Instituto Superior de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA

Desenvolvimento de Sistema Robótico com Visão para colocação de peças de metal num suporte

Relatório de Estágio para a obtenção do grau de Mestre em Engenharia Eletrotécnica

Especialização em Automação e Comunicações em Sistemas Industriais

Autor

João Pedro Santos Antunes

Orientador

Doutor João Paulo Morais Ferreira

Supervisor na empresa Real Robotic Systems

Eng. Luís Miguel Nolasco

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Coimbra, novembro 2024

AGRADECIMENTOS

"Todos os passos na vida são objetivos que temos de conquistar, independentemente das dificuldades que há para os subir. O importante é ir subindo."

Quero começar por agradecer ao Engenheiro Luís Nolasco e a toda a equipa da equipa da Real Robotic Systems, por terem dado a oportunidade de aplicar o que aprendi ao longo dos meus estudos, principalmente no curso de Mestrado em Engenharia Eletrotécnica, a partilha de conhecimentos quando as dúvidas surgiram, acrescentando uma mais-valia, não só a nível profissional, mas também a nível pessoal.

Aos meu colegas de curso Daniel Silva, João Ferreira e João Lemos por me terem aturado durante estes dois anos de curso, pela partilha de conhecimentos e cooperação na realização das tarefas, ficando com algumas histórias para contar. E essencialmente por nunca me terem deixado desistir, mesmo quando parecia difícil, e terem-me ajudado a estudar nas disciplinas que tive mais dificuldade.

Pretendo também, agradecer aos docentes do curso de Mestrado em Engenharia Eletrotécnica (MEE), pelo conhecimento e experiência profissional transmitido, sem isso não teria sido possível dar continuidade ao curso, e consequentemente ao estágio.

Por fim, e não menos importante, gostaria de agradecer a minha família mais concretamente aos meus pais, que sem eles nada disto teria sido possível, sempre a incentivarem. Que nos momentos menos bons, dizem para a próxima vai correr melhor, e com esforço não há impossíveis e tudo é possível.

RESUMO

O presente documento apresenta o desenvolvimento, a concretização e a finalização de um projeto de robótica que consistia na utilização de um robô na colocação de uma peça previamente escolhida, no objeto em questão.

O projeto mencionado foi realizado com a empresa Real Robotic Systems, inicialmente apresentado como um projeto modelo, proposto com o objetivo de ser incluído na linha de produção de uma empresa de forma a dinamizar o setor de produção e clarificar ideais concebidos.

Durante o processo de desenvolvimento de todo o projeto foi possível focar os conhecimentos e as aprendizagens que ao longo dos anos foram adquiridos, sendo que em particular, a área da robótica e da automação foram as mais exploradas.

Assim sendo, o projeto realizado vai ser composto por um objeto, que no caso, é um cabide que será colocado numa linha de produção, onde o robô terá de colocar uma peça predefinida no local disponível individualmente, com o auxílio de um equipamento de visão.

Ao realizar este projeto existiu uma necessidade de utilizar um sistema de visão 3D (três dimensões) para saber qual a posição e quais os pontos livres identificados no cabide, assim como o envio de coordenadas dos pontos livres; um robô para colocar a peça no lugar conforme as coordenadas enviadas pelo sistema de visão e para fazer o movimento inicial para verificação de espaços livres; finalizando com o autómato que estará responsável pela comunicação dos dois equipamentos anteriormente descritos e que também tem a função de tratar a informação recebida para o equipamento de destino processe com sucesso. E desta forma, as diversas peças inseridas no cabide durante a realização do processo, estarão nos respetivos locais, não existindo nenhum local disponível.

Para explicar o seu funcionamento irá mostrar-se, sempre que possível, a escolha dos equipamentos e a sua razão, as partes da programação/código utilizado para a execução, bem como imagens das mesmas.

O projeto foi concluído com a demonstração e apresentação do funcionamento em contexto real, para futuramente ser implementado em diversas empresas, de forma a existir uma evolução no setor.

Palavra-Chave: Robótica, *Visão*, Automação

ABSTRACT

This document presents the development, implementation and completion of a robotics project that consisted of using a robot to place a previously selected part on the object in question.

The project mentioned was carried out with the company Real Robotic Systems, initially presented as a model project, proposed with the aim of being included in a company's production line to streamline the production sector and clarify conceived ideals.

During the development process of the entire project, it was possible to focus on the knowledge and learning that had been acquired over the years, with the areas of robotics and automation being the most explored.

Therefore, the project carried out will consist of an object, which in this case is a hanger that will be placed on a production line, where the robot will have to place a predefined part in the available location individually, with the help of vision equipment.

When carrying out this project, it was necessary to use a 3D (three-dimensional) vision system to determine the position and free points identified on the hanger, as well as to send coordinates of the free points; a robot to place the part in place according to the coordinates sent by the vision system and to make the initial movement to check free spaces; finishing with the automaton that will be responsible for the communication between the two previously described pieces of equipment and that also has the function of processing the information received so that the target equipment can process it successfully. In this way, the various parts inserted into the hanger during the process will be in their respective places, with no other space available.

To explain its operation, whenever possible, the choice of equipment and its reason will be shown, the parts of the programming/code used for its execution, as well as images of them.

The project was completed with the demonstration and presentation of its operation in a real context, to be implemented in several companies in the future, to promote evolution in the sector.

Keyword: Robotics, Vision, Automation

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS

2D	Duas Dimensões
3D	Três Dimensões
ACSI	Automação e Comunicações em Sistemas Industriais
AGV	*Automated Guided Vehivcles*
COBOT	*Collaborative Robot*
CA	Corrente Alternada
CC	Corrente Continua
CPU	*Central processing unit*
CV	*Cleareance View*
Dint	*Double Int*
FBD	*Function Block Diagram*
FoV	*Field of View*
Grafcet	*Graphe Fonctionnel de Commande, Étapes Transitions*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
IL	*Instruction List*
IP	*Internet Protocol*
ISEC	Instituto Superior de Engenharia de Coimbra
LD	*Ladder*
MEE	Mestrado em Engenharia Eletrotécnica
MR	*Measurement Range*
RAM	*Random Access Memory*
ROM	*Read-only Memory*
RRS	Real Robotic Systems
SCADA	*Supervisory Control and Data Acquisition*
SFC	*Structured TextSequential Flow Chart*
SMED	*Single Minute Exchange of Dies*
TCP/IP	*Transmission Control Protocol/Internet Protocol*

1 INTRODUÇÃO

No presente capítulo pretende fazer-se um enquadramento do estágio tendo como base as motivações que levam as empresas a modernizarem-se. Além dos objetivos inicias do estágio e do ambiente envolvente da empresa, irá ser apresentada uma breve descrição dos capítulos que compõem o atual relatório.

1.1 Enquadramento

O presente documento é um relatório de estágio da componente de Projeto/Dissertação/Estágio, inserido no plano de estudos do Mestrado em Engenharia Eletrotécnica (MEE) na área de especialização de Automação e Comunicações em Sistemas Industriais (ACSI), ministrado pelo Instituto Superior de Engenharia de Coimbra (ISEC).

A empresa de estágio, Real Robotic Systems, fundada em 2018, em Coimbra, é uma empresa especialista em trabalhos de alta qualidade nas áreas de automação, robótica e de sistemas de visão (quer sejam eles em duas dimensões ou três dimensões), para instalações fabris.

Atualmente, com a necessidade dos produtores em obterem melhorias nos seus processos de produção, exige que empresas como a Real Robotic Systems implementem mecanismos de melhoria de processos de produção, de forma a existir um resultado mais eficaz e rentável tanto para a empresa produtora como para o consumidor.

A melhoria de processos de produção, pode ser obtida através da alteração do tipo de metodologia utilizada na produção (*Lean*, SMED (*Single Minute Exchange of Dies)*, *Kaizen*, entre outras) – metodologias que levam a um estudo da produção e com esse estudo melhorar ou simplificar processos de produção; a contratação e/ou atribuição de conhecimentos de trabalhadores, com um custo avultado, no entanto, o número de colaboradores da empresa, pode ser reduzido. e a automatização da mesma, que inicialmente tem um custo elevado (tempo de produção, materiais, menos colaboradores, entre outros), mas fará uma redução de custos de produção a curto/médio prazo.

O intuito do presente relatório de estágio, é abordar a automatização de fábrica, com vista a um aumento de produção e a custo reduzido.

Na realização de um projeto de automatização de fábrica tem de se ter em atenção diversos fatores, entre os quais:

- O tempo de ciclo, isto é, o tempo de demora desde a elaboração do produto, à entrada da máquina e o términus;
- O espaço que a máquina ocupa dentro das instalações da fábrica;
- O ambiente e as condições onde a máquina irá exercer as suas funções (se a sua funcionalidade será eficaz, a que velocidades poderá exercer a tarefa prossuposta, ou se irá realizar mais do que um produto em simultâneo);

- A autonomia do equipamento e se existe a necessidade de executar tarefas em colaboração com os operadores;
- A automatização pretendida, se automação, se robótica ou ambos;
- A análise dos custos do investimento - orçamento estipulado.

1.2 Objetivo do Estágio

No decorrer do estágio, foi dado a oportunidade, de entrar em contacto com diversos projetos de automação e/ou robótica, com diferentes tipos de equipamentos e métodos de programação, bem como aplicar os conhecimentos adquiridos, práticos e teóricos, durante o curso MEE.

Os objetivos principais do estágio foram:

- Operar em diferentes robôs;
- Compreender os tipos de movimentos e quais as limitações de um robô;
- Aprender a programar em vários tipos de autómatos;
- Fazer interface entre o homem e a máquina;
- Programar autómatos e sistemas de visão 3D (três dimensões).

Sendo que estes objetivos foram cumpridos em vários projetos:

- Linha de montagem de empacotamento de garrafas de plástico – neste projeto foi possível trabalhar com uma máquina em ambiente de produção, sendo que o principal objetivo seria colocar tampas nas respetivas garrafas e proceder ao embalamento em sacos plásticos. Após o processo concluído, prosseguiu-se ao armazenamento dessas mesmas garrafas. Com um número previamente predefinido, o robô teria de ir buscá-las à linha e colocá-las numa caixa. O robô utilizado para este efeito foi um robô colaborativo uma vez que, teria de estar a trabalhar em simultâneo, enquanto os operadores substituíam as caixas. Neste projeto foram aplicados conhecimentos de automação uma vez que a linha de montagem foi programada de início até ao fim, da robótica para o controlo do robô e o interface homem-máquina para controlar os estados e ações da linha/máquina. Neste relatório, no anexo I, encontra-se o interface homem-máquina realizado para este projeto utilizando o sistema *SCADA (Supervisory Control and Data Acquisition*), ou seja, rede de comunicação é responsável por transmitir dados entre os equipamentos e o software de supervisão, aonde é possível observar os vários menus e ações que o operador pode realizar nos diferentes menus.
- Linha de montagem de cortes de perfis – neste caso, a linha era responsável por fazer a esquadria do perfil e dividi-los nas dimensões pretendidas. Após o corte, avançava-se para o processo de embalamento e à colocação num carrinho de transporte. No projeto descrito, foi possível fazer correções a nível de automação com a linha em produção e a correção de algumas falhas no interface homem-máquina de forma a tornar-se mais percetível para o operador.
- Robô paletizador – neste caso, o robô teria de colocar caixas com diferentes dimensões numa palete, sabendo que esta poderia dispor de vários andares e dimensões. A informação da escolha do tamanho das caixas ou do tamanho das paletes teria de ser fornecida do autómato para o robô, e este com base na

programação realizada, teria de colocar as caixas no lugar adequado e predefinido anteriormente. O projeto possibilitou a realização do estudo de movimentos do robô conforme as posições das caixas predefinidas pelo cliente.

- Robô com sistema de visão 3D – podendo esta temática dividir-se em dois projetos distintos:
    - No primeiro projeto, o sistema de visão teria de identificar as peças em que se encontravam dentro de um contentor e enviar a informação para o robô. O robô tinha a principal a função de agarrar na peça e colocá-la numa linha de produção. A pedido do cliente foram implementadas medidas de segurança adicionais que consistiam, em caso de um operador se aproximasse de um robô operacional, este teria de diminuir a sua velocidade ou até mesmo parar. Assim, foi possível aprender configurações de velocidades conforme as indicações que vinham do sensor de segurança e a configurar sistemas de visão 3D.
    - O segundo projeto, que será explorado posteriormente neste relatório, consiste em colocar peças no cabide com a condicionante de não saber a disponibilidade do mesmo. Para este projeto utilizou-se um robô e um sistema de visão 3D. Assim sendo o movimento inicial a realizar será uma captura de imagem por parte do sistema de visão 3D com o auxílio do robô, uma vez que o sistema de visão estava posicionado na ponta da garra do robô. Após essa captura de imagem, o sistema de visão teria de observar os espaços livres disponíveis e enviar as coordenadas para o robô. Com os valores de coordenadas recebidas pelo robô, este terá de se deslocar até as coordenadas e largar a peça. Neste projeto, delinearam-se os seguintes objetivos específicos:
        - Programação do robô;
        - Estabelecer a comunicação entre robô e o sistema de visão 3D, usando um autómato para controlar;
        - Conceber um algoritmo de análise de imagem, capaz de interpretar uma imagem e devolver pontos;
        - Fazer um estudo dos movimentos do robô.

1.3 Estrutura do Documento

O presente relatório encontra-se dividido em cinco capítulos, sendo que os capítulos encontrar-se-ão divididos em seções, para dividir o tema em subcapítulos.

No segundo capítulo, será feita uma revisão literária dos principais temas abordados no relatório, tais como: a robótica, a automação e os sistemas de visão 3D (três dimensões). Sendo que os temas se encontram repartidos em conceitos básicos e história. Para concluir o capítulo, ainda serão abordados projetos semelhantes, assim como considerações finais.

No terceiro capítulo abordar-se-á o projeto e a sua envolvente, bem como o local e as tarefas a realizar, e todo o equipamento necessário e o seu porquê.

No quarto capítulo desenvolver-se-á o projeto nas áreas de robótica, automação e visão 3D, onde se irá explicar todo o trabalho desenvolvido e com isso explicar toda a programação relativa as áreas de desenvolvimento.

No quinto capítulo irá fazer-se uma conclusão do projeto, que contará com uma reflexão sobre o que poderia ser melhorado no projeto.

2 REVISÃO DA LITERATURA

O presente capítulo apresenta uma breve introdução à robótica, automação e sistemas de visão 3D, quer ao nível histórico, a sua evolução, e alguns conceitos e/ou expressões, de modo a facilitar a compreensão, nos capítulos seguintes.

Além do que foi mencionado anteriormente, irão ser abordados projetos semelhantes, seguido de umas considerações finais sobre a revisão da literatura.

2.1 Robótica

Denomina-se por robótica um conjunto de técnicas que dizem respeito ao funcionamento e utilização de autómatos (robôs) na execução de múltiplas tarefas em substituição do homem [1], com vista a melhorar as condições de trabalho ou aliviar o trabalho dos operadores.

2.1.1 História da Robótica

A robótica surge no século XX, não se sabendo a data em concreto dos primeiros robôs a existirem, sabe-se apenas, que foram baseados em princípios de Aristóteles sobre ferramentas automáticas, Henry Ford, Leonardo Da Vinci e Isaac Asimov [2].

O primeiro robô industrial conhecido foi desenvolvido em 1959 por George Devol e Joseph Engelberger. O robô pesava duas toneladas, utilizava atuadores hidráulicos, em que a sua programação era feita em coordenadas de junta, ou seja, os ângulos das várias juntas eram armazenados durante uma fase de ensino e reproduzidos na operação. [3].

Em 1961, foi utilizado o primeiro robô numa linha de produção pela GM Ternstedt, com o objetivo de fabricar puxadores de portas e janelas, produtos de iluminação e outras ferragens para os interiores dos automóveis. O robô respondia a comandos, passo a passo, armazenados num tambor magnético.

Seis anos depois, apareceu o primeiro robô na Europa, com o nome de "*Unimate*". que tinha como objetivo transportar peças fundidas, de uma linha de montagem e soldar essas peças em carrocerias de automóveis. O "*Unimate*" foi licenciado para que fosse produzido pela Nokia da Finlândia.

Com a evolução da tecnologia, também os robôs acompanharam essa evolução e em 1971, apareceram os primeiros robôs atuados por hidráulica, ou seja, compostos por fluidos de água ou óleo [4]. Após um ano, revelou-se o primeiro robô móvel com visão, que apresentava um sistema de navegação inteligente com a ajuda de marcadores.

Em 1973, apareceu o robô em que deixou de ser guiado por hidráulica e passou a ser composto por juntas eletromecânicas, desenvolvido pela KUKA [5].

Desse modo, com o avanço da tecnologia por consequência também os robôs foram evoluindo, em diversos aspetos tais como:

- deixaram de ser controlados por um tambor magnético para serem controlados por microcomputadores;
- uma maior precisão;
- um aumento de capacidade de carga e de menores dimensões;
- uma melhor adaptação com outros sistemas, outros robôs e principalmente com os operadores [6].

Atualmente, existe uma grande variedade de robôs e tecnologia, em que a escolha dos mesmos depende do local, da função necessária que se pretende realizar e do orçamento que a empresa tem disponível para investir.

2.1.2 Constituição de um robô

O robô para funcionar na sua plenitude tem de ser composto por vários componentes, sendo que a ausência de algum acaba por condicionar o funcionamento do robô.

Os componentes que constituem o robô, de forma que este se encontre operacional são:

- Robô (figura 2.1) – elemento que irá realizar o movimento, podendo variar o número de eixos e elos, dependendo do tipo e função pretendida. Através da cinemática podem definir-se a posição cartesiana (x, y, z) do robô, sendo que podem ter juntas de rotação - (R), ou prismática/de translação - (P)

Figura 2.1 Robôs industriais e colaborativos [7].

- Controlador (figura 2.2) – elemento onde estão presentes todas as configurações a realizar pelo robô, quer as indicações da garra quer as configurações de sistema. Para além disso, é responsável por armazenar os dados (pontos e/ou chamadas de função) do programa do robô, sendo por isso, capaz de ordenar o movimento ou paragem do mesmo. Pode ainda, comunicar com um autómato (PLC), com uma consola, e em certos casos, é capaz de comunicar com uma botoneira. Alguns controladores, podem até controlar mais do que um robô.

Figura 2.2 Controlador Motoman NX100 [8].

- Consola (figura 2.3) – A consola permite ao operador ou ao programador, realizar movimentos do robô ou executar programas predefinidos. Para que isso seja possível, a mesma terá de ter a capacidade de comunicar com o controlador do robô, que deverá estar ligado ao robô.

Figura 2.3 Consolas de vários fabricantes [9].

Para além disso, o robô tem de ter elementos de proteção e segurança, para prevenir e proteger os operadores ou pessoas que sejam curiosas. Os elementos de segurança podem ser físicos ou físicos programados, como se descreve abaixo:

- Gradeamentos (figura 2.4) – elementos físicos, quer sejam eles metálicos ou em acrílico, impossibilitando o acesso, de uma pessoa, à área de trabalho do robô, normalmente tem um formato de portas, mas com sensores de abertura. Estes sensores têm de estar de alguma maneira interligados ao robô e quando a porta

estiver aberta, o mesmo tem de funcionar a velocidades reduzidas ou nos piores dos casos não se movimentar.

Figura 2.4 Gradeamento [10].

- Scanner a laser de segurança (figura 2.5) – é composto por um dispositivo sensível eletricamente, que usa feixes de infravermelhos, para detetar se houve ou não uma violação da área de trabalho. Caso haja uma invasão na área de trabalho, o robô terá de abrandar a velocidade de trabalho ou até mesmo parar, dependendo do tipo de robô ou do tipo de trabalho e/ou programação realizada. Pode ainda ser programada a velocidade do robô, tendo em conta a distância da área de trabalho. [11]

a) b)

Figura 2.5 Exemplo de um a) scanner laser de segurança [12]; b) aplicação [13].

- Botoneira de emergência (figura 2.6) – programada ou ligada diretamente aos contactos de emergência, presentes no controlador do robô, ou em determinados casos ao PLC. A botoneira tem de ser capaz, em caso de emergência, de parar o funcionamento robô. A sua localização é estrategicamente colocada, para que toda

os operadores sejam capazes de acionar, em situações de emergências. Para além disso, tem de estar distribuídas pela área de funcionamento do robô.

Figura 2.6 Botoneira de emergência [14].

2.1.3 Conceitos da Robótica

Antes de prosseguir com o relatório de estágio de mestrado, é importante ressaltar alguns conceitos/constituição básicos da robótica (figura 2.7), pois sem eles, pode existir uma maior dificuldade de interpretação do conteúdo do mesmo. Assim sendo, de seguida descrevem-se aqueles de maior relevância:

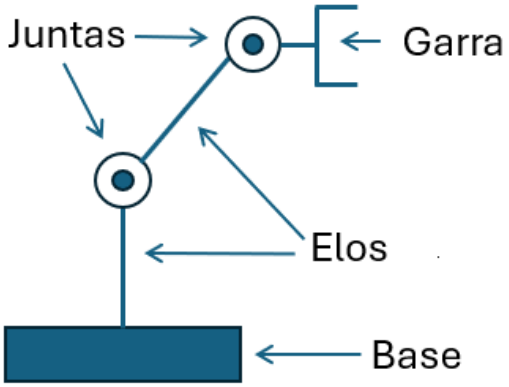


Figura 2.7 Constituição de um robô [15].

- Base – ponto onde o robô está colocado;
- Elos ou links – são pontos de ligação, entre as diferentes partes do corpo do robô, que podem ser mecânicas, elétricas ou eletrónicas;
- Graus de liberdade – conjunto de deslocamentos e/ou rotações que um robô consegue executar, normalmente é um número compreendido entre 1 e 6;
- Junta ou eixo (figura 2.8) - componentes móveis do robô, que resultam em movimentos relativos entre os eixos do robô. As juntas podem ser de vários tipos:

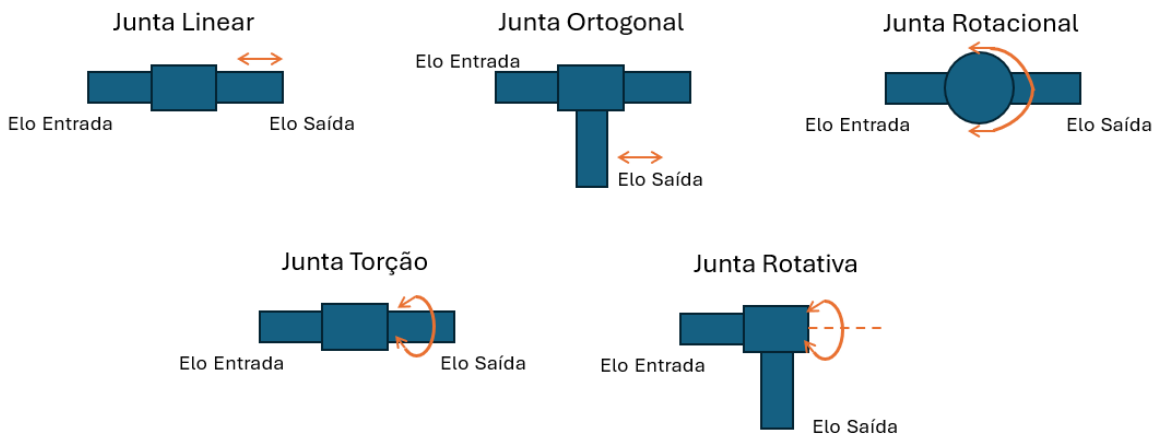


Figura 2.8 Tipos de juntas [16].

- Velocidade de movimentação - é a medida de rapidez, com que ele se desloca de um ponto a outro;
- Aceleração do movimento - é a taxa de variação da velocidade do robô, em relação ao tempo. Ou seja, é a capacidade de o robô aumentar a sua velocidade ou de alterar de direção rapidamente. A aceleração é importante para determinar a agilidade e eficiência de um robô, nas diferentes tarefas e ambientes. Quanto maior a aceleração de um robô, mais rápido ele poderá se deslocar e se adaptar às mudanças no ambiente ao seu redor.

Com a descrição da composição de um robô, já feita neste relatório de estágio, é importante também informar quais os tipos de movimentos que um robô consegue realizar, como se distinguem e quais as implicações que trazem.

Para isso vai-se dividir os movimentos em:

- Movimentos lineares;
- Movimentos ponto a ponto;
- Movimento *ARC* ou *CIRC*.

O movimento linear (figura 2.9) faz com que o robô se movimente linearmente. Isto é, o robô cria uma linha reta entre o ponto de partida e o ponto de chegada, e gira em todas as suas articulações, para manter o seu caminho correto. Ainda assim, pode existir algumas desvantagens, como vantagens dependendo da forma que se pretende aplicar o movimento.

Uma das principais desvantagens, de um movimento "Linear", é que ele tende a ser mais lento, do que um movimento ponto a ponto. Isto ocorre porque algumas articulações, precisam de se movimentar mais do que as outras, para manter o robô num caminho reto, dependendo da orientação específica e das suas posições articulares.

O movimento será sempre limitado, pela velocidade máxima da junta móvel mais lenta, pois cada um dos motores da junta terá uma velocidade máxima.

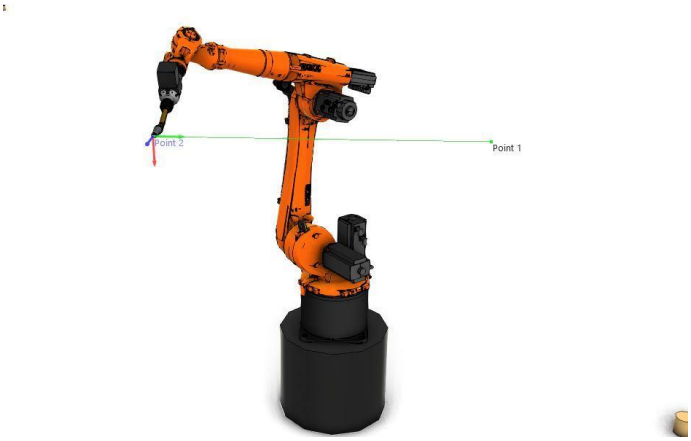


Figura 2.9 Robô a fazer movimento linear [17].

O movimento ponto a ponto ou em inglês *"joint"* (figura 2.10), é um movimento que ao contrário do movimento linear não faz uma linha reta entre o ponto A e o ponto B, mas sim um movimento que lhe trará maior rapidez de movimento.

Os movimentos ponto a ponto são vantajosos, pois permite que as articulações do robô se movam a uma maior velocidade programável, permitindo ao utilizador, otimizar a operação do robô e reduzir o tempo de ciclo. No entanto, esses movimentos são muito mais imprevisíveis do que os movimentos lineares.

Outra vantagem, é que os movimentos podem funcionar em aplicações onde o espaço é restrito, todavia os movimentos devem ser limitados, utilizando uma velocidade fixa numa área específica do programa e devem utilizar pontos auxiliares.

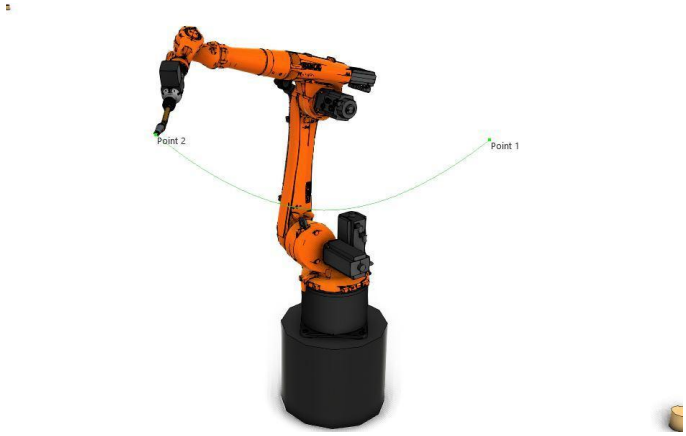


Figura 2.10 Robô a fazer movimento ponto a ponto [17].

O movimento *ARC* ou *CIRC*, também conhecido como movimento circular (figura 2.11), como o nome indica, o robô realiza um movimento circular. O movimento é usado para muitas aplicações padrão, incluindo fresar, colar, soldar, entre outras.

Este movimento é executado permitindo ao robô um mínimo de 2 posições: um ponto final e um ponto auxiliar. O robô mover-se-á em arco circular para a posição final através da posição auxiliar.

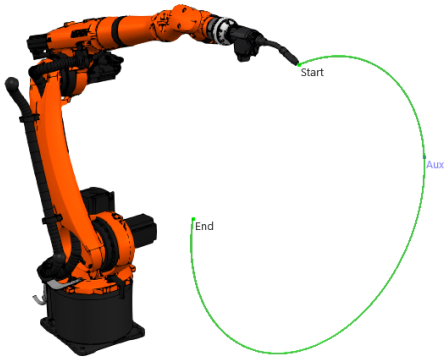


Figura 2.11 Robô a fazer movimento *CIRC* ou *ARC* [17].

Ainda pode haver a necessidade, em certos casos, de ter pontos de ajuda à execução do movimento, estes pontos têm o nome de pontos auxiliares ao movimento. Os pontos auxiliares (figura 2.12) ao movimento tem como objetivo ajudar a trajetória do movimento do robô.

A ajuda da trajetória tem de ser aplicada caso durante o deslocamento até ao ponto B, exista um obstáculo ou haja uma posição de entrega ou recolha com um ângulo de inclinação muito grande ou muito apertado.

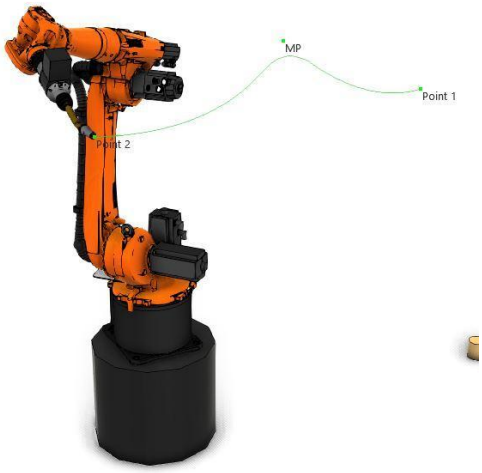


Figura 2.12 Robô a fazer um ponto de aproximação [17].

2.1.4 Áreas da Robótica

Atualmente, a robótica pode classificar-se de várias formas, como por exemplo quanto ao tipo de robôs; qual a área de intervenção onde vai ser aplicada ou qual a função que terá de efetuar.

Neste relatório vai classificar-se de acordo com os tipos de robótica existentes:

- Robótica móvel – área da robótica em que os robôs têm a liberdade total, não são fixos num determinado ponto. Aqui podemos dividir em:
    - Humanoides- robôs que têm um esqueleto e um funcionamento motor muito idêntico ao dos humanos;
    - AGV (automated guided vehivcles) – robô que tem um mapa em memória, conforme foi programado, atende às necessidades impostas, definindo o melhor caminho e desviando-se de obstáculos;
    - Robôs de exploração – são construídos para explorações espaciais, ou para ir a locais onde um humano não consegue chegar, quer seja pelo tamanho, quer pelas especificidades do ambiente;
    - Robótica Doméstica – como é o caso dos aspiradores automáticos, que conforme a programação definem o melhor caminho e indicam ou não ao proprietário, se conseguiu concluir o que foi programado;
    - Robótica nos brinquedos – robótica que foi projetada para o ensino de pessoas, ou que foi implementada num determinado brinquedo para responder a certos estímulos realizados por uma pessoa.
- Robótica fixa ou de manipulação – este tipo de robótica é projetado para permanecer a fixa num local definido, dividindo-se em:
    - Robótica médica – robôs construídos para realizar operações (cirurgia robótica), apresentam uma grande precisão, possibilitam uma resposta a qualquer movimento do médico, e essencialmente traduzem as sensações para o médico saber e sentir o que está a fazer;
    - Robótica industrial (figura 2.13) – como o nome indica são robôs construídos para aplicações industriais, podendo estes seres capazes de pegar em quase todo o tipo de pesos e realizar quase todo o tipo de movimentos;

Figura 2.13 Exemplo de robótica industrial [18].

    - Cobot ou robô colaborativo (figura 2.14) – é um robô que pode operar com segurança, ao lado de humanos e no mesmo espaço de trabalho. Pode funcionar como robô industrial, se assim for programado ou se a ferramenta que opera for perigosa. Tem sensores que medem a força, e quando sente alguma força contra o movimento do robô, o mesmo possui a capacidade de interromper o movimento que está a fazer e entra em erro ou em emergência. Para voltar a funcionar tem de se rearmar o robô e o mesmo fica pronto a funcionar.

Figura 2.14 Exemplo de Cobot [19].

2.2 Automação

A automação pode-se caracterizar como uma tecnologia relacionada, que aplica sistemas mecânicos, eletrónicos, e/ou baseados em computadores, com vista à operação e ao controlo da produção [20]. Quer com isto dizer que, a automação pode ser simplesmente o uso de um equipamento (mecânico, eletrónico ou computacional) que ajuda na realização de uma ou mais tarefas, por exemplo uma batedeira de bolos é um exemplo de automação, pois o pasteleiro deixa de misturar os ingredientes à mão, e assim pode fazer outras operações que sejam necessárias.

A automação pode ser classificada em diversas vertentes, entre os quais a doméstica e a industrial, sendo que o tipo de automação que se irá abordar neste relatório de estágio é a automação industrial.

A automação industrial é normalmente pensada para fábricas ou ambientes fabris, em que é necessário ter em conta o ambiente de funcionamento e as especificidades do trabalho. Para poder ser realizada, a automação industrial tem de utilizar três tipos equipamentos:

- Controlo – equipamento capaz de receber informação dos sensores e dar ordem à parte de atuação. Por exemplo: computador, autómato (PLC), entre outros;
- Atuação – equipamento que está sujeito a ações da parte do controlo. Por exemplo: tapetes de movimentação de peças elétricos, válvulas pneumáticas, motores elétricos, entre outros;
- Sensores – equipamento que transmite à parte de controlo as informações de estado sobre um determinado processo. Por exemplo: sensores fim de cursos, sistemas de visão (3D, infravermelhos, 2D, etc.), etc.

2.2.1 História Automação Industrial

A revolução industrial nos séculos XVII e XVIII, originou uma mudança de pensamento na forma de produção, e com isso a introdução do conceito de automação industrial. Nesse tempo, a automação consistia na produção de máquinas mecânicas que ajudassem o operador a realizar as diferentes tarefas.

No século seguinte (século XX), com o aparecimento da energia elétrica e posteriormente o computador, levou a que o tipo de automação começasse a levar outro rumo, mudando de mecânica para elétrica [21].

Com esta mudança, em 1948, John T. Parsons criou um sistema de automação que consistia no uso de cartões perfurados com informações. Os cartões continham informações que serviam para controlar os movimentos de uma máquina-ferramenta, sendo desenvolvido no Laboratório de Servomecanismos do Instituto Tecnológico de Massachusetts (MIT).

Passados seis anos (em 1954) apareceram os primeiros robôs com o nome de tcheco robota, que significa "escravo". Os robôs foram criados por George Devol, tendo como principal objetivo substituir a mão-de-obra no transporte de materiais perigosos, mas com o passado do tempo, a GM instalou robôs na sua linha de produção para a soldagem de carrocerias [22].

Com o passar dos anos a automação industrial dividiu-se em três grupos (figura 2.15):

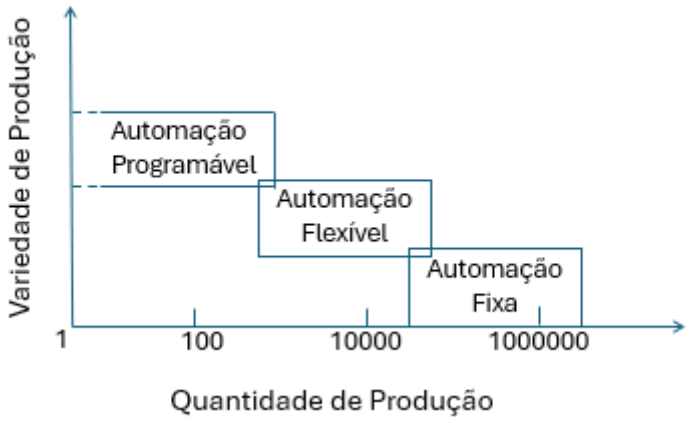


Figura 2.15 Tipos de automação.

- Automação Fixa – cada equipamento tem um conjunto de movimentos fixos, não havendo alteração dos mesmos;
- Automação Flexível – os equipamentos são projetados com capacidade de alterar a sequência das operações, conforme as especificidades dos produtos. Há a possibilidade de fazer alguns produtos diferentes;
- Automação Programável - o sistema é capaz de produzir uma variedade de peças ou produtos praticamente sem perdas de tempo. As únicas perdas de tempo estão relacionadas com as trocas de formato de um desenho ou modelo para o seguinte. Não é considerado tempo de produção perdido ao reprogramar o sistema e alterar ferramentas, equipamentos, configurações da máquina, mas sim tempo de funcionamento. [23]

Com a evolução da automação industrial e com a inteligência artificial, atualmente os autómatos e robôs quando bem programados, já são capazes de tomar decisões por si próprios, de acordo com a programação efetuada.

