Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA

Desenvolvimento do Software para Integração de Cobots nos Sistemas Cloison e Porteur

Trabalho de Projeto para a obtenção do grau de Mestre em Engenharia Eletrotécnica

Especialização em Automação e Comunicações em Sistemas Industriais

Autor

Francisco José Carvalho da Cunha

Orientadores

Professor Doutor Nuno Miguel Fonseca Ferreira

Professor Doutor João Paulo Morais Ferreira

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, dezembro 2025

AGRADECIMENTOS

A realização deste projeto não teria sido possível sem o contributo e apoio de várias pessoas e entidades, às quais expresso aqui o meu mais sincero agradecimento.

Em primeiro lugar, gostaria de agradecer ao Professor Nuno Miguel Fonseca Ferreira e ao Professor João Paulo Morais Ferreira pela orientação, disponibilidade e valiosos contributos científicos e académicos ao longo de todo o desenvolvimento deste projeto.

Agradeço também à minha equipa e amigos André Gonçalves, David Lopes, Fernando Lopes e Tiago Pereira, pelo constante apoio, companheirismo e pela prontidão com que sempre estiveram dispostos a ajudar.

À empresa Europneumaq, manifesto o meu reconhecimento pelo suporte e pelas condições proporcionadas para o desenvolvimento deste projeto. Um agradecimento especial ao Engenheiro Rui Bessa e ao Técnico Paulo Magalhães, pelo acompanhamento, disponibilidade e contributo técnico fundamental.

À minha namorada, Filipa Santos, expresso a minha profunda gratidão por todo o apoio emocional, incentivo, dedicação e ajuda prestada ao longo desta jornada.

Por fim, e de forma muito especial, agradeço à minha família, pelo amor, paciência e apoio incondicional desde o início. A confiança que sempre depositaram em mim e o seu apoio contínuo foram determinantes para a concretização deste objetivo.

A todos, o meu muito obrigado.

RESUMO

O presente relatório de projeto consiste no desenvolvimento de um software para a integração de robôs colaborativos (*cobots*) nos processos de montagem do Cloison e do Porteur, elementos presentes na linha de produção de automóveis na fábrica da Stellantis, em Mangualde. Este projeto foi desenvolvido no âmbito da iniciativa "Agenda GreenAuto: Green Innovation for the Automotive Industry", conduzida pelo Laboratório RoboCorp no ISEC em colaboração com a empresa Europneumaq, e está alinhado com os princípios da Indústria 5.0, que promove a colaboração entre os humanos e os robôs bem como a sustentabilidade nos processos industriais.

O trabalho foi estruturado em etapas bem definidas, iniciando-se com a análise dos requisitos funcionais e a seleção criteriosa dos equipamentos mais adequados, como robôs colaborativos, sistemas de visão e periféricos, incluindo aparafusadoras e sistemas de cubas vibratórias. Simulações em ambiente virtual, utilizando ferramentas como o software RoboDK, permitiram prever possíveis problemas e validar as soluções antes da sua implementação prática. Posteriormente, os sistemas foram montados e testados em ambiente real na Europneumaq, apresentando resultados expressivos, como maior eficiência no processo de montagem, aumento da precisão e significativa redução de erros e desperdícios, que viabilizou a sua implementação na fábrica.

Este estudo demonstrou a viabilidade da automação em processos industriais tradicionalmente manuais, provando que a integração de *cobots* melhora a produtividade e a qualidade final dos produtos, além de reduzir o esforço físico e o risco de erros humanos. A pesquisa também destacou a importância das tecnologias digitais, como os sistemas de visão, na transição entre o ambiente de simulação e o ambiente real.

Além de atingir os objetivos propostos, este trabalho sugere futuras expansões, como a implementação desta aplicação nas restantes áreas de produção, melhorias no sistema de visão e a utilização de inteligência artificial para maior eficiência e adaptação. Este projeto reafirma a relevância da automação sustentável e colaborativa na indústria automóvel moderna.

Palavras-Chave: automação de robótica colaborativa, simulação com *cobots*, programação de *cobots,* sistemas de visão, sistemas integrados.

ABSTRACT

The present report consists in the development of a software with the purpose of integrating collaborative robots (cobots) in assembling processes of Cloison and Porteur, elements present in the vehicle production line at the Stellantis factory in Mangualde. This project was developed within the scope of the initiative “Agenda GreenAuto: Green Innovation for the Automotive Industry”, conducted by the RoboCorp Laboratory at ISEC institution in collaboration with the company Europneumaq, and is also aligned with the principles of 5.0 Industry, which promote the collaboration between humans and robots as well as the sustainability of industrial processes.

This work was structured in well-defined steps, beginning with the analysis of the functional requirements as well as a meticulous selection of the most adequate equipment, such as collaborative robots, vision systems and peripherals, including mechanical screwdrivers and vibrating bowl systems. Simulations in a virtual environment, using tools such as the software RoboDk, enable the possible prediction of problems and the validation of solutions for these, before being implemented in a real context. After this, the systems were assembled and tested in a real environment in Europneumaq, showing expressive results, such as an increase in the efficiency and precision of the assembling process as well as a significant reduction of mistakes and waste, which made it viable for this system to be implemented in the installations of the factory.

This study demonstrates the viability of automation in industrial processes that usually operate on manual labor, proving that the integration of cobots improves productivity and the quality of the end product and reduces body fatigue and the risk of injury to the employees. The research conducted also highlights the importance of digital technologies, such as vision systems, in the transition between a simulation environment and a real one.

Besides achieving the proposed goals, this work also suggests future expansions, such as the implementation of this app in the remaining areas of production, improvements on the vision system and the use of artificial intelligence for greater efficiency and adaptability. This project reaffirms the relevance of sustainable and collaborative automation in the modern automobile industry.

Keywords: collaborative robotics automation, simulation with cobots, cobot programming, vision systems, integrated systems.

