclo tem-se ainda a apresentação da equação que permite incrementar o tempo, cujo valor de cada time-step de simulação é 50 ms. A variável de tempo inserida é a variável *j* e como se verá de seguida, as diferentes iterações de tempo ao longo do ciclo permitem a evolução do valor dos ângulos das juntas.

```
if j==1
%velocity constraints all joints
vi = 0;
vf = 0.02;
position = [0, 0.4965, 0.6356];
orientation= [0, 0, 0];
pose = [0, 0, 0, 0, 0, 0];
pose(1:3)=position(1:3);
pose(4:6)=orientation(1:3);

%position constraints
[joints_pegar] = InvKin(pose(1), pose(2), pose(3), pose(4), pose(5), pose(6));

%matrix for joints differential
[joints_differential] = armJoints_M - [joints_pegar];

%Compute bigger differential in joints in a trajectory
bigger_joints_differential = max(abs(joints_differential(:)));

% T - number of time steps for maximum velocity of 20 degrees/s
T = ceil( ((bigger_joints_differential)*(180/pi))/(20*0.05));


%theta1 and computing of its coefficients
theta_1_i = armJoints{1};
theta_1_f = joints_pegar(1);
coeff_1 = trajectory_coefficients(theta_1_i, theta_1_f, vi, vf, T);
```

Figura 231 - Integração de trajetória em *Matlab* para simulação em *CoppeliaSim* - Iteração para a definição das restrições e dos coeficientes da juntas

❖ Na primeira iteração do ciclo, ou seja, quando a variável *j* admite o valor de 1, um conjunto de diferentes parâmetros, especificamente para a trajetória do ciclo são calculados. Mais concretamente, a velocidade inicial, a velocidade final, as coordenadas da posição final, orientação do manipulador e ainda o *arm plane angle*. Além destes, um outro parâmetro que é calculado de enorme importância, é o número de incrementos necessários para que a trajetória se complete na totalidade. De forma a estabelecer esse valor, foi definido um valor de 20°/s para a velocidade máxima de rotação das juntas, que em conjunto com um valor de 50 ms de time-steps e o valor relativo ao diferencial máximo na variação do valor das juntas, tornando assim possível o cálculo do número de *time-steps* requeridos para que o movimento entre as devidas posições se concretize.

❖ É definida uma matriz com o valor das juntas desejadas, de maneira a que a posição final com a orientação e *arm plane angle* inseridos, seja atingida. O valor das juntas foi calculado recorrendo à função da cinemática inversa.

❖ Uma vez definidos os parâmetros, o cálculo dos diferentes coeficientes relativos à equação polinomial de geração de trajetórias torna-se possível. Os 4 coeficientes para a geração de trajetória de cada junta, foram definidos recorrendo à seguinte função:

```
function [coefficients] = trajectory_coefficients(teta_p_i,teta_p_f,vi,vf,T)

a0=teta_p_i;
a1=vi;
a2=(3/(T^2))*(teta_p_f - teta_p_i)-(2/T)*vi - (1/T)*vf;
a3=(-2/T^3)*(teta_p_f - teta_p_i) + (1/T^2)*(vf+vi);
coefficients=[a0,a1,a2,a3];
end
```

Figura 232 - Função para o cálculo dos coeficientes

Em que, com recurso aos parâmetros de entrada, os diferentes coeficientes são calculados e colocados posteriormente na respetiva matriz.

```
if j<=T
armJoints{1}= set_arm_joint(coeff_1,j);
armJoints{2}= set_arm_joint(coeff_2,j);
armJoints{3}= set_arm_joint(coeff_3,j);
armJoints{4}= set_arm_joint(coeff_4,j);
armJoints{5}= set_arm_joint(coeff_5,j);
armJoints{6}= set_arm_joint(coeff_6,j);
end

error = robot_arm.set_joints(armJoints) %send value for arm Joints in rad

[ReadArmJoints] = robot_arm.get_joints()

ReadArmJoints_M = cell2mat(ReadArmJoints);

if (norm (ReadArmJoints_M-joints_pegar)<=tol)
     test_agarrar=1;
     error = robot_arm.close_hand();
end

  pause(0.05);
end

 j=0;
```

Figura 233 - Integração de trajetória em *Matlab* para simulação em *CoppeliaSim* - Cálculo dos novos valores das juntas e envio ao simulador

❖ Por fim, definidos os coeficientes, a equação polinomial de geração de trajetória torna-se possível de ser elaborada para cada uma das juntas, sendo que a variável é o tempo, ou seja, a variável *j,* e novos valores são obtidos sempre que uma nova iteração é efetuada incrementando o valor de *j* .Para o cálculo dos novos valores das juntas foi utilizada então uma função onde, recorrendo aos coeficientes característicos de cada junta calculados anteriormente, os novos valores das juntas poderão ser obtidos. A função poderá ser visualizada de seguida:

```
function [armJoints_M] = set_arm_joint(coeff,j)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
armJoints_M = coeff(1)+coeff(2)*(j)+coeff(3)*(j^2)+coeff(4)*(j^3);

end
```

Figura 234 - Função definindo a equação polinomial de geração de trajetória

❖ O ciclo acaba com os novos valores de juntas a serem enviados para o Simulador, sendo de seguida iniciado novamente o ciclo com um valor incrementado de *j.* Quando, por fim, na última iteração, os valores lidos se igualam aos valores previamente calculados para a posição desejada, recorrendo à função da cinemática inversa, um novo valor de 1 é atribuído à variável associada relativa à trajetória seguinte a ser realizada, permitindo assim o avanço do programa para um novo ciclo while, em que todos os aspetos de código abordados até agora ao longo deste ciclo se repetem. É importante mencionar ainda, que devido a incertezas de cálculo, o ciclo acaba de ser finalizado quando a posição desejada é atingida dentro de uma determinada tolerância atribuída.

Os ciclos while correspondentes à trajetória *Grasp* e ainda trajetória *Transport and Release* utilizados ao longo do código dos diferentes modos de operação, obedecem à mesma formulação de código, modificando apenas as restrições de trajetória utilizadas (posições inicial e final assim como velocidades inicial e final) assim como as designações das matrizes utilizadas.

Interface Gráfica de Controlo

Recorrendo à ferramenta *App Designer* do *MATLAB,* foi possível conceber uma interface a ser disponibilizada ao utilizador, garantido não só a capacidade deste em escolher se pretende que o braço execute o seu funcionamento normal ou não, assim como executar funções complementares.