2.2.2 Constituição de um autómato

O autómato ainda não funciona totalmente por si só, e neste sentido, necessita de um conjunto de equipamentos de auxílio para uma funcionalidade mais eficaz (figura 2.16), entre eles destacam-se:

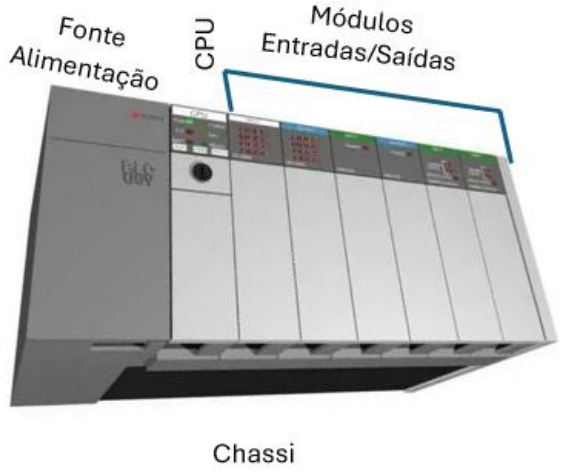


Figura 2.16 Exemplo da constituição de um autómato [24].

- *Rack* ou chassi - atua como a espinha dorsal do sistema, local onde irão estar colocados os autómatos e os equipamentos necessários.
- Módulo de Fonte de Alimentação - é usado para fornecer a energia necessária para todo o sistema PLC. Ele converte a energia CA (corrente alternada) disponível em energia CC (corrente continua), que é exigida pela CPU (*Central Processing Unit*) e pelo módulo de E/S (entada/saída). O PLC geralmente funciona utilizando uma fonte de 24V CC.
- Módulo CPU e memória – A CPU é o cérebro do PLC com um microprocessador octal ou hexagonal. Sendo uma CPU baseada num microprocessador, ele substitui temporizadores, relés e contadores. Os processadores de texto são usados para processar texto, dados numéricos, controlar e gravar dados. A CPU lê os dados de entrada dos sensores, processa-os e envia o comando para os dispositivos de controle. A nível de memórias temos a memória ROM (*Read-only Memory*) que inclui um sistema operacional, drivers e programas de aplicações, e a memória RAM (*Random Access Memory*) que é usada para armazenar programas e dados.
- Módulo de Entrada/Saída - ajuda a fazer a interface de dispositivos de entrada e saída com o microprocessador.
- Módulo de Interface de Comunicação - para transferir informações entre CPU e as redes de comunicação, módulos de E/S inteligentes que forem usados. Esses módulos de comunicação ajudam a conectar com os outros PLCs e computadores que são colocados em um local remoto.

2.2.3 Linguagem de programação de autómatos

Os autómatos para serem programados têm vários tipos de linguagem, sendo que as mais usuais são as que respeitam a norma IEC (*International Electrotechnical Commission*) 61131-3.

A norma IEC 61131-3 define quais são as linguagens utilizadas para programar um PLC para que haja um padrão e ser semelhante em todos os fabricantes, de modo a criar um *standard*. Assim sendo esta norma consiste nas seguintes linguagens [25]:

- *Structured Text* (ST) - Texto Estruturado;
- *Instruction List* (IL) - Lista de Instruções;
- *Ladder* (LD) - Linguagem Ladder;
- *Function Block Diagram* (FBD) - Diagrama de Bloco;
- *Sequential Flow Chart* (SFC) - Diagrama de Fluxo.

A norma IEC 61131-3 permite o uso de outras linguagens de programação (*Visual Basic, Flow Chart*, C++, entre outas), desde que obedeçam às mesmas formas de chamadas e trocas de dados.

2.2.3.1 *Structured Text* (ST)

O *Structured Text* (figura 2.17) é uma linguagem de programação PLC que se assemelha muito a C ou *assembly*. O programador insere linhas de código que serão executadas sequencialmente, avaliando funções específicas, verificando booleanas e energizando saídas apropriadas do PLC. Podendo ser facilmente manipulado em processadores de texto, tornando-o rápido de implementar sem a necessidade de *hardware*. [26]

```
1  IF #start = 1 THEN
2      //comment
3      "Max_nr" := #Array[0];
4      FOR #i := 1 TO 10 DO
5          // Statement section FOR
6          IF #Array[#i] > "Max_nr" THEN
7              "Max_nr" := #Array[#i];
8          END_IF;
9      END_FOR;
10 END_IF;
11
```

Figura 2.17 Exemplo de linguagem *Structured Text* [27].

O programa denomina-se *Main* e é definido mediante as instruções de início de programa e de fim. A linguagem é composta por declarações escritas, separadas por ponto e vírgula. As instruções usam indicações predefinidas para alterar as variáveis. Estas podem-se apresentar através de valores definidos explicitamente, variáveis armazenadas internamente ou entradas e saídas.

As vantagens da programação usando texto estruturado:

- Intuitivo para outras linguagens de programação - é de fácil aprendizagem para aqueles que estão habituados a programar noutras linguagens de programação. Apresenta as mesmas estruturas, paradigmas de programação e funções que se esperaria ver em C ou Java.
- Alta complexidade - permite uma maior flexibilidade do que outras linguagens e, assim, facilita a implementação de funcionalidades avançadas para aqueles que dominam a linguagem.

- Transferibilidade - é padronizado entre a maioria dos sistemas PLC, facilitando a migração entre plataformas. Pode ser implementado em plataformas de hardware e software.

Desvantagens da programação usando texto estruturado:

- Difícil de solucionar problemas - quando comparado à programação lógica *ladder*, o texto estruturado é muito mais complexo do ponto de vista da solução de problemas. Não há filas visuais, menos recursos visuais e, normalmente, mais código em uma única linha;
- Propenso a erros - proporciona maior flexibilidade ao programador. Os programadores devem usar as práticas recomendadas de programação para criar proteção de falhas e de deteção de erros.

2.2.3.2 *Instruction List* (IL)

As *Instruction List* (figura 2.18) são uma linguagem textual de baixo nível semelhante à linguagem *assembly*. Atualmente está a ser eliminado em favor de linguagens mais modernas.

Em termos de fluxo do programa da linguagem, cada linha especifica à instrução, bem como as condições e os resultados da execução. Em muitos aspetos, as Listas de Instruções estão mais próximas de como se implementam programas de lógica *ladder* do que o texto estruturado. No entanto, qualquer uma das linguagens é capaz de criar o mesmo fluxo de processo.

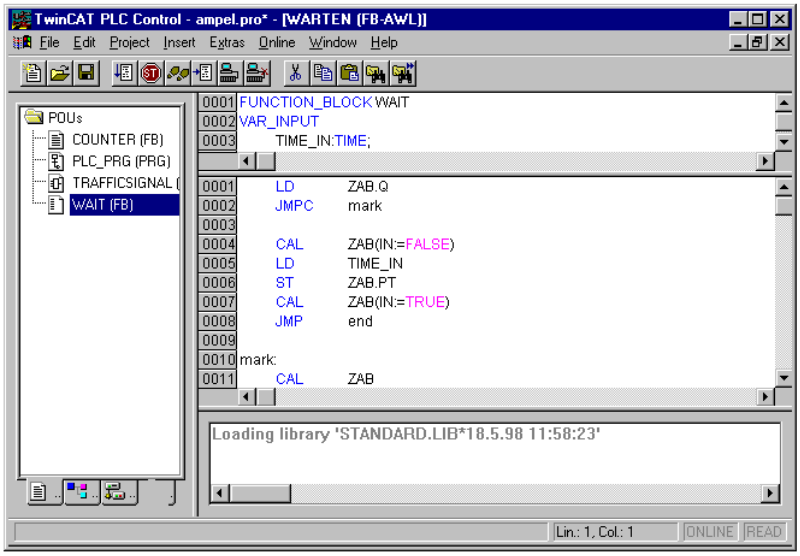


Figura 2.18 Exemplo de linguagem *Instruction List* [27].

As vantagens da programação usando listas de instruções:

- Altamente padronizada - seguem uma estrutura rígida que exige que o usuário crie variáveis explicitamente, especificando condições, e lista todas as instruções. Há pouca variação entre a implementação do programa, o que leva a um código de fácil compreensão.

- Instrução focada - há um alto nível de importância atribuído às instruções em vez do fluxo de dados. Este estilo de programação cria um nível de clareza de como os dados são processados no programa.

As desvantagens da programação usando listas de instruções:

- Indisponível na maioria das plataformas PLC - não são um método popular de programação, pois vêm de forma não natural para a maioria dos programadores.

2.2.3.3 *Ladder Logic* (LD)

A linguagem *Ladder* (figura 2.19) é uma linguagem gráfica de programação PLC, baseada em diagramas de circuitos de lógica de relé. Os relés conduzem cargas com base na lógica simples que é implementada através da ligação aos dispositivos.

A ligação desses dispositivos é delineada em desenhos elétricos e assumem o layout semelhante à de uma escada. À medida que os PLCs mais básicos foram introduzidos, a programação de PLCs com lógica *Ladder* foi projetada para imitar o layout de circuitos baseados em relés. Em outras palavras, a lógica *Ladder* foi uma das primeiras linguagens de programação PLC que ainda é usada hoje devido à sua simplicidade. [26]

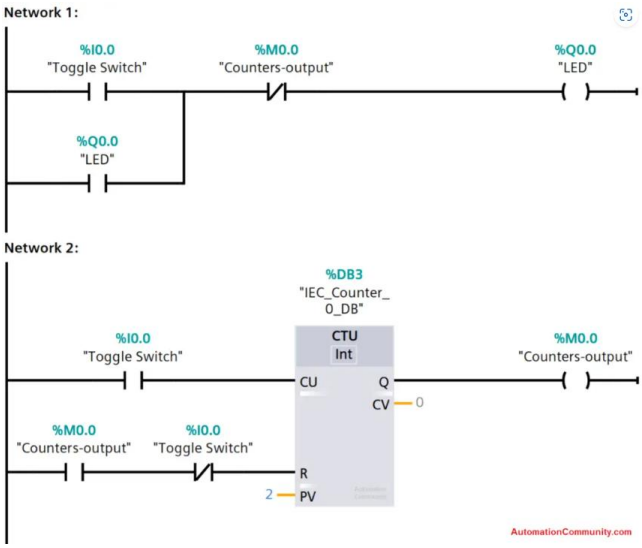


Figura 2.19 Exemplo de linguagem *Ladder* [27].

As vantagens da programação usando linguagem *Ladder*:

- Simples de implementar e solucionar problemas - é uma linguagem visual que fornece confirmações de estados para a maioria das instruções. Ou seja, é fácil para alguém com pouco conhecimento de um processo específico percorrer o programa e entender a lógica;
- Design modular - lógica pode ser facilmente modificada através da adição ou subtração da lógica. Cada degrau é uma condição separada e pode ser removido ou adicionado conforme necessário;
- Resiliência e Consistência - permite que o utilizador implemente variadas funções. No entanto, a linguagem é altamente padronizada e não há flexibilidade total, mantendo assim, o código consistente entre as diferentes implementações.

As desvantagens da programação usando linguagem *Ladder*:

- Curva de aprendizado Íngreme - é uma linguagem simples, mas não muito intuitiva para quem tem conhecimentos noutros tipos de linguagem de programação;
- Implantação lenta - devido à natureza visual da lógica *Ladder*, um programador leva mais tempo para criar a lógica que imaginou. Há uma necessidade de arrastar e soltar elementos que retardam o processo de desenvolvimento, em comparação com outras linguagens de programação modernas;
- Não intuitivo para aplicações complexas - esta linguagem é boa quando se trata de tarefas booleanas sequenciais. No entanto, quando se trata no controlo moderno que envolve controle de fluxo, sensores analógicos e *loops* de *feedback,* nem sempre é fácil de implementar e decifrar.

2.2.3.4 *Function Block Diagram* (FBD)

A linguagem *Function Block Diagram* (figura 2.20), é uma linguagem de programação desenvolvida pensando em processos químicos. Permite ao programador criar uma representação visual e fluxo do processo com transições apropriadas entre as instruções. O editor visual é amigável, intuitivo e cria uma maneira natural de implementar fluxos específicos.

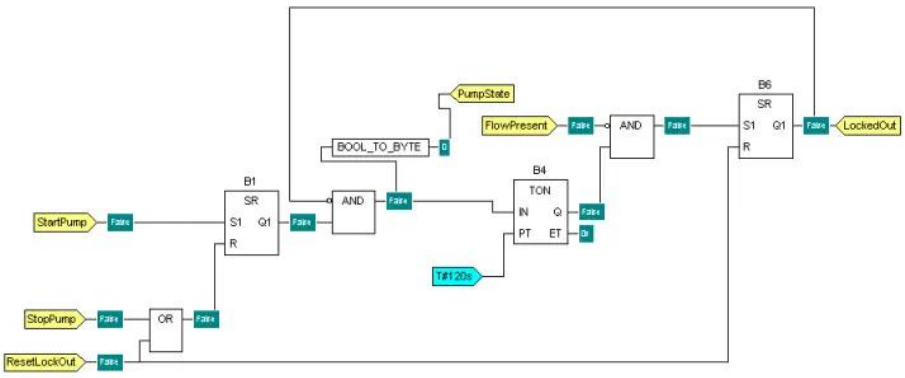


Figura 2.20 Exemplo de linguagem *Function Block Diagram* [27].

As vantagens da programação usando de diagramas de blocos:

- Editor visual flexível – a programação de Diagrama de Blocos de Função é muito amigável e fornece uma maneira simples de criar qualquer *layout*;
- Ideal para estruturas de programação complexas - na lógica *ladder*, o programador terá de usar vários degraus, enquanto uma única página de FBD chegará;
- Fácil utilização- o editor visual do FBD, vem naturalmente para a maioria dos programadores. Sendo o *layout* do processo ser recriado por meio de uma metodologia de arrastar e soltar.

As desvantagens da programação usando de diagramas de blocos:

- Difícil de padronizar - devido à flexibilidade no *layout*, é desafiador padronizar programas escritos em FBD. Cada programador de PLC terá uma abordagem diferente dos outros. Quem vier atrás terá dificuldade em entender o fluxo de informações;

- Problemático em escala - a FBD brilha quando se trata de pequenas implementações de áreas específicas de um processo. No entanto, à medida que o programa se torna complexo, é fácil de se perder em todas as folhas.

2.2.3.5 *Sequential Flow Chart* (SFC)

A linguagem *Sequential Flow Chart* (figura 2.21), é constituída por um conjunto de etapas e processos identificados como num *Grafcet*. Por exemplo, as etapas do processo são executadas numa sequência, tendo condições de início definidas e fluxo como o processo será executado na instalação de produção.

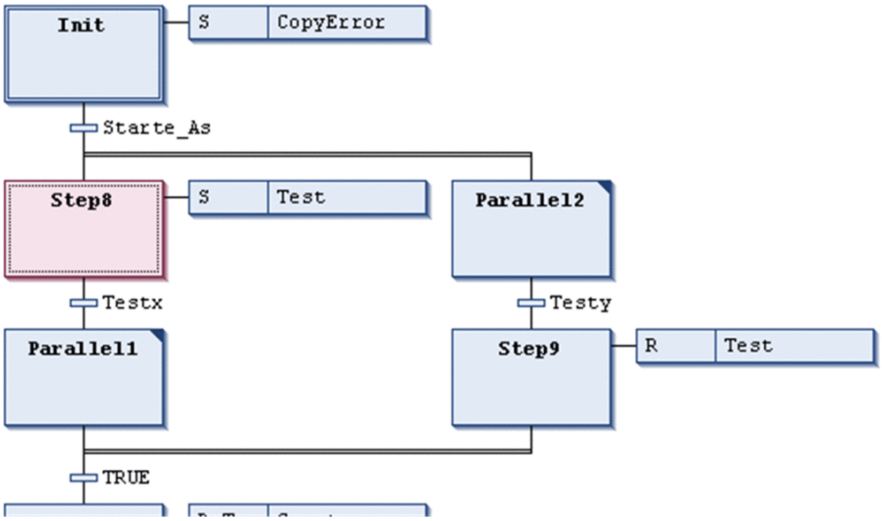


Figura 2.21 Exemplo de linguagem *Sequential Flow Chart* [27].

As vantagens da programação de gráficos de função sequencial PLC:

- Combinado com ST - A maioria dos editores SFC permite o uso de Texto Estruturado em casos específicos para criar fluxos lógicos avançados.

As desvantagens dos gráficos de função sequencial:

- Inaplicável na maioria das aplicações - é desafiador aplicar gráficos de função sequenciais a um processo que não é sequencial. Mas tem de se saber, que tem limitação no número de casos de uso;
- Fluxos paralelos são difíceis de implementar e solucionar problemas - pode se implementar uma quantidade ilimitada de fluxos de processo através de SFCs. No entanto, à medida que os caminhos do processo se dividem em vários fluxos, torna-se difícil implementar caminhos de fluxo separados e robustos.

2.3 Sistemas de visão 3D

Com o desenvolvimento da tecnologia industrial, houve a necessidade de inspecionar o produto final, inicialmente era efetuada pelos trabalhadores, e com o avançar da tecnologia começou-se a utilizar os sistemas de visão 3D. Esses sistemas, permitem detetar os defeitos do material produzido, inicialmente utilizando sistemas a 2D (duas dimensões), passando para sistemas a duas ou três dimensões.

Os sistemas de visão 3D (três dimensões) são compostos por uma câmara de profundidade, projetada especificamente para apresentar informações precisas de profundidade em três dimensões. [28]

2.3.1 Técnica captação de imagens

As técnicas de captação de imagens, dependem do tipo de sistema de visão, podendo ser as seguintes:

2.3.1.1 Triangulação a laser (Laser Triangulation)

A técnica de triangulação a laser é adequada para capturar de objetos a curtas distâncias. Utiliza um laser de linha, que é usado como elemento de projeção e uma câmara monocromática que é usada como elemento de deteção. Assim, que a linha laser atinge o objeto, a trajetória da linha é modificada. Essa modificação leva a que, a partir da triangulação trigonométrica, com a distância entre o laser-sensor e o sensor-objeto, seja possível obter o ângulo de projeção relacionado com a distância entre o objeto e o sensor. Desta forma consegue-se obter uma nuvem de pontos complexa.

As vantagens da Triangulação a laser**:**

- Esta técnica fornece alta resolução e precisão na nuvem de pontos obtidos.

As desvantagens da Triangulação a laser**:**

- As propriedades do objeto a analisar, pode afetar o processo de obtenção de imagem, uma vez que ao usar um laser, os objetos transparentes ou muito brilhantes são difíceis de digitalizar.
- Exige deslocamento do equipamento ou do objeto [29].

2.3.1.2 Imagens estereoscópicas (Stereo Vision)

A técnica da visão estereoscópicas funciona de uma maneira muito semelhante ao olho humano. Utiliza dois sistemas de visão 2D, que capturam duas imagens a partir de posições diferentes, calculando um mapa 3D usando o princípio da triangulação. Em vez de trabalhar com um sistema de visão e um laser, como o anterior, esta técnica trabalha com dois sistemas de visão simulando o olho humano. Capturam duas imagens e usam o ângulo conhecido entre os dois sistemas de visão, fazendo uma correlação de pontos entre as duas imagens.

Os pixels são emparelhados entre os dois conjuntos de imagens para produzir profundidade 3D. Os pontos coincidentes entre as imagens são baseados na variação da textura de forma a encontrar arestas e características distintas, o que pode causar problemas quando as superfícies do objeto não terem contraste suficiente ou são demasiado semelhantes.

As vantagens das Imagens estereoscópicas:

- Nível de precisão e velocidade de digitalização [29].

As desvantagens das Imagens estereoscópicas:

- É necessário obter imagens de referência de excelente qualidade para um bom resultado. A falta de textura nas superfícies pode fazer com que haja problemas.

2.3.1.3 Luz estruturada (Structured Light)

Nesta técnica, denominada luz estruturada, utiliza-se um projetor que projeta diferentes padrões de luz (proporcionando a sua própria textura) na superfície do objeto e grava a forma como o objeto distorce esses padrões.

A luz estruturada usa o mesmo princípio da triangulação trigonométrica, com a diferença de que o objeto tem de permanecer estático. Em vez de usar a técnica de projeção a laser de uma linha, é usado a projeção de padrão (Fringe Projection) geralmente binário. A luz estruturada é uma técnica rápida, que permite uma boa resolução.

As vantagens da Luz Estruturada:

- A precisão é a sua principal vantagem, uma vez que calculam milhões de pulsos de laser por picossegundo, e podem fazer a digitalização tanto de objetos grandes como de ambientes completos.

As desvantagens da Luz Estruturada:

- A morosidade de processamento, uma vez que o tempo necessário para medir cada ponto da imagem, varia em função do tamanho e da profundidade do objeto [29].

2.3.1.4 Pulso Laser (Time of Flight (ToF))

A técnica de pulso laser ou Time of Flight, baseia-se no tempo entre a emissão de luz infravermelha, demora a ir e a voltar ao embater num objeto. Conhecendo a velocidade constante da luz, pode-se obter a distância exata de um objeto, calculando o tempo que demora o infravermelho a ser emitido até ao seu retorno.

As vantagens do Pulso Laser:

- Velocidade de digitalização, incluindo grandes áreas, que também podem ser cobertas.

As desvantagens do Pulso Laser:

- É muito sensível às condições de iluminação do ambiente. Trabalhar ao ar livre num dia de sol é praticamente impossível [29].

2.3.2 Considerações ao escolher uma câmara de visão 3D

As principais considerações ao escolher uma câmara de visão 3D, são:

- Tipo de Aplicação – uma vez que a câmara é usada para permitir a deteção ou seleção ou colocação de objetos, pode-se dizer que conforme o ambiente de trabalho deve ser escolhida uma câmara mais ou menos robusta, ou seja, para um ambiente onde haja contacto com água ou possibilidade do mesmo, tem de se escolher uma câmara e equipamento auxiliar que seja capaz de suportar as exigências do ambiente.
- Precisão - para comparar a precisão entre câmaras 3D, é necessário entender a definição dos diferentes termos que são usados para medir a precisão. De acordo

com a ISO 5725, a definição de precisão é a combinação entre a precisão e a veracidade:

- Precisão - descrição de erros aleatórios, numa medida de variabilidade estatística.
- Veracidade - descrição de erros sistemáticos, numa medida de viés estatístico.
- Exatidão - descrição da combinação de erros aleatórios e sistemáticos (somatório entre a Precisão e a Veracidade).

Quanto maior a precisão de uma câmara 3D (precisão e veracidade), melhor adequa-se para curtas distâncias e pequenos objetos, uma vez que o erro pode originar problemas, caso haja colisões.

- Facilidade de uso - há três elementos para tornar o processo de desenvolvimento mais eficiente:
  - Hardware - uma câmara 3D deve ser flexível o suficiente para ser instalada no robô ou estacionária, dependendo do seu uso. No caso de qualquer força de torção, flexão e tração, os cabos precisam de ser projetados para aplicações de visão mecânica.
  - Software de visão mecânica – facilidade de configurar o algoritmo de análise de imagem, se suporta a linguagem de programação adicional, se tem algum recurso integrado para calibração e análise avançadas de imagem.
  - Documentação – manual de ajuda para iniciar a utilizar a câmara 3D e resolução de quaisquer problemas no futuro, de forma rápida.
- Segurança - uma câmara não baseada em laser pode ser vantajosa, pois evita que o operador seja exposto à luz laser, seja por exposição direta ou por reflexos de objetos brilhantes.

2.4 Projetos semelhantes

No presente capítulo são apresentadas pesquisas distintas de dissertações e artigos sobre as áreas que serão abordadas no projeto, com intuito de demonstrar projetos idênticos ao que vai ser descrito no capítulo 3.

A seguinte dissertação de mestrado, tem como base a utilização de um robô para realizar uma paletização (Hélder Torres Aguiar, Desenvolvimento de um Sistema de Paletização Robotizado [30]). O presente projeto, tem como princípio o estudo de movimentos de largada de peça, e com isto foi possível observar-se o pensamento dos movimentos que visam a utilização de pontos auxiliares e/ou de aproximação. Também é possível observar um estudo do modo de largada de caixas, e a sua a ordem de colocação em cima da palete.

A próxima dissertação (Teresa Cristina de Sousa Azevedo, Reconstrução e Caracterização de Estruturas Anatómicas Exteriores usando Visão Activa [31]) utiliza a visão 3D para o mapeamento de superfícies ou objetos, na qual pode-se concluir qual é a importância de uma boa distância entre o sistema de visão e o objeto em questão, e qual a inclinação

necessária do sistema de visão, para obter uma boa captura de imagem. Para além disso, consegue-se verificar como o sistema faz a construção de uma imagem 2D e transforma-a numa imagem 3D.

O artigo com o tema de sistemas robóticos guiados por visão com estratégias de controlo inteligentes para tarefas autónomas [32], utiliza sistema de visão na execução de tarefas autónomas requeridas. A finalidade é a utilização de um sistema de visão 3D (três dimensões) na ponta do robô, onde um objeto pré-definido é capturado pelo sistema de visão e com isso o robô dirigir-se autonomamente até à posição da peça (figura 2.22). Este artigo, tem como propósito fornecer informação detalhada sobre as diferentes fases da movimentação de um robô que segundo os autores, podem dividir-se por: perceção, localização, planeamento e servo visual. Nas diversas fases identificadas foram analisadas as diferentes ações e comportamentos do robô de forma a fornecer explicações sobre as variadas técnicas e fundamentos aplicados. Para além do projeto indicado no artigo anterior, foi também incluído um conjunto de controladores de alto nível, necessários para a fase de movimentação do robô baseadas em abordagens de otimização, com vista a redução de tempo de movimentação.

Figura 2.22 Ambiente de trabalho do robô [32].

O próximo artigo aborda uma ideia semelhante ao primeiro, sendo que o principal objetivo deste segundo projeto tem em vista a minimização do erro de estimativa em pontos apreensíveis [33], o aperfeiçoamento da eficiência através da utilização de diferentes técnicas, a precisão do planeamento do percurso realizado pelo robô para evitar obstáculos que possam ser identificados, e por fim a retenção do alvo numa posição estável e segura, utilizando diversos tipos de manipulação e captura de imagem. No mesmo são apresentadas cinco técnicas diferentes para a manipulação do robô, entre as quais: aquisição de dados 3D; reconhecimento de objetos; estimativa da posição; análise da posição da garra do robô; planeamento do movimento. Além do que foi referido anteriormente, com o auxílio de várias comparações e cenários de aplicações específicos, concluiu-se que:

- Os dados 3D são insuficientes, embora o custo seja mais reduzido e, atualmente na área da robótica sejam fundamentais, estes podem se tornar dispendiosos no que toca à recolha de dados 3D, especialmente com anotação, em comparação com os conjuntos de dados de imagens 2D (duas dimensões) tradicionais.

- A fusão de dados multisensores, por mais que seja um processo crucial na robótica, principalmente para que exista uma melhoria da qualidade do processo e uma maior utilidade dos dados obtidos, como por exemplo, reconhecer e apreender um objeto robusto com um único sensor, tal como discriminar se a garrafa está cheia ou vazia com a visão ou mesmo clarificar objetos com cores diferentes com um sensor tátil, o cruzamento de ambos os dados não seriam suficientes para a eficácia do processo. Uma vez que com o auxílio do cérebro humano existe uma melhor capacidade de perceção de diferentes modalidades.

- O reconhecimento geral de objetos 3D que apresentam características como a opacidade e rigidez, nos dias que decorrem, ainda é uma adversidade, pois de acordo com diversos estudos, a maioria dos objetos do quotidiano são na verdade objetos não rígidos, como por exemplo roupas e livros; outros são transparentes, por exemplo as janelas ou copos de vidro; e ainda objetos sem textura tal como os que possuem uma superfície de metal ou plástico. Com isto, entende-se que existe uma grande dificuldade na identificação destes tipos de objetos.

- A manipulação de objetos orientada para a realização de tarefas específicas ainda é um grande desafio, pois existe uma dependência entre a forma como o objeto é manipulado e a função do mesmo. Como se pode verificar, a atribuição de tarefas para certos objetos pode causar dificuldades durante o processo.

- A ligação entre a segurança e a interação homem-robô durante o processo, pode ser uma vantagem para muitas empresas, uma vez que já existe ambientes industriais onde a relação entre o homem e o robô é real, e que para isso tem de haver equipamentos de segurança definidos conforme o ambiente ou o tipo de trabalho. Neste caso, o robô identifica o movimento e a intenção humana com sensores táteis, por exemplo, sensores de toque e pele artificial, e sensores sem contacto físico, de forma a evitar colisões entre robôs e trabalhadores e/ou obstáculos em tempo real.

O seguinte artigo demonstra como o cálculo de estimativa erro entre o sistema de visão 3D e o robô pode afetar o resultado da movimentação do robô [34]. Para a realização do projeto foi utilizado um robô Mitsubishi PA-10, que continha sete juntas de rotação, conhecidos como "*joints*", sendo que o controlo escolhido foi o de velocidade. Neste caso, o robô movimenta-se utilizando comandos de velocidade para cada junta. A câmara de visão 3D foi posicionada na ponta do robô PA-10, antes da realização da tarefa, foi necessária uma calibração utilizando um algoritmo.

Na prática, este trabalho consistiu num papel branco com nove pontos pretos, utilizados na estimativa de posição do alvo fixo. O centro da área de cada ponto foi usado como um ponto de destaque ao processar a imagem dentro de um sistema de visão. Sabendo que a taxa de *frames,* já incluindo o tempo de processamento de imagem, foi cerca de 20 *frames* por segundo, e que o tempo de otimização de resposta de todo o sistema fosse eficiente, os ganhos do controlador visual do robô foram ajustados manualmente. Uma vez que o objetivo principal era encontrar um alvo fixo (pontos pretos), para que fosse calculado com base em estimativa erro, de modo a obter uma resposta concisa, foi adicionado um erro entre o *frame* da câmara e a ponta do robô. Os resultados da experiência foram comparados com as informações obtidas utilizando cálculos de cinemática da posição do alvo. Para isso, a posição do alvo foi fixada em X, Y e Z e a sua orientação em *Raw, Pitch* e *Yaw.*

Concluiu-se que a realização do projeto utilizando a estimativa de posição do alvo obtém-se informações tridimensionais sobre a relação entre o alvo fixo e a câmara com uma precisão aceitável. Além disso, o sistema de visão baseado na posição de controlo, os seus sinais são demonstrados no sensor na câmara e, consequentemente, este tipo de sistema com base na posição é mais percetível a erros no robô e também a erros de calibração.

2.5 Considerações finais

Uma vez realizada a revisão literária no âmbito do estágio e da elaboração do presente relatório, vão ser apresentadas umas breves considerações sobre os temas abordados (robótica, automação e sistemas de visão 3D).

A nível da robótica, no estágio houve uma interação com robôs, quer industriais quer colaborativos. Sendo que neste relatório o tipo de robótica escolhida foi a robótica industrial visto que não havia a necessidade de ter um operador perto do robô. Para o projeto escolhido, o robô era composto por 6 juntas e 4 elos, uma vez que vão ser utilizados vários tipos de movimentos tais como, movimentos lineares e movimentos de juntas.

No tema da automação, a linguagem utilizada no autómato é a *Ladder Logic*, uma vez a programação e a verificação/localização dos erros de programação torna-se mais rápida e eficiente.

A nível da visão 3D utilizar-se-á um sistema de visão que utiliza a triangulação a laser, pois no projeto, o objeto com alvo de captura de imagem encontra-se parado, enquanto o sistema de visão esta em constante movimento.

Com os artigos e dissertações apresentados anteriormente, foi possível ter uma perspetiva de como a robótica e a visão 3D podem servir de auxílio à indústria fabril e quais as condições que se devem ter em conta para uma melhor produção. Os projetos mencionados acima vão servir de apoio ao projeto que será descrito no capítulo seguinte, isto é, uma melhor compreensão dos erros que possam ocorrer durante a sua realização bem como as diferentes formas de solucionar as questões ou entraves que possam surgir.

3 PROJETO

Neste atual capítulo será descrito detalhadamente o projeto realizado, sendo que o mesmo se encontra dividido em duas partes:

- Descrição completa do projeto – neste subcapítulo irá fazer-se uma descrição completa do objetivo do projeto;
- Escolha do material e quais as suas razões – neste subcapítulo explicar-se-á toda a escolha de material, bem como quais as condições que tiveram de ter em conta para as mesmas.

3.1 Descrição do projeto

O projeto desenvolvido é um projeto modelo, isto é, um protótipo de máquina concebido para apresentar um produto ou uma ideia ao consumidor. Além desta função este protótipo pode ser utilizado como fonte de estudo para a própria empresa de forma a compreender a sua viabilidade e eficácia na sua produção.

Este tem como objetivo desenvolver uma máquina capaz de colocar uma peça de metal num cabide, cabide esse que estará integrado numa linha de produção de uma fábrica (figura 3.1).

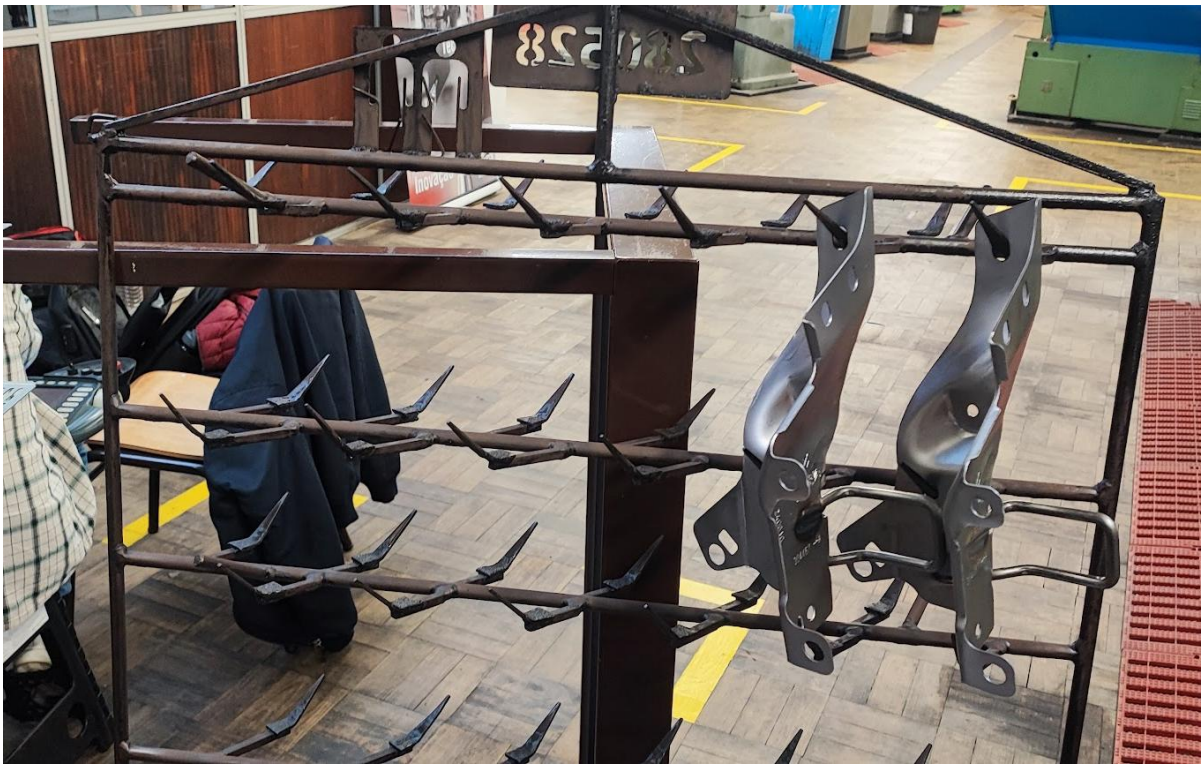


Figura 3.1 Cabide com as peças.

A peça é composta por diversos furos como se pode observe na figura 3.2, sendo que os que estão destacados com a cor vermelha serão essenciais para colocar a peça e com isso segurar a mesma no cabide.

Figura 3.2 Peça a ser colocada no cabide.

O cabide (figura 3.3) vai estar suspenso dentro de uma linha de produção, podendo ter alguma rotatividade, assim sendo, o cabide não terá sempre a mesma posição ou rotação. Para além disso, não se sabe quantos espaços vazios há no cabide, existindo sempre a necessidade de fazer uma identificação de espaços livres, assim como a análise da rotação e da posição do cabide.

Figura 3.3 Cabide das peças.

Na figura 3.4., abaixo identificada, está exemplificado um cabide com altura de 1300mm separado por ganchos de 1700mm; apresenta um comprimento de 800mm que se encontra separado por 100mm.