Índice

Conteúdo

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIGLAS E ABREVIATURAS

AGV	Automated Guided Vehicle
AMR	Autonomous Mobile Robot
CPS	Cyber-Physical Systems
CAD	Computer-Aided Design
Cobots	Robôs Colaborativos
HRC	Human-Robot Colaboration
IA	Inteligência Artificial
IoE	Internet of Energy
IoP	Internet of People
IoS	Internet of Services
IoT	Internet of Things
PLC	Programmable Logic Controller
TCP	Tool Center Point
TCP/IP	Transmission Control Protocol/Internet Protocol
UR	Universal Robots

CAPÍTULO 1 – Introdução

Neste primeiro capítulo será apresentado o enquadramento do projeto, bem como os objetivos e metodologia e por fim a estrutura.

1.1 – Enquadramento

Este projeto enquadra-se numa bolsa de investigação com o intuito da realização do projeto "Agenda GreenAuto: Green innovation for the Automotive Industry" que está a ser desenvolvido pelo Laboratório RoboCorp situado no Instituto Superior de Engenharia de Coimbra juntamente com a empresa Europneumaq. O projeto consiste primeiramente em trabalho de investigação e posteriormente na implementação de robôs colaborativos em duas secções, sendo estas a secção de montagem do Cloison e a secção de montagem do Porteur, de uma fábrica de montagem de veículos, a Stellantis em Mangualde.

Depois de passar pelas três grandes revoluções da indústria, conhecidas como indústria 1.0, indústria 2.0 e indústria 3.0, que alteraram por completo o meio de funcionamento das empresas, é cada vez mais notável a adesão destas à indústria 4.0 e recentemente 5.0, o que irá providenciar novas tecnologias e metodologias. Com a implementação destas indústrias, espera-se a criação de sistemas mais flexíveis e ágeis, capazes de utilizar de forma eficiente a informação e os dados recolhidos para apoiar a tomada de decisão e a execução de processos. Uma das principais características desta implementação será conseguir executar de forma autónoma as decisões e processos para termos sistemas cada vez mais organizados. Por sua vez, a implementação destas indústrias tão sofisticadas necessitará de tempo para a mudança uma vez que trará necessidades educacionais e qualificativas dos operadores a nível de perfis de trabalho [1] .

Sendo a robótica colaborativa um dos principais ramos da indústria 5.0, é cada vez mais notória a introdução de robôs colaborativos (*cobots*) nas empresas devido a uma das suas principais características, a segurança, que é bastante superior à dos robôs tradicionais, também conhecidos por robôs industriais. Isto possibilita a colaboração homem-máquina que permite o trabalho conjunto com funcionários, podendo este ser direto ou indireto [2] .

Combinando os pontos fortes destes *cobots* nomeadamente a segurança, a sua grande precisão e eficiência, com os pontos fortes dos operários, como a capacidade de resolver problemas e a possibilidade de se adaptar a qualquer situação, torna estes robôs uma ótima opção para as empresas, aumentando assim a sua produtividade, eficiência e eficácia, além de ser uma opção mais acessível economicamente [3] .

Neste caso, o objetivo seria a implementação desta dita indústria 5.0 na fábrica, fazendo a automação dos apertos mencionados acima com a implementação de *cobots* juntamente com sistemas auxiliares para a execução correta e precisa do processo pretendido, processo esse que até à sua implementação é feito manualmente pelos trabalhadores.

Com a automação destes processos espera-se um aumento bastante significativo tanto a nível de eficiência como eficácia, reduzindo o tempo necessário para o aperto manual, tal como diminuindo erros e falhas humanas que possam existir devido à percentagem de erro de um *cobot* ser bastante menor num trabalho repetitivo e monótono.

1.2 – Objetivos

Este projeto tem como principal objetivo aprofundar os conhecimentos ao nível da robótica industrial, adquiridos ao longo do mestrado, e da robótica colaborativa que estarão presentes ao longo do projeto.

O projeto que este documento relata tem como principal objetivo a automatização do aparafusamento de 2 pontos de montagem numa fábrica, os quais são, a montagem do Cloison (separador que se encontra entre a zona do condutor e a zona da carga de uma carrinha de mercadorias) e a montagem do Porteur (elemento da roda). Ambas as montagens são efetuadas manualmente por operadores e o objetivo será instalar *cobots* que possam efetuar esse trabalho em ambas as montagens, tornando-as mais eficientes e autónomas.

Plano de trabalho para a realização do projeto:

- Recolha de informação e requisitos funcionais dos protótipos a desenvolver;
- Elaboração de um estudo intensivo de pesquisa para verificar quais os melhores equipamentos para a elaboração do projeto, como por exemplo, robôs, câmaras de visão, sistema de alimentação de porcas, suportes necessários, etc…;
- Realização de desenhos e simulações 2D e 3D;
- Estudo e desenvolvimento de testes e ensaios do sistema de visão artificial e interligação com os robôs colaborativos;
- Concretização de demonstradores de ambos os projetos;
- Aplicação temporária em chão de fábrica para a elaboração de testes;
- Recolha de informação para melhorias e realização das mesmas;
- Possível implementação definitiva na fábrica.

1.3 – Estrutura do projeto

O presente projeto encontra-se organizado nos capítulos descritos de seguida.

- **Capítulo 1 – Introdução:** É apresentado o enquadramento do projeto tal como os seus objetivos.
- **Capítulo 2 – Revisão da Literatura:** Encontra-se a revisão da literatura sobre temas relacionados com o projeto apresentado ao longo deste documento, a evolução da indústria, a robótica ao longos dos anos, a robótica industrial, a robótica colaborativa e sistemas integrados.

- **Capítulo 3 – Descrição do projeto:** Neste capítulo é descrito detalhadamente cada um dos dois projetos existentes, o aperto do Cloison e o aperto do Porteur.
- **Capítulo 4 – Desenvolvimento do projeto:** É descrito todo o desenvolvimento e apresentado os resultados do projeto, passando pela escolha dos periféricos, os programas utilizados em todo o desenvolvimento, simulações em ambiente virtual, montagem e testes em ambiente real, programação dos periféricos e implementação em chão de fábrica do projeto.
- **Capítulo 5 – Conclusões e trabalhos futuros:** São apresentadas as principais conclusões do projeto tal como os objetivos alcançados e respetivos trabalhos futuros.

CAPÍTULO 2 – Revisão da Literatura

Neste capítulo será apresentar a revisão da literatura de alguns temas que são pertinentes para a realização deste projeto. Irei desenvolver temas como a evolução da indústria e as suas revoluções, robôs industriais e robôs colaborativos e por sua vez as diferenças entre ambos. Irei também fazer uma breve menção de sistemas de visão e aparafusamento.