Desta forma, pode-se distinguir 2 áreas distintas ao longo da elaboração da interface gráfica:

Cálculo da Cinemática Inversa - O utilizador introduz os valores da pose, obtendo assim as coordenadas das juntas.

Tipo de Funcionamento Desejado - Permite ao utilizador aceder à escolha do tipo de funcionamento que deseja. Foram definidas as seguintes opções:

- **Modo Automático:** Equivalente ao comportamento *Pick&Place* ao longo de um ciclo de operação. As diferentes etapas envolvidas na trajetória são adotadas de forma automática.

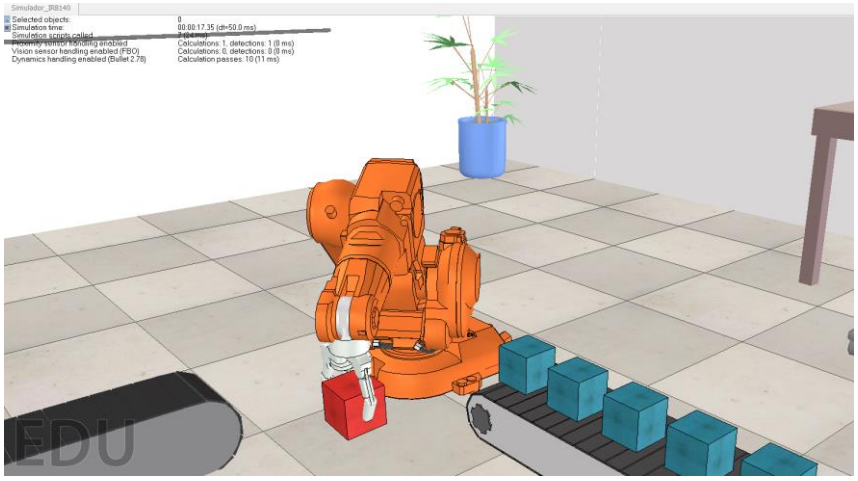
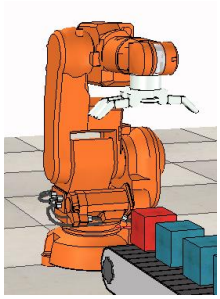
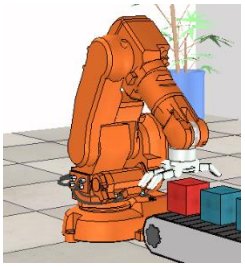
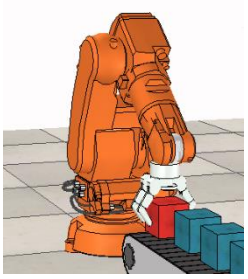
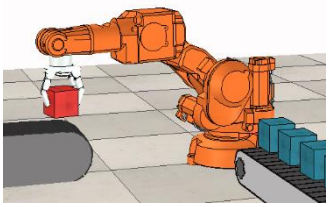


Figura 235 - *Pick&Place* - Modo automático

- **Modo Manual:** Equivalente ao comportamento *Pick&Place* ao longo de um ciclo de operação, no entanto o utilizador dita através da interface os momentos em que as diferentes etapas de trajetórias são calculadas e, de forma consequente, implementadas.

Tabela 32 - Ilustração das diferentes etapas presentes no modo manual

Modo Manual			
Centrar	**Aproximar**	**Agarrar**	**Transporte**
			

Como anteriormente mencioado, através da *App Designer* do *MATLAB,* um **GUI (** Graphical User Interface) foi criado. O mesmo poderá ser visualizado na Figura 236.