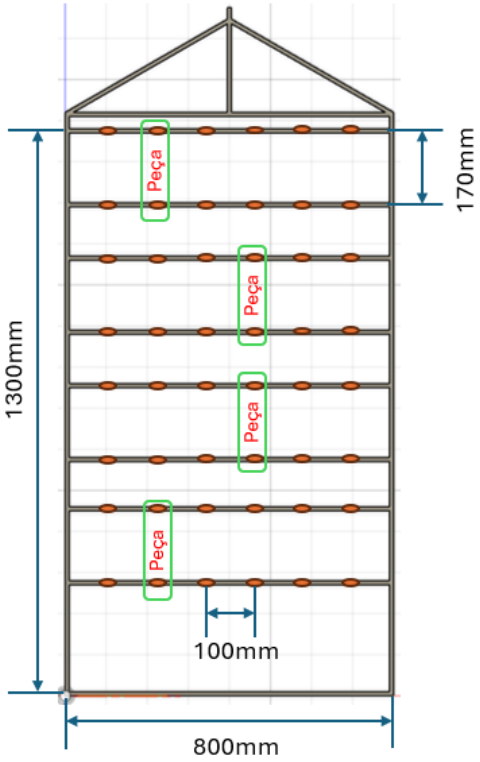


Figura 3.4 Dimensões do cabide das peças.

Para além dos ganchos já existentes no cabide, este também terá de possuir pontos indexadores (marcados a vermelho na figura 3.5) junto do mesmo, a fim de permitirem a identificação da rotação.

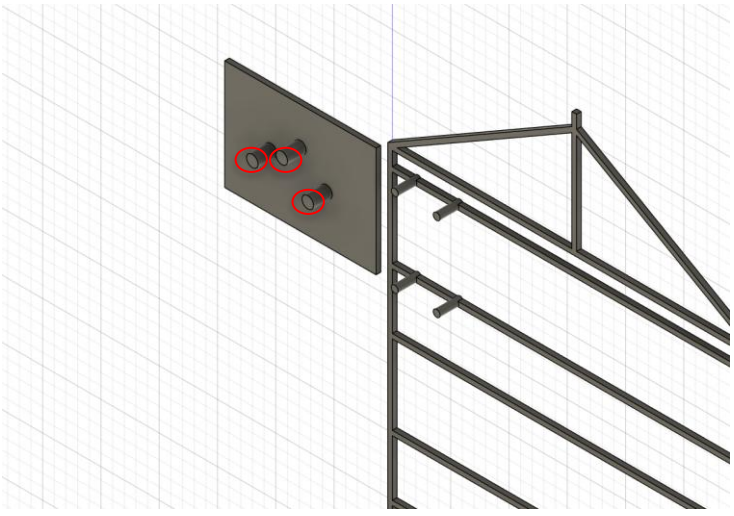


Figura 3.5 Cabide das peças com os pontos indexadores.

Neste projeto em específico, sendo apenas um protótipo existirá eventualmente diferenças relativamente ao robô, que não terá pinça para largar ou agarrar a peça, ou seja, a peça nunca vai sair da ponta do robô. Esta situação deve-se ao facto de este ser um projeto modelo e não existir essa necessidade, uma vez que o objetivo seria apenas simular e compreender o funcionamento em contexto real.

3.2 Escolha de equipamento para o projeto

Para o projeto e tendo em vista a condicionante de não se conhecer a rotatividade do cabide e qual a disponibilidade do cabide foi necessário a utilização de um sistema de visão 3D. O sistema de visão tem de ser capaz de detetar a rotação do cabide e quais os espaços livres, para além disso, tem a função de enviar os pontos de largada de peça para o robô, depois de aplicado o algoritmo de processamento de imagem.

O robô tem de possuir uma garra capaz de segurar e movimentar a peça até ao ponto de largada, que é definido pelo sistema de visão 3D. O robô também tem de ser capaz de deslocar-se até pelo menos a meio do cabide para a realização de um varrimento de imagem do sistema de visão. Tudo isto sabendo que o sistema de visão vai estar posicionado na ponta da garra do robô.

Para a parte de segurança e de comunicação entre equipamentos, foi colocado um autómato, com vista a isolar os equipamentos e assim reduzir as falhas. Sendo que o mesmo poderá ser implementado numa linha de produção e com isso trabalhar em conjunto com outros equipamentos, de modo a uma melhor forma de integração de novos equipamentos, ou até mesmo, criar um interface homem-máquina para trabalhar com a máquina.

3.2.1 Escolha do robô

A característica necessária para a escolha do robô foi o peso da garra, sendo que a mesma não tem uma pinça para largar ou agarrar a peça, uma vez que este projeto tinha como propósito unicamente uma simulação

A garra demonstrada na figura 3.6 é composta pelo sistema de visão 3D e pela peça que se terá de colocar no cabide, que se encontra posicionada na ponta do robô, apresentando um peso aproximado de 2 quilogramas. Para além do mais, o robô precisará de chegar pelo menos a meio do cabide e nesta situação o robô apresenta um comprimento máximo de pelo menos 400mm.

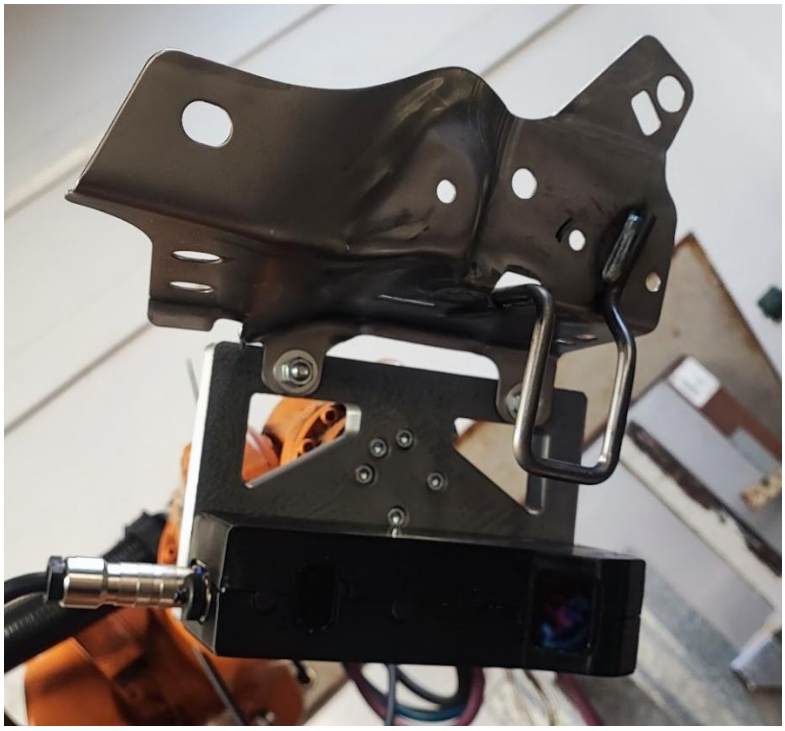


Figura 3.6 Garra do robô com a peça.

Além do que foi descrito anteriormente, o robô terá de ter um protocolo de comunicação *PROFINET*, pois era exigido como protocolo de comunicação visto que a linha de produção utilizaria esse mesmo protocolo.

3.2.1.1 Robô ABB IRB1100

Uma das opções utilizadas no projeto foi o robô ABB IRB 1100 (figura3.7), sendo um robô industrial da marca ABB.

Figura 3.7 Robô ABB IRB 1100 [33].

Este robô tem a capacidade de carga de 4 quilogramas e tem como principal protocolo de comunicação *PROFINET* para a ligação ao autómato, condições que eram exigidas para o projeto em questão.

O equipamento apresentado, seria uma opção viável, visto que o robô estava montado e pronto para ser programado, apesar da versão disponível ser o IRB 1100-4/0.58, com um alcance máximo de 0,58 metros (como se pode ver no anexo II).

Para além disso, este robô para funcionar de maneira correta necessita de um controlador, que neste caso em específico, podia ser o OmniCore C30/C90/E10 (figura 3.8).

Figura 3.8 Controlador Robô ABB IRB 1100 [34].

3.2.1.2 KUKA KR6

Outra opção viável de robô era um KUKA KR6 (figura 3.9), sendo este também um robô industrial da marca KUKA.

Figura 3.9 Robô KR6.

O robô apresentado na figura anterior possui uma capacidade de carga de 6 quilogramas, tendo ainda um comprimento máximo de 1,611 metros (anexo III). Para o caso da comunicação *PROFINET* para a ligação ao autómato, é inevitável a existência de uma carta para que haja a ligação, carta essa, demonstrada no controlador.

O controlador para este robô tem de ser o KRC2 (figura 3.10) e como referido anteriormente tem de ter uma carta de comunicação *PROFINET*.

Figura 3.10 Controlador KUKA KRC2.

Este robô encontrava-se disponível, mas não se encontrava totalmente funcional uma vez que as condições do pavimento não se encontravam no seu melhor estado, dessa forma não garantiam a estabilidade precisa para utilização, para além das baterias que armazenam as configurações também não estavam nas melhores condições (não funcionavam).

3.2.1.3 Robô escolhido

Com as duas opções de robôs descritas, foi necessária uma comparação de ambos. Para a comparação dos dois robôs vai-se utilizar uma tabela (tabela 3.1), para ser mais simples de se relacionar e comparar, tendo como principais características: o controlador, a capacidade de carga, o comprimento máximo, a comunicação *PROFINET* e por fim, o estado do robô.

Tabela 3.1 – Comparação dos robôs.

Robô	ABB IRB 1100	KUKA KR6
Controlador	OmniCore C30/C90/E10	KRC2
Capacidade de carga (kg)	4	6
Comprimento máximo (mm)	580	1611
Comunicação *PROFINET*	Sim	Sim, se tiver a carta
Estado do robô	Pronto a utilizar	Faltava estar preso ao chão e faltavam baterias que guardam configurações

Ao analisar a tabela acima, pode-se concluir que se fosse pretendido que o robô chegasse ao comprimento máximo do cabide (800mm), a opção de escolha seria o robô da KUKA. Caso pretenda-se chegar ao meio do cabide com era o exemplo deste projeto, qualquer um dos robôs servia para este projeto.

Agora, tendo em vista o estado em que se encontravam os robôs, conclui-se que o robô mais indicado era o da ABB, visto que estava pronto a utilizar e tem o protocolo de comunicação pretendido. Ao contrário do KUKA que por não estar bem fixo ao chão, podia ser um grande risco para o operador, bem como para os equipamentos ou para o robô; e também pelo estado das baterias, uma vez que não estavam em condições, e ter-se-ia de estar sempre a configurar os eixos, por essas razões decidiu-se não utilizar o KUKA.

3.2.2 Escolha do autómato

Na escolha do autómato havia dois requisitos a serem seguidos: a possibilidade de ligação a módulos de expansão de entradas/saídas e a necessidade de ligação à rede industrial *PROFINET*.

A rede industrial *PROFINET* é uma rede que foi desenhada para a troca de dados entre autómatos e os dispositivos (entrada/saída), utilizando o standard IEC 61158. Este protocolo de comunicação garante uma comunicação em tempo real, não sendo interrompida pela comunicação padrão baseada em TCP/IP, e os requisitos de tempo sejam atendidos de forma confiável. A rede utilizada teria de ter uma velocidade de 10 ou 100Mbit/s [35].

As opções disponíveis no começo do projeto eram todas da marca Siemens e dentro dessas opções, só havia uma que cumpria todos os requisitos. Os outros autómatos não cumpriam as condições, pelo simples facto de não terem o protocolo de comunicação que se pretendia utilizar, assim sendo o autómato escolhido foi o S7-300 da Siemens (figura 3.11).

Figura 3.11 Autómato Siemens S7-300 [36].

O autómato S7-300 necessita de ter uma fonte de alimentação entre os 19.2V e os 28.8 V CC e uma corrente mínima de 2 A (anexo IV). O autómato escolhido tem a possibilidade de programação em dois *softwares* TIA Portal ou STEP 7. A escolha do software depende da preferência do programador, que será mais elaborado no capítulo da programação do autómato (secção 4.2).

Para além das características identificadas, este autómato tem a possibilidade de ligação à rede *PROFINET* e/ou *PROFIBUS*.

3.2.3 Escolha de sistema de visão 3D

Para a escolha do sistema de visão 3D existia a necessidade deste, ser capaz de detetar a rotação do cabide, quais os espaços livres do cabide e indicar quais as deslocações e/ou rotações do cabide ao robô, tendo como base o processamento de imagens que realizou.

3.2.3.1 Sistema de visão Gocator 2140D

Uma das opções dadas foi o sistema de visão a Gocator 2140D (figura 3.12), sendo o robô industrial da marca Gocator.

Figura 3.12 Sistema de visão 3D Gocator [37].

O sistema de visão Gocator tem uma distância ao objeto antes de conseguir medir (clearance distance, CD) de 190mm, após essa distância tem-se um alcance de medidas (measuremennt range, MR) de 210mm, podendo o campo de visão variar entre 96 e 194 mm como se pode ver no anexo V. Para além do mais, a Gocator tem ligação à tecnologia de comunicação pretendida (*PROFINET*).

O sistema de visão ainda tem a necessidade de ser alimentada por uma fonte de alimentação externa, neste caso tem de ser compreendida entre 24 ou 48 V CC com uma potência de 13 W.

3.2.3.2 Sistema de visão Sick V3T12S-MR32A7

Outra opção de sistema de visão era uma Sick V3T12S-MR32A7 (figura 3.13), sendo este também um robô industrial da marca Sick.

Figura 3.13 Sistema de visão 3D Sick [38].

O sistema de visão Sick tem uma distância ao objeto antes de conseguir medir, de 141mm e após essa distância tem-se um alcance de medidas de 541mm, podendo o campo de visão variar entre 100 e 270 mm como se pode ver no anexo VI. A Sick tem ligação a tecnologia de comunicação pretendida (*PROFINET*).

O sistema de visão ainda tem a necessidade de ser alimentada por uma fonte de alimentação externa, neste caso tem de ser compreendida entre 24 CC, com uma potência de 11 W.

3.2.3.3 Sistema de visão escolhido

Com as duas opções de sistemas de visão 3D descritas, faltava agora comparar as mesmas. Para a comparação dos sistemas de visão vai-se utilizar uma tabela (tabela 3.2), para ser mais simples de se relacionar e comparar as características.

Tabela 3.2 – Comparação dos sistemas de visão.

Sistema de visão	Gocator 2140D	Sick V3T12S-MR32A7
Distância até começar a ver o objeto (mm)	190	141
Alcance máximo (mm)	210	541
Campo de visão (mm)	96 - 194	100 - 270
Comunicação *PROFINET*	Sim	Sim
Estado do sistema de visão	Pronto a utilizar	Pronto a utilizar

Ao verificar e analisar a tabela anterior conclui-se que o sistema de visão recomendado para este projeto seria o da Sick, uma vez que tem um melhor alcance e um o campo de visão mais largo do que o da Gocator.

No entanto como este era um projeto modelo e pretendia-se explorar o funcionamento do algoritmo de processamento de imagem de sistema de visão Gocator, optou-se por escolher esse mesmo modelo, apesar de apresentar um menor alcance e um campo de visão menor.

3.2.4 Fonte de alimentação

A fonte de alimentação vai ser necessária para alimentar o autómato e o sistema de visão 3D. A fonte tem de transformar 230V de CA (corrente alternada), em +24V de CC (corrente continua), pois é a tensão que irá dar um melhor funcionamento dos equipamentos acima descritos, faltando apenas saber a corrente da mesma (figura 3.14).

Figura 3.14 Representação da ligação dos equipamentos.

Para os cálculos sabe-se que a tensão de alimentação do sistema é de +24V CC e que vai calcular-se primeiro a resistência do sistema de visão, e depois a resistência do autómato, mediante os parâmetros que se sabe de cada equipamento.

Assim sendo, para o cálculo da resistência do sistema de visão (Rsistemavisao), utilizou-se a potência do sistema de visão, dada pelo fornecedor (13W), e a tensão de alimentação do sistema. Passando para o cálculo da resistência do autómato (Rautomato), de acordo com o manual técnico, sabe-se a tensão de alimentação e a corrente necessária para o autómato funcionar (2 A).

Com todas as resistências calculadas, fez-se um cruzamento de informação (quadro 3.3), ou seja, calculou-se a resistência equivalente e de seguida a corrente total necessária para o sistema funcionar na sua plenitude.

Tabela 3.3 – Cálculos realizados esquematizados.

Nome do equipamento	Tensão alimentação	Potencia	Corrente	Resistência
Sistema de Visão	+24V DC	13W		44,308 Ω
Autómato	+24V DC		2 A	12 Ω
Total	+24V DC		2,54 A	9,443 Ω

Assim, conclui-se que a fonte de alimentação a utilizar tem de ter pelo menos uma corrente de 2.54 A. Assim sendo, a fonte escolhida foi uma fonte de alimentação com uma corrente de 5 A e uma tensão de +24 V CC (figura 3.15).

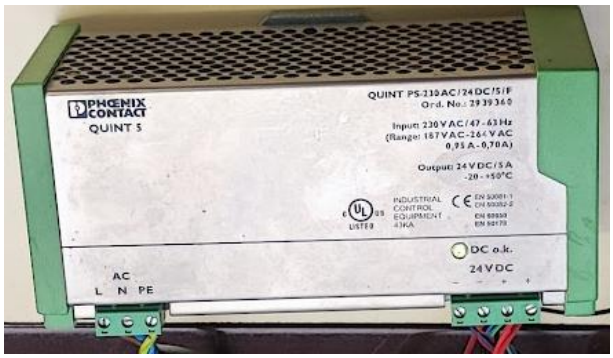


Figura 3.15 Fonte de alimentação escolhida.

3.2.5 Switch

Com os equipamentos ligados à rede de energia, faltava ligá-los de maneira que pudessem comunicar entre si. Tendo em conta esta situação, era necessário um equipamento que fosse capaz de juntar todos equipamentos num só e que se utilizasse portas RJ45, para isso o equipamento escolhido foi um *switch*.

O *switch* (figura 3.14) é um equipamento de rede equipado com diversas portas de comunicação (RJ45 ou fibra ótica) que conecta os elementos de rede, para a transmissão de dados. Trata-se de um dispositivo que recebe os pacotes de dados enviados por qualquer dispositivo da rede, redirecionando-os para o respetivo destino [41].

Para este projeto o *switch* tem de possuir as seguintes características:

- Portas de comunicação RJ45 - o *switch* terá de ter portas de comunicação RJ45 porque todos os equipamentos anteriormente escolhidos têm PROFINET, que utiliza este tipo de portas de comunicação;
- Número de portas de comunicação terá de ser pelo menos 4 (figura 3.16) - pois terá de haver uma para cada equipamento (uma para o robô, uma para o sistema de visão e uma para o autómato) deixando uma de reserva. Sendo que a de reserva vai ser sempre guardada para a necessidade de fazer alterações no programa ou em algum equipamento ligado, efetuadas pelo programador para conectar o computador.

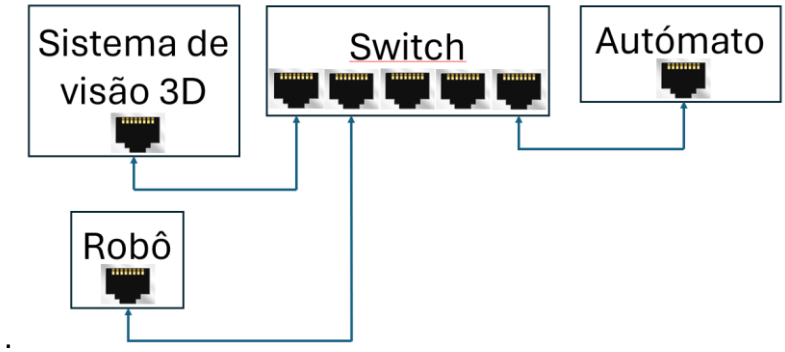


Figura 3.16 Esquema do número de portas do *switch*.

Para cumprir com as características mencionadas anteriormente, o equipamento *switch* escolhido foi um da TP-LINK (figura 3.17) com 5 portas e com possibilidade de velocidade de comunicação 100 Mbps.

Figura 3.17 *Switch* escolhido [39].

Concluindo, para a realização do projeto mencionado, foi necessária a escolha de bons equipamentos que funcionassem na sua plenitude, uma vez que, sem as características apresentadas, o projeto não teria sido bem-sucedido. Apesar dos obstáculos ocorrentes de alguns equipamentos fornecidos, encontraram-se soluções mais eficazes, de forma que todo o trabalho fluísse de melhor forma, e desse modo, o objetivo inicial fosse cumprido.

4 DESENVOLVIMENTO DO *SOFTWARE* DO PROJETO

O presente capítulo inicia-se com uma explicação de um projeto a nível da programação que vai ser descrito de acordo com a ordem presente no relatório, isto é, uma divisão em três temas: robótica; automação, e sistemas de visão

Iniciando na área da robótica, será apresentado um estudo de movimentos realizado com o auxílio de um simulador de um robô, sendo que a sua programação irá ser realizada na consola já existente no robô.

Segue-se o tema da automação, aonde será explicada toda a parte de troca de informações entre os equipamentos e alguma programação suplementar relativa aos pedidos de informação.

Finalizando com a programação de sistemas de visão 3D, aonde será explicado e descrito o algoritmo de processamento de imagem e qual o seu modo de funcionamento de acordo com o sistema de visão.

Para além disso é necessário salientar que este programa está feito só para uma captura de imagem e respetiva entrega de peça, ou seja, caso se pretenda fazer uma nova captura de imagem terá de se fazer uma nova inicialização do sistema.

4.1 Desenvolvimento do sistema robótico

Este capítulo vai dividir-se em dois temas principais: a simulação e estudo de movimentos; e uma breve apresentação das alterações que foram feitas para a realidade em relação a simulação, bem como uma apresentação de parte do código do robô.

4.1.1 Simulação da parte da robótica

O projeto iniciou-se com o estudo de movimentos no simulador, mesmo antes de se iniciar a parte de programação do robô em contexto real.

Neste subcapítulo abordar-se-á o desenvolvimento no software de simulação e quais as conclusões resultantes do mesmo. Para essa análise, vai utilizar-se o software RobotStudio da ABB, que durante a sua utilização foi desenvolvido um manual que se encontra no anexo VII.

Com o uso do simulador pretendeu-se simular o movimento de captação de imagem com a ajuda do sistema de visão, e também simular o movimento de largada de peça no cabide, sendo que os pontos de movimentação terão coordenadas diferentes dos usados na aplicação real. Para conseguir desenvolver estas simulações foi necessário a criação de um ambiente semelhante à realidade.

A fim de simular os movimentos, foi necessário criar-se um cabide semelhante ao que se pretende utilizar, criar uma garra semelhante á que se vai usar no robô, que neste caso para simplificar optou-se por um paralelepípedo, uma vez que a estrutura e as dimensões se assemelham àquela que vai ser utilizada no robô em contexto real (figura 4.1).

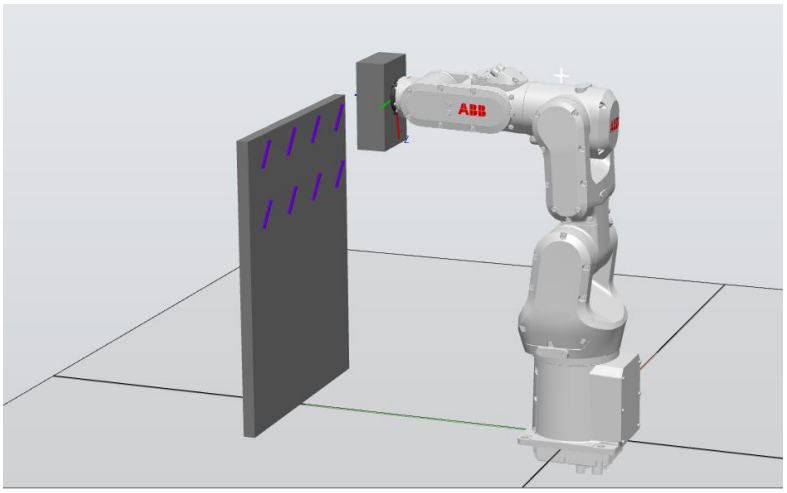


Figura 4.1 Ambiente de simulação.

Com o ambiente criado, passou-se ao estudo dos movimentos, tendo o mesmo sido iniciado pelo movimento de captura de imagens do sistema de visão.

O movimento do sistema de visão 3D, de acordo com as especificações do mesmo, tem de ser executado da esquerda para a direita (figura 4.2 – linha amarela, deslocamento do ponto A para o B), com uma rotação de forma que o sistema de visão fique na parte debaixo da garra, logo o movimento de captura de imagens terá de ter as mesmas características, uma vez que o sistema de visão vai estar na garra do robô.

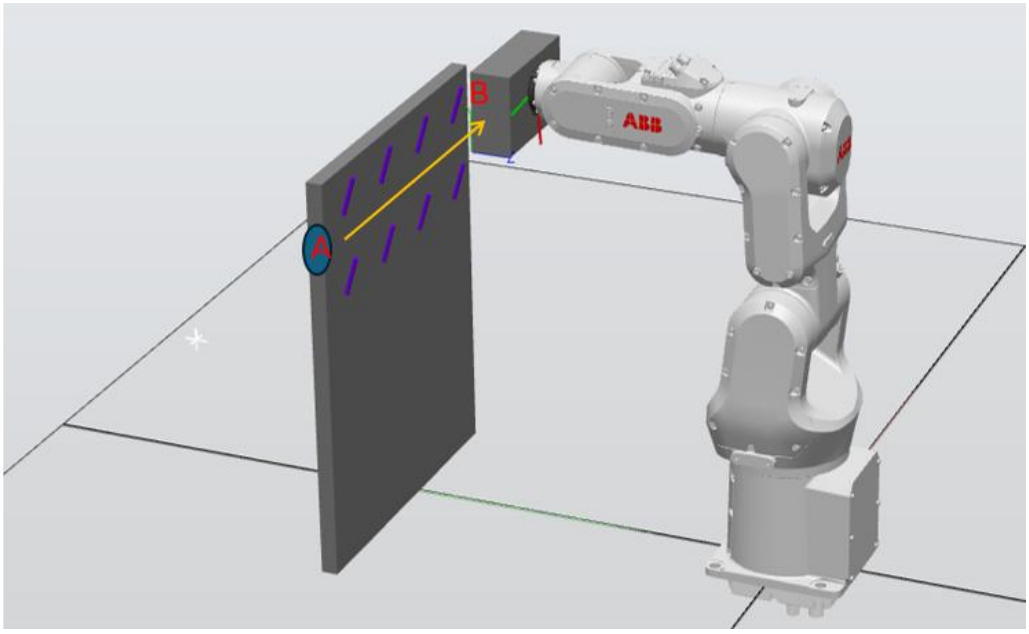


Figura 4.2 Simulação do movimento de captura de imagens.

Com a simulação do movimento de captura de imagens, concluiu-se que é fundamental fazer-se uma rotação da garra, de modo que o sistema de visão 3D fique virado para baixo. Depois da rotação, o robô terá de se movimentar até ao ponto de início de captura de imagem (ponto A da figura 4.2). Quando o robô chegar ao ponto final de captura de imagem (ponto B da figura 4.2), o mesmo terá de voltar à posição de origem.

Para a simulação do movimento de largada de peça, não se vai utilizar as posições dadas pelo sistema de visão, porque no simulador não há a possibilidade de simular o sistema de visão. No entanto, irá utilizar-se pontos marcados que terão o mesmo efeito.

Ainda antes de começar a marcar pontos, teve de se delinear uma estratégia de largada de peça. Esta situação deveu-se ao facto de as peças no cabide, só entrarem se forem colocadas debaixo para cima e da direita para a esquerda do cabide.

No simulador e com o tamanho real dos objetos, reparou se que para ser colocada a peça no cabide irá ser necessário criar pontos auxiliares. Pontos esses que serão importantes para que não haja colisões (peça-cabide ou garra-cabide). Ou seja, antes de cada largada de peça irá haver um ou mais ponto(s) de aproximação para que durante o movimento não haja colisões.

Com estas considerações, o movimento de largada de peça começará com uma rotação da garra, de maneira que a peça fique na vertical. Com a peça na orientação vertical, o robô deslocar-se-á até um ponto auxiliar (ponto C na figura 4.3). Este ponto estará perto do cabide e vai servir para posicionar a peça o mais centrada possível, com o ponto de largada de peça. Quando chegar ao ponto de aproximação (ponto B na figura 4.3), o robô terá de se deslocar para uma posição ligeiramente mais baixa e estará na posição correta que é a de largada (ponto A na figura 4.3). Após a largada de peça, o robô vai repetir o processo, mas neste caso utilizando a ordem de movimentos inversa (figura 4.3).

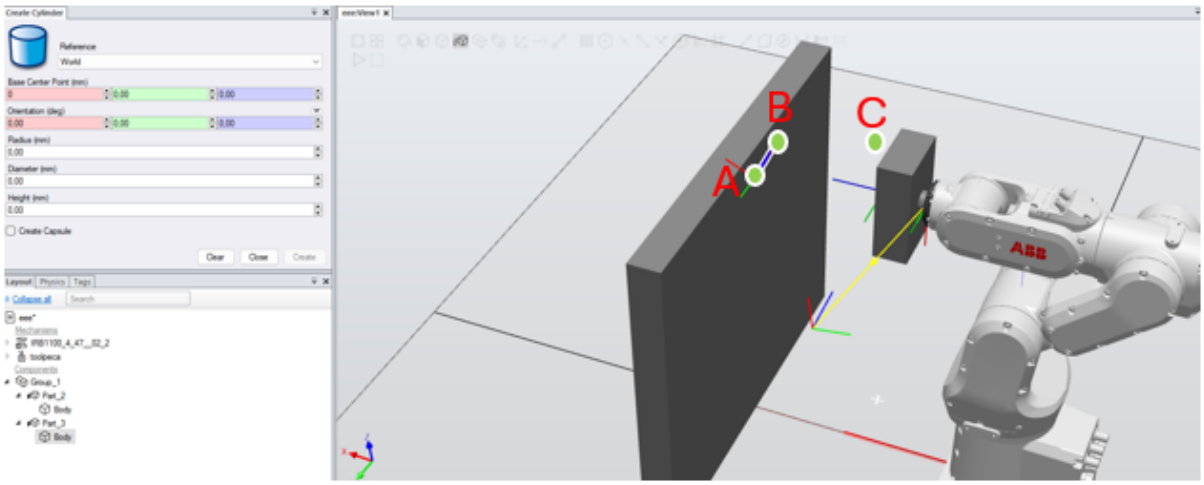


Figura 4.3 Simulação do movimento de largada de peça no cabide.

Com as simulações, concluiu-se que a primeira deslocação do robô vai fazer uma passagem para fazer a captura de imagens e o sistema de visão vai verificar quais são os lugares disponíveis e as suas coordenadas, e depois sim, fazer a entrega da peça ao destino.

Após este processo passou-se para a aplicação em contexto real.

4.1.2 Programação do robô

Com os testes realizados no simulador, foi necessário criar um fluxograma para facilitar a programação do robô e com isso aplicar as conclusões retiradas na simulação. Com o

fluxograma (figura 4.4) pretende-se explicar os movimentos ou ações que o robô irá ter, e assim prosseguir para a programação do mesmo em contexto real.

O fluxograma poder-se-á dividir em três partes. Na primeira parte, vai verificar se estão reunidas as condições de segurança do robô, e se o mesmo está na posição de HOME (posição de referência inicial/final), posições de referência de largada que servirão para ajustar o ponto largada de peça, sendo que também é necessário seguir para a função captura de imagens por parte do sistema de visão 3D. Quando esta função (captura de imagem) terminar o programa tem de seguir para a função de largada de peça.

A segunda parte do fluxograma, está relacionada com a captura de imagem, onde terá de se posicionar o robô num ponto auxiliar, ponto esse que estará numa posição mais atrás do que o de captura. Quando o robô estiver no ponto auxiliar, irá iniciar a capturar de imagem. uma vez que o sistema de visão demora algum tempo a processar. Segue depois para o ponto inicial de captura de imagem que é um ponto predefinido na fase inicial. Este ponto não sofre qualquer alteração a nível da posição, visto que está definido desde o início do programa quando for aplicado em contexto real.

Assim que chegar ao ponto final de captura de imagem, ponto este que estará definido no começo do programa, tal como o anterior, o sistema de visão desliga-se por ordem dada e movimenta o robô até a posição de *HOME*.

A última parte do fluxograma, está relacionada com a receção de dados do sistema de visão e a execução do movimento de largada de peça por parte do robô. Sempre que o robô recebe os valores dos pontos, faz a correção dos mesmos, devido à diferença que existe entre o ponto zero do sistema de visão e o ponto zero da ponta do robô, pois os equipamentos têm pontos de origem em localizados em diferentes zonas.

Com a correção efetuada, o robô irá calcular os pontos de aproximação e os pontos auxiliares. Com os cálculos efetuados, o robô movimenta-se até ao ponto auxiliar, posteriormente reduz a velocidade, e movimenta-se para o ponto de aproximação, e subsequentemente para o ponto de largada de peça.

Após a largada de peça, o robô faz os movimentos pela ordem contrária descrita anteriormente, ou seja, ao estar no ponto de largada de peça, vai deslocar-se para o ponto de aproximação, depois movimenta-se para o ponto auxiliar com uma velocidade maior.

Assim que chegar ao ponto auxiliar, verifica se o sistema de visão enviou mais algum ponto, se sim volta a repetir o procedimento desde a correção de valores, sem precisar de fazer uma nova captura de imagem. Caso não haja nenhum envio de valores por parte do sistema de visão, o robô prossegue até á primeira parte do fluxograma na posição *HOME*.

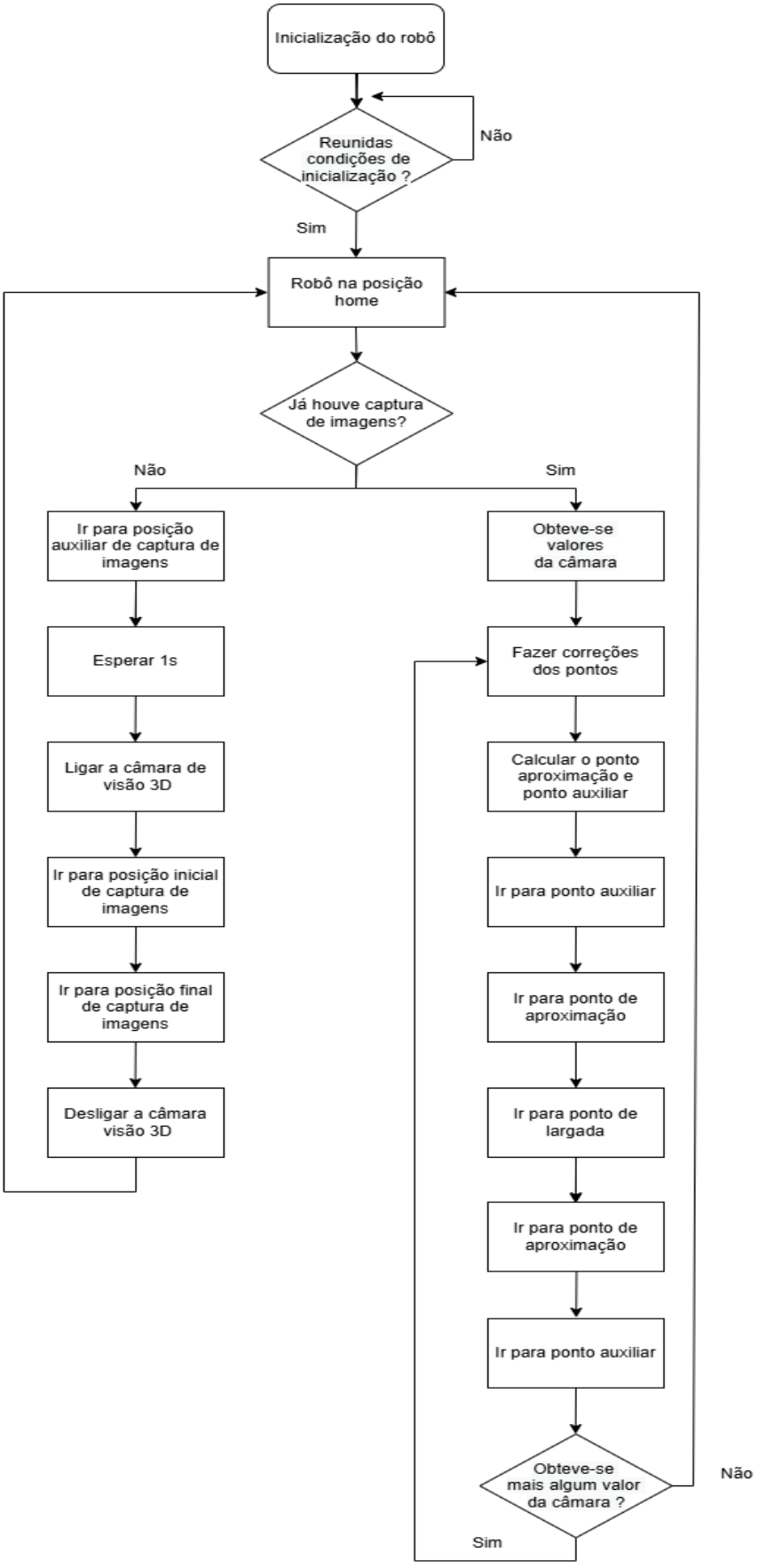


Figura 4.4 Fluxograma do movimento do robô.