2.1 – A evolução da indústria

Ao longos dos anos, devido à sua grande evolução, o setor da indústria tem desempenhado um papel cada vez mais importante para o país devido ao seu desenvolvimento tecnológico, principalmente a nível económico. Esta revolução veio alterar o método de fabricação da maioria dos materiais, que, anteriormente produzidos de forma artesanal, começaram a ser fabricados por indústrias de produção. Com o avanço da tecnologia foram surgindo novos métodos de produção com a principal função de reduzir o esforço humano, o que levou a um aumento significativo da produção e da economia [4] .

Até ao ano de 2023 mencionavam-se apenas quatro grandes revoluções da indústria que percorreram os últimos séculos, começando na indústria 1.0 e terminando na indústria 4.0 [5] . Por sua vez, no começo de 2023 deu-se o início da introdução da indústria 5.0 que prevê a valorização dos recursos que um humano pode oferecer e pretende aplicar o conceito de trabalho homem-máquina, que consiste na integração dos robôs colaborativos (*cobots*) na área de trabalho de um operador [6] .

Atualmente, tanto a indústria 4.0 como 5.0 têm sido alvo de implementação pelas empresas, apesar de esta última ainda estar numa fase inicial. O principal objetivo da indústria 5.0 é colocar o bem-estar dos trabalhadores no centro dos sistemas de produção, proporcionando o trabalho colaborativo entre os trabalhadores e os *cobots*, para um aumento significativo da produção e da eficiência [7] .

A primeira revolução industrial, indústria 1.0 (Figura 1), foi iniciada na Inglaterra no início dos anos 1760, Século XVIII, que posteriormente se expandiu para a França, Alemanha, Estados Unidos, entre outros países [8] . Esta revolução durou até ao início de 1860, Século XIX e ficou marcada pela utilização de vapor ou água como fonte de energia para a maquinaria presente nas fábricas, o que aumentou bastante a produtividade humana [9] . A nível de transporte também sofreu uma evolução bastante grande, devido aos navios a vapor e alguns anos depois devido às locomotivas, que faziam com que as entregas dos produtos demorassem muito menos tempo. A indústria 1.0 veio melhorar principalmente as indústrias da mineração, do transporte e da agricultura, mas nenhuma destas chegou perto dos resultados positivos tanto a nível de custo de material como de tempo de produção das indústrias têxteis, que foram as que mais beneficiaram desta revolução [4] .

Figura 1 - Indústria 1.0 [10]

Relativamente à indústria 2.0 (Figura 2), ocorreu no final do século XIX e surgiu de uma evolução e aprimoramento das tecnologias criadas na indústria 1.0. Ficou conhecida através do desenvolvimento de novas fontes de energia como o petróleo e a eletricidade, o motor a combustão interna e o crescimento da indústria química [11] .

Outro ponto bastante importante desta revolução foram as linhas de montagem. Henry Ford desenvolveu uma ideia de produção em massa em que, ao invés de a linha de montagem cobrir a montagem integral do carro, os veículos eram construídos parcialmente ao longo do tapete rolante, o que levou a uma produção bastante mais rápida e com menor custo. Com esta inovação começou-se a verificar uma grande afluência às linhas de montagem como o método padrão de produção em massa. Ainda nesta indústria deu-se o desenvolvimento de dispositivos eletrónicos e a construção da primeira linha de integração [4] .

Figura 2 - Indústria 2.0 [10]

Passando à terceira grande revolução industrial, indústria 3.0 (Figura 3), esta existiu durante o século XX, mais precisamente de 1950 a 1970, e ficou conhecida como a revolução técnico-científica. Esta era foi marcada pelo grande avanço tecnológico nas áreas das telecomunicações, informática, eletrónica, mas sobretudo na área da robótica. Esta indústria evoluiu ainda mais as linhas de produção através da automação, o que permitiu às indústrias acompanharem a crescente procura no mercado [9] .

Em termos mais específicos, nesta revolução existiu uma enorme procura e, por sua vez, uma enorme produção de componentes ligados à lógica digital como transístores MOS, chips de circuitos integrados como computadores, telefones sem fios, microprocessadores e a internet. Com a inclusão destes transístores e chips nas máquinas, estas tornaram-se possíveis de automatizar, o que levou a uma redução do esforço, um aumento da velocidade e a um aumento da precisão [4] . Importa ainda destacar um marco bastante importante desta era, que foi o desenvolvimento do *Programmable Logic Controller* (PLC), criado para o propósito de controlo e monitorização de processos das linhas de montagem e das máquinas.

Figura 3 - Indústria 3.0 [10]

O conceito de indústria 4.0 (Figura 4) surgiu no século XXI, no ano de 2011, na Alemanha, e foi desenvolvida com o intuito de criar valor para as empresas com a inserção de conceitos como "*Cyber-Physical Systems*" (CPS), "*Internet of Things*" (IoT), "*Internet of Services*" (IoS), "*Internet of People*" (IoP) e "*Internet of Energy*" (IoE). Com a integração destes conceitos e utilizando sistemas de redes digitalizados, interligados e descentralizados, as empresas conseguiram aumentar a sua produtividade e fiabilidade com um desempenho sustentável [12] bem como obter um maior controlo e flexibilidade.

O CPS é um sistema utilizado pelas máquinas para que estas tenham uma comunicação mais inteligente independentemente de onde se encontram no ambiente físico. É um sistema onde os seus componentes, tanto físicos como de software, estão altamente interligados e são capazes de operar em diferentes locais e níveis temporais, conforme o contexto em que sejam necessários [4] .

A IoT, IoS, IoP e IoE consistem na interligação, como os seus nomes indicam, das coisas, dos serviços, das pessoas e da energia com a internet, o que leva a um controlo muito superior de todos estes pontos de uma indústria. Este sistema veio revolucionar o controlo e a supervisão remota de todos os conceitos que são importantes numa empresa, devido à possibilidade de ser tudo monitorizado e supervisionado via internet.