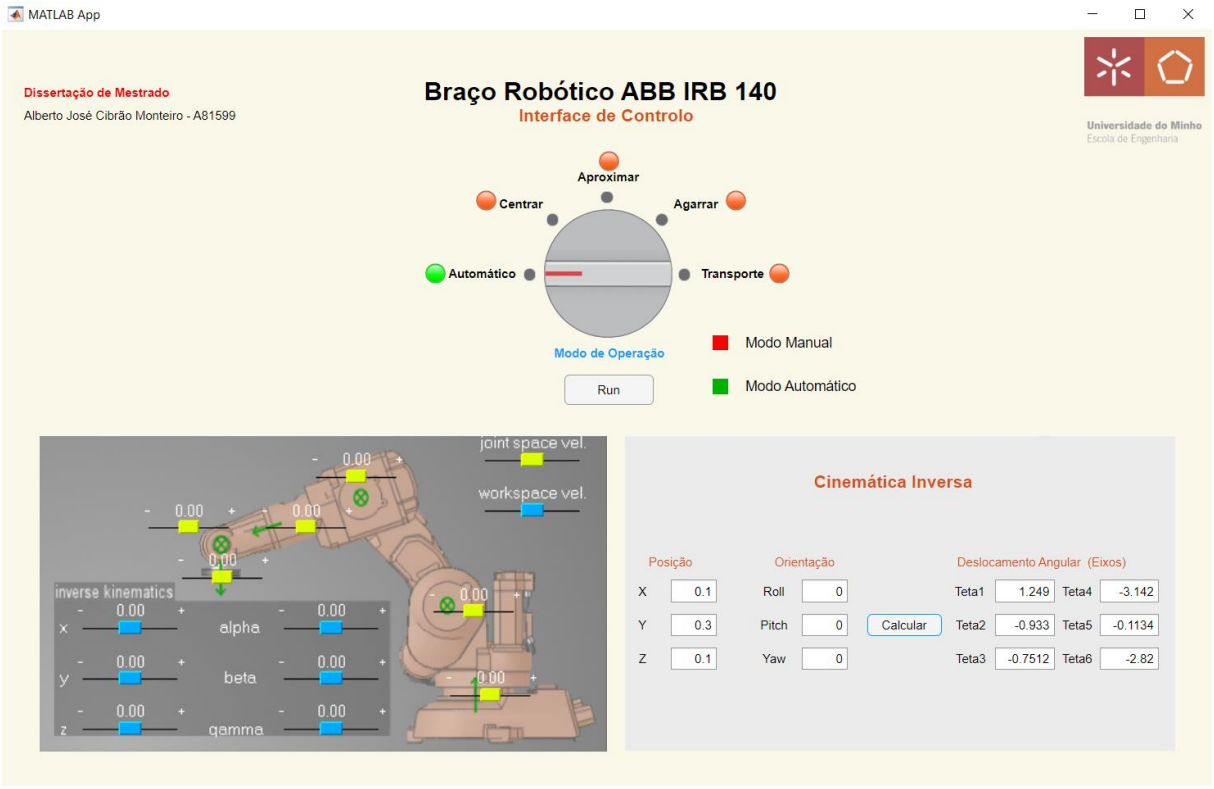


Figura 236 - Interface de Controlo: ABB IRB 140

➢ Callbacks

De forma a garantir que as funções necessárias para a apresentação dos valores ou simulações desejadas sejam processadas nos momentos requeridos, é necessário "chamá-las" sempre que a condição estabelecida seja atuada. Para o nosso caso, as condições definidas foram os botões de cálculo que permitem calcular os valores da cinemática e ainda o "*Discrete Knob*" por forma a estabelecer o funcionamento desejado e assim iniciar a reprodução da respetiva simulação e/ou função associada.

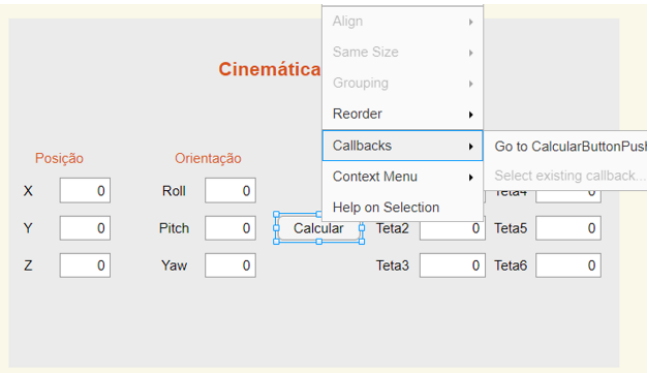


Figura 237 - Definição de um *Callback* para a Cinemática Direta

De maneira a conseguir estabelecer essa ligação entre os componentes atuados na interface e as funções associadas, é necessário configurar o componente através de um *Callback*. Quando premido, este redireciona automaticamente para a secção no código onde uma função poderá ser introduzida e assim, obter-se a associação desejada.

Exprimindo o exemplo da Cinemática Inversa, a função *InvKin* é chamada quando o Botão de Cálculo é premido, sendo posteriormente processada e permitindo a apresentação dos resultados na interface.

Tabela 33 - *Callback* e Função associada

Callback	**Função**

`% Callbacks that handle component events`
`methods (Access = private)`
 	
`% Button pushed function: CalcularButton`
`function CalcularButtonPushed(app, event)`
`InvKin(app)`
`end`
	`methods (Access = private)`
	
	`function results = InvKin(app)`
	`%Link length`
	`L = [0.352 0.07 0.177 0.36 0.177 0.38 0.065];`
	`%L(1, 1) = 0.34 - 0.0787; %Coppelia referencial adjustment`
	`%L(1, 4) = 0.126 + 0.0238; %Tool adjustment`
	
	`d1 = L(1);`
	`a1 = L(2);`
	`a2 = L(4);`
	`d4 = L(6);`
	`d6 = L(7);`
	`(...)`

É importante referir que para todas as opções disponíveis ao longo da interface, as funções associadas já estavam parcialmente elaboradas, pelo que o processo ficou bastante simplificado. Foi ainda necessário elaborar alguns ajustes, de forma a que os resultados a serem apresentados na interface fossem devidamente associados aos indicadores gráficos.

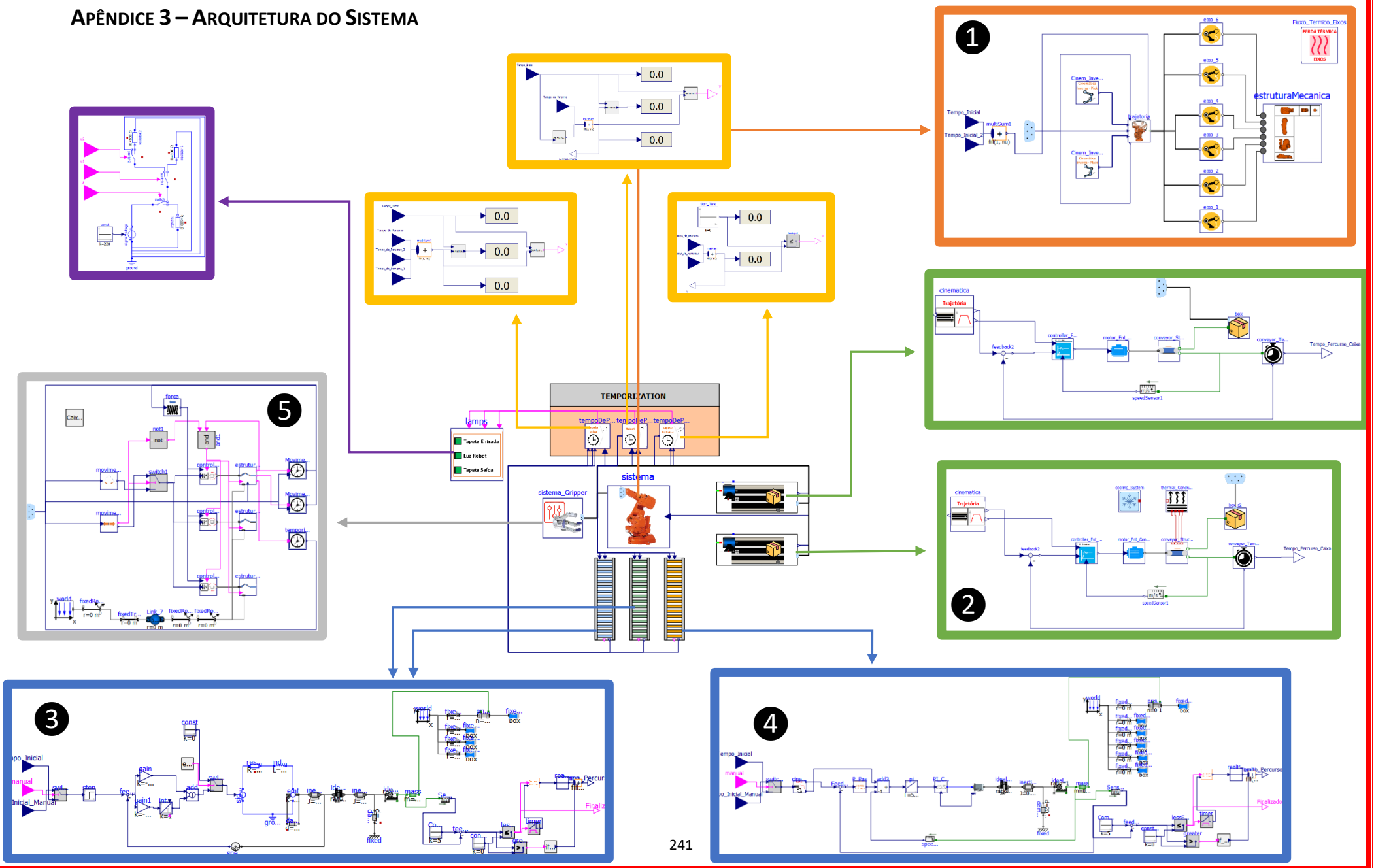
```
(...)

joints = [theta1 theta2 theta3 theta4 theta5 theta6];
disp((DH_theta(2)+theta2));

app.Teta1EditField.Value = joints(1);
app.Teta2EditField.Value = joints(2);
app.Teta3EditField.Value = joints(3);
app.Teta4EditField.Value = joints(4);
app.Teta5EditField.Value = joints(5);
app.Teta6EditField.Value = joints(6);
```