Com os movimentos anteriormente explanados, passou-se para a programação do robô em contexto real. Para isso, foi criado um programa para o robô, de acordo com o fluxograma descrito, ou seja, programa esse que vai estar dividido em quatro funções para facilitar a programação e a análise de erros.

A primeira função tem o nome de *MainModule* (figura 4.5), que irá assumir a função principal. Nesta função está declarado a ferramenta (o centro da garra), os pontos auxiliares de largada da peça e o ponto *HOME,* e complementarmente a inicialização de outras funções conforme a ação necessária.

O *MainModule* inicia-se após a verificação das condições de segurança, de imediato segue para a função captura de imagens (GocatorVarrimento). Com a função de captura de imagens terminada, o programa segue para a função de largada de peça (GoToPos).

```
MODULE MainModule
    !defenição das ferramentas
    PERS tooldata ToolPeca:=[TRUE,[[50.4202995300293,-174.951995849609,203.432998657227],[1,0,0,0]],[5,[0,0,20],[1,0,0,0],0,0,0]];
    PERS tooldata ToolAux:=[TRUE,[[-117.648,-6.72186,145.043],[1,0,0,0]],[5,[0,0,0],[1,0,0,0],0,0,0]];
    !declarar variaveis
    VAR Dnum PosXreal;       !variavel posição de X
    VAR Dnum PosYreal;       !variavel posição de Y
    VAR Dnum PosZreal;       !variavel posição de Z
    VAR Dnum PosRXreal;      !variavel rotação em X
    VAR Dnum PosRYreal;      !variavel rotação em Y
    VAR Dnum PosRZreal;      !variavel rotação em Z
    VAR orient aux_orient;   !variavel auxiliar de orientação
    !pontos de referencia
    PERS robtarget PosDeixarPeca:=[[452.102,157.896,-76.911],[0.19744,0.787075,0.441273,0.383156],[0,0,-2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget PosDeixarPeca_ref:=[[458.06,165.66,-45.76],[0.118928,0.776229,0.468278,0.405019],[0,0,-2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget PosAproxDeixarPeca:=[[450.04,168.74,-27.11],[0.556714,0.712719,0.287966,0.314924],[1,1,-3,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS wobjdata RefGocator:=[FALSE,TRUE,"",[[-348.770050048828,665.8369140625,212.241180419922],
        [0.672083,0.740438,-0.00657774,-0.00346431]],[[0,0,0],[1,0,0,0]]];
    PERS loaddata load1:=[5,[0,0,200],[1,0,0,0],0,0,0];
    !função principal
    PROC main()
        !chamada de função de captura de imagem
           GocatorVarrimento;
        !chamada da função de largada de peça
           GoToPos;

    ENDPROC
```

Figura 4.5 Código do principal de acesso às funções do robô.

A função que tem como objetivo fazer a captura de imagem é a GocatorVarrimento (figura 4.6). Nesta função, o robô, vai realizar um movimento de junta até ao ponto auxiliar, após a sua chegada faz o pedido de captura de imagem ao sistema de visão 3D. Este pedido é feito antes de chegar ao ponto de início porque o sistema de visão 3D demora tempo a iniciar a captura de imagem, e no caso, pretende-se ter um sistema otimizado a nível de tempo. De seguida, o robô movimenta-se linearmente para o ponto de início de captura de imagem, e deslocar-se-á até ao ponto de fim de captura de imagem, movendo-se continuamente até ao mesmo. Assim que chegar ao ponto final de captura, a captura de imagem do sistema de visão 3D, fica em pausa e o robô movimenta-se até ao ponto HOME. Nestas deslocações realizadas utilizou-se como eixo de referência a ponta do robô.

```
PROC GocatorVarrimento()
    !movimento para posição auxilar
    MoveJ [[-373.45,375.00,574.64],[0.432505,-0.818332,0.34786,0.149218],[1,1,-2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v300,fine,tool0;
    !espara que acabe o movimento
    WaitTime\InPos,1;
    !liga o sistema de visão
    SetDO DO_LigarCamara,1;
    !movimento para posição de inicio de captura de imagem
    MoveL [[365.71,374.93,574.59],[0.432492,-0.818341,0.34787,0.149184],[0,2,-4,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v50,fine,tool0;
    !movime to para posição de fim de captura de imagem
    MoveL [[-67.21,372.86,574.56],[0.432459,-0.818352,0.347877,0.149202],[1,1,-3,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v300,fine,tool0;
    !desliga o sistema de visão
    SetDO DO_LigarCamara,0;
    WaitTime\InPos,5;
    !movimento para a posição Home
    MoveL [[0,0,0],[0,0,0,0],[0,2,-4,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v3000,fine,tool0;
ENDPROC
```

Figura 4.6 Código do movimento de capturar as imagens por parte do robô.

Quando existe uma captura de imagem, a função principal chama-se GoToPos (figura 4.7), esta tem como objetivo fazer a movimentação do robô para os pontos fornecidos pelo sistema de visão 3D, mas antes de realizar a movimentação, é necessário aplicar uma correção de distâncias, pois há uma divergência entre o centro da garra e o centro do sistema de visão 3D. Essa correção de valores, encontra-se na função ObterValoresCamera, ou seja, é necessário enviar os valores para a mesma.

Quando os valores retornam da função de correção, o robô movimenta-se linearmente para o ponto de aproximação (PosDeixarPeca), e ao chegar ao ponto de aproximação, vai continuar a mover-se continuamente e com uma velocidade reduzida para o ponto auxiliar, que estará um pouco atrás do ponto de largada (neste caso estará 15mm no eixo do Z), de seguida movimenta-se sequencialmente com uma velocidade ainda mais reduzida que a anterior, até ao ponto de largada de peça. Nestas deslocações utilizou-se como eixo de referência a câmara de visão.

Com a largada de peça realizada, o robô volta a fazer os mesmos movimentos, só que com a ordem inversa, ou seja, começa no ponto de largada de peça, passa pelo ponto auxiliar, vai ao ponto de aproximação, até chegar ao ponto *HOME*.

```
PROC GoToPos(
    !envia os valores recebidos do automato para a função ObterValoresCamera
    ObterValoresCamera;
    !espera pelo valores recebidos da função ObterValoresCamera
    ConfL\Off;
    !movimento para a posição  de aproximação de deixar peça
    MoveL PosAproxDeixarPeca,v100,z1,ToolPeca\WObj:=RefGocator;
    !movimento para a posição auxiliar de deixar peça
    MoveL Offs(PosDeixarPeca,0,0,15),v50,z0,ToolPeca\WObj:=RefGocator;
    !movimento para a posição de deixar peça
    MoveL PosDeixarPeca,v20,fine,ToolPeca\WObj:=RefGocator;
    !movimento para a posição auxiliar de deixar peça
    MoveL Offs(PosDeixarPeca,0,0,15),v50,z0,ToolPeca\WObj:=RefGocator;
    !movimento para a posição  de aproximação de deixar peça
    MoveL PosAproxDeixarPeca,v100,z1,ToolPeca\WObj:=RefGocator;
    !movimento para a posição Home
    MoveL [[0,0,0],[0,0,0,0],[0,2,-4,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v3000,fine,tool0;
ENDPROC
```

Figura 4.7 Código do movimento de deixar a peça por parte do robô.

A função denominada ObterValoresCamera (figura 4.9) faz a conversão dos valores obtidos pela captura de imagens, por parte do sistema de visão 3D, que foram recebidas e processadas pelo autómato, e posteriormente enviadas para o robô.

Para que isso aconteça, o robô vai receber os valores individualmente do autómato e dividir por 1000, pois o robô move-se usando como unidade de referência, os quatriliões. Após a divisão dos valores para ficarem de acordo com os do robô, é necessário verificar o sinal do mesmo. Essa verificação é feita com recurso a uma variável auxiliar (SinalPos), que é enviada pelo autómato, informação essa que varia conforme o sinal do valor. Ou seja, se for um valor negativo, o valor desta variável terá de ter o valor de 1, caso contrário terá de ter o valor de 0. Toda esta informação vem à parte do valor, de forma a assegurar o bom funcionamento do processo e que não haja perda de informação.

Com esta verificação, no caso da variável SinalPos aparente o valor 1(um), é necessário multiplicar-se por -1, ao valor que anteriormente foi dividido por 1000, sendo que este processo tem de se realizar para todos os eixos (X, Y, Z) e ângulos de rotação dos eixos (RX, RY, RZ).

O próximo passo, é a construção de uma variável que vai conter toda a informação do ponto para aonde o robô tem de se deslocar. Essa variável vai receber todos os parâmetros individualmente, e vai aplicar um fator de correção dependendo do mesmo. O fator de correção vai ser aplicado, devido a divergência do centro do sistema de visão para o centro da ponta do robô.

Por exemplo, para o eixo X é fundamental a conversão de *doublenum* para *num,* do valor de PosXReal, visto que o robô só consegue ler valores *num*. Após esta conversão, é preciso fazer-se a correção do valor de X, que neste caso é uma subtração de 5 cm, uma vez que o centro do sistema de visão encontra-se deslocado 5 cm do centro da ponta do robô, no eixo X e no sentido negativo (figura 4.8). Após este processo, insere-se os valores finais na variável PosDeixarPeca utilizando a função trans.x, que vai armazenar os valores X na posição correta.

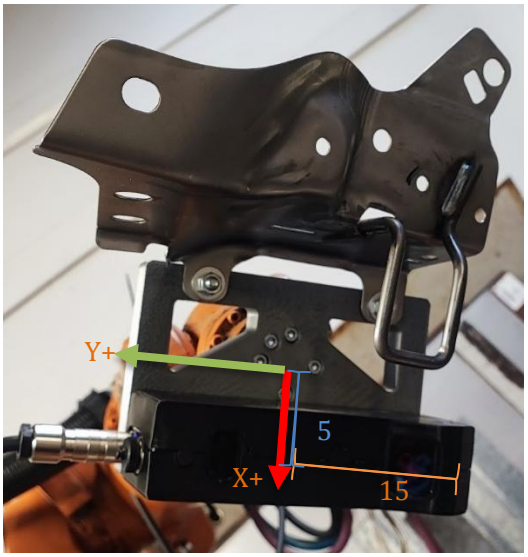


Figura 4.8 Distanciamento do sistema de visão e do centro *end effector* do robô.

No caso do eixo Y, vai primeiro converte-se de *doublenum* para *num* do valor de PosYReal, visto que o robô só consegue ler valores *num*. Depois desta conversão vai fazer-se a correção do valor de Y que neste caso é uma adição 15 cm, uma vez que o centro do sistema de visão encontra-se deslocado 15 cm do centro da ponta do robô, no eixo Y e no sentido positivo (figura 4.8). Após isso insere-se os valores finais na variável PosDeixarPeca utilizando a função trans.y, que vai armazenar os valores Y na posição correta.

Já no caso do eixo z, é necessária a conversão de *doublenum* para *num* do valor de PosYReal, visto que o robô só consegue ler valores *num*. Depois desta conversão não se vai fazer correção do valor 0 (zero), porque se quer manter a altura de referência do sistema de visão (figura 4.8), uma vez que a peça e o sistema de visão têm a mesma altura. Após o processo, insere-se os valores finais na variável PosDeixarPeca utilizando a função trans.z, que vai armazenar os valores Z na posição correta.

Depois dos eixos, foi indispensável relacionar os ângulos e as distâncias aplicadas, de maneira a ser possível a movimentação correta do robô e a sua própria orientação. Para isso foi necessário utilizar a função implementada de Euler. Sendo que no caso em específico da ABB, a ordem de colocação dos ângulos e das distâncias segue a ordem de Z, Y e X.

Assim sendo com este relacionamento de informação dada, consegue-se definir a orientação dos valores de cada ângulo e das distâncias dos pontos de deslocação do robô.

```
PROC ObterValoresCamera()
    !divisão por mil (unidade de referencia do robo)
    PosXreal:=GInputDnum(PosX)/1000.0;      !divisao por da posição X
    PosYreal:=GInputDnum(PosY)/1000.0;      !divisao por da posição Y
    PosZreal:=GInputDnum(PosZ)/1000.0;      !divisao por da posição Z
    PosRXreal:=GInputDnum(PosRx)/1000.0;    !divisao por da rotação X
    PosRYreal:=GInputDnum(PosRy)/1000.0;    !divisao por da rotação Y
    PosRZreal:=GInputDnum(PosRz)/1000.0;    !divisao por da rotação Z

    !verifica o valor do bit X
    IF SinalPosX=1 THEN  !caso  o valor for 1
        PosXreal:=-1*PosXreal;     !mulitplica por -1 o valor da posição X
    ENDIF
    !verifica o valor do bit X
    IF SinalPosY=1 THEN  !caso  o valor for 1
        PosYreal:=-1*PosYreal;     !mulitplica por -1 o valor da posição Y
    ENDIF
    !verifica o valor do bit X
    IF SinalPosZ=1 THEN  !caso  o valor for 1
        PosZreal:=-1*PosZreal;     !mulitplica por -1 o valor da posição Z
    ENDIF
    !verifica o valor do bit X
    IF SinalPosRX=1 THEN  !caso  o valor for 1
        PosRXreal:=-1*PosRXreal;   !mulitplica por -1 o valor da rotação X
    ENDIF
    !verifica o valor do bit X
    IF SinalPosRY=1 THEN  !caso  o valor for 1
        PosRYreal:=-1*PosRYreal;   !mulitplica por -1 o valor da rotação Y
    ENDIF
    !verifica o valor do bit X
    IF SinalPosRZ=1 THEN  !caso  o valor for 1
        PosRZreal:=-1*PosRZreal;   !mulitplica por -1 o valor da rotação Z
    ENDIF

    !atribuir compensação de deslocamento
    PosDeixarPeca.trans.x:=DnumToNum(PosXreal)-5000;
    PosDeixarPeca.trans.y:=DnumToNum(PosYreal)+15000;
    PosDeixarPeca.trans.z:=DnumToNum(PosZreal);

    !atribuir compensação do angulo
    PosDeixarPeca.rot:=OrientZYX(DnumToNum(PosRZreal)+EulerZYX(\Z,PosDeixarPeca_ref.rot),DnumToNum(PosRYreal)+
        EulerZYX(\Y,PosDeixarPeca_ref.rot),DnumToNum(PosRxreal)+EulerZYX(\X,PosDeixarPeca_ref.rot));
    !PosAproxDeixarPeca.trans := PosDeixarPeca.trans;
ENDPROC
```

Figura 4.9 Código do movimento de largada da peça por parte do robô.

Com estas funções disponíveis, foi possível colocar o robô a dar ordens de início e de fim de captura de imagens do sistema de visão 3D, de receber os valores e movimentar o robô para os pontos recebidos.

4.2 Automação

Neste capítulo irá abordar-se tudo o que está envolvido no autómato, ou seja, será explicado o pensamento, prosseguindo para a programação, quais as configurações da programação do autómato para o funcionamento e a programação aplicada no mesmo.

4.2.1 Configurações da programação

A programação do autómato neste projeto vai servir para encaminhar a informação entre o robô e o sistema de visão 3D, para isso a programação tem de ser capaz de realizar a troca de informações, sem que haja perdas (figura 4.10).

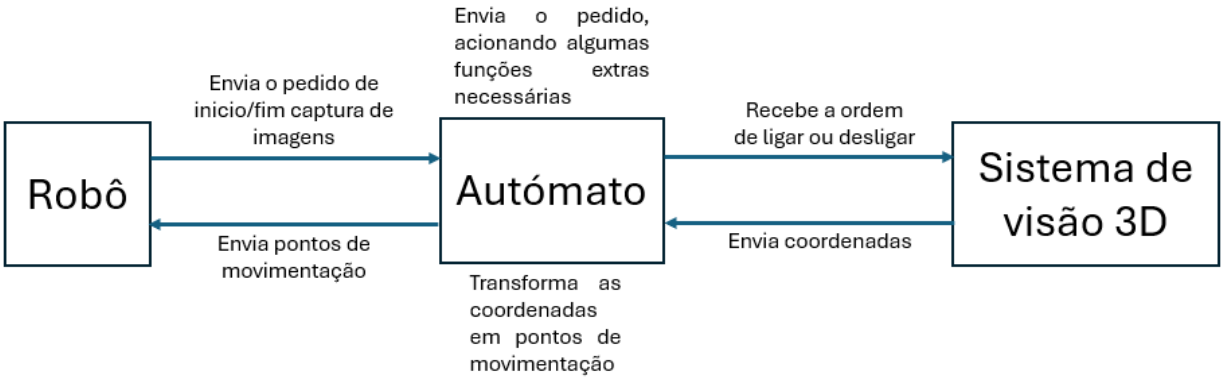


Figura 4.10 Troca de informação do autómato ordem de ligar e desligar.

Para além do autómato ter a capacidade de trocar a informação e encaminhá-la para o local correto, é necessário proceder a ajustamentos da mesma, de forma que os diferentes equipamentos sejam capazes de a interpretar e utilizar.

De acordo com as informações descritas anteriormente, existem duas formas distintas de analisar a programação do autómato, como se pode verificar na figura 4.10:

- Na primeira análise, verifica-se que o robô envia um pedido de início ou fim de captura de imagem, transmitindo essa informação diretamente ao autómato. O autómato, por sua vez, após rececionar a informação dá ordem de ligar ou desligar o sistema de visão;
- Na segunda análise, inicia-se com o sistema de visão 3D a enviar coordenadas dos pontos obtidos através da análise da imagem capturada para o autómato. Este, recebe as coordenadas, transformando-as em ponto de movimentação, que serão posteriormente enviados para o robô. Assim que os pontos de movimentação forem rececionados pelo robô, este irá deslocar-se para esses mesmos pontos.

Concluídas as análises acima descritas sobre as ações e funções do autómato de acordo com comportamento do robô ou do sistema de visão, é dado início à fase de programação do autómato em contexto real.

4.2.2 Programação do autómato

Antes do início da programação, tendo em conta as considerações da secção anterior (4.2.1) desenvolveu-se dois fluxogramas com o intuito de tornar o processo da

programação mais simples e eficiente, sendo que os mesmos se encontram em anexo no presente documento (anexo VIII).

A programação do autómato apresenta-se utilizando a linguagem LADDER, que estará dividida em *network* em vez de linhas. O autómato é representado por dois tipos de variáveis:

- As externas (anexo IX) que mantém interações entre os diferentes equipamentos (robô ou sistema de visão) -autómato; e autómato- equipamentos.
- As internas que servem de auxílio para a programação, ou seja, ajudam a obter resultados para as variáveis externas: autómato-equipamentos (anexo X).

Com as variáveis já definidas, inicia-se a fase do código, que contém todas as instruções. Assim sendo, tem de se iniciar quando estiverem reunidas as condições de segurança e/ou de inicialização. Neste caso será apenas a inicialização das entradas e do programa do sistema de visão 3D (figura 4.11), uma vez que não é necessário juntar as condições de segurança do robô, porque não existe esse objetivo. Há sim, o intuito dos sistemas funcionarem de forma independente, exceto na troca de informação de dados e/ou pedidos.

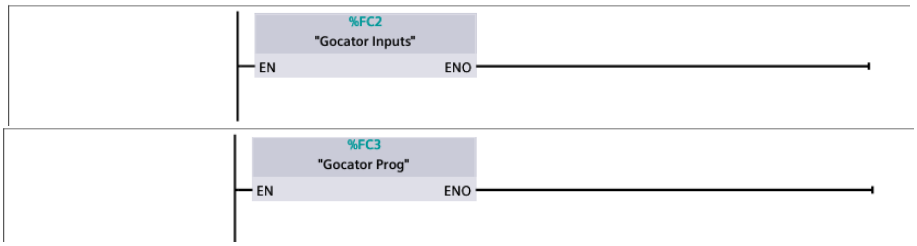


Figura 4.11 Condições de inicialização do autómato.

Após o sistema estar inicializado, seguiu-se para as ações importantes, que neste caso serão a leitura de uma entrada digital, que vem da comunicação do robô que acontecerá quando houver pedido de captura de imagem (iLigarCamara).

Assim que haja esse pedido, acontece duas ações distintas (figura 4.12), a primeira ação dá o *trigger* ao sistema de visão 3D. A segunda ação impede que haja outro equipamento a usar o sistema de visão, e segue a ordem para o sistema de visão ligar, da parte do autómato. Estas duas ações só acontecem quando existir o pedido de captura de imagem da parte do robô.

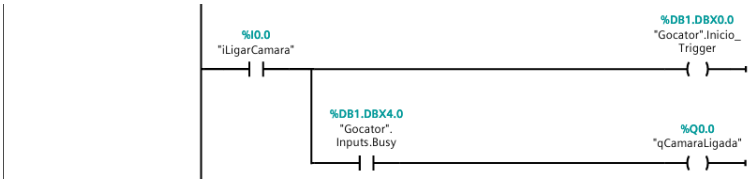


Figura 4.12 Ações a realizar quando houver o pedido de ligar o sistema de visão.

Com o sistema de visão a capturar imagens, procedeu-se ao acionamento do comando de registo, com vista a armazenar e saber o número de *frames*. O comando de registo vai variar entre 0 e 1, conforme o valor de *trigger* (figura 4.13). Isto é, o comando de registo fica com o valor de 1 (um) caso exista o pedido de *trigger,* caso não exista o pedido de

trigger, o valor apresentado é 0 (zero). O comando de registo vai apenas servir como indicador para o sistema de visão.

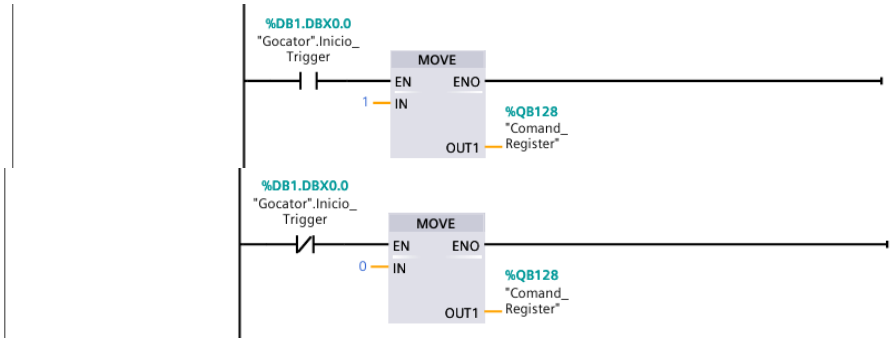


Figura 4.13 Comando de registo e contabilização de *frames*.

Com o comando de registo enviado para o sistema de visão, o autómato passa a receber uma variável com o número de *frames*, cabendo ao autómato contabilizar o número de *frames* que recebeu.

Após a contagem de *frames* e ter sido verificado que é igual a um, o autómato vai rececionar as coordenadas do sistema de visão para variáveis internas que serão utilizadas posteriormente, de modo que o robô seja capaz de se movimentar para a posição pretendida (figura 4.14). Tudo isto é necessário, pois o sistema de visão e o robô trabalham com unidades diferentes um do outro.

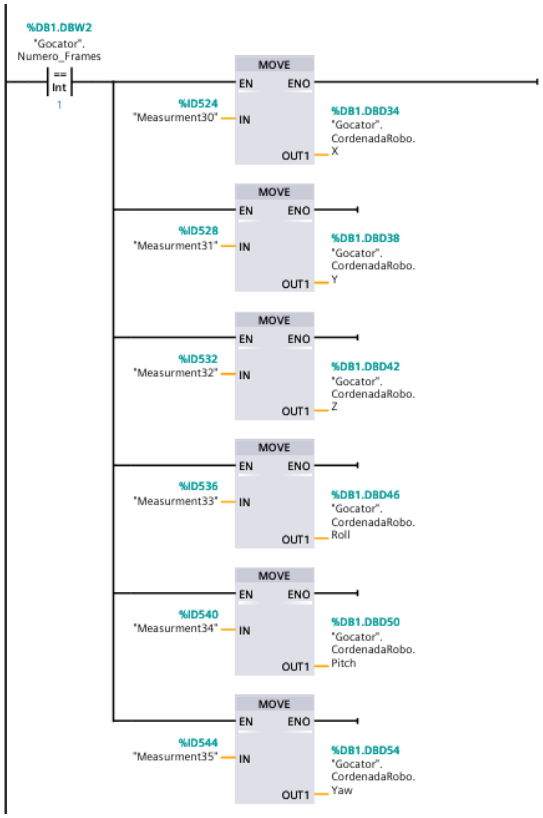


Figura 4.14 Recebe os valores do sistema de visão e envia para o robô.

O sistema de visão pode funcionar de diversas formas, tendo em conta que se pretende utilizar um algoritmo criado no sistema de visão e com ordem de ligar e desligar por parte

do robô. Neste caso em particular, optou-se pelo *Runtime* 3 como descrito na tabela 4.1 (de acordo com as especificações do sistema de visão), com a particularidade de os *bytes* apresentarem uma ordem diferente da que o robô consegue interpretar, devido ao facto do sistema de visão enviar aqueles que serão mais significativos (12-15).

Tabela 4.1 – Configurações do autómato, definição dos valores de algoritmo e dos valores de decisão [40].

Indexador de *byte*	Nome da variável	Descrição
0-3	variável de tempo de execução 0 (Runtime Variable 0)	Guarda o valor pretendido na variável de tempo de indexador 0
4-7	variável de tempo de execução 1 (Runtime Variable 1)	Guarda o valor pretendido na variável de tempo de indexador 1
8-11	variável de tempo de execução 2 (Runtime Variable 2)	Guarda o valor pretendido na variável de tempo de indexador2
12-15	variável de tempo de execução 3 (Runtime Variable 3)	Guarda o valor pretendido na variável de tempo de indexador3

Na programação do autómato, preparou-se este, para os diferentes modos de funcionamento, tendo em consideração a informação anterior, devido à opção de escolha do *Runtime* 3 foi indispensável optar pelo *Measurment* 3, caso a escolha tivesse incidido sobre um valor distinto do aplicado, não se teria obtido valores concretos.

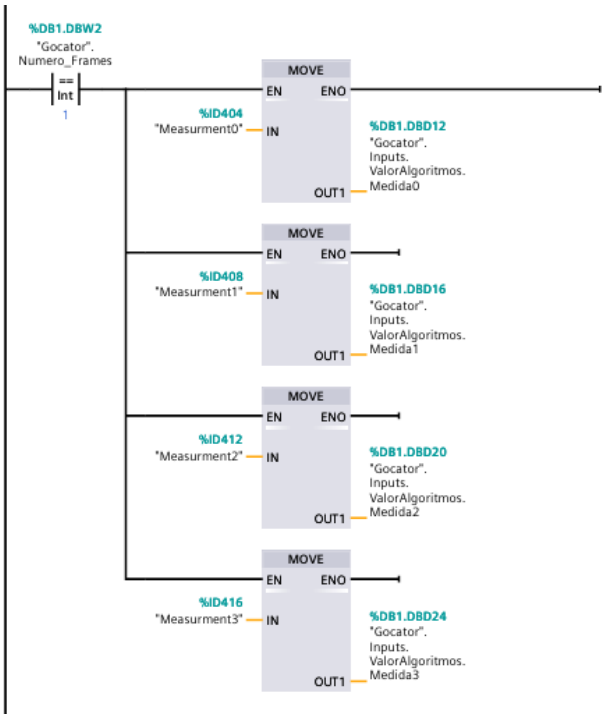


Figura 4.15 Recebe os valores do algoritmo.

Com os valores de algoritmo já definidos, o passo seguinte é definir os valores de decisão (figura 4.16), sendo que este tipo de valores só será utilizado pelo sistema de visão e guardados unicamente no autómato. Neste caso, com o impacto da escolha do *Runtime* 3 (três), pela regra o valor de decisão terá de ser o 3.

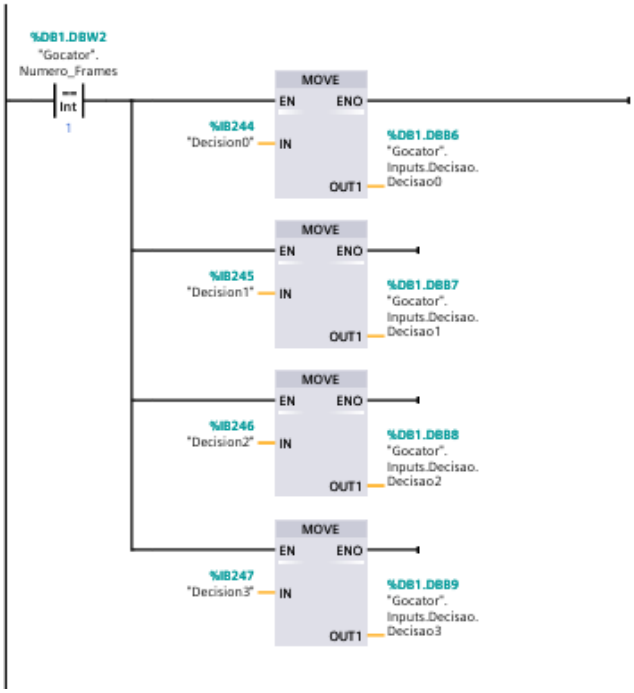


Figura 4.16 Recebe os valores de decisão.

Com todas as informações retiradas da captura de imagens, por parte do sistema de visão, e com o número de *frames* igual a um, fez-se um *reset* à variável *trigger*, pois não existia a necessidade de capturar mais imagens (figura 4.17).

Figura 4.17 *Reset* do *trigger*.

O autómato após receber o valor de X, Y e Z do sistema de visão, vai enviar para a função conversão de valores, para que os valores possam ser convertidos de maneira que o robô consiga fazer uma leitura e interpretação correta. Após o envio dos valores para a função conversão de valores, o programa recebe o *bit* do sinal e o valor (figura 4.18).

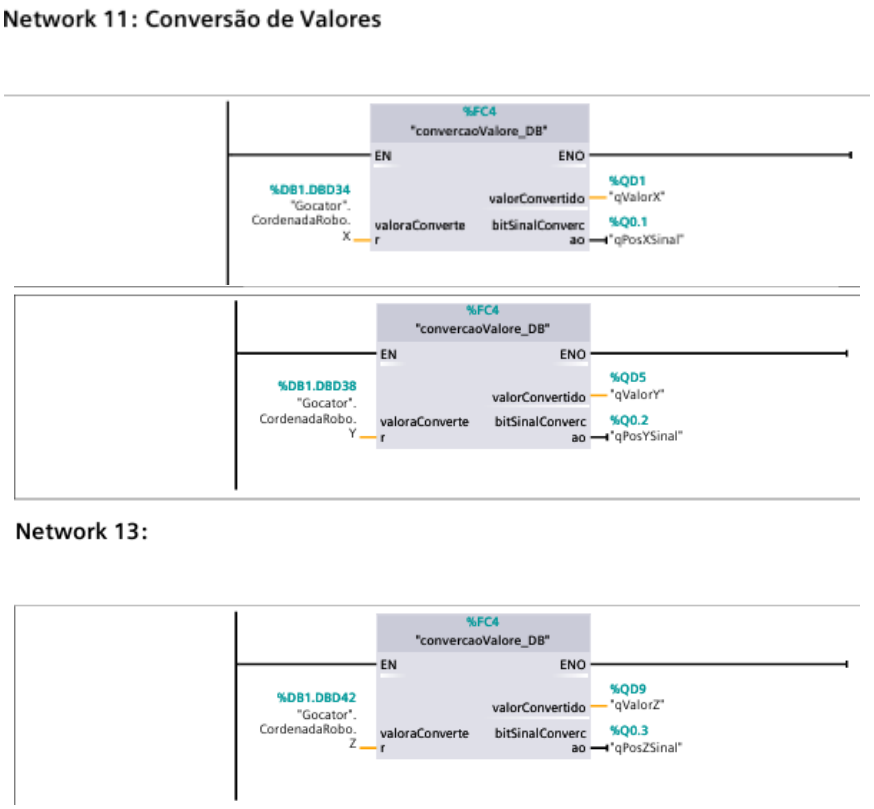


Figura 4.18 Aplicação da função conversão de valores (X, Y e Z).

A função converter valores, como o nome indica, vai realizar a conversão de valores que recebe por parte do sistema de visão, convertendo-os de forma que o robô os consiga interpretar. Quando a função recebe um valor do sistema de visão, verifica o sinal do mesmo, caso o valor tiver o sinal positivo a função continua o programa, ou seja, o *bit* de sinal (figura 4.19) fica com o valor de 0 (zero). Se o valor recebido tiver o sinal negativo, o *bit* de sinal passa a ter o valor de 1 (um), e procede com a sua função. Esta verificação do valor do sinal tem como objetivo a proteção do sinal de valores e a inexistência de constrangimentos ou perdas de informações, uma vez que existe a alteração de valores que permitem uma leitura acertada por parte do robô.

Figura 4.19 Verificação do sinal do valor a converter.

Com o *bit* sinal do valor já adquirido, pretendia-se trabalhar o valor do código principal, e com isso transformá-lo em absoluto (figura 4.20). Para isso, na programação do autómato foi necessário fazer a conversão de valores, uma vez que não era possível fazer diretamente, devido ao tipo de valor que se estava a utilizar. Ou seja, era preciso passar de um valor *Dint*, para um valor Real, com este último valor conseguiu-se obter o valor absoluto. Após a obtenção do valor absoluto, voltou-se a repetir o processo, mas no sentido inverso, de valor Real para um valor *Dint*.

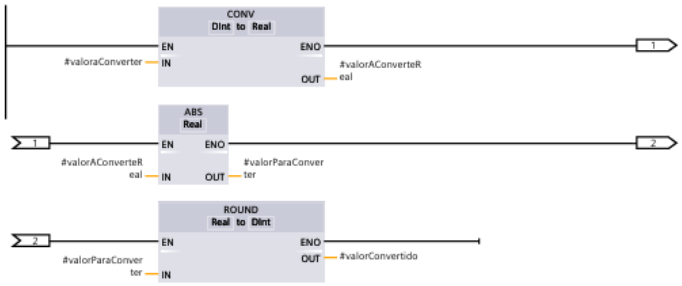


Figura 4.20 Valor absoluto.

Uma vez que o valor de leitura do robô é diferente da forma como o sistema de visão envia, é imprescindível a inversão de ordem dos *bits*. Quer com isto dizer que, o sistema de visão pode enviar o número 1345, mas o robô para receber esse número tem de ser 5431. Para isso, dividiu-se o valor que era composto por 32 *bits,* em duas variáveis (*word*[0] e *word*[1]), sendo que cada uma delas é composta por apenas 16 *bits* (figura 4.21).

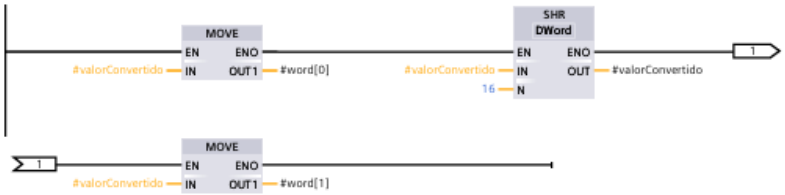
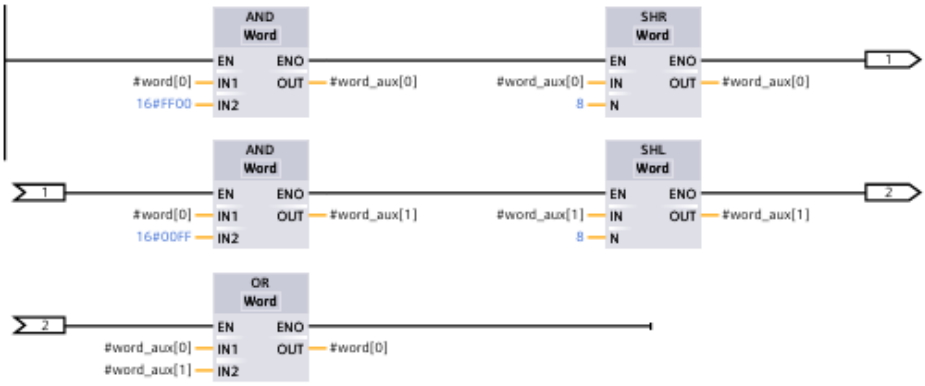


Figura 4.21 Valor recebido decomposto em dois conforme o número de *bits*.

Com o valor dividido em duas variáveis de 16 *bits* cada, ainda era preciso dividir em variáveis com um menor número de *bits*, isto porque ainda não estava no formato que permitisse a leitura correta. Assim sendo, voltou-se a dividir em mais duas variáveis (word_aux[0] e word_aux[1]), sendo que o word_aux[1] fica com os 8 *bits* mais significativos (primeiros 8 *bits*) e o word_aux[0] com os menos significativos (últimos 8 *bits*). Após isso, agregou-se as variáveis numa só com o nome de word[0], sendo que os primeiros 8 *bits* eram os da variável word_aux[0] e os últimos da variável word_aux[1] (figura 4.22).