Para além destes conceitos, a indústria 4.0 também ficou conhecida por tecnologias como "*Cloud Computing*", a "*Big Data*", a cibersegurança, os Robôs Autónomos, a Simulação e a Realidade Aumentada [13] .

Com a implementação destas tecnologias deu-se o aparecimento de um novo conceito, o "*smart factory*", ou fábrica inteligente, que se define como um sistema ciberfísico integrado que possibilita a troca de informações e de dados com diferentes características, oferecendo uma operação eficiente e estável [14] .

Figura 4 - Indústria 4.0 [10]

Por fim temos a quinta revolução industrial (Figura 5), também conhecida como indústria 5.0, que será uma evolução da indústria 4.0 [15] . Com o seu aparecimento ainda bastante recente, é muito superficial a informação que existe sobre a mesma, mas sabe-se que esta terá uma ênfase na sustentabilidade, na resiliência, e sobretudo na indústria centrada no ser humano. Pode-se dizer que a quinta revolução industrial irá trazer a sinergia de trabalho entre os humanos e as máquinas autónomas, o tão conhecido sistema homem-máquina, que por sua vez tentará tirar o melhor dos dois mundos, isto é, tentará juntar a capacidade intelectual e a criatividade da parte do ser humano com os fluxos de trabalho dos sistemas inteligentes [16] . A combinação deste sistema homem-máquina, onde teremos máquinas de alta potência em conjunto com técnicos altamente treinados, irá permitir às empresas um processo de produção mais eficiente, sustentável e seguro [4] .

Podemos ainda afirmar que algumas das principais tecnologias que possibilitarão que a indústria 5.0 se possa expandir são os materiais avançados, melhorando as propriedades e criando novas funcionalidades para os materiais, a fabricação e processamento inteligente, que consiste na análise de dados em tempo-real com a utilização da Inteligência Artificial (IA) e o *machine learning* para tentar prever possíveis eventos críticos e atuar de acordo com a situação de forma a evitar possíveis problemas, a nanotecnologia com o desenvolvimento de materiais e dispositivos em

nano escala que irão melhorar a economia sustentável, conservando recursos e protegendo o ambiente e por fim a fabricação sustentável [15] .

Figura 5 - Indústria 5.0

2.2 – A robótica

Nesta secção irei elaborar uma introdução sobre a robótica, explicando o seu início, passando para o seu surgimento e a evolução dos robôs ao longo do tempo, terminando com alguns exemplos dos robôs mais conhecidos atualmente.

2.2.1 – Início da robótica

Ao longo da evolução da indústria, tornou-se cada vez mais notória a necessidade de melhorar o processo de produção tornando-o mais prático, rápido e eficiente, surgindo a área da robótica no início do século XX, possibilitando a melhoria destes aspetos.

Em 1921, na peça de teatro Os Robôs Universais de Rossum (R.U.R. – Rosumovi Umeli Roboti) do escritor checo Karel Capek, surgiu pela primeira vez a palavra robô que vem do termo robota que significa trabalhos forçados ou servidão [17] .

Já em 1950, Isaac Asimov surgiu com as três leis da robótica, definidas por [18] :

1. Um robô não pode causar dano ao ser humano ou, por inação, permitir que o ser humano se fira;
2. Um robô deve obedecer às ordens que lhe são dadas pelo ser humano, exceto quando estas entram em conflito com a 1ª Lei;
3. Um robô deve proteger a sua própria existência, desde que essa mesma proteção não interfira com a 1ª e 2ª Lei.

Como podemos verificar, estas leis visam o bem-estar e a segurança do ser humano e, apesar de muito simples e básicas, devem estar sempre presentes na programação do robô para manter o bom funcionamento e segurança dos trabalhadores à sua volta.

Uns anos mais tarde, já no seculo XXI, Mark W. Tilden apresentou outras três leis da robótica, nomeadamente [19] :

1. Um robô deve proteger a sua existência a todo o custo;
2. Um robô deve obter e manter acesso à sua própria fonte de energia;
3. Um robô deve procurar continuamente por uma fonte de energia melhor.

No que diz respeito a estas leis, é notório o foco exclusivo na proteção do robô, negligenciando a segurança humana. Isto acaba por gerar uma controvérsia significativa entre os especialistas ao compararem as leis de ambos os autores, dada a sua considerável contradição.

2.2.2 – Surgimento e evolução dos robôs ao longo do tempo

Um robô industrial é defino como "*uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial*" (ISO (International Organization for Standardization) 10218:2011).

Um robô consegue realizar atividades físicas, em modos operacionais distintos, mais automatizadas ou não, auxiliando as tarefas humanas em vários meios. Estes surgem como um grande auxílio aos recursos humanos uma vez que podem efetuar tarefas repetitivas, com um grau elevado de esforço, ou até mesmo tarefas perigosas que são inadequadas do ponto de vista ergonómico, isto é, podendo prejudicar a saúde e o bem-estar dos seres humanos [20] .

Em 1954, George Devol, também conhecido por "Avô da Robótica", fez o registo de uma patente de um manipulador que foi desenvolvido para realizar tarefas repetitivas. Esse manipulador foi chamado de Unimate, e foi considerado o primeiro robô industrial (Figura 6). Dois anos depois, numa festa, Devol e Joseph Engelberger, que ficou conhecido por "Pai da Robótica", encontraram-se e, ao discutirem robótica e as leis de Isaac Asimov, traçaram o plano que mudaria o mundo da manufatura.

Desta junção nasceu a empresa Unimation que foi a responsável pela criação deste robô, que por sua vez, em 1961, vendeu o primeiro exemplar para a empresa General Motors onde foi implementado numa linha de montagem em Trenton. A sua função passava apenas por mover e ordenar as peças de metal fundido de um ponto para o outro [21] . Com a ideia de expandir o negócio, em 1966, Engelberger licenciou o robô à empresa Nokia of Filand para que este pudesse ser produzido na Escandinávia e na Europa Oriental, e em 1969 fez o mesmo com a Kawasaki Robotics para que pudesse ampliar a sua produção e comercialização para a Ásia [22] .