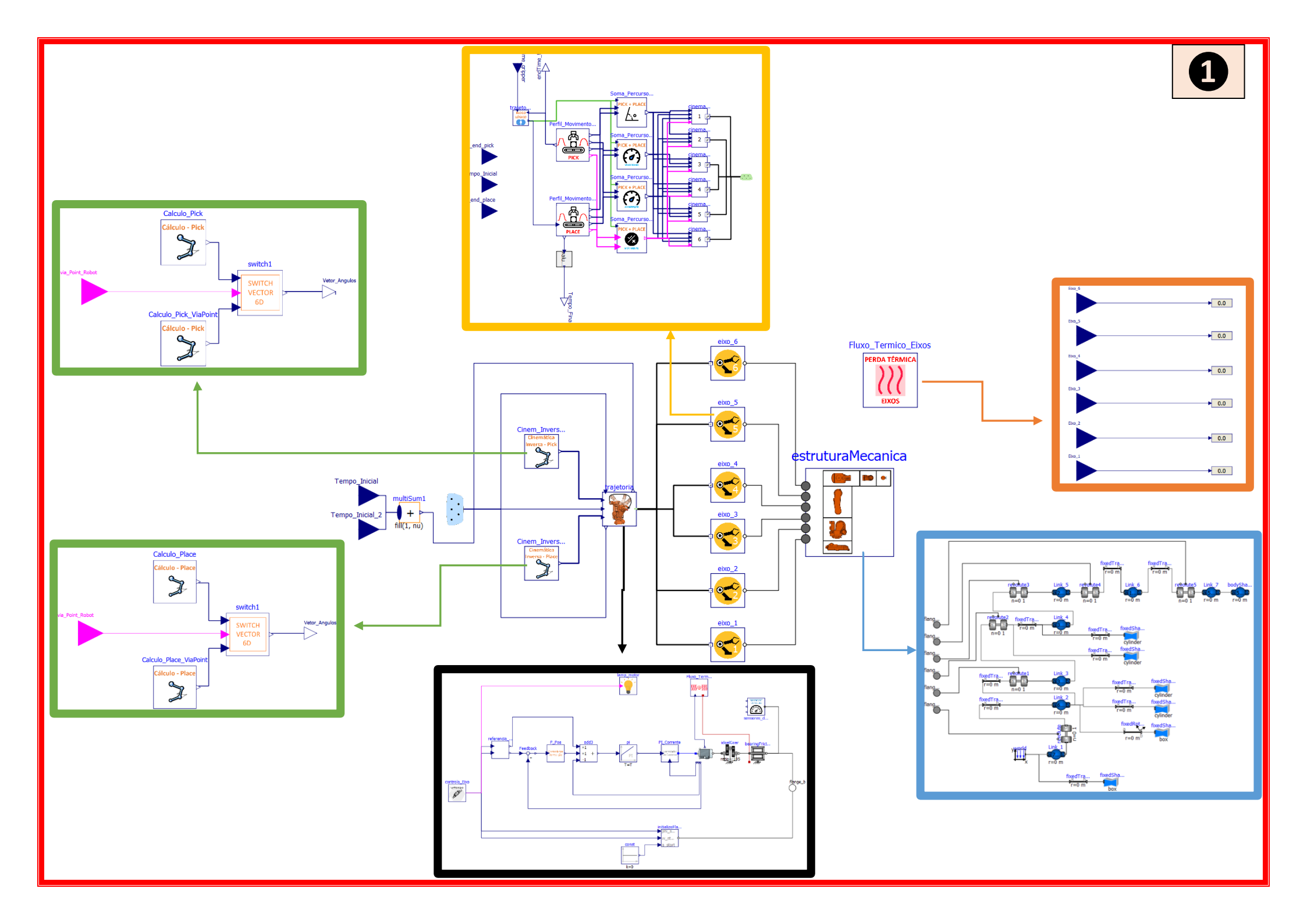
Figura 238 - Atribuição de valores aos indicadores gráficos