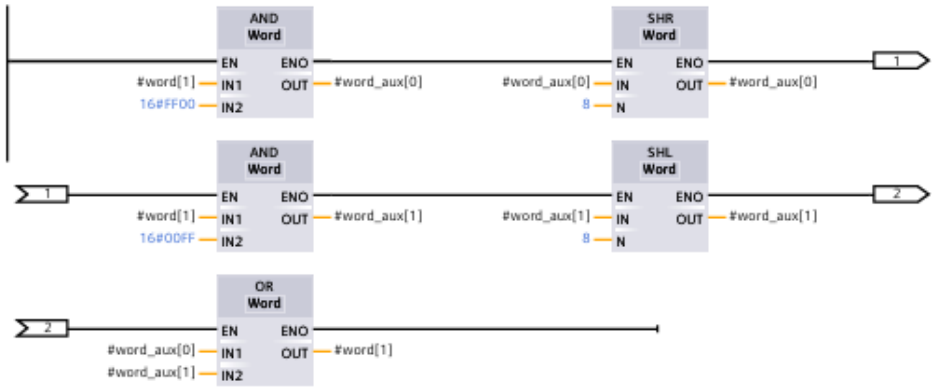


Figura 4.22 Alteração da ordem dos *bites* das variáveis de 16 *bits*.

Com a ordem das variáveis de 16*bits,* faltava apenas juntar essas variáveis para formar uma com 32 *bits*. Assim sendo colocou-se a variável word[1] em primeiro lugar e a variável word[0] em segundo, ou seja, os primeiros *bits* vão ser os correspondentes ao da variável word[0] e os últimos da variável word[1] (figura 4.23).

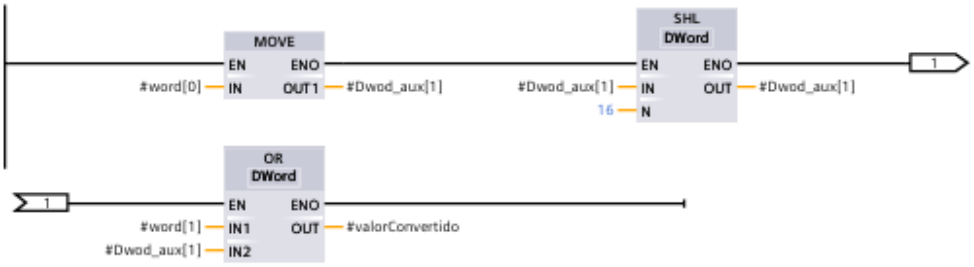
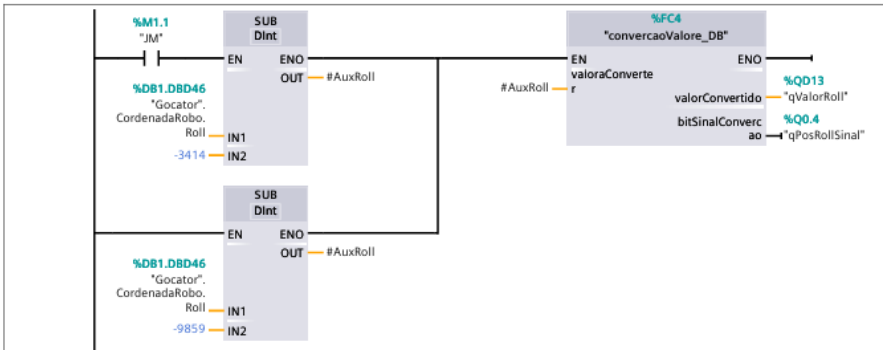


Figura 4.23 Alteração da ordem dos *bits* das variáveis de 32 *bits*.

Com esta alteração, os valores regressam ao código principal, sendo compostos pelo *bit* de sinal e o valor convertido.

No código principal com os valores das coordenadas X, Y e Z já definidos, apenas faltavam os valores de Roll, Pitch e Yaw, que são os valores de rotação angular. Sendo que, neste caso, o autómato recebe os valores e faz uma subtração com os valores de correção, valores esses que são os valores angulares de quando o robô se encontra a fazer a captura de imagem.

Com a aplicação desta correção, o valor vai ser convertido de modo que o robô as consiga ler e interpretar. Assim, com o envio dos valores para a função conversão de valores, o programa recebe o *bit* do sinal e o valor (figura 4.24).



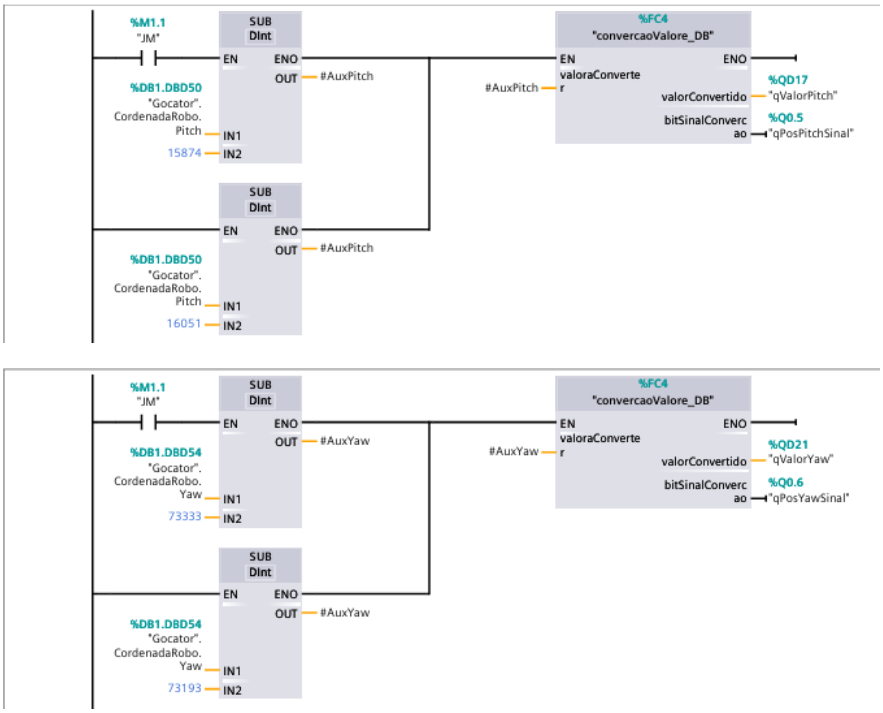
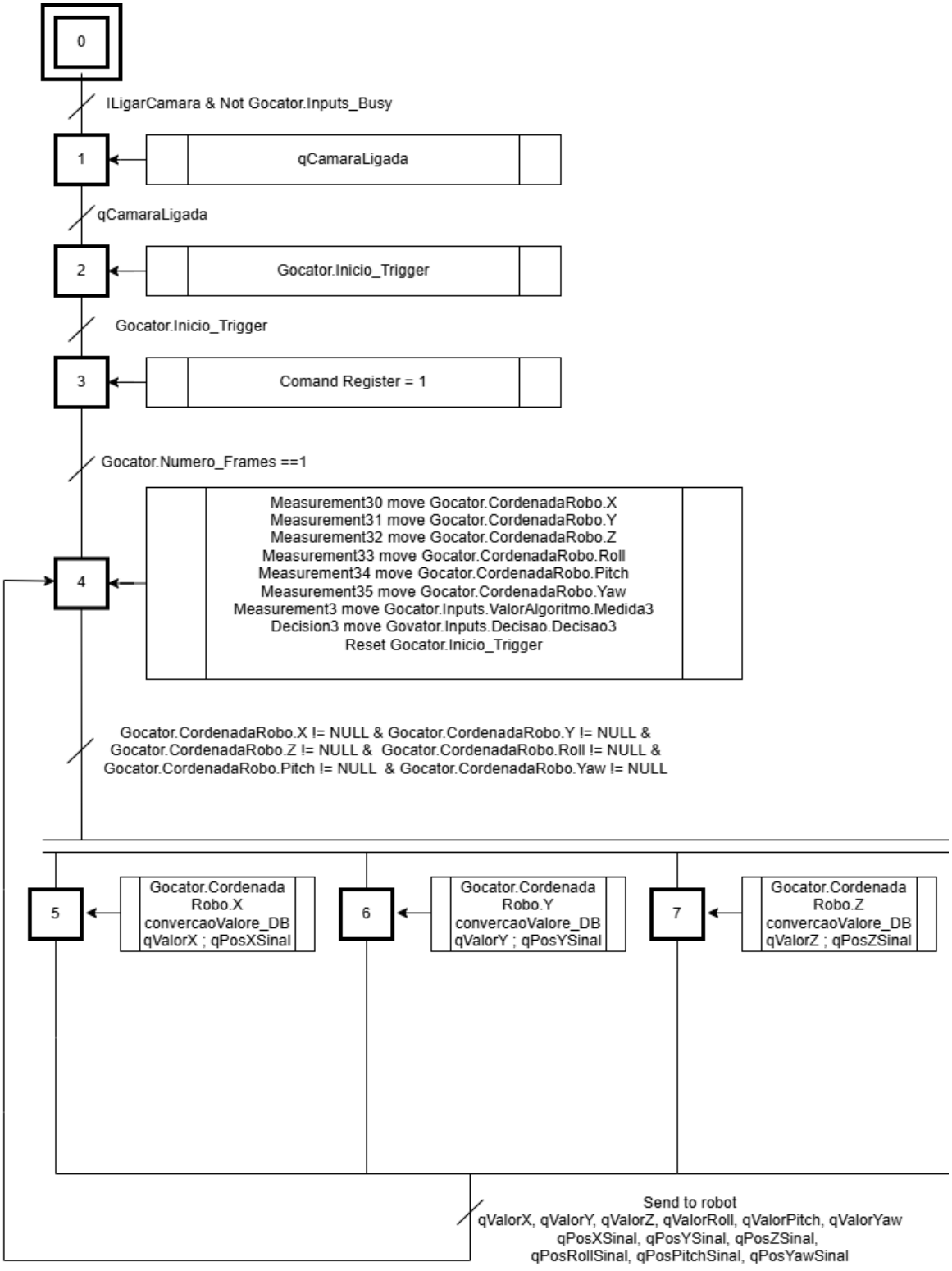


Figura 4.24 Aplicação da função conversão de valores (Roll, Pitch e Yaw).

Após esta conversão de valores, os mesmos são enviados para o robô.

Para a realização desta programação acima descrita foi necessário fazer um *grafcet* (figura 4.25), onde estará toda a informação necessária sobre as variáveis e ações a executar.


0
lLigarCamara & Not Gocator.Inputs_Busy
1
qCamaraLigada
qCamaraLigada
2
Gocator.Inicio_Trigger
Gocator.Inicio_Trigger
3
Comand Register = 1
Gocator.Numero_Frames ==1
4
Measurement30 move Gocator.CordenadaRobo.X
Measurement31 move Gocator.CordenadaRobo.Y
Measurement32 move Gocator.CordenadaRobo.Z
Measurement33 move Gocator.CordenadaRobo.Roll
Measurement34 move Gocator.CordenadaRobo.Pitch
Measurement35 move Gocator.CordenadaRobo.Yaw
Measurement3 move Gocator.Inputs.ValorAlgoritmo.Medida3
Decision3 move Govator.Inputs.Decisao.Decisao3
Reset Gocator.Inicio_Trigger
Gocator.CordenadaRobo.X != NULL & Gocator.CordenadaRobo.Y != NULL &
Gocator.CordenadaRobo.Z != NULL & Gocator.CordenadaRobo.Roll != NULL &
Gocator.CordenadaRobo.Pitch != NULL & Gocator.CordenadaRobo.Yaw != NULL
5
Gocator.Cordenada Robo.X convercaoValore_DB qValorX ; qPosXSinal
6
Gocator.Cordenada Robo.Y convercaoValore_DB qValorY ; qPosYSinal
7
Gocator.Cordenada Robo.Z convercaoValore_DB qValorZ ; qPosZSinal
Send to robot
qValorX, qValorY, qValorZ, qValorRoll, qValorPitch, qValorYaw
qPosXSinal, qPosYSinal, qPosZSinal,
qPosRollSinal, qPosPitchSinal, qPosYawSinal

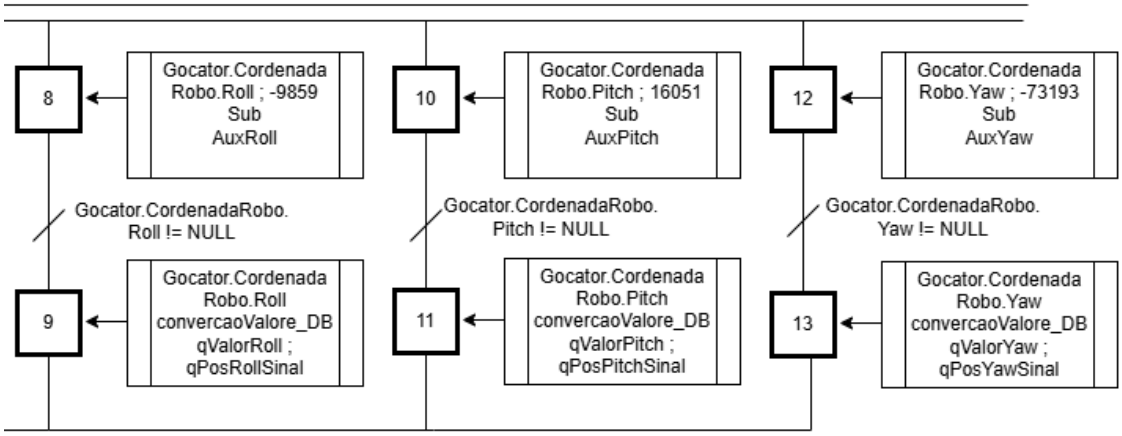


Figura 4.25 *Grafcet* da programação do autómato.

Com o *grafcet* e a programação do autómato realizada, vai mostrar-se a programação do sistema de visão 3D.

4.3 Sistema de visão 3D

Neste presente capítulo será abordado o modo de funcionamento do sistema de visão 3D e o algoritmo de processamento de imagem.

4.3.1 Modo de funcionamento de sistema de visão 3D

O funcionamento do sistema de visão 3D vai ser explicado neste subcapítulo (figura 4.26), para isso o sistema encontra-se dividido em três partes:

- Inicialização do sistema e verificação de existência de captura de novas imagens. Dependendo do resultado da verificação, o programa terá um comportamento diferente;
- Captura de imagens de acordo com o pedido do robô enviado pelo autómato;
- Processamento de imagens faz a análise da captura de imagem, com recurso a um algoritmo de análise da mesma e envia os valores do ponto para o autómato.

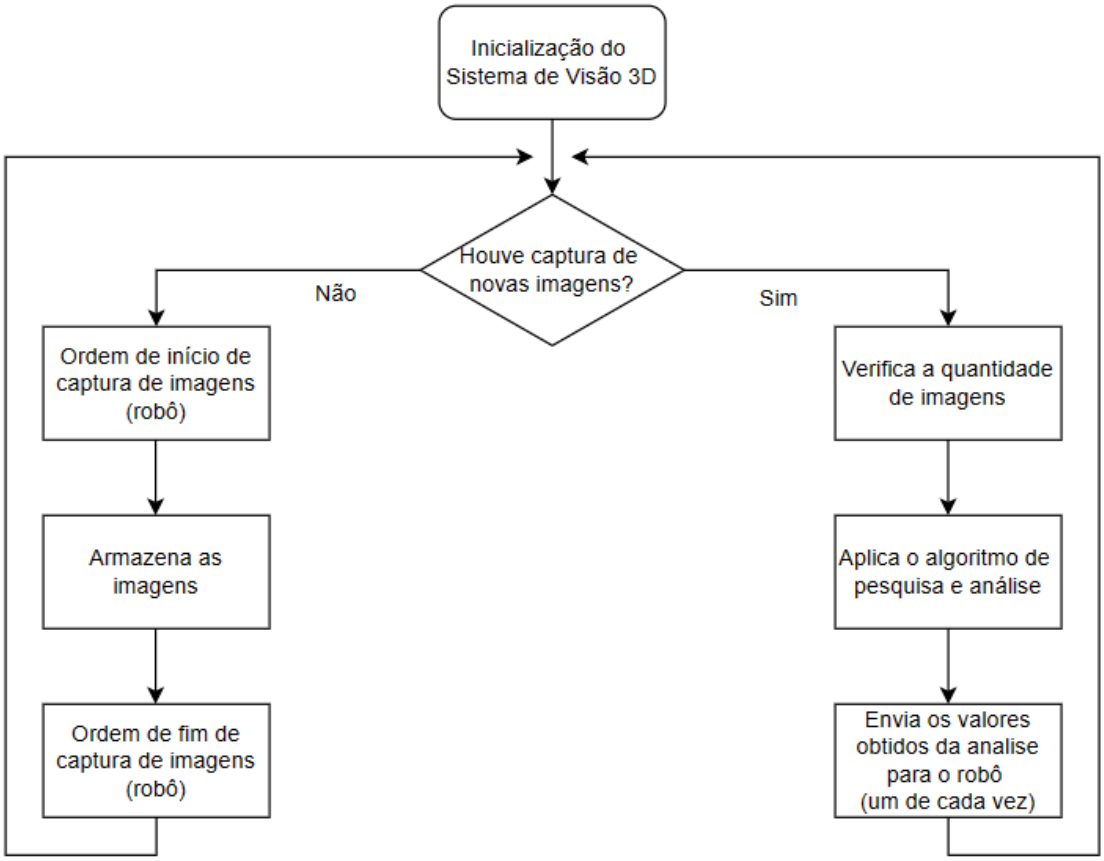


Figura 4.26 Modo de funcionamento do sistema de visão.

O sistema de visão só irá funcionar e iniciar quando estiverem reunidas todas as condições de inicialização, que neste caso são condições implícitas aos outros elementos que compõem a máquina. Todas as vezes que o sistema de visão seja iniciado, terá de capturar uma nova imagem, pois este não tem qualquer memória, e além disso o sistema apenas consegue analisar individualmente cada imagem. No entanto, verifica sempre se o sistema tem alguma imagem, na sua inicialização.

Caso não haja imagem passa-se para a parte da captura de imagem em que, o sistema de visão aguarda a ordem do robô de início de captura e o sistema armazena as imagens. Assim que o robô dá a ordem de fim de captura, o sistema tem de parar de gravar imagens e voltar para a parte de verificação de existência de alguma imagem anterior.

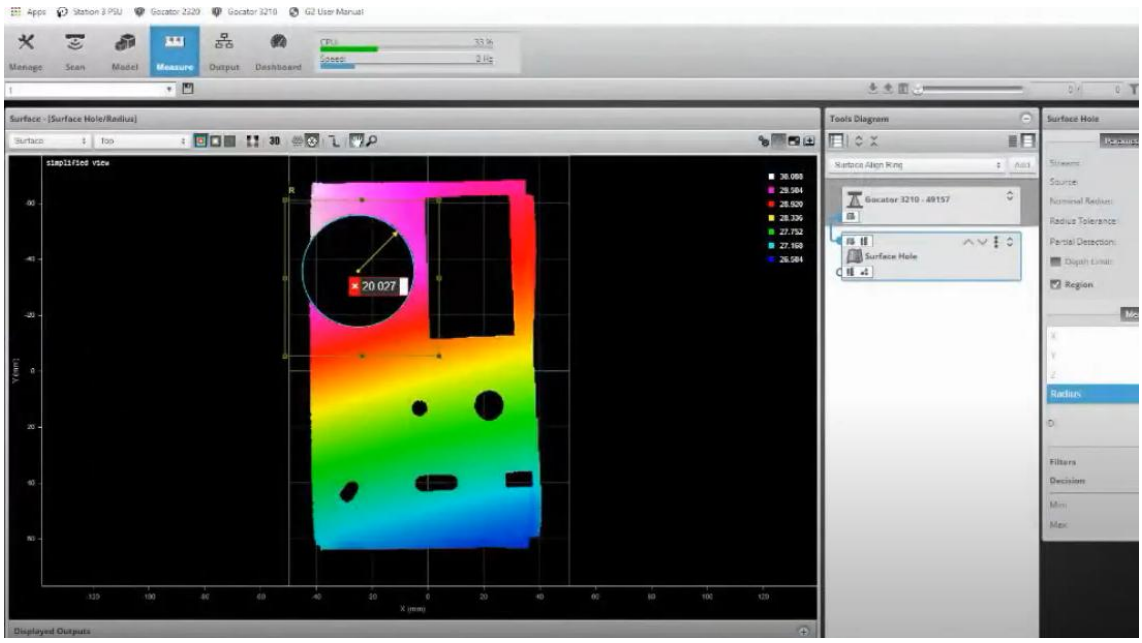


Figura 4.27 Exemplo de uma imagem processada pelo sistema de visão.

Com a captura de imagem, dá-se início ao processamento de imagens. O sistema verifica a quantidade de *frames* e aplica o algoritmo de análise (manual desenvolvido para fazer o algoritmo presente no anexo XI), após o processamento (figura 4.27), o sistema de visão envia os valores para o robô, sabendo que o sistema de visão envia só um valor de cada vez.

4.3.2 Algoritmo de processamento de imagem

O algoritmo de processamento de imagem do sistema de visão 3D vai ser capaz de:

- Corrigir algumas vibrações que possam existir devido ao deslocamento do robô onde se encontra o sistema de visão;
- Detetar os pontos máximos, pontos com maior altura, que serão o local onde o robô terá de colocar a peça no cabide, independentemente da posição do cabide;
- Enviar os valores para o autómato.

Para cumprir estas necessidades do sistema de visão criou-se um fluxograma de forma a organizar as necessidades por ordem de prioridade (figura 4.28).

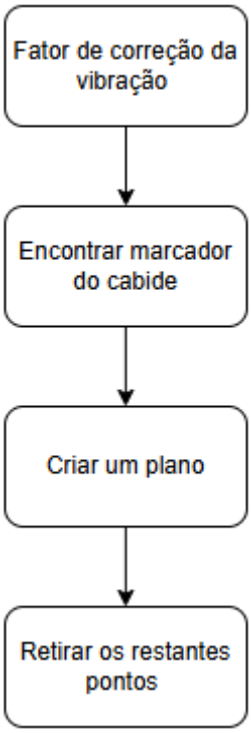


Figura 4.28 Fluxograma da programação do sistema de visão.

A prioridade principal é analisar a imagem e compreender se a mesma está desfocada, para isso escolheu-se o fator de correção de vibração (que varia entre 0 e 100%) como a primeira ação a realizar no algoritmo. Deste modo, garante-se a obtenção de imagens com maior qualidade possível e uma redução significativa do nível de imprecisões. Este fator de correção é definido conforme a vibração do robô, uma vez que para aplicar este fator de 37%, foi necessário fazer diversas captações de imagens e analisar a qualidade da imagem adquirida.

Após essa correção de vibração com os primeiros pontos captados pelo sistema de visão, vai definir-se os pontos que correspondem ao marcador do cabide (figura 4.29). Esse marcador, vai servir como referência para o algoritmo, para entender se houve rotação do cabide ou não. O marcador vai ser constituído por um conjunto de 3 (três) pontos, que por sua vez um deles será considerado o início do plano.

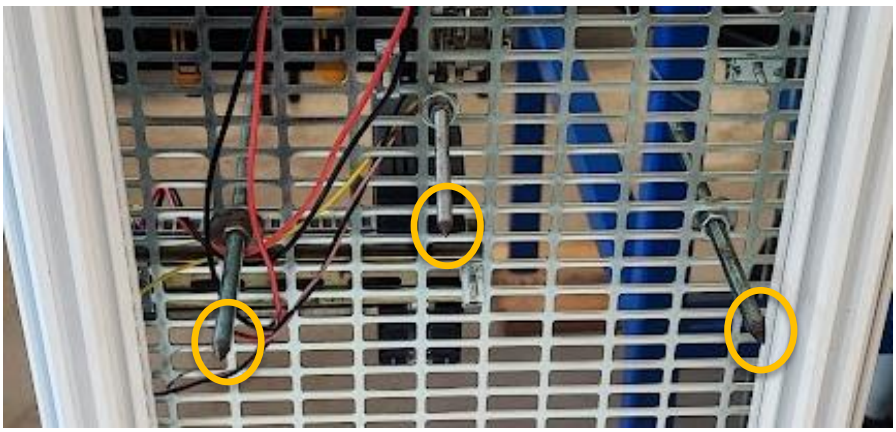


Figura 4.29 Imagem do marcador de início do plano.

O plano terá como função dar as coordenadas dos pontos de largada de peça, sabendo que o robô e o sistema de visão utilizam esses pontos como início do plano.

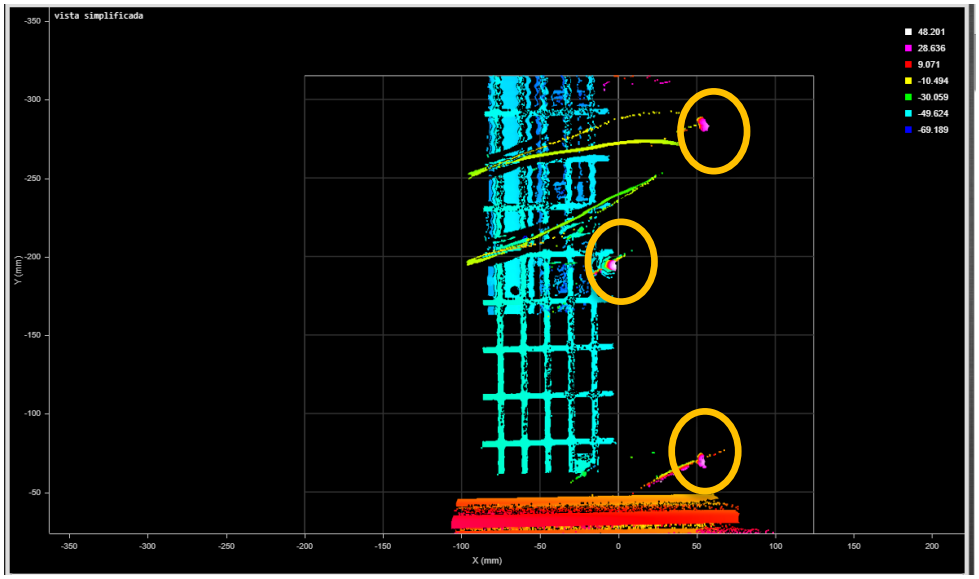


Figura 4.30 Imagem dos pontos para a criação do plano.

Como se pode observar na figura 4.30, os 3 pontos para a criação do plano, definidos com círculos amarelos centrados nos pontos roxos, pretendeu-se criar um plano onde a origem do referencial é um ponto que se encontra fora do cabide, mas que tem a função de marcador. Esse marcador vai servir para referência, cada vez que aparecer um novo cabide na linha de montagem.

Com o plano criado, passou-se a uma análise da associação de pontos, de imagens capturadas e para a verificação dos pontos acima do plano, observando apenas os pontos onde existe variação do valor, no eixo do Z (figura 4.31).

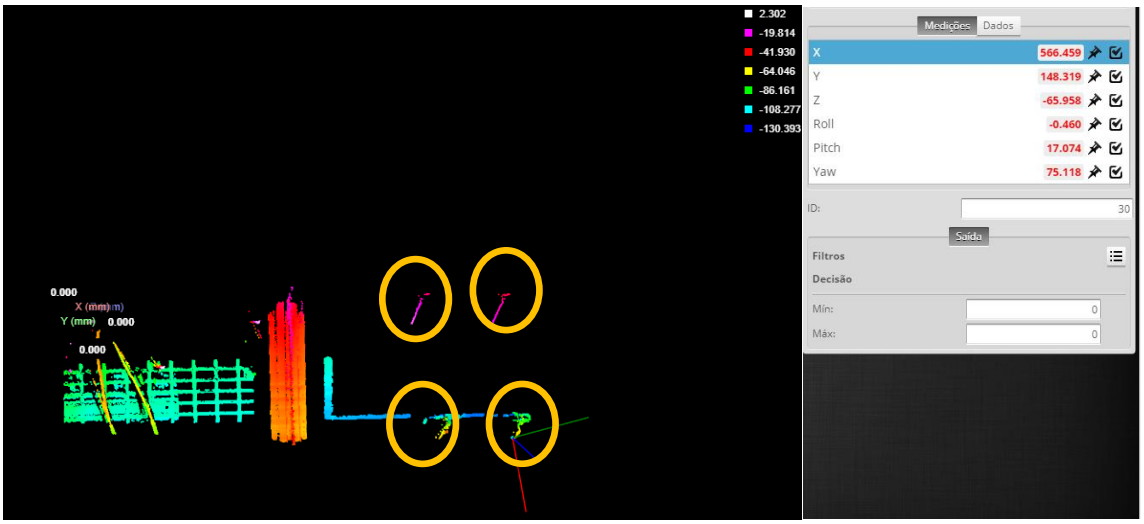


Figura 4.31 Imagem dos pontos para a criação do plano.

Se houver pontos ou parte da imagem abaixo do plano, o sistema de visão descarta essa informação. Se houver pontos acima do plano, o sistema de visão tenta procurar o ponto com uma altura maior, como se pode verificar na figura 4.31. Sendo que o sistema de visão

cria um ponto no local onde o eixo do Z tem o valor de coordenadas mais alto, ou seja, no local de entrega de peça (representados com círculos amarelos na figura 4.31).

Como o algoritmo de análise de imagem realizado, inicia-se a parte de teste em contexto real, para isso simulou-se o ambiente, e apurou-se o resultado da captura do sistema de visão 3D (figura 4.32). Nessa captura, observou-se os pontos que servirão de base para a criação do plano e os pontos de largada da peça.

Sendo que os pontos de largada da peça estão mais perto do sistema de visão, em relação aos outros pontos ou partes do ambiente que o sistema de visão possa capturar.

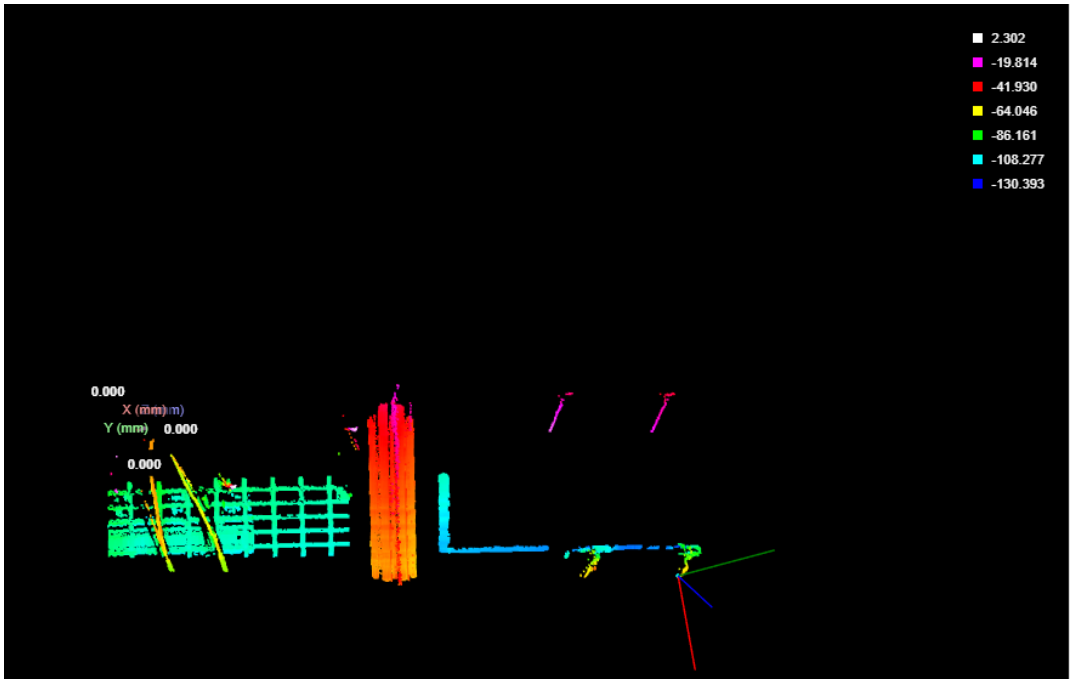


Figura 4.32 Imagem processada pelo sistema de visão.

Depois de serem localizados os pontos de largada de peça, o sistema de visão envia os dados para o autómato de uma forma automática, enviando prontamente os mesmos para o robô, fazendo-o deslocar-se até aos pontos definidos.

5 CONCLUSÃO

Para a realização do presente relatório, foi escolhido o projeto mais relevante no âmbito do estágio curricular, do mestrado em engenharia eletrotécnica, com o intuito de demonstrar o trabalho desenvolvido nas áreas de automação, robótica e visão 3D.

5.1 Conclusões do projeto

O projeto consiste num sistema de visão 3D, seguro a uma garra do robô, sendo que para a realização deste projeto em particular foi necessário um equipamento robusto, para adequar o ambiente envolvente e também as condições impostas.

Com o equipamento certo, foi imprescindível fazer um estudo de como seriam os movimentos do robô. De forma a tornar o sucesso do projeto mais certo, procedeu-se a simulações, para entender qual o melhor sentido de movimentação do robô para que existisse uma boa captura de imagem, tornando-o mais eficiente

Na parte da robótica, apesar do robô escolhido não ter sido o mais indicado, foram cumpridos os objetivos iniciais, sendo que um desses objetivos era o robô conseguir chegar pelo menos a dois pontos de largadas da peça. Para isso foi necessário criar uma divisão no programa que apresenta duas fases distintas: a captura de imagem, onde o robô terá de fazer uma rotação e de seguida uma deslocação até ao ponto inicial e final; e a largada de peça, onde o robô recebe os dados do sistema de visão. Após a receção dos dados, este movimentasse até ao ponto auxiliar, seguindo para o ponto de aproximação e finalmente ao ponto de largada da peça. Terminado todos estes processos, o robô retorna à posição inicial, passando por todos os outros pontos, no sentido inverso.

No sistema de visão foi possível perceber que sem o fator de correção de imagem, não se conseguia obter uma imagem de captura com qualidade, sendo que para isso foi necessário fazer vários testes para ajustar o fator de acordo com a vibração do robô.

Para a captura de imagens e obtenção de pontos, foi necessário aplicar-se um algoritmo de visão, exigindo um entendimento na área de programação e qual o seu impacto na obtenção de resultados. Assim sendo, teve de se realizar um estudo sobre o *software* e as respetivas funções, bem como houve a necessidade de relembrar como se faz um plano para obtenção de uma referência dos pontos de largada.

Neste sentido, e após todos os processos necessários terem sido cumpridos, a realização do projeto foi concluída com sucesso, possibilitando o êxito todos os objetivos definidos no início do estágio.

5.2 Melhorias a realizar no projeto

O projeto apesar de ser um projeto modelo, que tinha como objetivo fazer um estudo em contexto real do sistema de visão 3D, aplicado diretamente na garra do robô, constatou-se que foi concluído com sucesso, mesmo ter sido apresentado como um projeto modelo.

Após a conclusão dos testes, a implementação do projeto em contexto real, terá de ter melhorias em alguns aspetos, ou até mesmo, adicionar equipamento(s) extra(s). Algumas das melhorias sugeridas, são as seguintes:

- Substituição do robô – apesar de o robô cumprir o objetivo pretendido, caso fosse aplicado num contexto fabril teria de ser escolhido um com maior alcance, para chegar à ponta do cabide;
- Programação – tornar a programação capaz de trabalhar em modo automático ou manual, ou seja, quando escolhido o modo manual, o programador conseguir acionar um equipamento de cada vez, ou só o sistema de visão ou só o robô;
- Encontrar uma solução alternativa para a garra – a garra teria de ser pensada para conseguir agarrar a peça à mesma altura do sistema de visão, para não interferir na captura de imagem;
- Colocação de um interface homem-máquina – com esta interface, pretende-se que um operador consiga monitorizar o estado de funcionamento da máquina, sendo que a qualquer momento, poderá ver o funcionamento, ou observar algum alarme;
- Implementar botões (automático/manual, início, fim e de emergência) e sinalizadores visuais de funcionamento – uma das funções que não foram implementados, neste projeto, está relacionado com a criação de botoneiras de funcionamento e sinalizadores visuais, pois são mecanismos primordiais de segurança e de regulação do funcionamento da máquina, em contexto real;
- Barreiras de segurança – as barreiras, sendo um mecanismo de proteção direta, de pessoas e bens, tendo como função evitar acidentes, e seria um ponto crucial a acrescentar.

REFERÊNCIAS

[1] D. d. P. Robótica, Porto Editora, 2024.

[2] Robotnik, *History of robots and robotics*, https://robotnik.eu/history-of-robots-and-robotics/, 2021.

[3] A. Misiti, *A History of Industrial Robots*, https://www.wevolver.com/article/a-history-of-industrial-robots, 2020.

[4] L. S. A. Gasparetto, *A Brief History of Industrial Robotics in the 20th Century*, https://www.scirp.org/journal/paperinformation?paperid=90517, 2019.