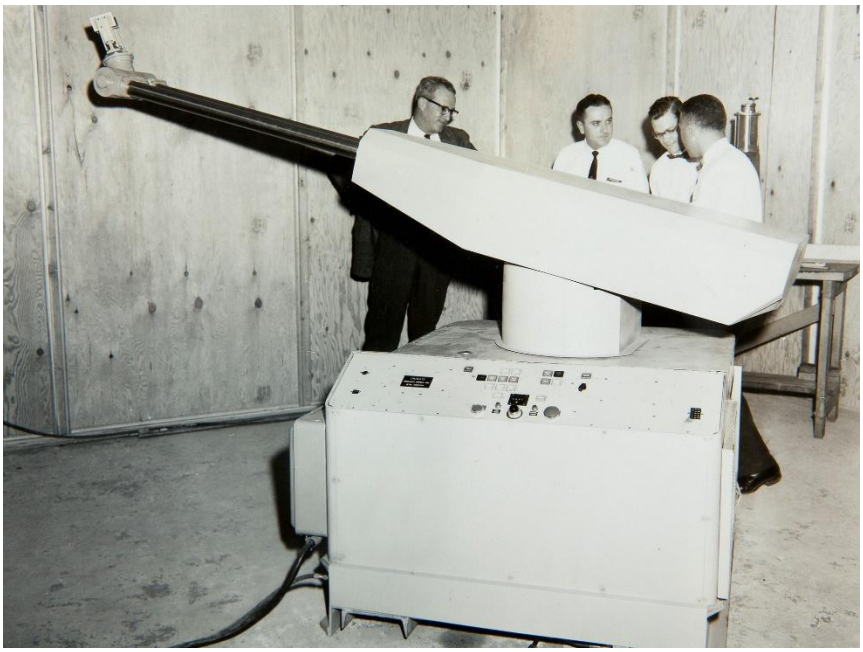


Figura 6 - Primeiro Robô Industrial, Unimate 1954 [21]

Entre 1966 e 1972 foi desenvolvido no “Artificial Intelligence Center of SRI” o primeiro robô móvel, que era dotado de IA, o Shakey (Figura 7), capaz de navegar de forma inteligente no ambiente envolvente.

Já em 1973, a empresa KUKA deixou de usar o Unimate e começou a desenvolver os seus próprios robôs, surgindo o primeiro robô com seis eixos acionados electromecanicamente, o Famulus (Figura 8).

Em 1979, o Japão desenvolveu o primeiro robô a motor, o Nachi (Figura 9), que veio substituir os antigos robôs hidráulicos, o que deu origem á era dos robôs elétricos.

No fim do século XX, a Motoman introduziu o primeiro sistema de controlo de robô (MRC) que possibilitava o controlo sincronizado de dois robôs. O MRC proporcionou também a opção de criar e editar a lista de trabalho do robô a partir de um computador comum.

Em 2003 foram enviados os primeiros veículos robóticos, o Spirit e o Opportunity (Figura 10), para Marte. Três anos depois foi apresentado pela KUKA o primeiro robô leve (Figura 11). Este pesava apenas 16kg, em contraste com o Unimate, que pesava 1,5 toneladas.

Por sua vez, em 2011, foi lançado na Estação Espacial Internacional o primeiro robô humanoide, o R2B (Figura 12). Já em 2016, foi apresentado o Meca500 (Figura 13), o robô industrial mais pequeno com seis eixos, este pesava apenas 4,6kg.

Entre estes existem muitos mais robôs desenvolvidos e implementados. Uma das empresas que tem vindo a destacar-se bastante nos últimos anos é a Doosan Robotics. Esta desenvolve *cobots* (Figura 14) para várias aplicações, contando com

quatro séries diferentes de robôs, as séries H (Hight Power), M (Masterpiece), A (Agile) e E (Edge).

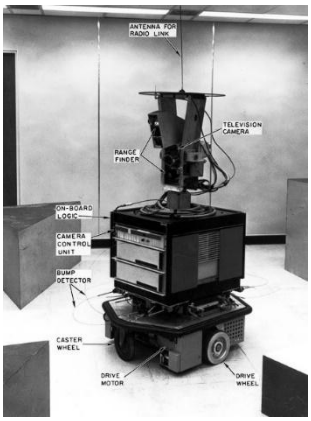


Figura 7 - Shakey 1972 [23]

Figura 8 - Famulus 1973 [24]

Figura 9 - Nachi 1979 [25]

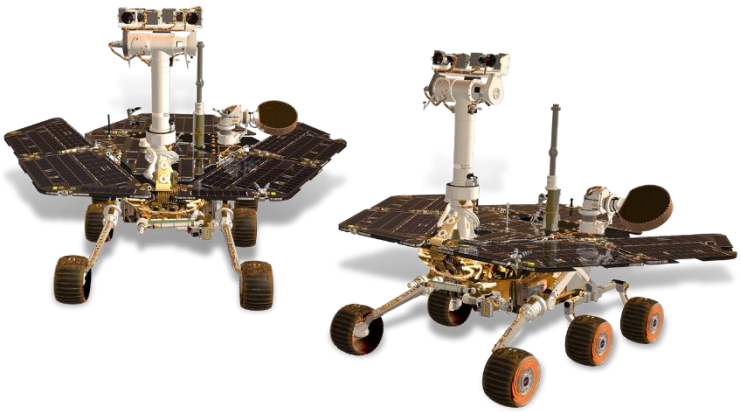


Figura 10 - Spirit e Opportunity 2003 [26]

Figura 11 - Robô leve da KUKA 2006 [27]

Figura 12 - Robonaut (R2B) 2011 [28]

Figura 13 - Meca500 2016 [29]

Figura 14 - Doosan M1013 [30]

2.2.3 – Robôs mais conhecidos da atualidade

A nível da robótica, é bastante notável a evolução e o desenvolvimento que têm vindo a mostrar ao longo dos anos. Atualmente são vários os exemplos de robôs que existem para ajudar nas várias áreas do quotidiano. Na Figura 15 podemos verificar alguns dos robôs mais conhecidos da atualidade.

Um destes robôs é o Atlas, desenvolvido pela Boston Dynamics, que é dotado de potência e equilíbrio que o fazem ser capaz de demonstrar técnicas avançadas de agilidade e atletismo. É constituído por 28 juntas com atuação hidráulica, é capaz de se movimentar a 2,5 m/s, pesa 89kg e tem 1,5 metros de altura. Este humanoide foi criado com o objetivo de se conseguir mover com velocidade e destreza tendo ainda a mobilidade, perceção e inteligência para ser capaz de coexistir com os humanos [31] .