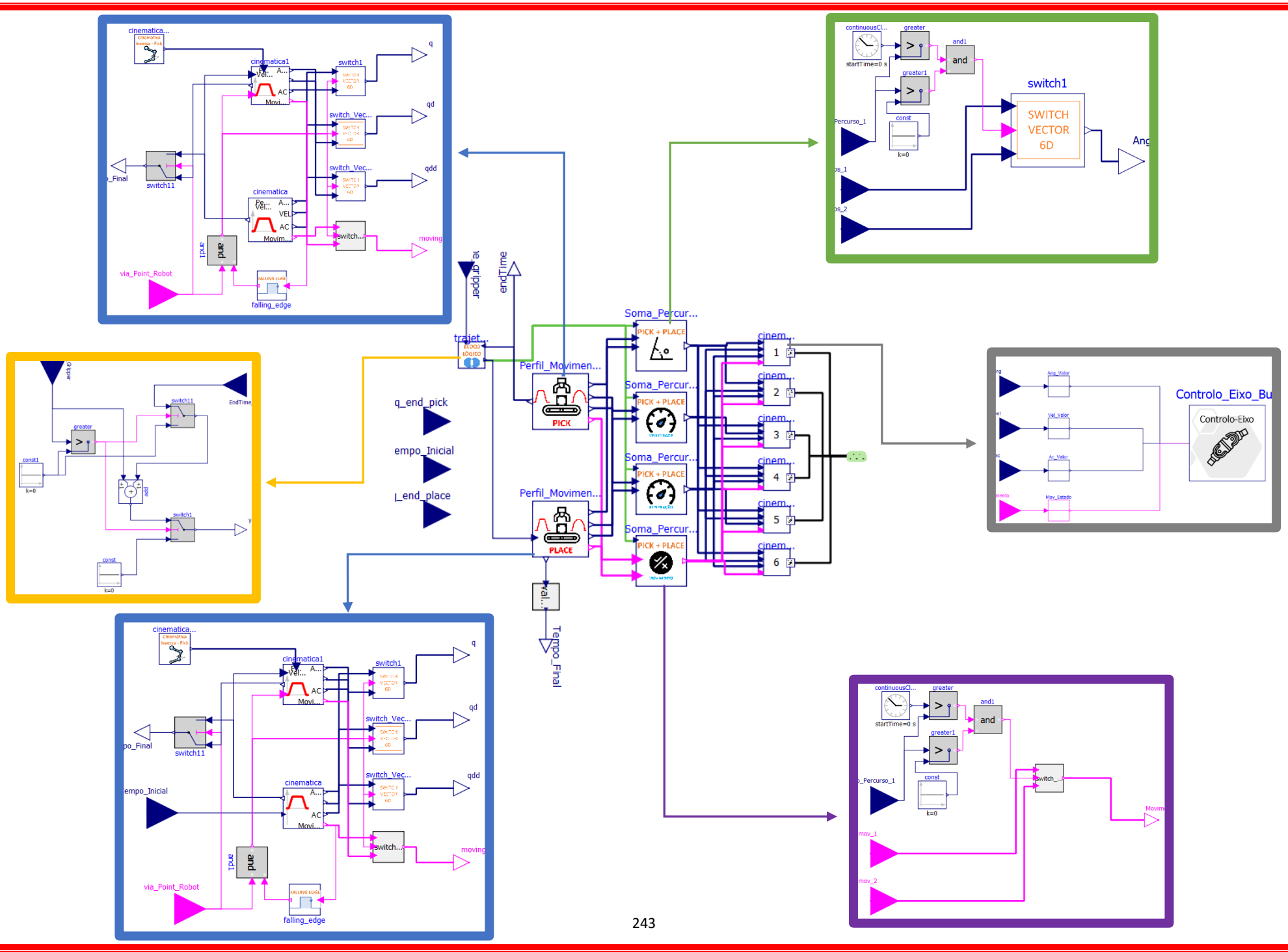
Nesse sentido, as variáveis relativas aos valores dos indicadores, como se pode observar no extrato de código em cima dentro dos retângulos azuis, são igualadas aos valores das variáveis calculadas na própria função, dentro dos retângulos vermelhos.

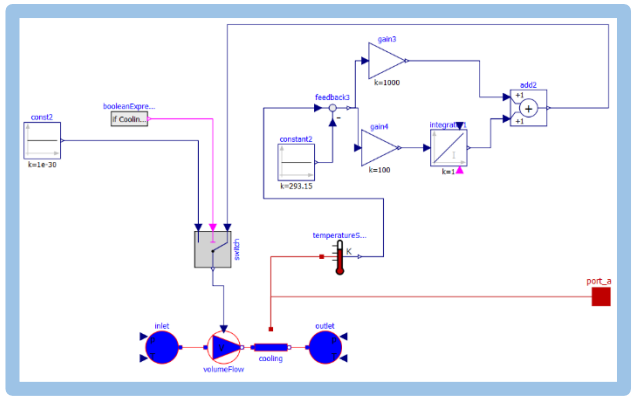
Já para o caso das simulações adjacentes ao tipo de funcionamento do robot, basta que o *Callback* seja definido através da função previamente elaborada e o modo de operação selecionado, e a respetiva simulação correrá paralelamente no *Coppelia*, caso o ficheiro do programa se encontre aberto, uma vez que as funções que permitem a comunicação com o programa simulador já se encontram incorporadas na função do modo de operação.