[5] IEEE, *Techincal Commitee for Humanoid Robotics*, https://www.ieee-ras.org/humanoid-robotics.

[6] R. D. Right, *History of the Industrial Robot*, https://robotsdoneright.com/Articles/history-of-the-industrial-robot.html, 2024.

[7] C. B. Silveira, Os 6 Principais Tipos de Robôs Industriais, https://www.citisystems.com.br/tipos-de-robos/, 2019.

[8] RobotsDoneRight, Yaskawa Motoman NX100 Controller, https://robotsdoneright.com/Motoman/Yaskawa-Motoman-NX100-controller.html.

[9] Indusprime, As 6 Melhores Marcas dos Teach Pendants e seus Robôs Industriais, https://www.indusprime.com.br/teach-pendant-robos-industriais-6-marcas.

[10] SEESistemas, Robôs industriais: Quais são os tipos que existem?, https://www.seesistemas.com.br/robos-industriais.

[11] SICK, *Understanding Safety Mats vs. Safety Laser Scanners*, https://sickconnect.com/nowsmaller-smarter-safety-laser-scanners/, 2014.

[12] Eltra-trade, Sick S30A-6011DA, https://eltra-trade.com/products/sick-s30a-6011da.

[13] HES, *Leuze electronic's Safety Laser Scanner with added "Motion*, https://www.hazeng.com/editorial/health-and-safety/leuze-electronics-safety-laser-scanner-with-added-motion.

[14] CalcadaeAmorim, Botão Botoneira de Emergencia Telemecanique, https://calcadaeamorim.com/produtos-categoria/material-eletrico/automacao-e-comando/a-comando-e-sinalizacao/.

[15] J. Ferreira, Sistemas Robóticos, ISEC, 2023.

[16] MFGRobots, Tipos De Juntas De Robôs:Um Guia Detalhado, https://pt.mfgrobots.com/equipment/robot/1004016491.html.

[17] N. Henrique, *Understanding Industrial Robot Motion Types,* https://www.solisplc.com/tutorials/industrial-robot-motion-types.

[18] Robots.com, *Industrial Robot History*, https://www.robots.com/articles/industrial-robot-history, 2017.

[19] Interplast, Estratégias de segurança para cobots: aplicações de colaboração seguras e eficazes, https://interplast.pt/Artigos/303187-Estrategias-de-seguranca-para-os-cobots-aplicacoes-de-colaboracao-seguras-e-eficazes.html, 2020.

[20] Dicionario, Automação, Porto Editora, 2024.

[21] N. Bong, *The Evolution of Automation*, https://www.progressiveautomations.com/blogs/news/the-evolution-of-automation, 2022.

[22] G. Santos, O que é Automação Industrial?, https://www.automacaoindustrial.info/o-que-e-automacao-industrial/, 2017.

[23] D. A.Al-Hatab, *MT308 Industrila Automation*, https://pt.slideshare.net/drkhali_l2012/module-1-lecture-1-introduction-to-automation-in-production-systemsppt.

[24] V. Muthukrishnan, *Programmable Logic Controllers (PLCs): Basics, Types & Applications*, https://www.electrical4u.com/programmable-logic-controllers/?utm_content=cmp-true, 2023.

[25] J. Faylor, *PLC Programming Languages: Go Beyond Ladder Logic,* https://inductiveautomation.com/blog/plc-programming-languages-go-beyond-ladder-logic, 2022.

[26] V. Romanov, *PLC Programming Language Specifications*, https://www.solisplc.com/blog/plc-programming-languages.

[27] L. Cope, *PLC Programming Languages: A Comparative Guide*, https://engineerfix.com/electrical/plc/plc-programming-languages-a-comparative-guide/?utm_content=cmp-true, 2023.

[28] Ø. Borgan, *A beginner's guide to 3D machine vision cameras,* https://medium.com/zivid/a-beginners-guide-to-3d-machine-vision-cameras-1bb116ab98a9, 2020.

[29] S. Imaging, Inspeção de pneus com visão 3D, https://pt.stemmer-imaging.com/blog-pt-pt/visao-3d-conceitos-basicos-2/.

[30] H. T. Aguiar, Desenvolvimento de um Sistema de Paletização, Instituto Politécnico de Viseu: https://repositorio.ipv.pt/bitstream/10400.19/2045/1/TESE_Helder-10-9-2013.pdf, 2023.

[31] T. C. d. S. Azevedo, Reconstrução e Caracterização de Estruturas Anatómicas Exteriores usando Visão Activa, Universidade do Porto: https://web.fe.up.pt/~tavares/downloads/publications/teses/MSc_Teresa_Azevedo.PDF, 2002.

[32] V. K. R. K. Abhilasha Sing, *A survey on vision guided robotic systems with intelligent control strategies for autonomous tasks*, https://www.tandfonline.com/doi/full/10.1080/23311916.2022.2050020#d1e207, 2022.

\[33] Y. Cong, R. Chen, B. Ma, H. Liu, D. Hou e C. Yang, *A Comprehensive Study of 3-D Vision-Based Robot Manipulation,* https://ieeexplore.ieee.org/document/9541299, 2023.

[34] M. S. a. M. H. A. Jr., *3D Vision-Based Control On An Industrial Robot,* National University of Singapore, 2001.

[35] ABB, Product specification IRB 1100, 2023.

[36] ABBRobotics, IRB 1100 Product Management, 2018.

[37] N. Ayllin e H. Hunter, *How PROFINET Works: A Beginner's Guide,* https://us.profinet.com/wp-content/uploads/2020/10/How-PROFINET-Works-Complete.pdf.

[38] Siemens, Data sheet 6ES7315-2AH14-0AB0, https://docs.rs-online.com/ada7/0900766b8172fcef.pdf, 2019.

[39] Basler, LMI Gocator 2100 - LMI Sensor Gocator 2140, https://www.baslerweb.com/ko-kr/shop/lmi-sensor-gocator-2140/.

[40] sbindustrialsupply, NEW SICK V3T12S-MR32A8 TRISPECTOR 1060427 V3T12SMR32A7, https://sbindustrialsupply.com/shop/new-sick-v3t12s-mr32a8-trispector-1060427-v3t12smr32a8/.

[41] A. Benfica, O que é um Switch?, https://www.palpitedigital.com/o-que-e-um-switch/.

[42] Worten, Switch TP-LINK TL-SF1005D (5 Portas Fast Ethernet - 100 Mbps), https://www.worten.pt/produtos/switch-tp-link-tl-sf1005d-5-portas-fast-ethernet-100-mbps-4342195.

[43] KUKA, Technical Data KR6, 2003.

ANEXO I – PROJETO INTERFACE HOMEM-MÁQUINA

Além do projeto anteriormente apresentado, este relatório vai ser composto por mais um projeto, sendo que o tema abordado será o interface homem-máquina executado durante o período de estágio.

Este projeto consistiu numa máquina que tinha como principal objetivo, agarrar uma rolha, colocá-la numa garrafa e, por fim, embalar a garrafa. Após o processo de embalamento da garrafa, esta teria de ser colocada numa caixa. Para a realização deste processo, a máquina foi dividida em:

- Parte de entrada – entrada das garrafas na máquina no seguimento de uma linha de produção;
- Enrolhamento – processo de colocação da tampa na garrafa;
- Ensacamento– processo que consiste na colocação da garrafa dentro de um saco, podendo este, ser com ou sem soldadura (saco selado);
- Transportador de saída – quando existe um número predefinido de garrafas, agrupa-se e transfere-se para a saída da máquina;
- Saída – colocação das garrafas prontas, de forma que o robô as consiga agarrar para ser realizado o transporte para a caixa;
- Robô – controle da garra do robô;
- Tapetes - controlo de velocidades de alguns tapetes da máquina.

Com a descrição da máquina realizada, passou-se para a interface, que foi realizada de modo que qualquer operador pudesse trabalhar com a máquina. Assim sendo, a interface teria de possuir as seguintes características:

- Menu principal – menu selecionado para a escolha das ações em cada modo, observação de alarmes caso estes existam, e realização de alterações nas receitas predefinidas caso seja administrador;
- Menu manual – menu para a realização de alterações manualmente de cada parte da máquina;
- Menu automático – menu predefinido para as alterações de receita, em modo automático e habilitar ou desabilitar algumas partes da máquina;
- Alarme –visualização de alarmes em caso de existência dos mesmos;
- Receitas – as configurações que a máquina vai ter caso seja alterado o tipo de garrafa a utilizar.

O menu principal (figura I.1), conta com a identificação da máquina e das empresas quer a proprietária quer da empresa que construiu a máquina, com a identificação do dia e da hora, com a apresentação de alarme caso haja e com o menu manual, automático, alarmes e receitas.

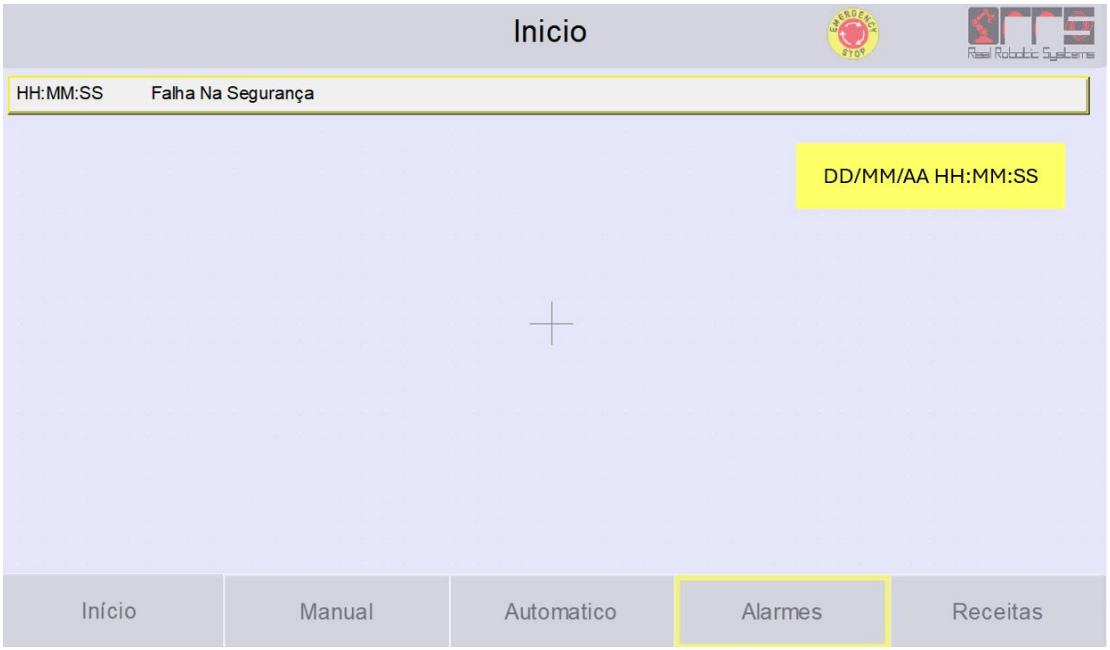


Figura I.1 Menu Principal.

Menu Manual

O menu Manual (figura I.2) está dividido de acordo com a caracterização da máquina realizada no princípio deste capítulo, assim sendo encontra-se dividida em estação de entrada, estação de enroscar, estação de ensacar, transportador de saída, estação saída, robô e tapetes. Ainda que dentro destes submenus ainda se possa dividir em mais menus, de forma a especificar e poder trabalhar em caso de necessidade de reparações.

Para o menu Manual funcionar o seletor da botoneira da máquina terá de estar colocado no modo Manual, existindo assim algumas ações que possam não funcionar, uma vez que a máquina está protegida com encravamentos mecânicos e com encravamentos no autómato, que não permitem certas ações caso as condições ambientais não estejam reunidas, de forma que não exista nenhuma colisão e subsequentemente material partido ou empenado.

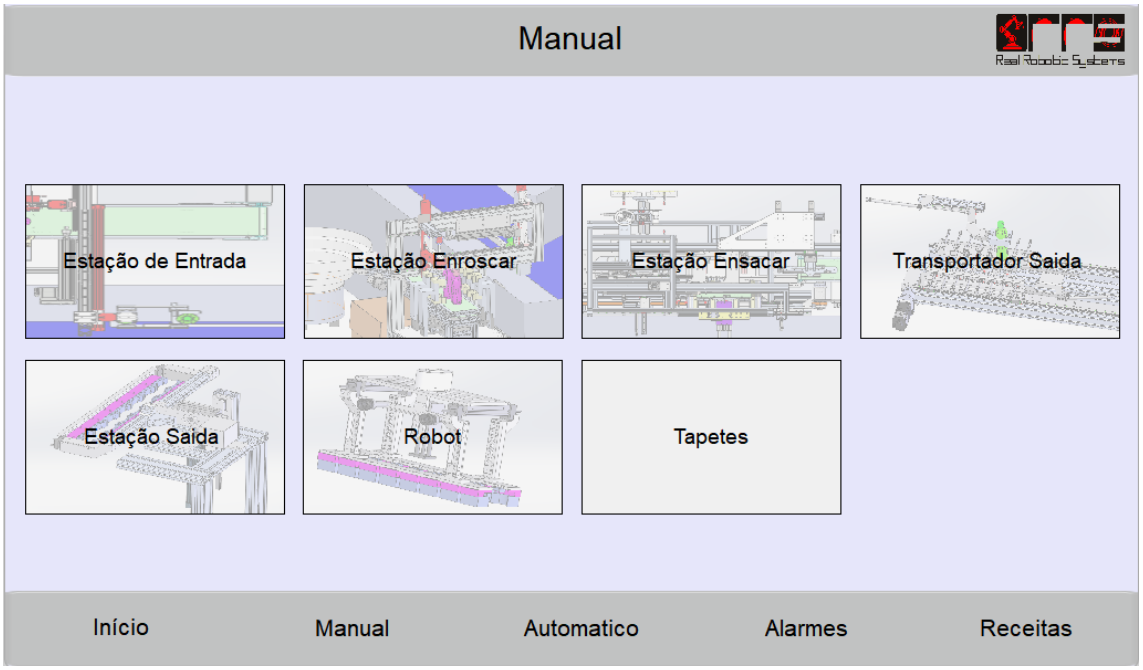


Figura I.2 Menu Manual.

Para além da divisão feita dentro deste menu, devido a extensão da máquina e a complexidade da mesa, em alguns casos dividiu-se em mais menus, para que se tornasse o menu mais fácil de interpretar e localizar a parte da máquina que se pretende testar ou localizar.

No menu de estação de entrada (figura I.3) dividiu-se em submenus:

- gota a gota – com este menu pretendia-se visualizar e modificar todos acionamentos relativos a separação das garrafas que vinham da linha de produção;
- transporte entrada máquina – aqui pretendia-se visualizar e modificar os acionamentos responsáveis por agarrar as garrafas separadas que vinham da linha de produção (parte gota a gota) e colocá-las no tapete de entrada da máquina.

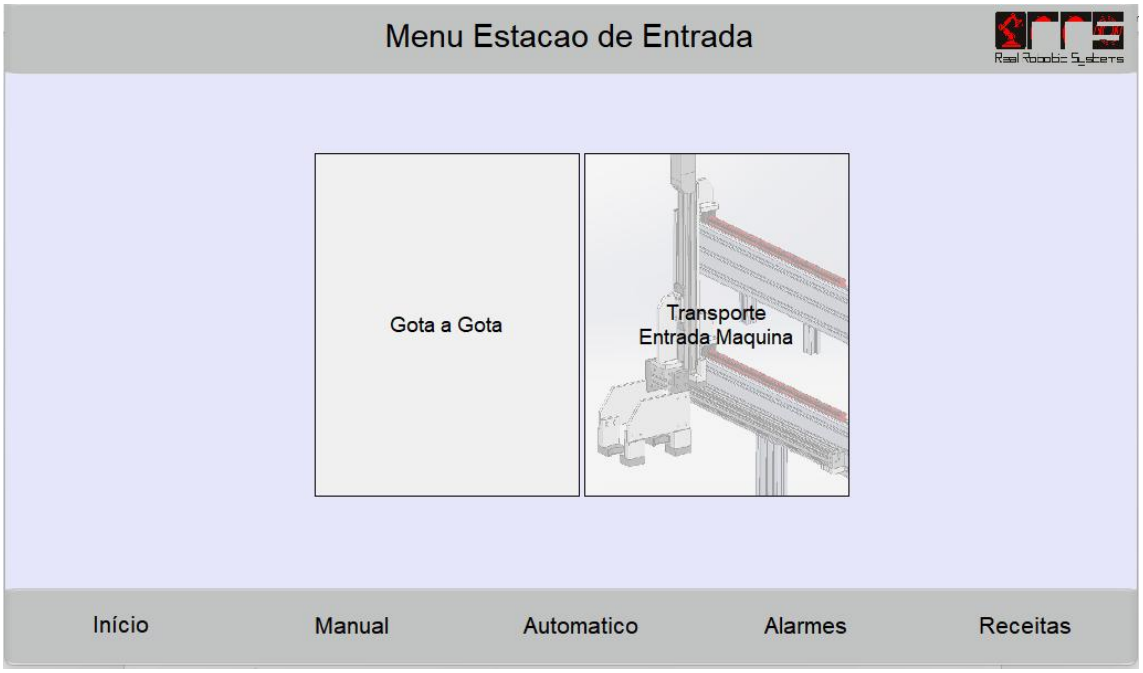


Figura I.3 Menu Estação de Entrada.

O menu de enroscar (figura I.4) está dividido em fornecimentos de tampas e roscagem, porque considerou-se que eram duas ações diferentes, assim sendo o fornecimento de tampas disponibiliza toda a informação sobre o abastecimento de tampas à máquina, e a roscagem toda a informação relativamente ao processo de colocação de tampas na garrafa.

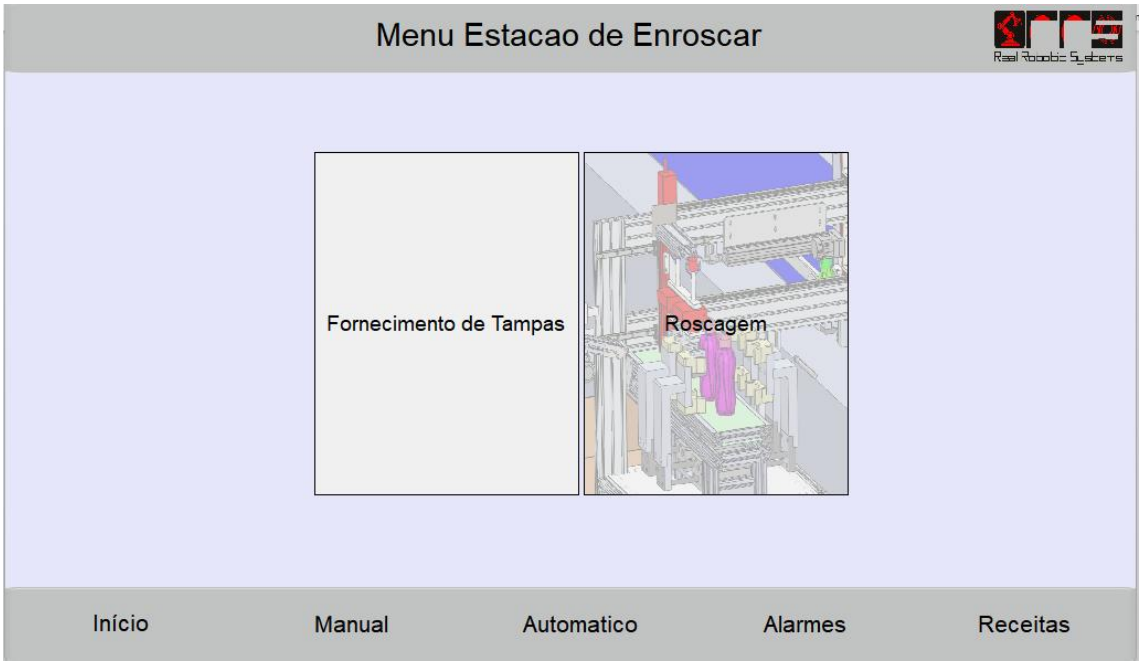


Figura I.4 Menu Estação de Enroscar.

O menu de ensacamento das garrafas (figura I.5) dividiu-se em diversas partes devido a complexidade do processo, ou seja, para realizar o processo de ensacamento era necessário movimentar muitos eixos e válvulas e como forma de tornar o menu mais

acessível para testar os componentes individuais, dividiu-se menus mais pequenos com as funções especificas e com a respetiva sinalização dos sensores.

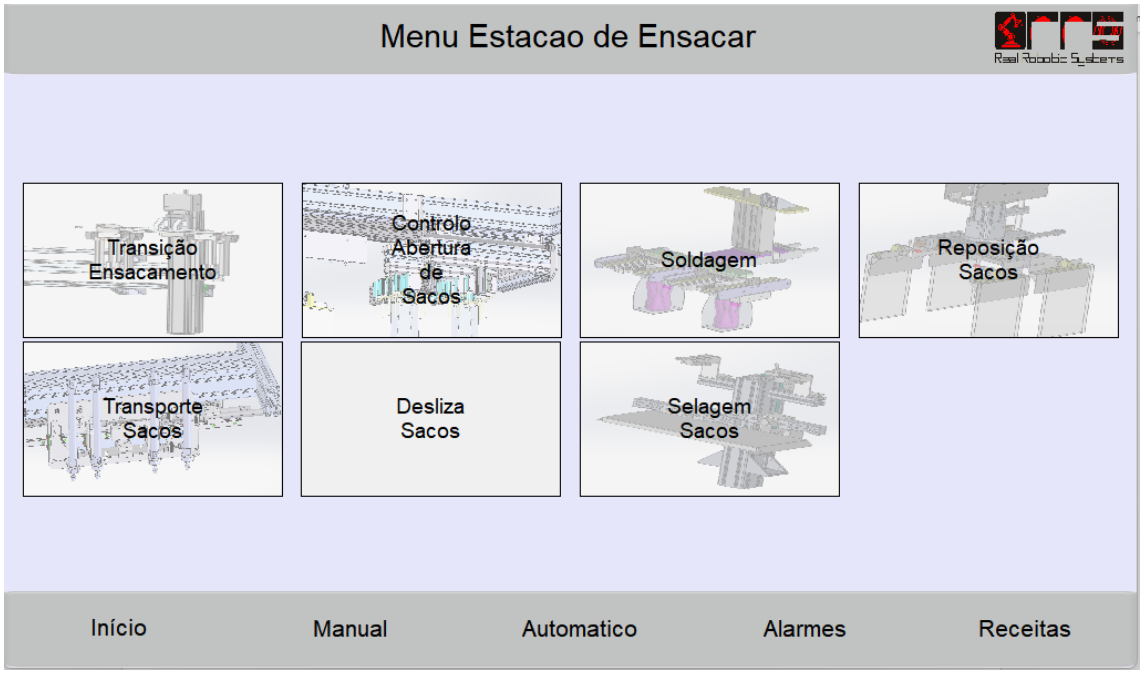


Figura I.5 Menu Estação Ensacar.

Com o menu transportador de saída (figura I.6), consegue perceber-se quais os cilindros utilizados para o transportador que vai levar as garrafas até à estação de saída.

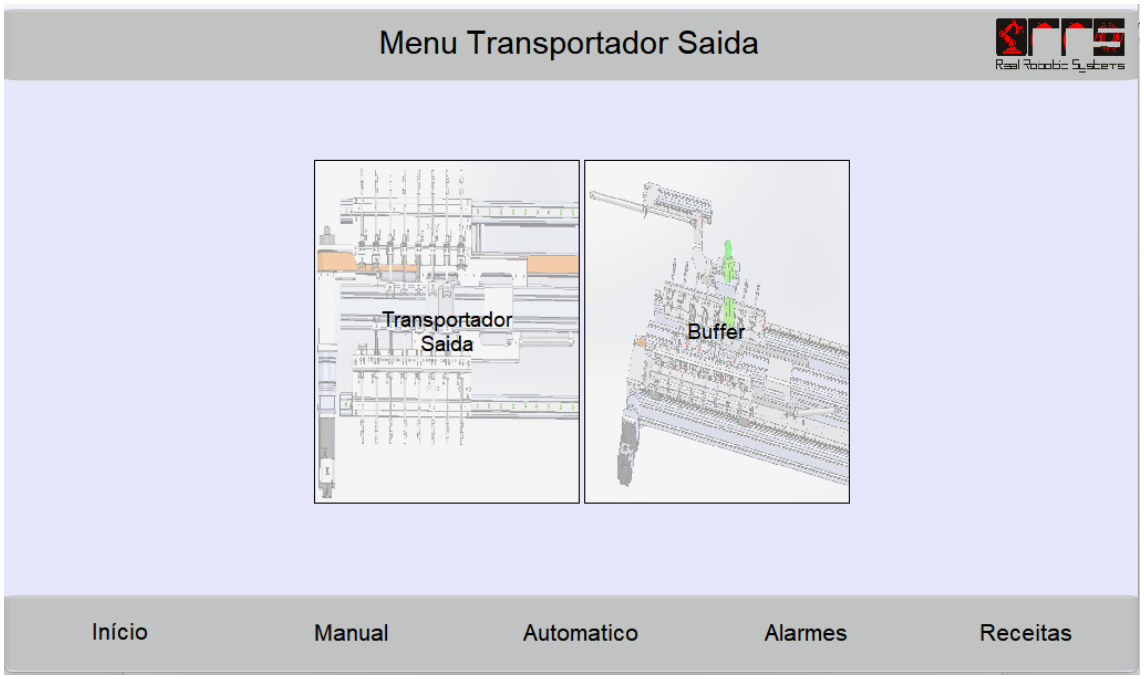


Figura I.6 Menu Transportador Saída.

No menu estação saída robot (figura I.7) pretendia-se alterar ou verificar o funcionamento da estação de saída, assim, consegue-se ensaiar a ação de preparar as garrafas para o robô as ir buscar.

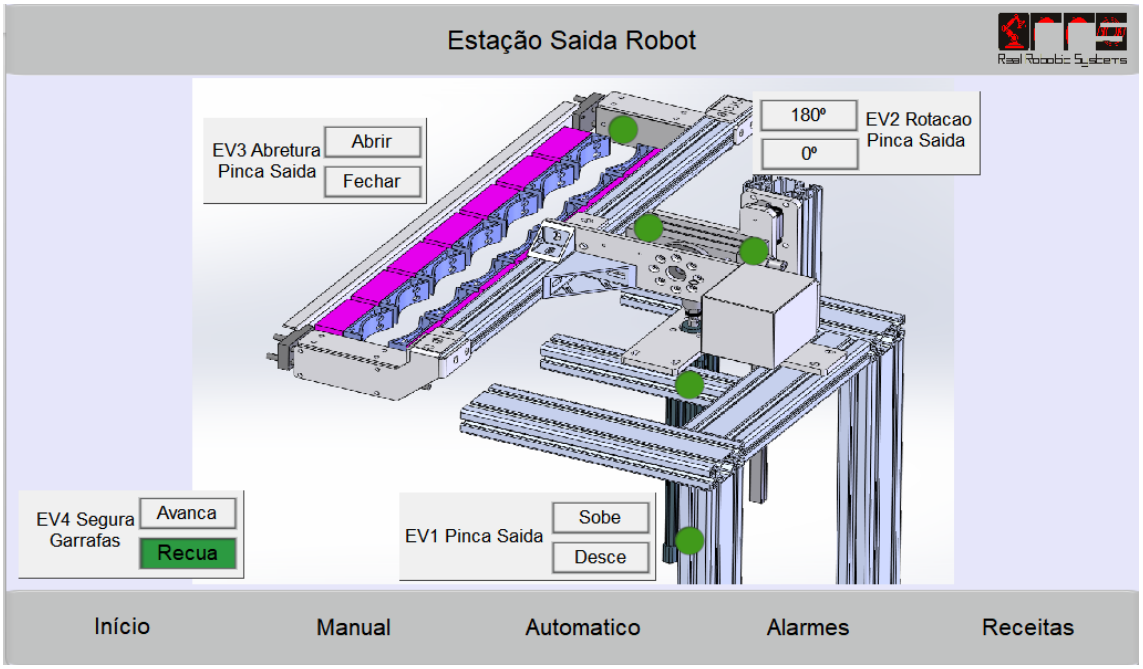


Figura I.7 Menu Estação Saída Robot.

O menu tapetes (figura I.8) foi criado unicamente para poder controlar a velocidade do tapete de entrada, de forma que quando as garrafas fossem colocadas não caíssem.

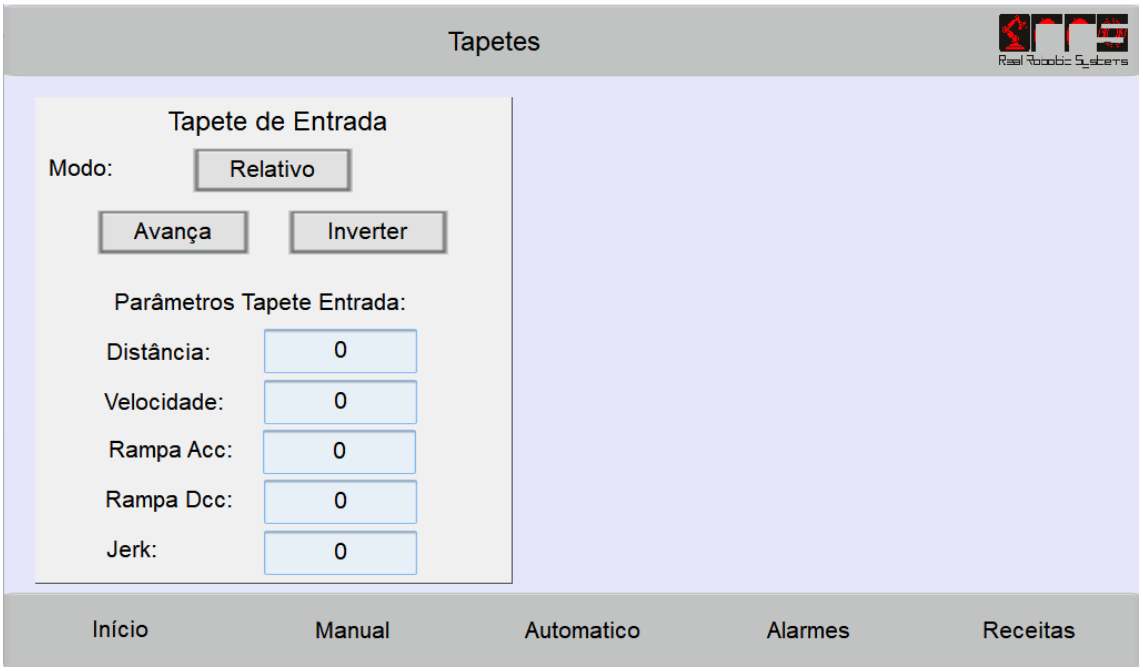


Figura I.8 Menu Tapetes.

Todos estes menus descritos anteriormente dentro do menu manual, contam com o acionamento das válvulas ou dos motores, bem como a sinalização dos sensores e a visualização através de um led caso estejam ativos.

Menu Automático

No menu automático (figura I.9) pode-se gerir e programar a máquina no modo automático, sendo que todas as alterações realizadas neste menu têm de ser realizadas com a máquina parada.

Assim sendo, este menu pode-se dividir em:

- Visualização – permite visualizar o funcionamento da máquina, ou seja, é possível verificar se há saturação da entrada (se há muitas garrafas na entrada da máquina), o número de sacos que estão colocados na máquina, se existe caixas e em caso de haver qual das caixas está a ser utilizada para colocar as garrafas. Também é possível verificar quantas garrafas o robô vai agarrar para colocar na caixa;
- Alteração do modo de funcionamento – é possível alterar a receita em produção e do robô, definir o número de sacos e habilitar ou desabilitar certas partes do robô (enroscar rolhas, ligar/desligar a soldadora dos sacos dependendo do tipo de sacos, e ligar/desligar o robô).

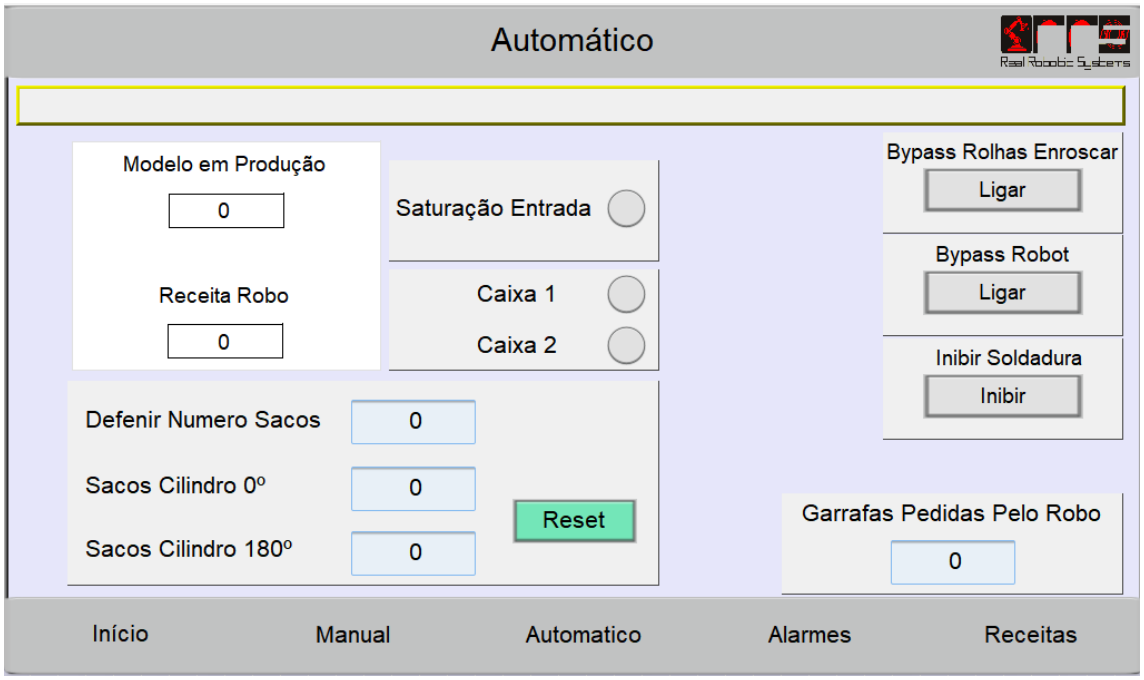


Figura I.9 Menu Automático.

Menu Alarmes

Neste menu encontrar-se-á os alarmes caso haja e a sua localização (figura I.10), por exemplo se houve falhar na leitura de um sensor de avançar ou de recuar de um sensor de um cilindro vai aparecer a dizer defeito no cilindro A, sendo que o mesmo pode ser devido ao cilindro não ter conseguido chegar a posição pretendida ou porque há falhar de uma leitura de um sensor. Mediante o tipo de erro a máquina pode parar ou não, cabendo essa decisão ao autómato e a programação realizado no mesmo.

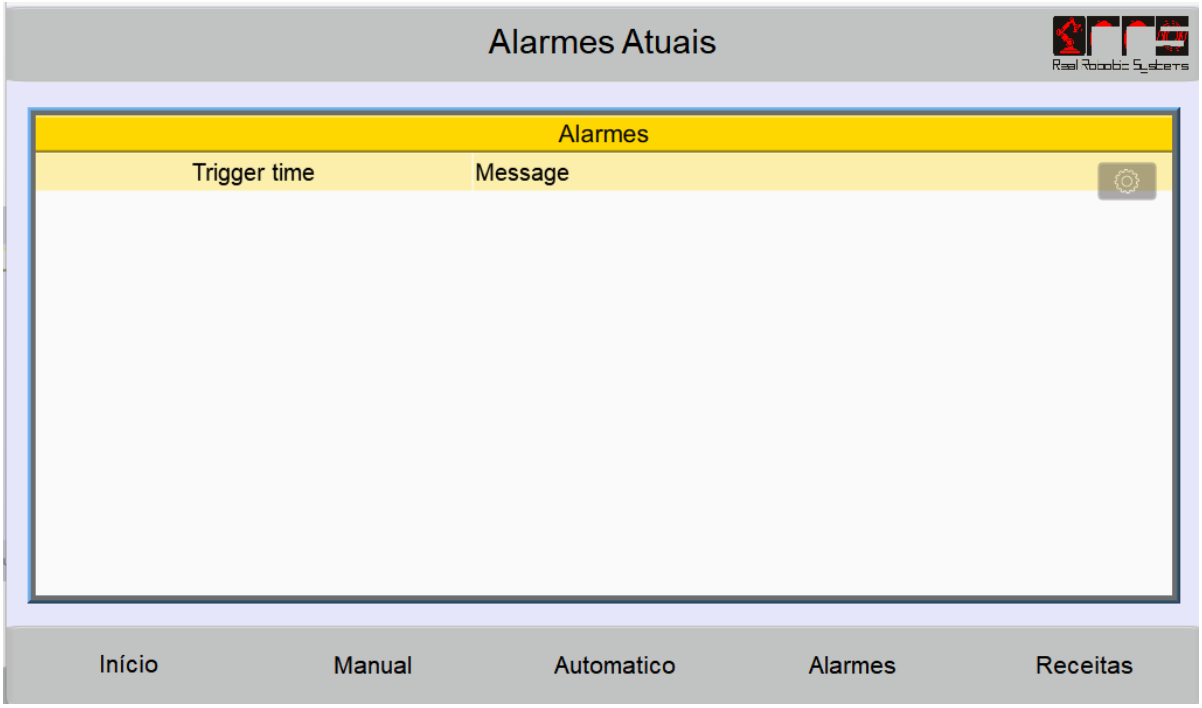


Figura I.10 Menu Alarmes.