De seguida temos a Sophia, que foi desenvolvida pela Hanson Robotics, esta é uma humanoide altamente avançada que é dotada de IA, mas o que chamou a sua atenção internacionalmente foi a sua aparência e capacidades muito próximas das dos humanos. Esta tem a capacidade de interagir e comunicar com as pessoas aprendendo com as experiências, isto é, à medida que vai comunicando vai melhorando a sua inteligência devido à IA, também é dotada de visão computacional e é capaz de fazer 62 expressões faciais o que torna mais “humano” conversar com ela sendo que esta é capaz de manter uma conversa fluida, fazendo e respondendo a perguntas [32] .

Com algumas semelhanças à Sophia, temos a Pepper, uma humanoide desenvolvida pela Aldebaran Robotics, que em 2015 foi adquirida pela SoftBank. Como a própria empresa diz, esta foi concebida para as pessoas, isto é, para comunicar, assistir e partilhar conhecimentos com elas, podendo ajudar tanto em casa, como no trabalho sendo capaz de criar relações reais com as pessoas à sua volta [33] .

Passando para outro espectro, foram desenvolvidos alguns robôs para a área da medicina, a Moxi*,* da Diligent Robotics, foi desenvolvida com o intuito de ajudar a equipa médica nos hospitais passando por assistente clínica fazendo tarefas indiretas com os pacientes, como por exemplo, recolher e distribuir medicamentos, equipamento médico, materiais, entre outras coisas, fazendo com que os médicos não tenham de perder tempo a fazer essas tarefas podendo dar mais atenção e cuidado aos pacientes. Esta possui um braço robótico, IA, sensores de segurança para a sua navegação e um armazém que permite transportar uma grande variedade de medicamentos e ferramentas [34] .

Outro robô muito conhecido neste ramo é o Da Vinci, criado pela Intuitive Surgical, este é mais antigo tendo sido desenvolvido em 1999, mas continua a ser um dos robôs mais conhecidos e utilizados na área da medicina. O Da Vinci é um robô cirúrgico que foi desenvolvido para ajudar em procedimentos pouco invasivos. Este é constituído por quatro braços equipados com equipamentos cirúrgicos e câmaras que são

controlados pelo médico remotamente a partir de uma consola onde este controla os movimentos tanto dos braços como dos equipamentos e das câmaras [35] .

Por fim, relativamente à robótica móvel que é responsável pelo transporte, existem os AGVs (*Automated Guided Vehicle*). Estes são veículos de condução autónoma que são amplamente utilizados na indústria para realizar o transporte dos materiais de forma totalmente independente num ambiente predefinido. São equipados com diversos tipos de sensores que permitem a navegação de forma segura, evitando obstáculos e possíveis colisões de forma autónoma.

Atlas | Sophia | Pepper

Moxi | Da Vinci | AGV

Figura 15 - Robôs mencionados [36]

2.3 – Robótica industrial

Relativamente à robótica industrial, um robô industrial é, de acordo com a ISO 8373, um manipulador multifuncional, automaticamente controlado, reprogramável e programável em três ou mais eixos, podendo ser fixado num local ou fixado a uma plataforma móvel, que por sua vez, é utilizado em aplicações de automação em ambientes industriais [37] .

Normalmente, estes robôs efetuam atividades repetitivas ou perigosas para o operador, tendo de ser máquinas bastante fortes para conseguirem carregar cargas extremamente pesadas, como por exemplo a montagem de um automóvel, ou serem extremamente precisos e rápidos para efetuarem tarefas como a construção de componentes eletrónicos. Uma vez que têm de ser bastante fortes ou extremamente

precisos e rápidos o que por norma acontece é que estes são bastante grandes e perigosos para as pessoas, o que leva a que as indústrias tenham de cercar o robô com cercas para a proteção dos operadores como podemos verificar na Figura 16 [38] . Com estas cercas os operadores nunca entram na zona de trabalho do robô, sendo que, para o fazer terão de carregar no botão de paragem que existe na cerca para parar o trabalho do robô ou ainda, como forma de segurança adicional, a cerca contém uma porta para a entrada do operador, aquando da abertura dessa porta é enviado um sinal para o robô para o mesmo parar a sua tarefa para não existir perigo para o trabalhador.

Figura 16 - Cerca de Segurança [39]

Um sistema robótico é constituído por três componentes principais, os quais são, o manipulador, o controlador do robô e o *teach pendant,* quando disponível, como é possível verificar na Figura 17 [40] . Normalmente este sistema é integrado com equipamentos adicionais tais como o *gripper,* que dependendo da aplicação tem inúmeras formas diferentes, o *conveyor*, que é um sistema de transporte que é utilizado para o transporte de peças de um ponto para outro de forma autónoma, elevadores, entre vários outros objetos para completar a linha de produção.

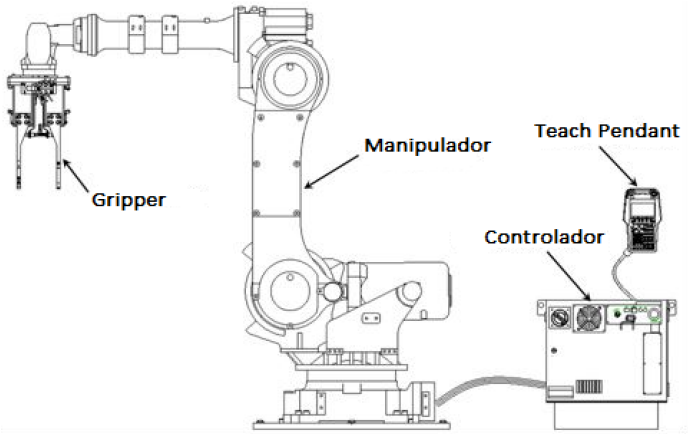


Figura 17 - Sistema robótico [40]

Relativamente ao manipulador, considerando a estrutura física do robô, este é constituído por juntas, elos que ligam as juntas, atuadores e sensores. Como referido anteriormente, estes são constituídos por três a seis eixos, sendo o mais usual de seis eixos, onde o braço robótico pode ser comparado com um braço humano, uma vez que seria esse o seu intuito.