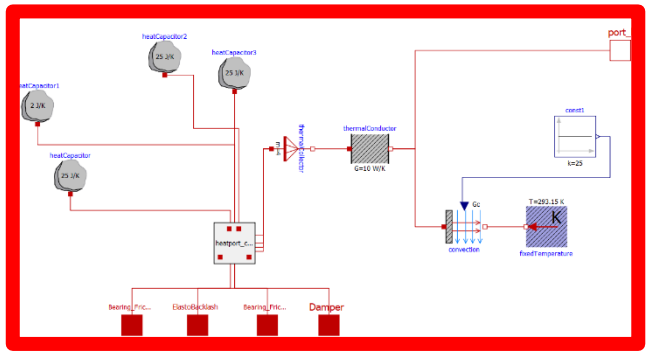
APÊNDICE 3 – ARQUITETURA DO SISTEMA

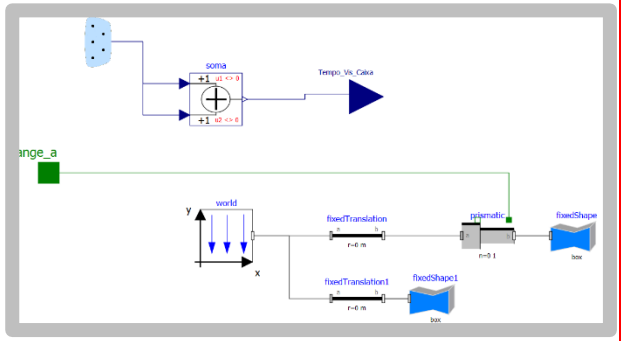


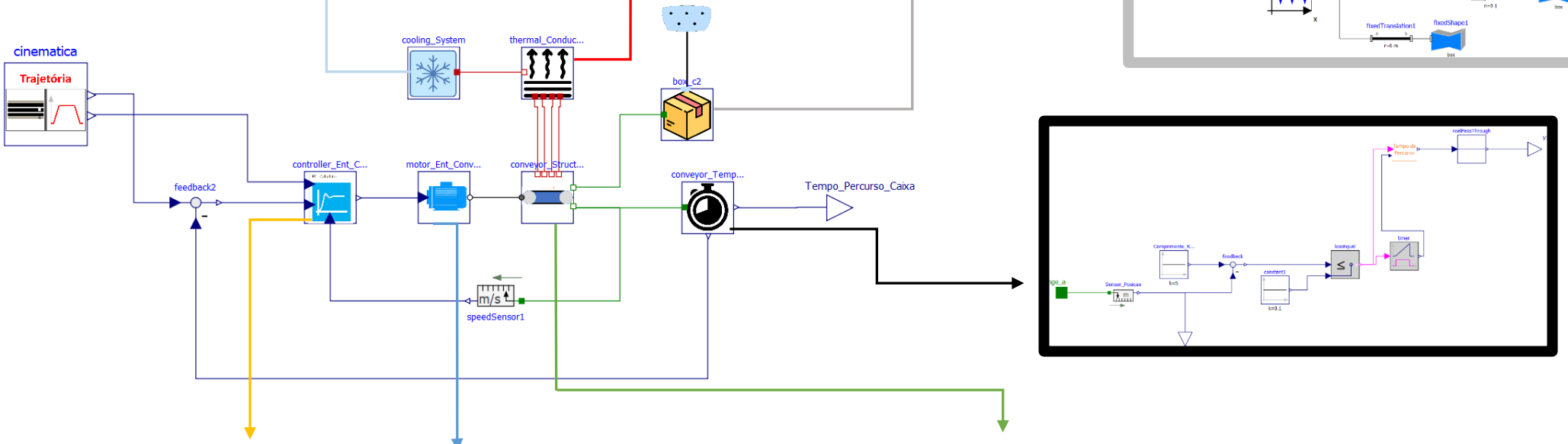

1
Calculo_Pick
Cálculo - Pick
switch1
SWITCH VECTOR 6D
Vetor_Angulos
via_Point_Robot
Calculo_Pick_ViaPoint
Cálculo - Pick
Soma_Percurso...
PICK + PLACE
Perfil_Movimento...
PICK
PLACE
cinema...
Tempo_Final
Fluxo_Termico_Eixos
PERDA TÉRMICA
EIXOS
eixo_6
eixo_5
eixo_4
eixo_3
eixo_2
eixo_1
estruturaMecanica
Cinem_Invers...
Cinemática Inversa - Pick
Cinemática Inversa - Place
Tempo_Inicial
Tempo_Inicial_2
multiSum1
fill(1, nu)
trajetoria
Calculo_Place
Cálculo - Place
Calculo_Place_ViaPoint
Link_1
Link_2
Link_3
Link_4
Link_5
Link_6
Link_7
cylinder
box


cinematica...
cinematica1
switch1
switch_Vec...
switch11
cinematica
switch...
and1
via_Point_Robot
falling_edge
moving
q
qd
qdd
switch1
SWITCH
VECTOR
6D
continuousCl...
greater
greater1
and
startTime=0 s
const
k=0
Percurso_1
Ang
gripper
endTime
traject...
Perfil_Movimen...
PICK
PLACE
Soma_Percur...
PICK + PLACE
cinem...
1
2
3
4
5
6
q_end_pick
empo_Inicial
l_end_place
Tempo_Final
Controlo_Eixo_Bu
Controlo-Eixo
Ang_Valor
Vel_Valor
Ac_Valor
Mov_Estado
EndTime
switch11
greater
const1
k=0
add
const
cinematica...
cinematica1
switch1
switch_Vec...
switch11
cinematica
switch...
and1
empo_Inicial
via_Point_Robot
falling_edge
moving
q
qd
qdd
continuousCl...
greater
greater1
and
startTime=0 s
const
k=0
Percurso_1
switch...
mov_1
mov_2

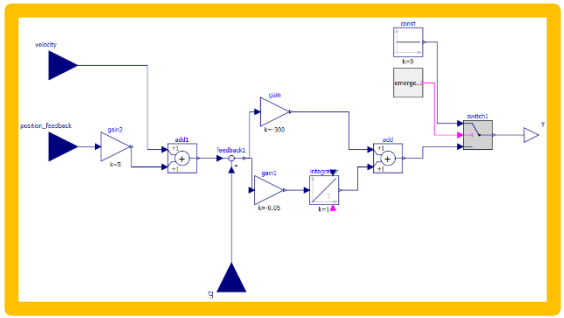

gain3
k=1000
booleanExpre...
if Coolin...
feedback3
add2
const2
k=1e-30
constant2
k=293.15
gain4
k=100
integrator
k=1
temperatureS...
switch
port_a
inlet
outlet
volumeFlow
cooling

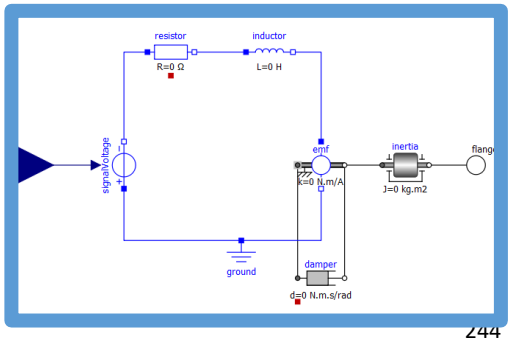

heatCapacitor2
25 J/K
heatCapacitor3
25 J/K
heatCapacitor1
2 J/K
heatCapacitor
25 J/K
thermalConductor
G=10 W/K
const1
k=25
port_
heatport_c...
convection
T=293.15 K
fixedTemperature
Bearing_Fric...
ElastoBacklash
Bearing_Fric...
Damper