Os alarmes que vão aparecer no menu de alarmes, foram configurados de acordo com a variável que veio do autómato e foi anexo a sua localização ou a razão (figura I.11). Os alarmes configurados tiveram todos a mesma prioridade, porque no autómato conforme o erro a máquina vai parar, e neste caso só se precisava de mostrar o alarme.

No.	Category	Text	Mode	Condition	Read
1	0: Category 0	Falha Na Segurança	BIT	ON	OMRO
2	0: Category 0	Robô em ERRO	BIT	ON	OMRO
3	0: Category 0	Eixo Vertical Ensacar Paragem de Emergência	BIT	ON	OMRO
4	0: Category 0	Eixo Vertical Ensacar em Alarme	BIT	ON	OMRO
5	0: Category 0	Eixo Horizontal Entrada (Festo) sem Home	BIT	ON	OMRO
6	0: Category 0	Eixo Horizonta Entrada (Festo) em ERRO	BIT	ON	OMRO
7	0: Category 0	Eixo Horizontal Ensacar (Festo) sem Home	BIT	ON	OMRO
8	0: Category 0	Eixo Horizontal Ensacar (Festo) em ERRO	BIT	ON	OMRO
9	0: Category 0	Eixo Vertical Entrada em Alarme	BIT	ON	OMRO
10	0: Category 0	Paragem de Emergência Robô	BIT	ON	OMRO
11	0: Category 0	Motores Rotativos sem Home	BIT	ON	OMRO
12	0: Category 0	Eixo Vertical Roscagem Paragem de Emergência	BIT	ON	OMRO
13	0: Category 0	Eixo Vertical Roscagem em Alarme	BIT	ON	OMRO
14	0: Category 0	Eixo Vertical Entrada Paragem de Emergência	BIT	ON	OMRO
15	0: Category 0	Maquina sem Sacos	BIT	ON	OMRO
16	0: Category 0	Defeito Cilindro Entrada Separa Garrafas	BIT	ON	OMRO
17	0: Category 0	Defeito Cilindro Pinca Transportadora Entrada	BIT	ON	OMRO

Figura I.11 Alarmes existentes.

Menu Receitas

No menu receitas (figura I.12) vão ser apresentadas todas as receitas programadas, quer sejam elas da máquina, quer sejam elas do robô. Também neste menu pode-se fazer alterações às receitas caso a máquina esteja parada e o operador que queira fazer as alterações tenha as senhas de administrador ou de programador.

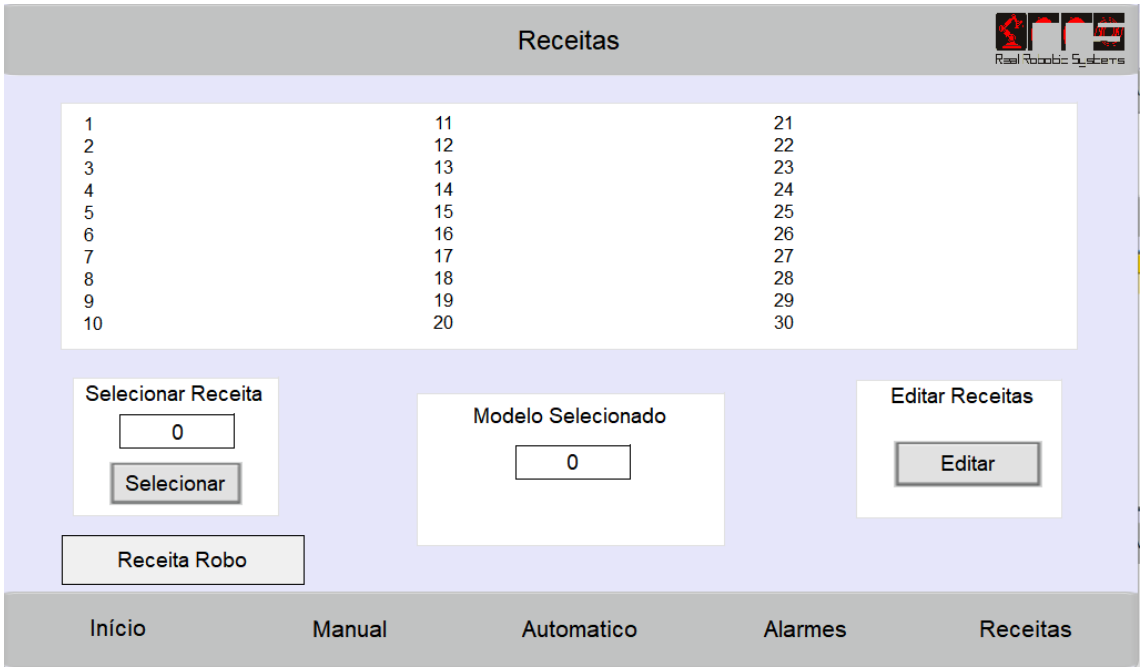


Figura I.12 Menu Receitas.

O operador ao carregar no editar e com as credenciais certas, vai poder mover todo o tipo de configurações e poder alterar de acordo com o pretendido (figura I.13). Ou seja, o poderá alterar as posições dos eixos que vai levar a que a máquina tenha um funcionamento diferente.

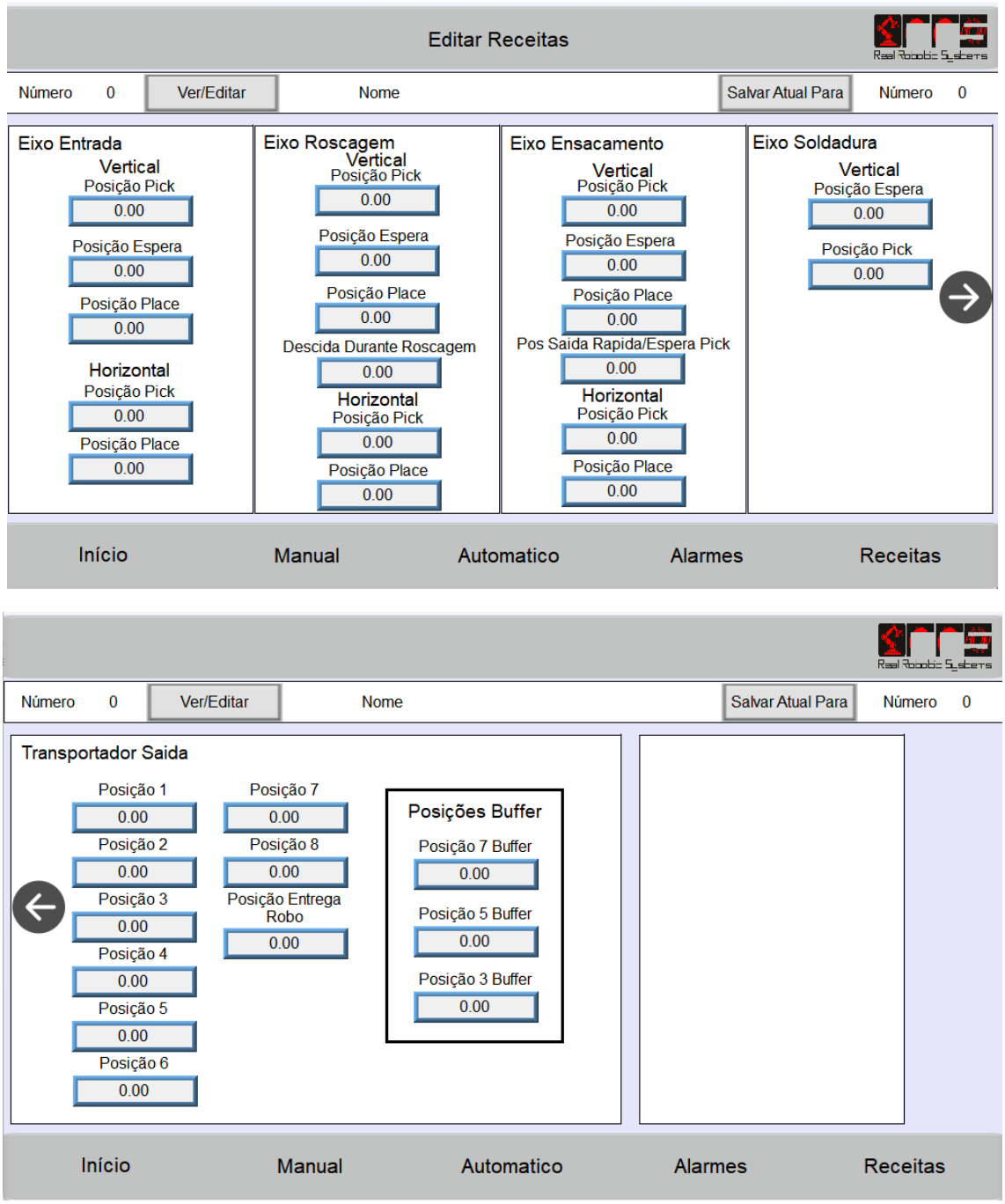


Figura I.13 Menu Alterar Receitas.

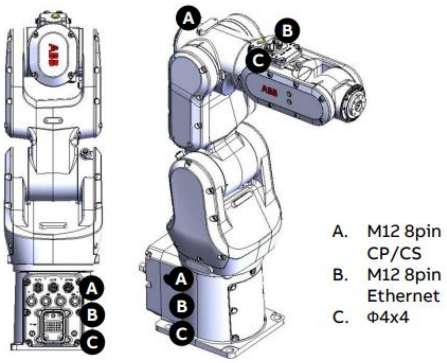
No final das alterações pretendidas o operado terá de carregar no botão salvar e colocar a interface no menu início. Caso pretenda iniciar a máquina nas novas configurações terá de rearmar a máquina na botoneira e selecionar o modo automático, após isso terá de verificar se a receita que alterou encontra-se selecionada no menu automático e depois sim carregar no botão de início.

ANEXO II - DADOS ROBÔ ABB

Features

IRB 1100 Preview

Item	Specification
Reach (mm)	475/580
Payload (kg)	4
Position repeatability (mm)	TBC
Protection	IP40
Cleanroom	CR4
Footprint (mm)	160 x 160
Mounting	Any angle

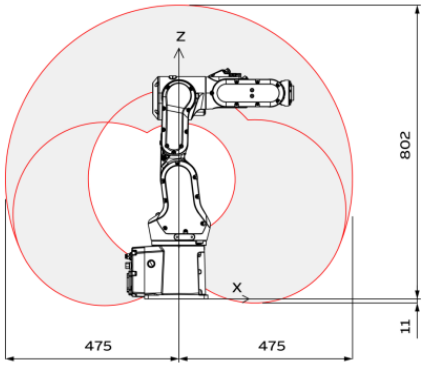


 | Slide 4 ABB

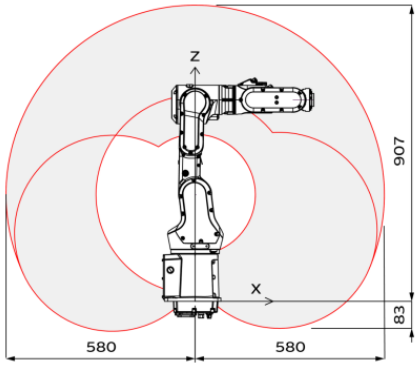
Main dimensions

Working Range

IRB 1100 – 4/0.475



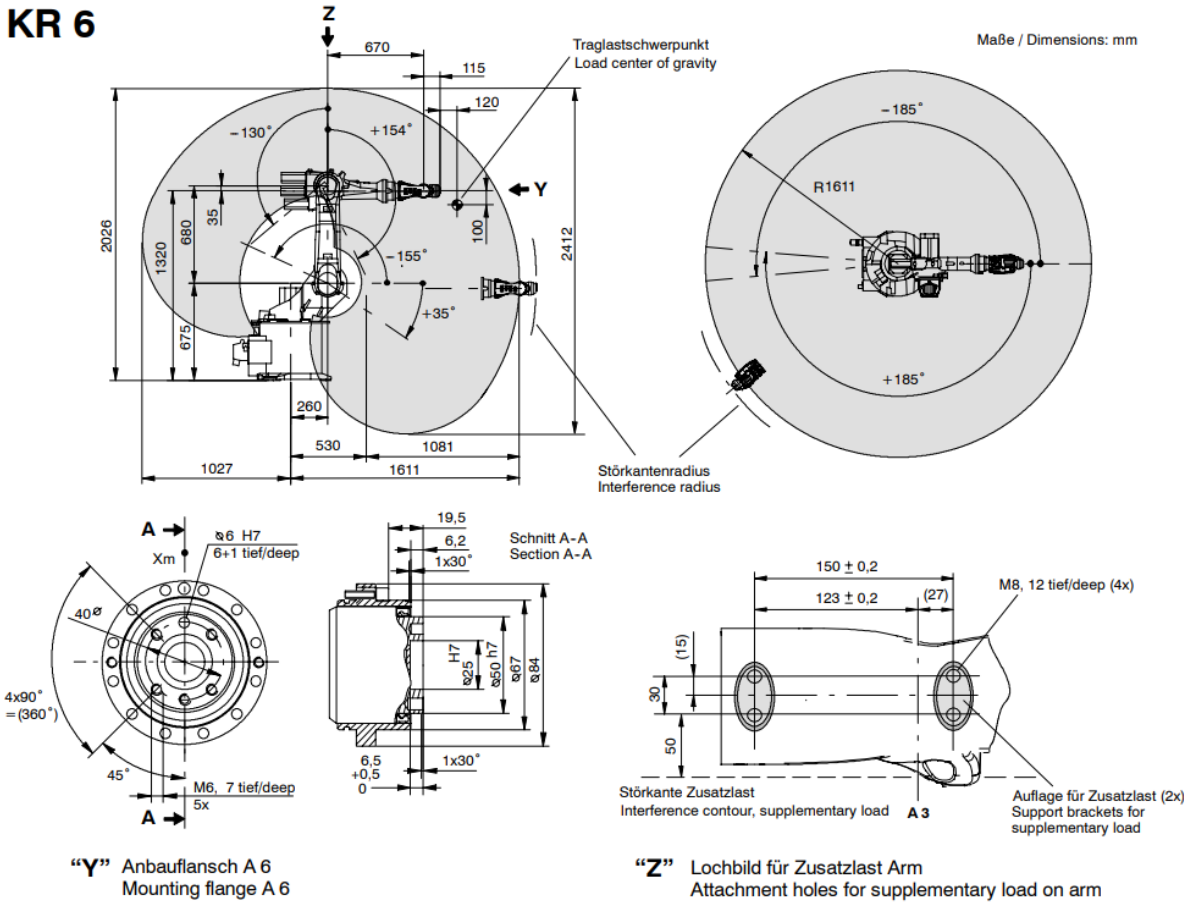
IRB 1100 – 4/0.58



 | Slide 9 ABB

Figura II.1 Ficha técnica robô ABB IRB 1100

ANEXO III – DADOS ROBÔ KUKA



"Y" Anbauflansch A 6
Mounting flange A 6

"Z" Lochbild für Zusatzlast Arm
Attachment holes for supplementary load on arm

Traglast / Payload:	6 kg	
Zusatzlast Arm / Schwinge / Karussell Supplementary load on arm / link arm / rotating column:	10 kg / variabel/variable / 20 kg	
Max. Gesamtlast / Total distributed load:	36 kg	
Anzahl der Achsen / Number of axes:	6	
Handvariante / Wrist variant:	Zentralhand 6 kg / In-line wrist 6 kg	
Anbauflansch A 6 / Mounting flange A 6:	DIN ISO 9409-1-A40	
Einbaulage / Mounting position:	Boden, Wand, Decke / Floor, wall, ceiling	
Wiederholgenauigkeit (ISO 9283) / Repeatability (ISO 9283):	± 0,05 mm	
Steuerung / Controller:	KR C2	
Gewicht (ohne Steuerung) ca. / Weight (excl. controller) approx.:	235 kg	
Arbeitsraumvolumen / Work envelope volume:	14,5 m³ 1)	
Achsdaten / Axis data:	Bereich (Software) / Range (software)	Geschwindigkeit / Speed
Achse / Axis 1 (A 1)	± 185° 2)	156°/s
Achse / Axis 2 (A 2)	+ 35° / -155°	156°/s
Achse / Axis 3 (A 3)	+ 154° / -130°	156°/s
Achse / Axis 4 (A 4)	± 350°	343°/s
Achse / Axis 5 (A 5)	± 130°	362°/s
Achse / Axis 6 (A 6)	± 350°	659°/s

Figura III.1 Ficha técnica robô KUKA KR6

ANEXO IV – DADOS AUTÓMATO

General information	
HW functional status	01
Firmware version	V3.3
Engineering with	
• Programming package	STEP 7 V5.5 + SP1 or higher or STEP 7 V5.2 + SP1 or higher with HSP 218
Supply voltage	
Rated value (DC)	
• 24 V DC	Yes
permissible range, lower limit (DC)	19.2 V
permissible range, upper limit (DC)	28.8 V
external protection for power supply lines (recommendation)	2 A min.
Mains buffering	
• Mains/voltage failure stored energy time	5 ms
• Repeat rate, min.	1 s
Input current	
Current consumption (rated value)	850 mA

	6ES7 312-1AE14-0AB0	6ES7 314-1AG14-0AB0	6ES7 315-2AH14-0AB0	6ES7 315-2EH14-0AB0	6ES7 317-2EK14-0AB0
Product-type designation	CPU 312	CPU 314	CPU 315-2 DP	CPU 315-2 PN/DP	CPU 317-2 PN/DP
S7 communication					
• supported	Yes	Yes	Yes	Yes	Yes
S5-compatible communication					
• supported	Yes; via CP and loadable FC	Yes; via CP and loadable FC	Yes; via CP and loadable FC	Yes; via CP and loadable FC	Yes; via CP and loadable FC
Web server					
• Web server				Yes; Read-only function	Yes; Read-only function
• Number of HTTP clients				5	5
Open IE communication					
• TCP/IP				Yes; via integrated PROFINET interface and loadable FBs	Yes; via integrated PROFINET interface and loadable FBs
- Number of connections, max.				8	16
• ISO-on-TCP (RFC1006)				Yes; via integrated PROFINET interface and loadable FBs	Yes; via integrated PROFINET interface and loadable FBs
- Number of connections, max.				8	16
- Data length, max.				32 768 byte	32 768 byte
• UDP				Yes; via integrated PROFINET interface and loadable FBs	Yes; via integrated PROFINET interface and loadable FBs
- Number of connections, max.				8	16
- Data length, max.				1 472 byte	1 472 byte

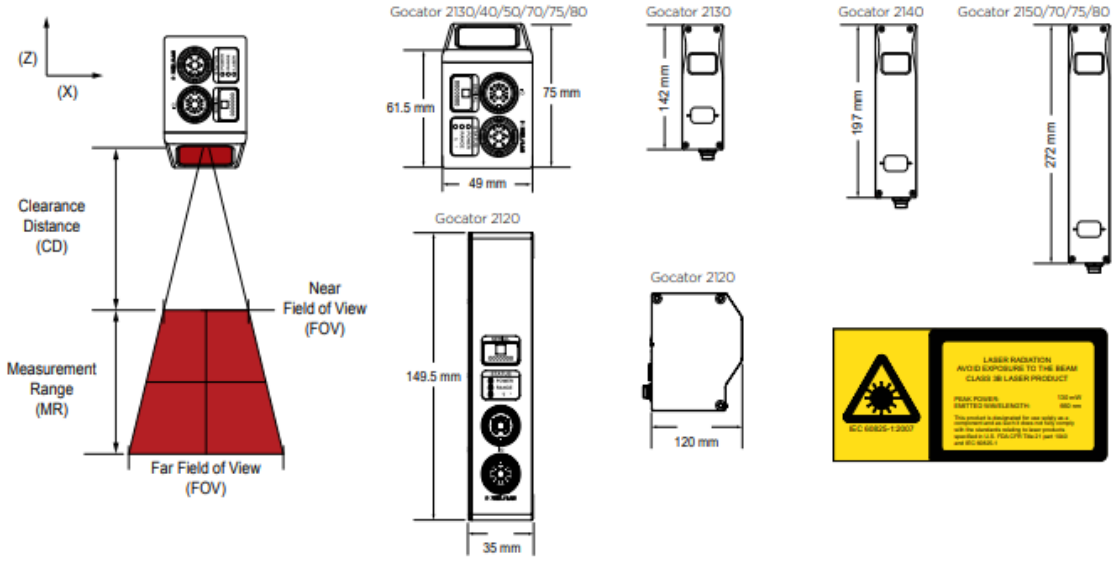
Figura IV.1 Ficha técnica autómato Siemens S7-300

ANEXO V – DADOS SISTEMA DE VISÃO 3D DA GOCATOR

GOCATOR 2100 SERIES MODELS	2120	2130	2140	2150	2170	2175	2180
Data Points / Profile	1280	640	640	640	640	640	640
Linearity Z (+/- % of MR)	0.01	0.01	0.01	0.01	0.04	0.03	0.04
Resolution Z (mm)	0.0018 - 0.0030	0.006 - 0.014	0.013 - 0.037	0.019 - 0.060	0.055 - 0.200	0.175 - 0.925	0.092- 0.488
Resolution X (mm) (Profile Data Interval)	0.028 - 0.042	0.088 - 0.150	0.19 - 0.34	0.3 - 0.6	0.55 - 1.10	0.51 - 1.58	0.75 - 2.20
Repeatability Z (µm)	0.4	0.8	1.2	2	8	12	12
Clearance Distance (CD) (mm)	40	90	190	300	400	650	350
Measurement Range (MR) (mm)	25	80	210	400	500	1350	800
Field of View (FOV) (mm)	18 - 26	47 - 85	96 - 194	158 - 365	308 - 687	324 - 1010	390 - 1260
Laser Classes	2, 3R	2, 3R	2, 3R, 3B	2, 3R	2, 3R	2, 3R	2, 3R
Dimensions (mm)	Side Mount 35x120x149.5	Top Mount 49x75x142	Top Mount 49x75x197	Top Mount 49x75x272	Top Mount 49x75x272	Top Mount 49x75x272	Top Mount 49x75x272
Weight (kg)	0.8	0.74	0.94	1.3	1.3	1.3	1.3

Optical models, laser classes, and packages can be customized. Contact LMI for more details.
Specifications stated are based on standard laser classes. Linearity Z, Resolution Z, and Repeatability Z may vary for other laser classes.
Refer to specifications in the Gocator Line Profile Sensor user manual for more details.

ALL 2100 SERIES MODELS	
Scan Rate	Approximately 170 Hz to 5000 Hz
Interface	Gigabit Ethernet
Inputs	Differential Encoder, Laser Safety Enable, Trigger
Outputs	2x Digital output, RS-485 Serial (115 kBaud)
Factory Communication	PROFINET, Modbus, EtherNet/IP, ASCII, Gocator
Input Voltage (Power)	+24 to +48 VDC (13 Watts); Ripple +/- 10%
Housing	Gasketed aluminum enclosure, IP67
Operating Temperature	0 to 50°C
Storage Temperature	-30 to 70°C
Vibration Resistance	10 to 55 Hz, 1.5 mm double amplitude in X, Y and Z directions, 2 hours per direction
Shock Resistance	15 g, half sine wave, 11 ms, positive and negative for X, Y and Z directions
Scanning Software	Browser-based GUI and open source SDK for configuration and real-time 3D visualization. Open source SDK, native drivers, and industrial protocols for integration with user applications, third-party image processing applications, and PLCs.



Sensor State Information

Byte Index	Name	Type	Description
0-3	Runtime Variable 0	32s	Stores the intended value of the Runtime Variable at index 0.
4-7	Runtime Variable 1	32s	Stores the intended value of the Runtime Variable at index 1.
8-11	Runtime Variable 2	32s	Stores the intended value of the Runtime Variable at index 2.
12-15	Runtime Variable 3	32s	Stores the intended value of the Runtime Variable at index 3.

Figura V.1 Ficha técnica sistema de visão Gocator 2140

ANEXO VI – DADOS SISTEMA DE VISÃO 3D DA SICK

Task	Detecting - Standard objects Detecting - Level Measuring - Dimension, contour and volume Measuring - Number Monitoring and controlling - Quality Determining position - 3D position determination
Technology	3D triangulation
Product category	Configurable
Toolkit	Shape Area detection Blob Finder Volume Find plane Fixed plane
Working distance	141 mm ... 541 mm
Example field of view	270 mm x 100 mm
Illumination	Integrated
Illumination color	Red, laser, Visible, 660 nm, ± 7 nm
Laser class	2 (EN 60825-1:2014+A11:2021; IEC 60825-1:2014, Complies with the FDA performance requirements for laser products except for conformance with IEC 60825-1 Ed.3., as described in Laser Notice 56 dated May 8, 2019.)
Factory calibrated	✓
Width at minimum operating distance	90 mm
Width at maximum operating distance	330 mm
Maximum height range	400 mm
Imaging angle	65°
Offline support	Emulator

Connection type	M12, 12-pin male connector, A-coded (voltage supply, I/O) M12, 8-pin female connector, X-coded (Gigabit Ethernet) M12, 8-pin female connector, A-coded (encoder)
Connector material	Nickel plated brass
Supply voltage	24 V DC, ± 20 %
Ripple	< 5 V_{pp}
Power consumption	≤ 11 W

Figura VI.1 Ficha técnica sistema de visão Sick V3T12S-MR32A7

ANEXO VII – MANUAL DE UTILIZAÇÃO DO ROBOTSTUDIO

Criar o ambiente de trabalho

Com o robô criado e escolhido aquando da criação do projeto, resta agora criar o ambiente de trabalho semelhante ou simplificado em relação ao ambiente onde se pretende inserir o robô.

Para isso vai-se ao simulador RobotStudio e no separador MODELING (em cima do lado esquerdo, sendo o terceiro botão) no submenu CREATE (figura VII.1).

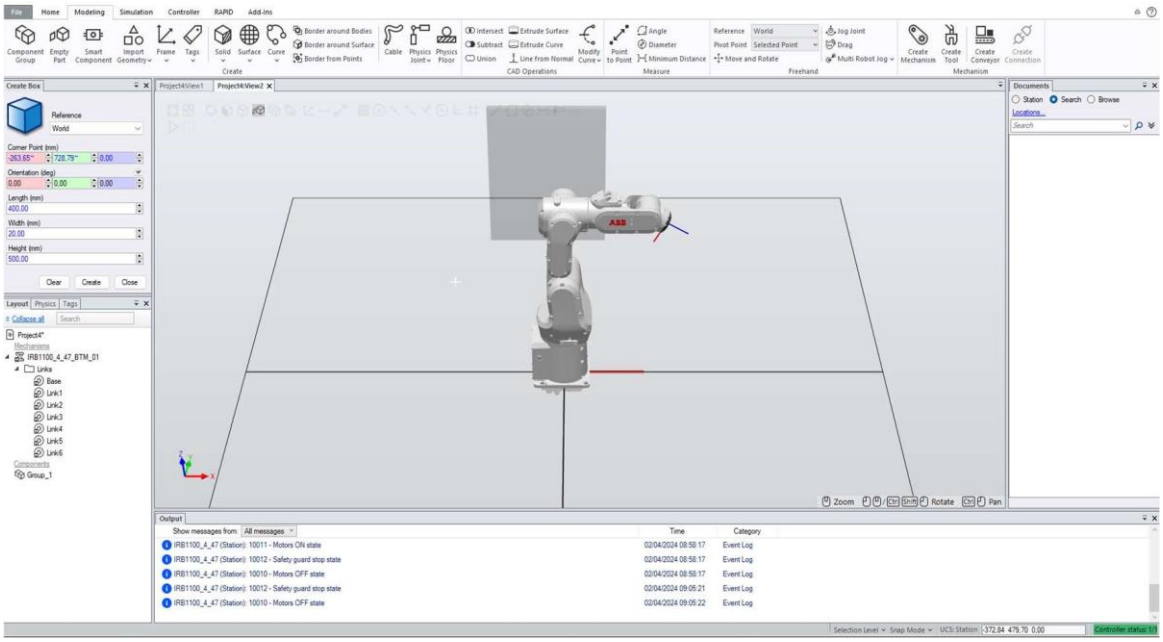


Figura VII.1 Criação do ambiente de trabalho

No submenu CREATE (figura VII.2) pode-se criar objetos de quase todos os tipos para criar um ambiente muito parecido a realidade e assim marcar uns pontos muito próximos da realidade ou conseguir fazer um estudo dos movimentos.

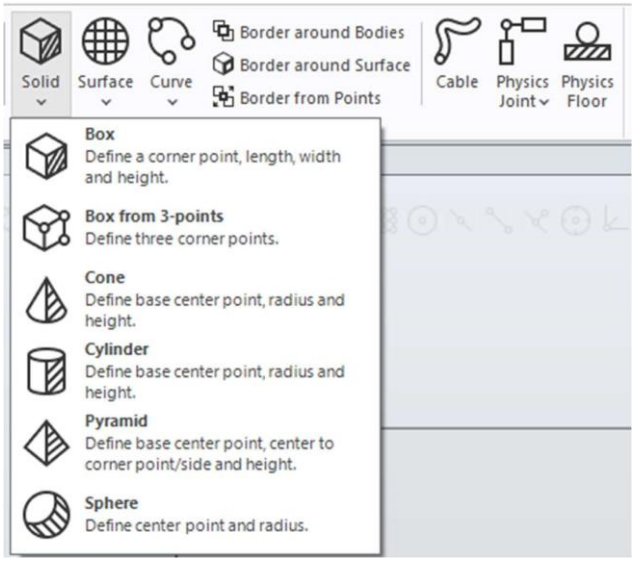


Figura VII.2 Menu CREATE

Para dar um exemplo de um objeto vai-se criar um retângulo (figura VII.3).

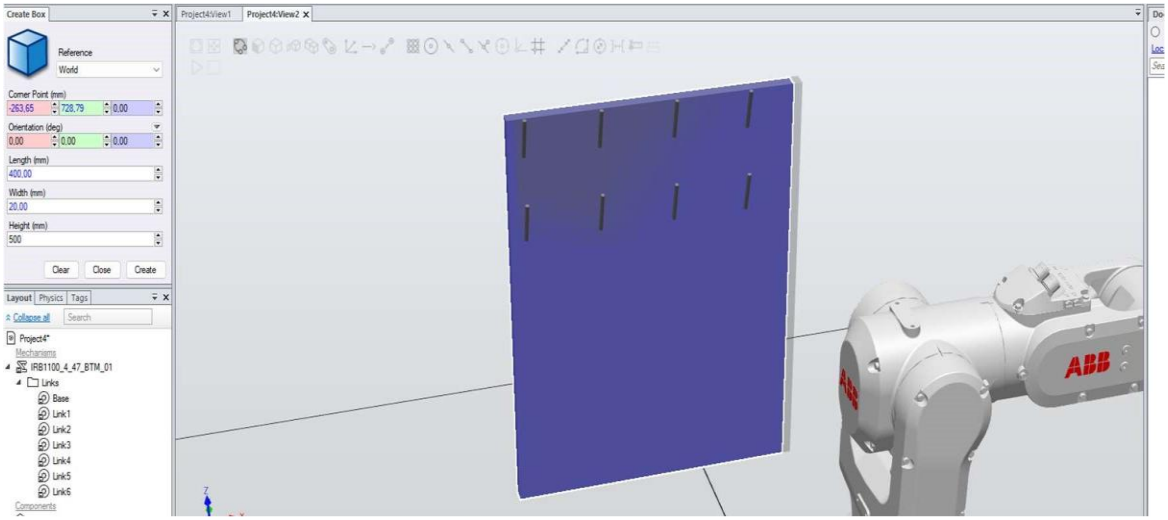


Figura VII.3 Criação de um retângulo

Caso queira-se duplicar o objeto carrega-se na barra lateral esquerda em cima do nome do objeto e carrega-se no DUPLICATE, depois escolhe-se em que direção (figura VII.4).

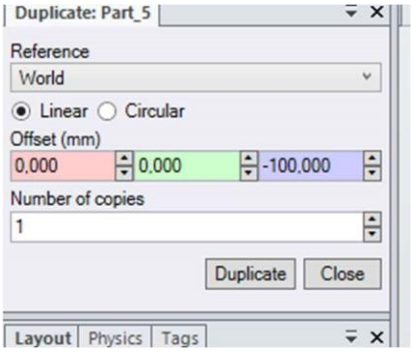


Figura VII.4 Menu DUPLICATE

Marcação de pontos ou trajetórias

Com o ambiente semelhante passa-se então a marcação de pontos. Para isso mexe-se o robô no menu HOME (em cima do lado esquerdo, sendo o segundo o botão) no submenu FREEHAND (figura VII.5).

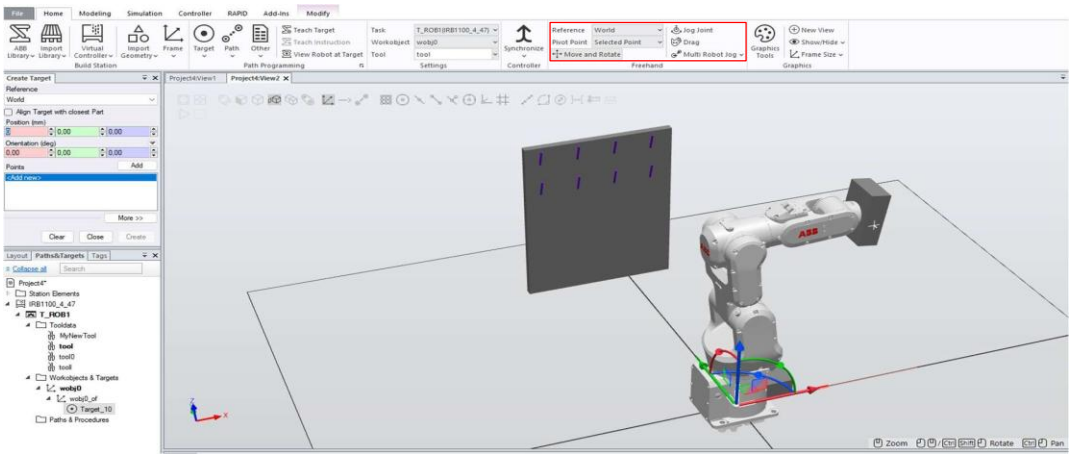


Figura VII.5 Menu Home

Podendo mover o robô junta a junta (JOG JOINT no simulador) ou arrasto (Drag no simulador) ou mexer o robô em várias juntas (MULTI ROBT JOG no simulador), estas opções estão assinaladas com quadros vermelhos na figura 5. Neste submenu pode-se ainda alterar o tipo de ferramenta, bem como o ponto pivot de ajuda.

Depois de guardar falta só mesmo guardar o ponto, para guardar o ponto vai-se a o menu HOME (em cima do lado esquerdo, sendo o segundo o botão) no submenu PATH PROGRAMMING (figura VII.6).

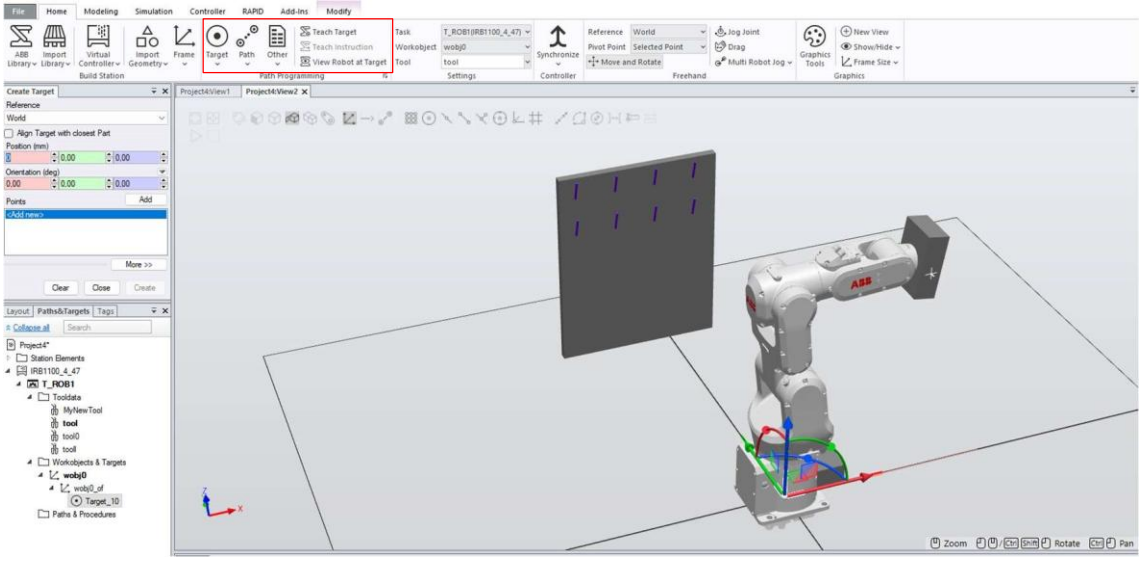


Figura VII.6Menu Home

Com os pontos marcados (figura VII.7 letra A) tem de se arrastar para as trajetórias (figura VII.7 letra B) e ainda escolher o tipo de movimento a efetuar, para criar-se trajetórias para o robô ou para o simulador.