Um manipulador de seis eixos é constituído por base, cintura, ombro, braço, cotovelo, antebraço e pulso e pode ser identificado no robô como se pode verificar na Figura 18. As duas características mais importantes na altura de escolher qual o modelo do robô que se quer adquirir é a distância que o manipulador consegue alcançar e o *payload*, o peso que o braço robótico consegue suportar.

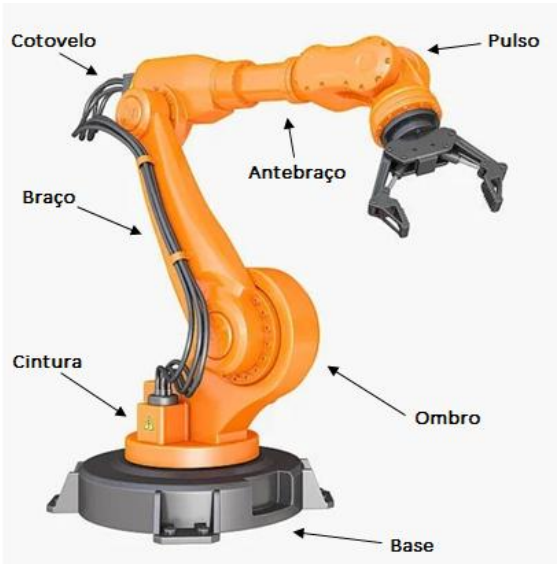


Figura 18 - Constituição do Manipulador (Adaptado de [41])

No controlador é onde se dá todo o controlo do sistema robótico, constituído pela fonte de energia que vai energizar todo o sistema robótico, local onde é armazenado e

executado o programa elaborado para o robô e onde ficam localizados os sinais de entrada e saída do sistema, resumindo, é o cérebro de todo o sistema.

Figura 19 – Controlador [42]

Por fim, a maioria dos sistemas robóticos são preparados para que a sua programação possa ser feita usando o *teach pendant* (um dispositivo de controlo portátil) onde é possível que, um operador que tenha a formação para tal, faça a programação da tarefa que o robô terá de efetuar. Esta programação é feita de modo manual no *teach pendant* que, por sua vez, está conectado ao controlador que receberá a tarefa que o operador realizou. Neste dispositivo o trabalhador fica com o controlo total do robô, mas para a realização da tarefa, terá de estar na zona de trabalho do robô, logo irá sempre estar a correr riscos, daí ser necessário que o operador tenha obtido a formação necessária para a sua utilização.

Figura 20 - Teach Pendant [43]

Existem inúmeros tipos de robôs industriais para todas as aplicações existentes de acordo com a *International Federation of Robotics,* uma organização que estuda a tendência da indústria robótica. De entre eles os mais utilizados são [44] :

1. **Robôs cartesianos**: também conhecidos por robô de pórtico, são desenvolvidos com um sistema de coordenadas retangulares como se fossem os eixos XYZ de um gráfico. Apresentam uma mecânica linear constituída por três juntas deslizantes com um formato muito idêntico aos eixos XYZ como mencionado anteriormente. Uma vez que se movem em linhas retas estes robôs são uma ótima opção para aplicações onde seja necessário mover objetos devido à sua alta precisão e repetibilidade.

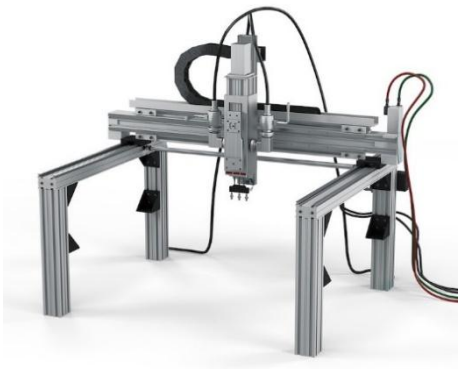


Figura 21 - Robô Cartesiano [45]

2. **Robôs *SCARA***: são constituídos por três eixos sendo que, de forma distinta dos cartesianos, estes têm dois eixos rotacionais e um eixo linear, onde os eixos rotacionais são relativos a movimentos horizontais, normalmente nos eixos XY, e o eixo linear refere-se a movimentos verticais, no eixo Z. Conforme as aplicações, estes planos podem ser invertidos relativamente aos eixos rotativos e ao eixo linear. Estes são capazes de realizar movimentos rápidos o que os torna muito adequados para trabalhos de montagem de objetos eletrónicos.

Figura 22 - Robô SCARA [46]

3. **Robôs articulados**: também conhecidos pela sua forma idêntica a um braço humano, estes robôs possuem pelo menos 3 juntas rotativas, podendo ser constituídos por mais, o que irá aumentar a amplitude do seu movimento. Normalmente, o mais comum é ver estes robôs com seis juntas, o que lhes permite alcançar qualquer ponto no seu espaço de trabalho. Sendo tão móveis e flexíveis, estes robôs são perfeitos para tarefas de montagem, embalamento, pintura e até mesmo trabalhos na área da saúde, como, por exemplo, cirurgias.

Figura 23 - Robô articulado [47]

4. **Robô Delta/Paralelo**: são máquinas construídas com três braços que são controlados individualmente e que têm uma base triangular. São robôs conhecidos pela sua aparência parecida com a de uma aranha, sendo bastante rápidos e ágeis. Devido à sua velocidade e agilidade são normalmente utilizados em tarefas que necessitem de velocidade de forma repetitiva, como operações de seleção de objetos ou embalamento. O Omron Quattro é o robô Delta mais rápido que existe na indústria, constituído por quatro braços e não três, como os robôs Delta usuais, sendo este capaz de fazer trezentas escolhas por minuto.

Figura 24 - Robô Delta e Omron Quattro, respetivamente [48] [49]

5. **Robô cilíndrico**: possui pelo menos uma junta rotativa na sua base, daí o seu nome, e uma junta prismática, que faz a conexão entre os dois elos, a base e a garra. A junta rotativa tem um movimento rotacional ao longo do eixo da junta enquanto a junta prismática tem um movimento linear. Estes robôs têm a capacidade de trabalhar em espaços apertados e são capazes de transportar cargas pesadas, o que os leva a aplicações como soldadura, embalamento e manutenção.