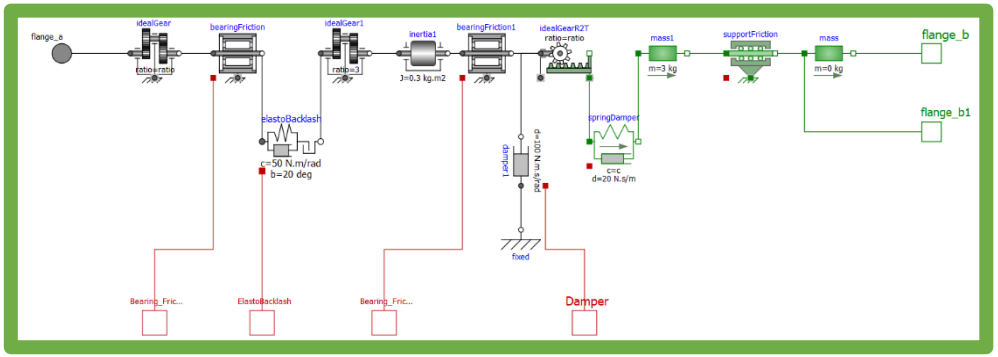

soma
Tempo_Vis_Caixa
ange_a
world
fixedTranslation
prismatic
fixedShape
fixedTranslation1
fixedShape1
box


cinematica
Trajetória
cooling_System
thermal_Conduc...
box_c2
feedback2
controller_Ent_C...
motor_Ent_Conv...
conveyor_Struct...
conveyor_Temp...
Tempo_Percurso_Caixa
speedSensor1
m/s

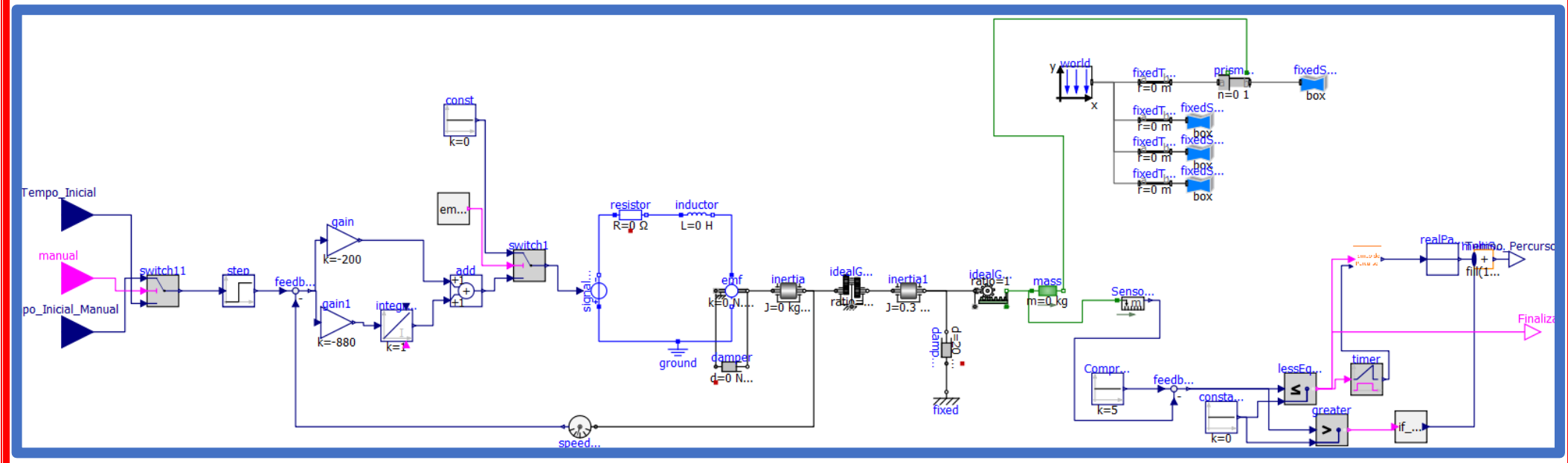
realPassThrough
Tempo de Percurso
Comprimento_R...
feedback
constant1
k=0.1
lessEqual
timer
Sensor_Posicao


velocity
const
k=3
emerge...
position_feedback
gain2
k=-5
add1
gain
k=-300
feedback1
add
gain1
k=-0.05
integrator
k=1
switch1

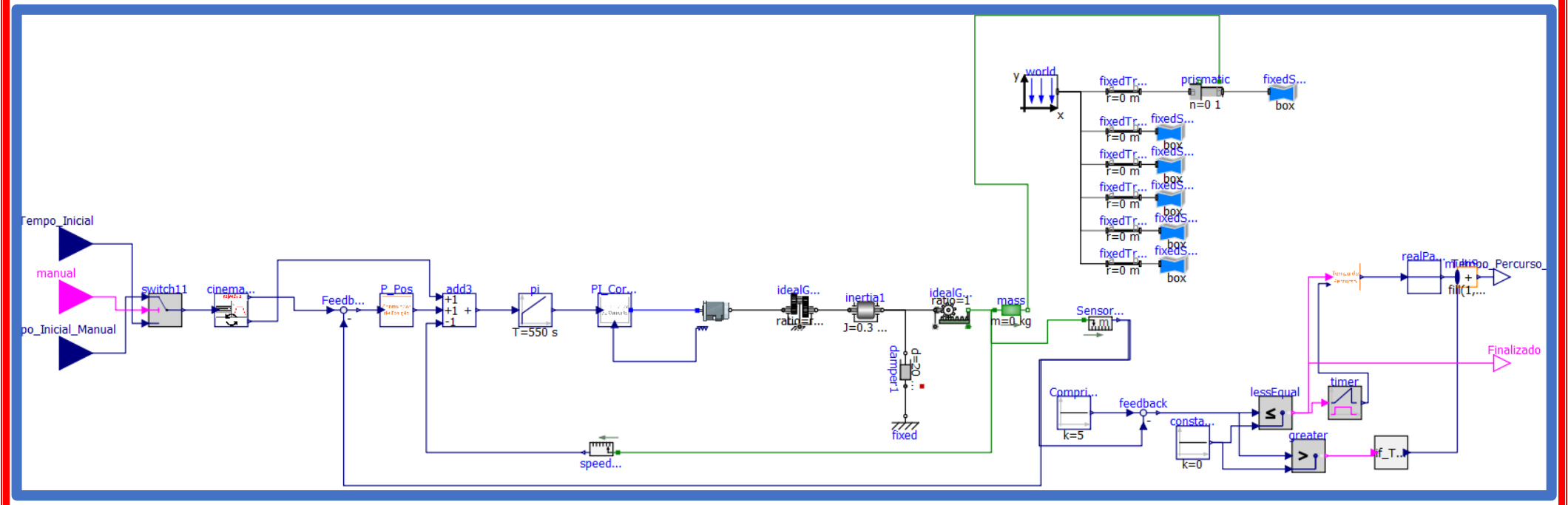

resistor
R=0 Ω
inductor
L=0 H
signalVoltage
emf
k=0 N.m/A
inertia
J=0 kg.m2
flange
ground
damper
d=0 N.m.s/rad