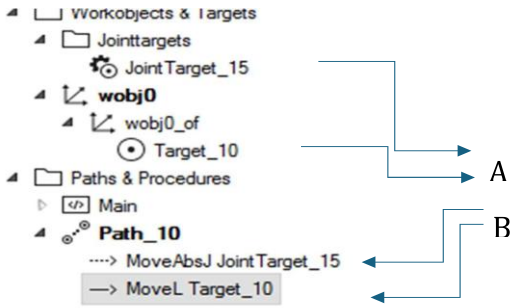


Figura VII.7Pontos e trajetórias

Visualização de movimentos e testes

Com os pontos e as trajetórias marcadas segue se a importação para o menu RAPID, para isso no menu HOME carrega-se no SYNCHRONIZE (figura VII.8).

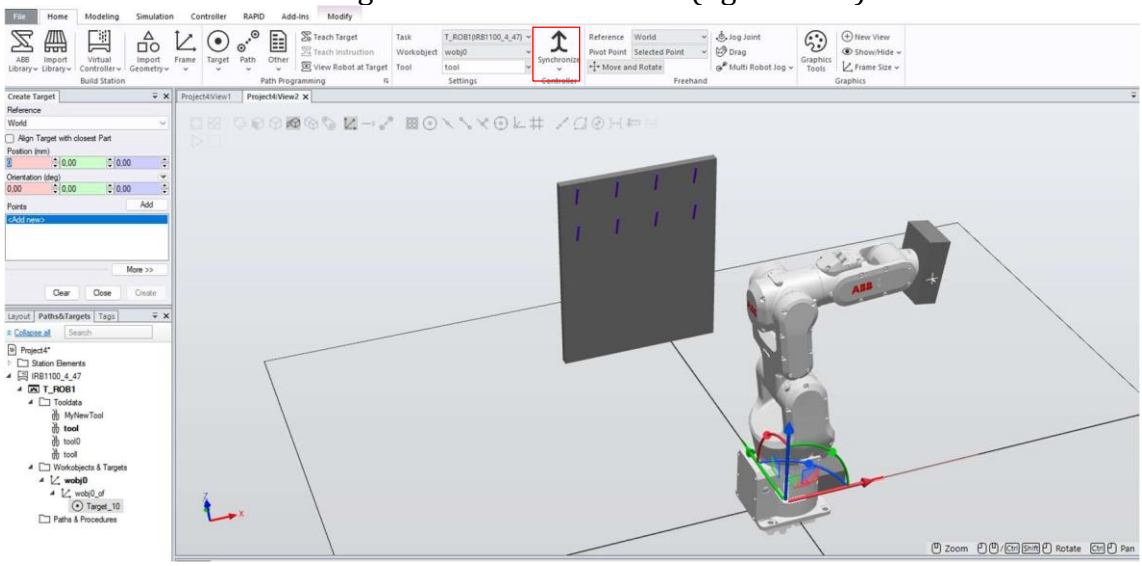


Figura VII.8Menu Home

Aparece um menu (figura VII.9) e nesse menu deve preencher-se todos os quadros com as configurações que se alteraram ou se pretender com todas para garantir que a infirmação passa completa.

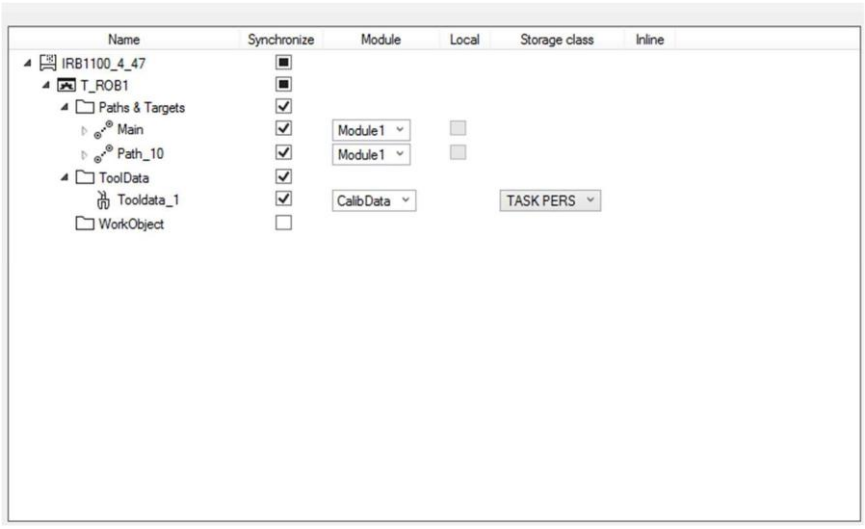


Figura VII.9 Menu SYNCHRONIZE TO RAPID

Para simular é so ir ao menu RAPID (em cima do lado esquerdo, sendo o quinto botão) no submenu TEST AND DEBUG (figura VII.10).

Figura VII.10 Menu TEST AND DEBUG

Se der algum erro deve ir ao código RAPID que se encontra na lateral esquerda (figura VII.11)

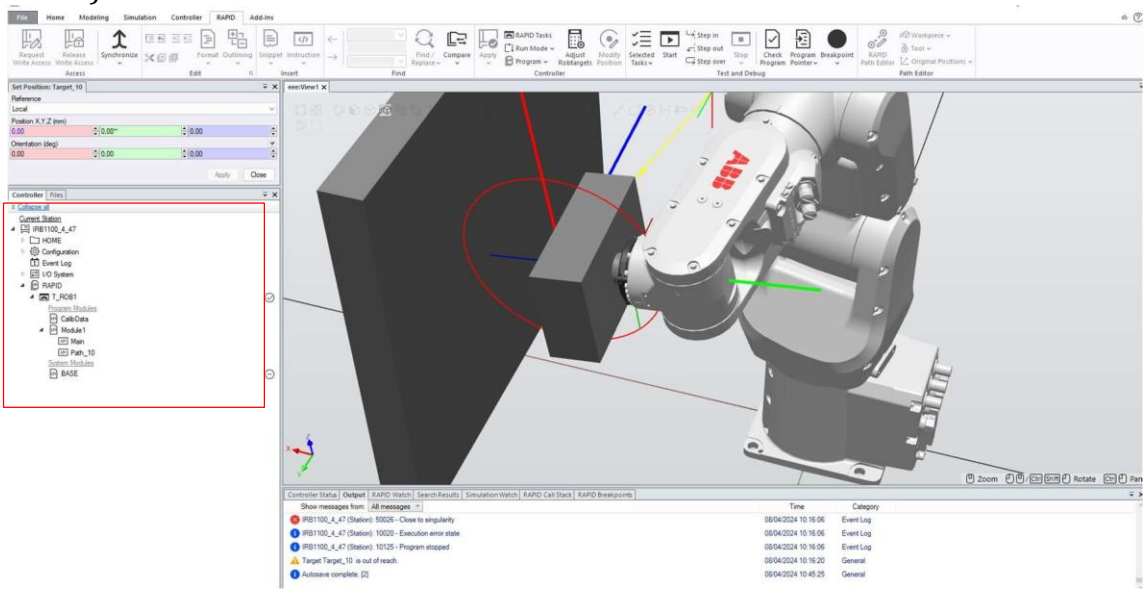


Figura VII.11 Localização do código RAPID

Dentro deste código tem pode-se encontrar o erro que estará sublinhado (figura VII.12), mas é preciso ter em atenção que nas primeiras vezes que se corre as trajetórias não estão no MAIN e é necessário colocar (figura VII.13).

```
MODULE Module1
        CONST jointtarget JointTarget_15:=[[0,-50,50,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[188.5653103,309.6189829,485.804915089],[0.527847635,0.530237701,-0.456685576,0.481313972],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    PROC Main ()
        Path_10;
    ENDPROC
    PROC Path_10()
        MoveAbsJ JointTarget_15,v1000,z100,Tooldata_1\WObj:=wobj0;
    MoveJ Target_10,v1000,z100,Tooldata_1\WObj:=wobj0;



    ENDPROC
ENDMODULE
```

Figura VII.12 Exemplo de erro (sublinhado a verde)

```
MODULE Module1
        CONST jointtarget JointTarget_15:=[[0,-50,50,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[188.5653103,309.6189829,485.804915089],[0.527847635,0.530237701,-0.456685576,0.481313972],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    PROC Main ()
        Path_10;
    ENDPROC
    PROC Path_10()
        MoveAbsJ JointTarget_15,v1000,z100,Tooldata_1\WObj:=wobj0;
        MoveL Target_10,v1000,z100,Tooldata_1\WObj:=wobj0;



    ENDPROC
ENDMODULE
```

Figura VII.13 Exemplo de código RAPID

ANEXO VIII – FLUXOGRAMAS DE AUXÍLIO À PROGRAMAÇÃO DO AUTÓMATO

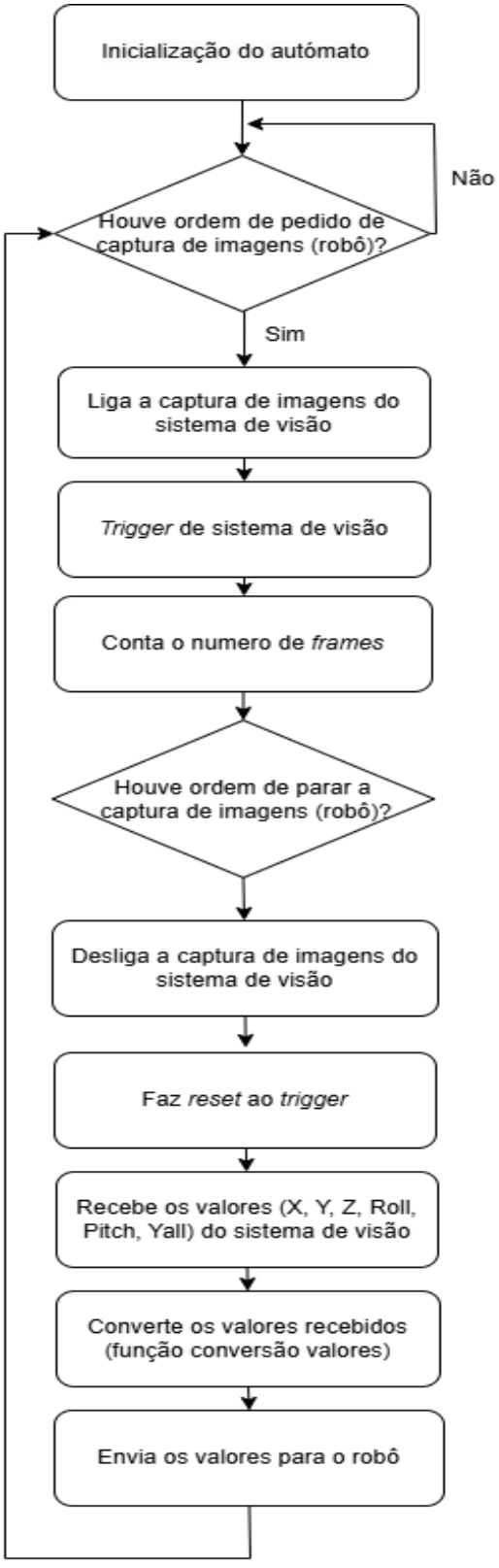


Figura VIII.1 Fluxograma do código principal

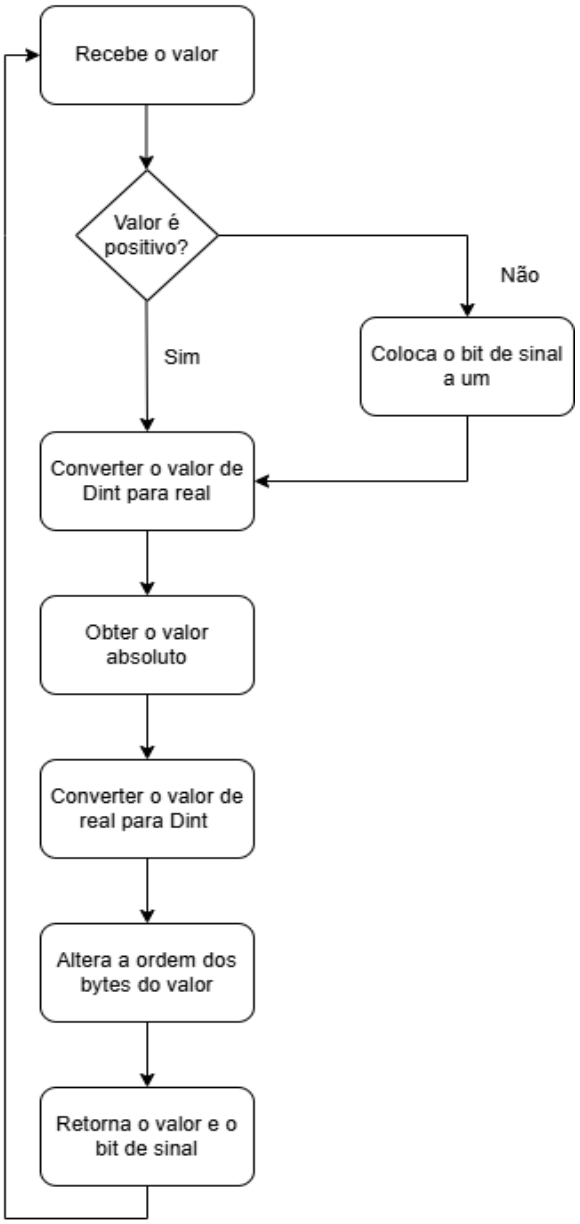


Figura VIII.2 Fluxograma da função de conversão de valores

ANEXO IX – TABELA DE VARIÁVEIS EXTERNAS DO AUTÓMATO

Tabela IX.1 - Tabela de variáveis externas do autómato

Nome da variável	Origem	Destino	
Decision (0,1,2,3)	Sistema de visão	Autómato	
Gcator.Inputs.Decisao.Decisao (0,1,2,3)	Autómato	Sistema de visão	
Gocator Inputs	Autómato	Sistema de visão	Inicialização do sistema de visão
Gocator Prog			
Gocator.CordenadaRobo.Pitch	Sistema de visão	Autómato	Envio de valor Pitch
Gocator.CordenadaRobo.Roll			Envio de valor Roll
Gocator.CordenadaRobo.X			Envio de valor X
Gocator.CordenadaRobo.Y			Envio de valorY
Gocator.CordenadaRobo.Yaw			Envio de valor Patch
Gocator.CordenadaRobo.Z			Envio de valor Z
Gocator.Inicio_Trigger	Autómato	Sistema de visão	
Gocator.Inputs.Busy	Sistema de visão	Autómato	
Gocator.Inputs.ValorAlgoritmo. Medida (0,1,2,3)	Autómato	Sistema de visão	
ILigarCamara	Robô	Autómato	
Measurment (0,1,2,3)	Sistema de visão	Autómato	
qCamaraLigada	Autómato	Sistema de visão	*Flag* de sistema e visão ligado
qPosPitchSinal	Autómato	Robô	Bit de sinal Pitch
qPosRollSinal			Bit de sinal Roll
qPosXSinal			Bit de sinal X
qPosYawSinal			Bit de sinal Yaw
qPosYSinal			Bit de sinal Y
qPosZSinal			Bit de sinal Z
qValorPitch	Autómato	Robô	Valor de Pitch
qValorRoll			Valor de Roll
qValorX			Valor de X
qValorY			Valor de Y
qValorYaw			Valor de Yaw
qValorZ			Valor de Z

ANEXO X – TABELA DE VARIÁVEIS INTERNAS DO AUTÓMATO

Tabela X.1 - Tabela de variáveis externas do autómato

Nome da variável	Origem	Destino	
bitSinalConvercao	Autómato	Autómato	*Flag* de número ímpar na função para o código principal
valorAConverterReal	Autómato	Autómato	Variável para converter em número real
valorConverter	Autómato	Autómato	Valor recebido na função
valorConvertido	Autómato	Autómato	Valor enviada pela função para o código principal
word []	Autómato	Autómato	Matriz com variáveis de tamanho de 16 *bits*
word_aux []	Autómato	Autómato	Matriz com variáveis de tamanho de 8 *bits*

ANEXO XI – MANUAL DE UTILIZAÇÃO DO SISTEMA DE VISÃO

Configurações da câmara

Antes sequer de ligar a câmara ao sistema, foi necessário ligar a câmara ao computador para fazer configurações de sistema. As configurações serão importantes pois sem elas não se consegue comunicar com o sistema.

Sendo assim começou-se então por ligar a câmara de visão 3D ao computador e começar por configurar o IP (*Internet Protocol*), para isso ligou-se a câmara a fonte de alimentação e o cabo de rede ao computador. O cabo de alimentação e o cabo de rede já vinham com a câmara e vinham num cabo único isolado com malha de ferro por causa do ruido que podia advir de outras máquinas.

Com a câmara já conectada passou-se então a configuração do IP da câmara, para isso abriu-se uma página de um explorador web do computador e num separador colocou-se o IP de fábrica da mesma.

Com o separador carregado, para configurar as configurações da câmara com o PLC é necessário ir ao menu OUTPUT (figura XI.1), este menu encontra-se no canto superior esquerdo sendo o quarto ícone.

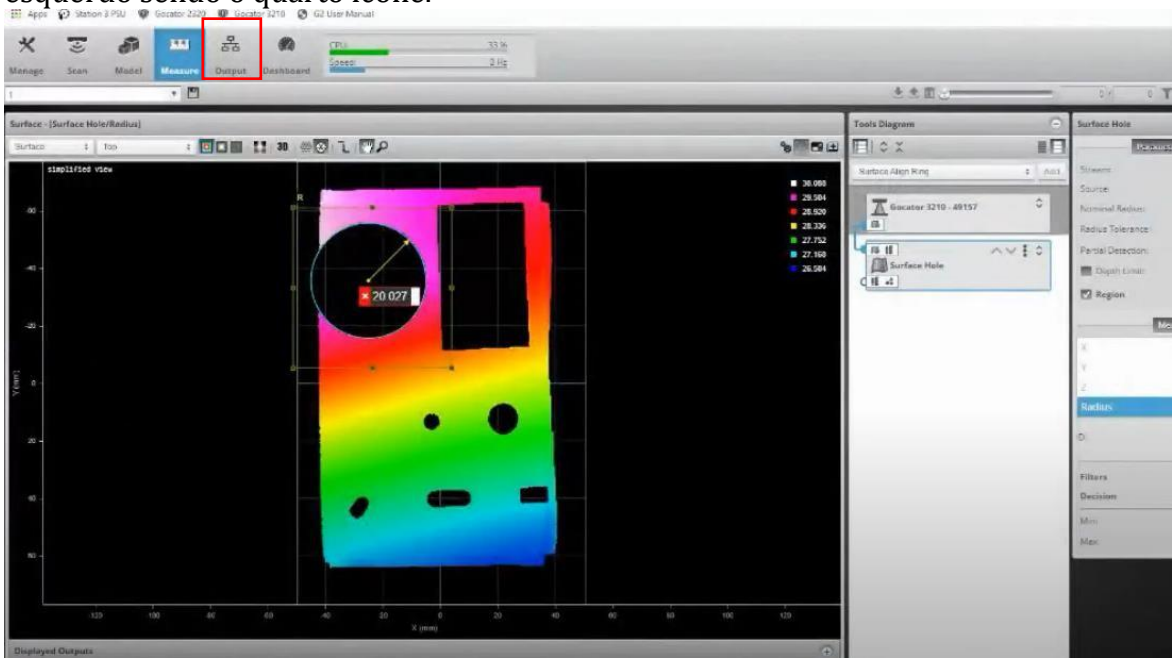


Figura XI.1 Menu principal da câmara

No menu OUTPUT (figura 2) pode configurar-se a ethernet, saídas digitais ou analógicas e saídas series.

Neste caso como pretende-se comunicar a câmara com o PLC escolhe-se o submenu Ethernet. Dentro do submenu ethernet escolheu-se o protocolo de comunicação (figura XI.2 letra A) que se pretende utilizar e qual o alinhamento de bit 16 ou 32 bits (figura XI.2 letra B).

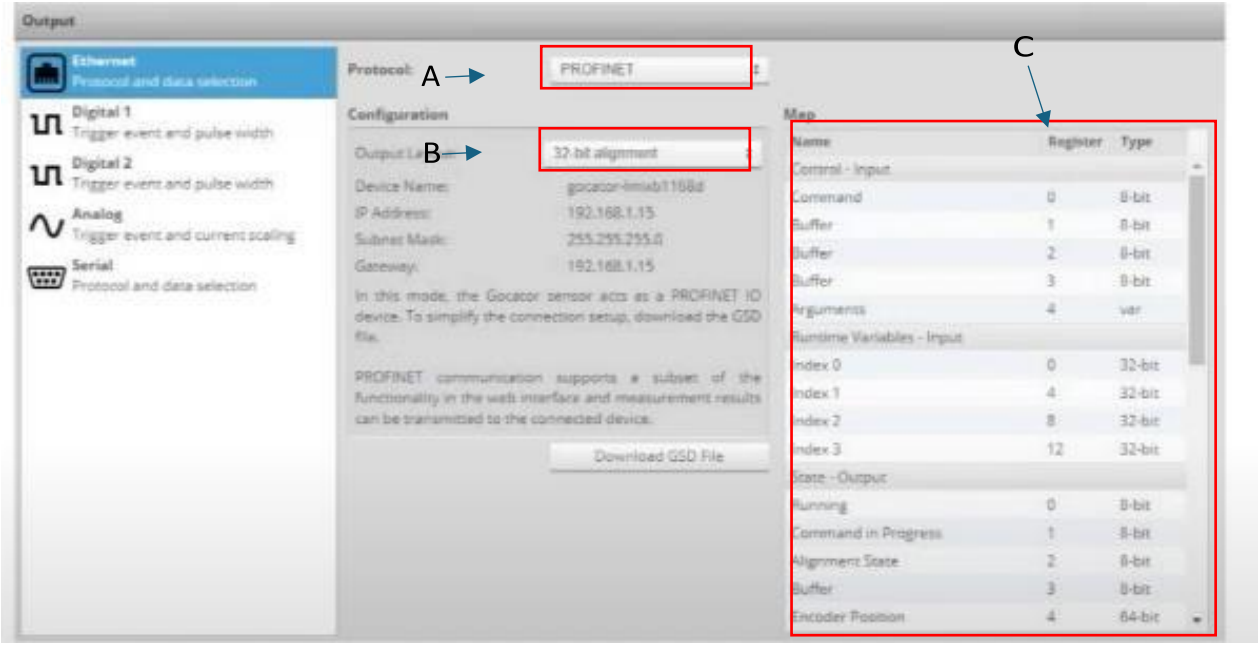


Figura XI.2 Menu *OUTPUT*

Mediante o protocolo de comunicação o mapeamento (map) das variáveis vão sendo alteradas (figura 2 letra C), o mapeamento é importante pois é vai ser descrito as variáveis necessárias para comunicação e/ou para ser dada alguma ordem entre a câmara o autómato e o robô. Por exemplo através destas variáveis pode-se dar inico e/ou fim a captura de imagens, entre outras funções (figura XI.3).

Figura XI.3 Mapeamento de variáveis da câmara de visão 3D

Na figura 5.3 pode-se reparar que conforme a variável e a função pode ter um peso diferente, por essa razão o número de bits vai variando. Antes de passar para a programação, faltava apenas transferir o ficheiro GSD (figura XI.4) do menu OUTPUT para a memoria do computador para conseguir programar a câmara no autómato.

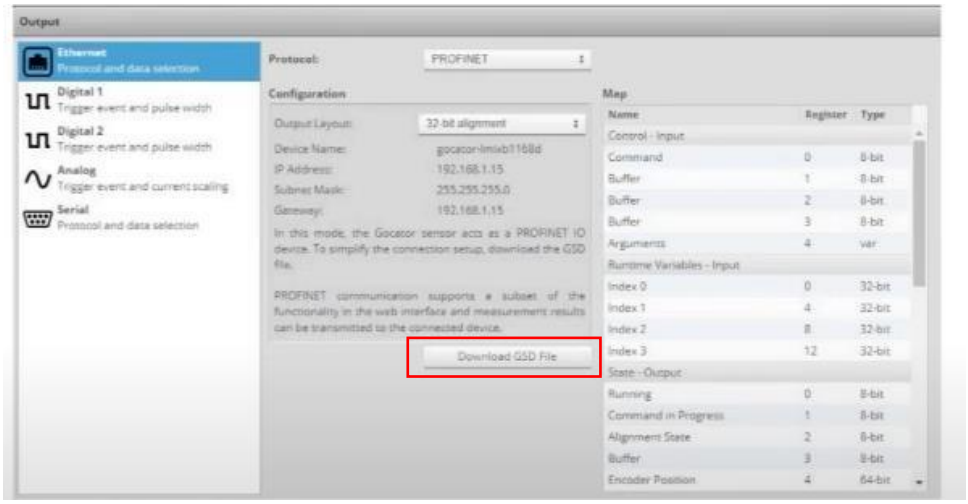


Figura XI.4 Menu OUTPUT download GSD file

Após a transferência da do ficheiro deve-se extrair os mesmos, pois a pasta vem num ficheiro comprimido que deve ser colocado no software do autómato.

Com as configurações todas feitas passa-se então para a programação da câmara de visão 3D.

Ajustar velocidade da Gocator com a velocidade de Robô

Com as configurações feitas faltava agora ajustar a velocidade de processamento do robô com a velocidade de processamento da câmara, para isso é necessário ir ao menu MANAGE (figura XI.5), este menu encontra-se no canto superior esquerdo sendo o primeiro ícone.

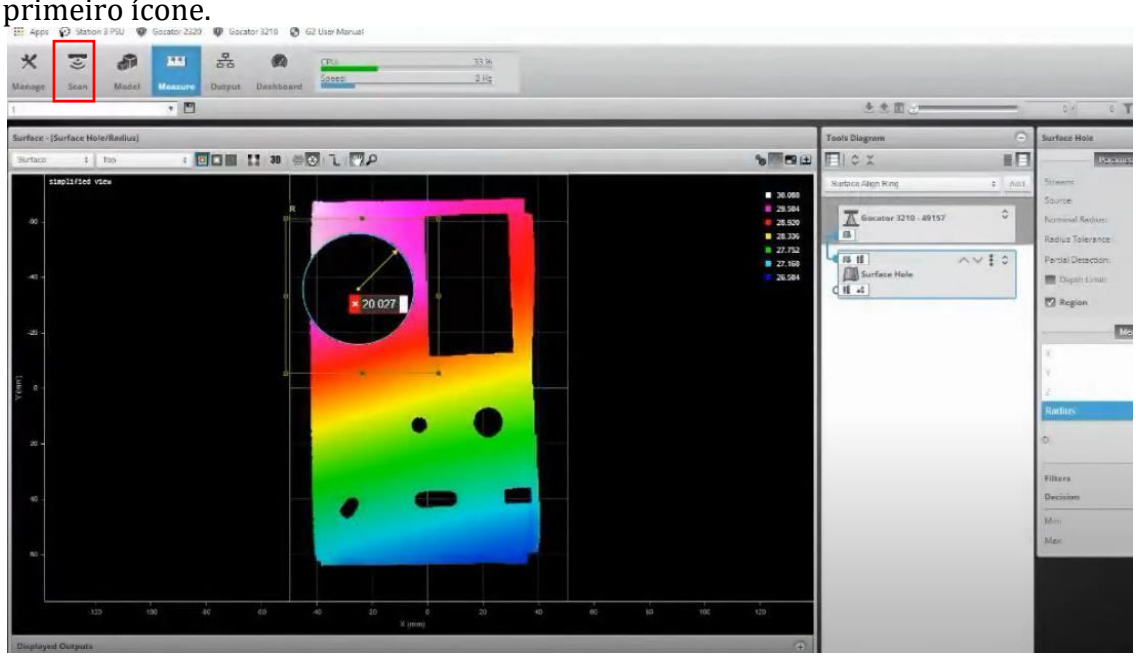


Figura XI.5 Menu principal

Dentro do menu MANAGE (figura XI.6) tem-se a possibilidade de alterar parâmetros câmara como:

- Sensor de sistema - mudar a orientação de deslocamentos da câmara;
- Layout dos dispositivos - caso haja mais que uma câmara;
- Networking - aspectos relacionados com as configurações do IP da câmara;
- Motion and Alignment - onde se pode alterar a velocidade de captura de imagens da câmara;
- Jobs - pode-se carregar programas antigos ou fazer download daquele que se está a trabalhar. Este menu é mais utilizado para fazer backups ou fazer alterações nos mesmos;
- Security - menu onde se pode bloquear algumas funções da câmara ou definir utilizadores;
- Maintenance - fazer resets de fábrica ou fazer alguma atualização ao sistema caso haja;
- Support - ajuda.

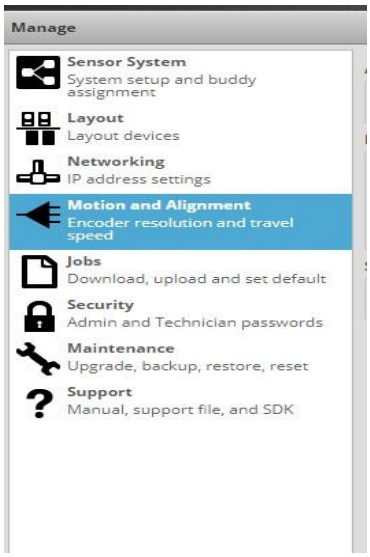


Figura XI.6 Menu Manage

Neste caso pretende-se a ajustar a velocidade, portanto vai-se ao menu Motion and Alignment (figura XI.7). Aqui irá mexer-se na Travel Speed (velocidade de viagem) para uma velocidade próxima ou igual a do robô.

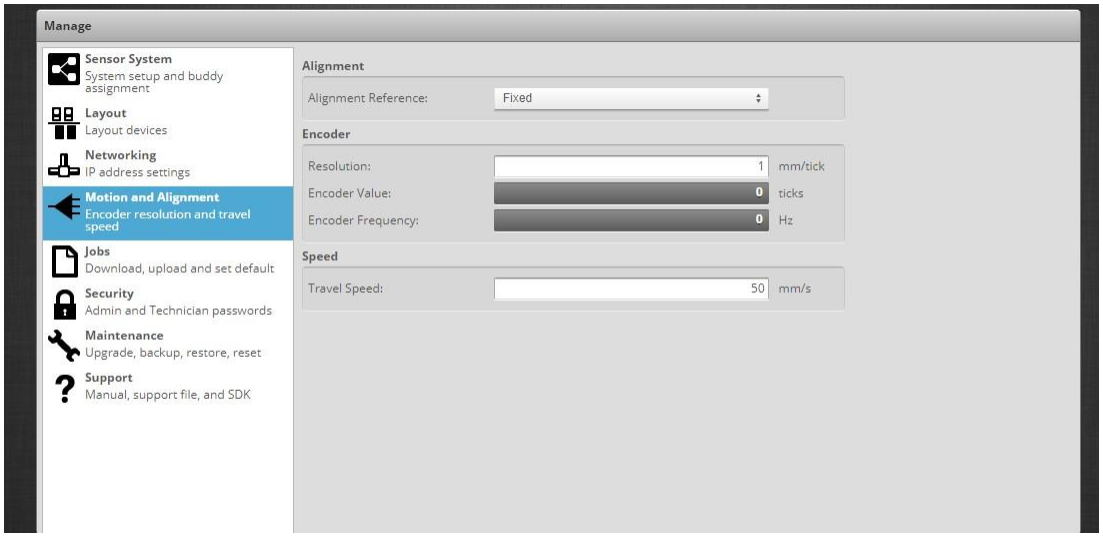


Figura XI.7 Menu Manage, submenu Motion and Alignment

Nesta situação em específico a velocidade de aquisição de imagens foi de 50 mm/s que foi a mesma velocidade configurada no robô. Após a definição da velocidade passou-se aos testes que foram concluídos com sucesso e foram retiradas imagens

Ajustar da área de processamento

Para ajustar a área que a câmara tem de processar com a área real do objeto tem de se ir ao menu SCAN (figura XI.8), este menu encontra-se no canto superior esquerdo sendo o primeiro ícone.

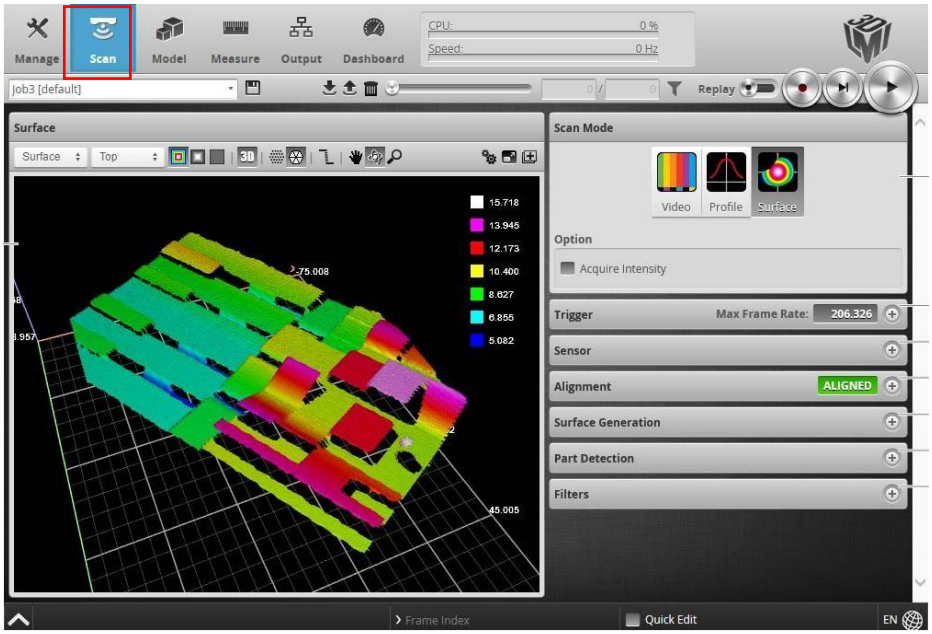


Figura XI.8 Menu principal

No menu SCAN tem de se abrir o submenu SENSOR (figura XI.9) para colocar as dimensões da peça a analisar, podendo também o deslocamento de houver.

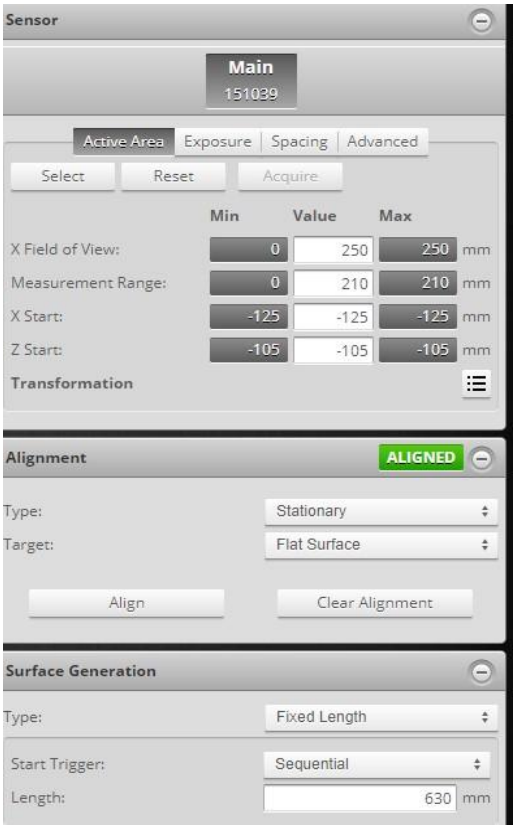


Figura XI.9 Menu SCAN, submenu Sensor

Dentro deste menu SCAN pode ainda alterar:

- Alinhamento – menu Alignment
- Tipo de superfície (tem comprimento fixo ou variável ou rotacional) – menu Surface Generation
- Pode analisar-se dados de varredura para identificar objetos discretos – menu Part Detection
- Aplicar filtros- Filtre Painel

Nesta situação vai-se utilizar os valores máximos que a câmara, ou seja, os valores de campo de visão máxima. A nível de alinhamento escolheu-se uma superfície direita, de modo a ser possível detetar as irregularidades do ambiente. A nível de geração de superfície com uma superfície com comprimento fixo e um disparo de câmara sequencial com um comprimento de 630 mm (valor definido pelo programa ao escolher um disparo sequencia e o comprimento fixo).

Instituto Superior de Engenharia
Politécnico de Coimbra