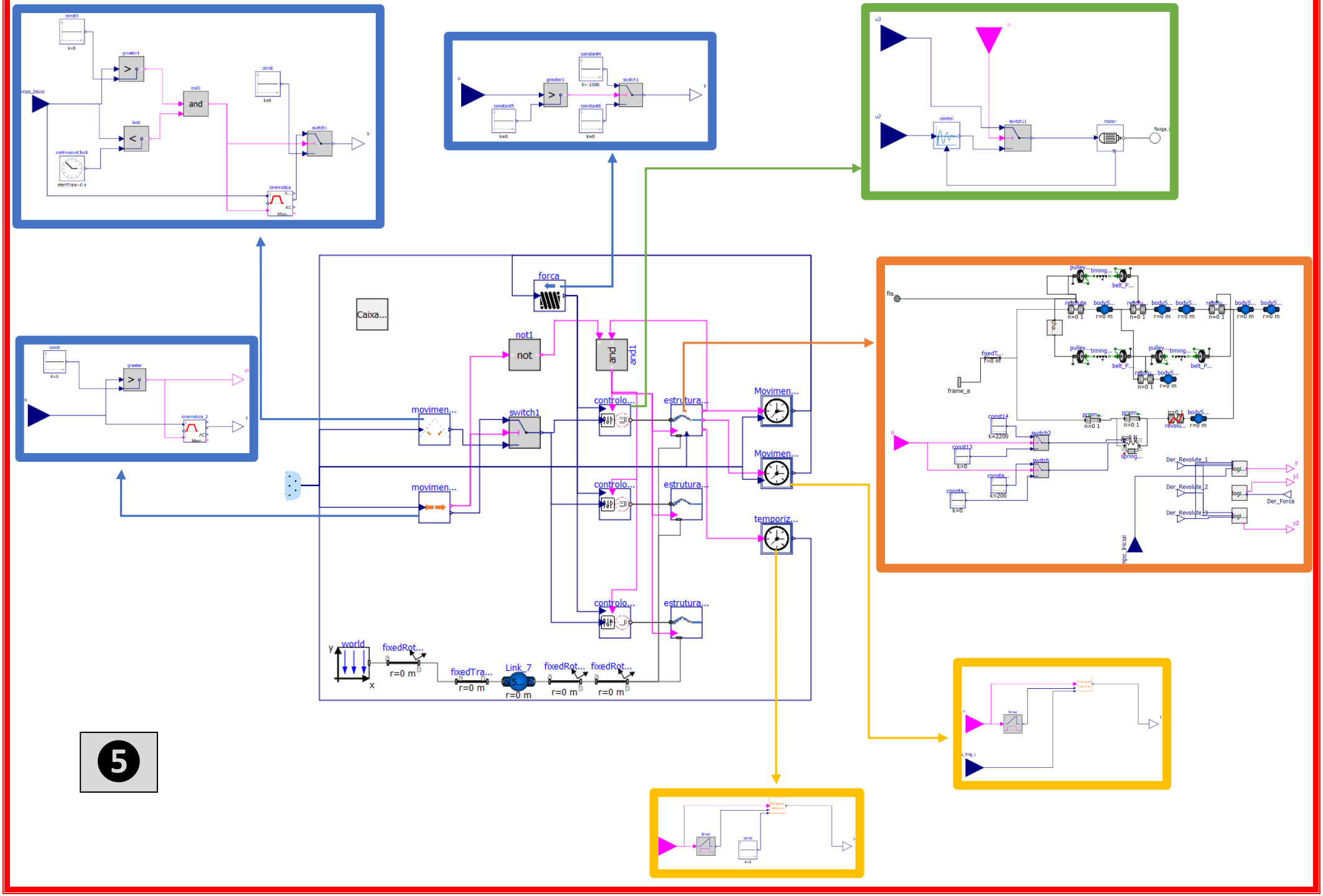

flange_a
idealGear
ratio=ratio
bearingFriction
idealGear1
ratio=3
inertia1
J=0.3 kg.m2
bearingFriction1
idealGearR2T
ratio=ratio
mass1
m=3 kg
supportFriction
mass
m=0 kg
flange_b
flange_b1
elastoBacklash
c=50 N.m/rad
b=20 deg
springDamper
c=c
d=20 N.s/m
damper
d=100 N.m.s/rad
fixed
Bearing_Fric...
ElastoBacklash
Bearing_Fric...
Damper

3


Tempo_Inicial
manual
po_Inicial_Manual
switch11
step
feedb...
gain
k=-200
gain1
k=-880
integr...
k=1
add
const
k=0
em...
switch1
resistor
R=0 Ω
inductor
L=0 H
signal...
ground
emf
k=0 N...
damper
d=0 N...
inertia
J=0 kg...
idealG...
ratio=...
inertia1
J=0.3 ...
idealG...
ratio=1
damp...
d=20...
fixed
mass
m=0 kg
world
x
fixedT...
f=0 m
prism...
n=0 1
fixedS...
box
Senso...
Compr...
k=5
feedb...
consta...
k=0
lessEq...
timer
greater
if_...
realPa...
Finaliza
speed...

4


Tempo_Inicial
manual
po_Inicial_Manual
switch11
cinema...
Feedb...
P_Pos
add3
+1
+1
-1
pi
T=550 s
PI_Cor...
idealG...
ratio=r...
inertia1
J=0.3 ...
idealG...
ratio=1
mass
m=0.1 kg
damper1
d=20
fixed
speed...
world
y
x
fixedTr...
r=0 m
prismatic
n=0 1
fixedS...
box
Sensor...
Compri...
k=5
feedback
consta...
k=0
lessEqual
timer
greater
realPa...
Tempo_Percurso_
fill(1,...
Finalizado


Caixa...
forca
not1
not
and1
and
movimen...
switch1
controlo...
estrutura...
Movimen...
temporiz...
world
fixedRot...
fixedTra...
Link_7
r=0 m
5