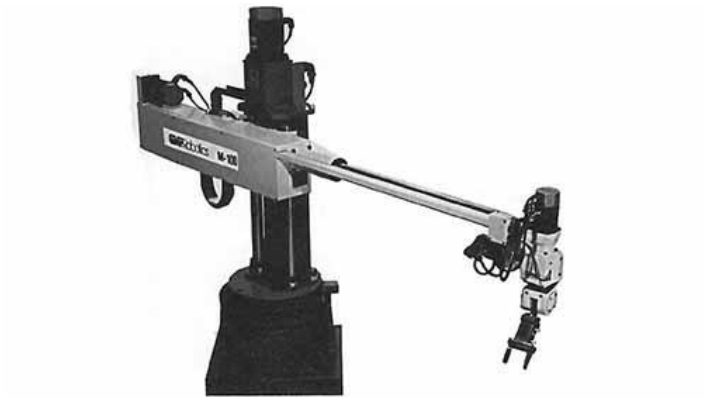


Figura 25 - Robô Cilíndrico [50]

6. **Robô polar**: também conhecidos por robôs esféricos, consistem num braço que é capaz de realizar movimentos multidirecionais a partir de uma base fixa. Este é constituído por uma junta de torção que liga o braço à sua base e uma combinação de duas juntas rotativas com uma junta prismática. São robôs que têm movimentos esféricos e os seus eixos formam um sistema de coordenadas polares, daí o seu nome. Têm um alcance grande à sua volta, tendo a limitação de ter um alcance vertical curto.

Figura 26 - Robô Polar [51]

Atualmente, os robôs industriais são os mais utilizados na indústria. São máquinas perfeitas para trabalhos repetitivos e precisos pois são capazes de efetuar uma tarefa vezes sem conta, sempre da mesma maneira, com uma chance de erro muito baixa,

ao contrário de um humano que ao fim de algumas horas começa a ficar cansado, o que vai afetar a sua atenção e precisão. Podem ainda fazer trabalhos perigosos sem correr risco de ferimento, tal como, trabalhar durante horas sem qualquer descanso, o que vai diminuir o risco de lesão para os operadores em certos trabalhos e vai aumentar a produtividade.

Relativamente às suas vantagens e desvantagens, estes têm os seus prós e contras. Dentro das vantagens, os robôs industriais são conhecidos por [52] :

1. A sua **eficácia e produtividade**, pois, a sua implementação permite o trabalho contínuo durante dias inteiros. Estes podem ser programados para realizar tarefas rápidas, confiáveis, precisas e repetitivas, o que leva a um aumento da produtividade e, por sua vez, a um aumento na produção global da própria fábrica;
2. Têm uma **grande adaptabilidade no sistema**, isto é, rapidamente é possível mudar o robô para um novo posto de trabalho com um novo plano de trabalho;
3. Fornecerá um **maior padrão nos produtos,** uma vez que os robôs industriais foram desenvolvidos para efetuar tarefas repetitivas e sem variação, como mencionado anteriormente, o que leva a uma percentagem de erro muito pequena, o que por sua vez faz com que a uniformidade do produto seja maior, mantendo a sua qualidade, o que não acontece com um operador devido à chance de erro do operador ser superior à de um robô industrial;
4. Com a implementação dos robôs industriais há uma **maior economia nos custos**, pois estes podem ajudar a reduzir os custos totais de fabricação de 20% a 60%, reduzindo os tempos de produção e aumentando a produtividade geral. Apesar do grande investimento que se tem no início ao adquirir um robô, o mesmo, a longo prazo, irá ter um retorno muito maior devido às reduções mencionadas;
5. Existe também um **aumento na segurança no trabalho,** pois os robôs são desenvolvidos para realizar tarefas que são perigosas e arriscadas para operadores, tais como aplicações com ferramentas perigosas ou grandes pesos. Podemos afirmar que para além destas ferramentas os robôs estão seguros contra exposição a fogo, fumos, toxinas e lasers, algo que iria afetar diretamente a saúde dos trabalhadores. Assim, ao atribuir estas tarefas às máquinas, protegendo os seres humanos, promover-se-á a saúde no trabalho a longo prazo, devido à possível diminuição de acidentes ou lesões no local de trabalho;
6. Por fim, podemos ainda dizer que os robôs permitem que os operadores **se foquem em tarefas mais importantes ou com maior valor** para a empresa, uma vez que, sendo os robôs a fazerem as tarefas repetitivas e perigosas, os trabalhadores poderão focar o seu tempo em tarefas mais importantes que os robôs não consigam realizar, aumentando assim a eficácia da fábrica.

Por outro lado, esta implementação também traz algumas desvantagens como:

1. O **custo inicial** que, como já mencionado, é um custo elevado relativo à compra, implementação e programação do robô, que terá de ser notado tal como alguns custos a longo prazo, como possíveis componentes adicionais e manutenções. Normalmente este investimento inicial é mais prejudicial para empresas pequenas por não terem tantas possibilidades de investir numa máquina que não é propriamente barata;
2. A sua **segurança** também pode ser um ponto negativo uma vez que os robôs são grandes, movem-se a velocidades elevadas e os mais antigos não têm um sistema de segurança muito desenvolvido, o que leva a possíveis lesões aos operadores. Daí muitas das fábricas usarem as chamadas “gaiolas” e divisórias, que consistem numa cerca de rede de ferro em volta da área de trabalho do robô, para não haver contacto direto com o robô por parte do operador;
3. Uma desvantagem também bastante considerável é o **nível de formação** que os operadores da empresa necessitam, pois, com a aquisição de um robô, acresce a necessidade da sua programação, o que não é algo que um operador sem formação seja capaz de efetuar. Assim, a empresa terá de contratar ou formar os seus trabalhadores para que sejam capazes de trabalhar com o robô, o que irá encarecer ainda mais a aquisição desta máquina.

Apesar de existirem algumas desvantagens ao adquirir um robô industrial, acredito que a sua aquisição fornecerá mais vantagens do que desvantagens. Atualmente são raras as fábricas que funcionam integralmente por operadores, pois cada vez tem sido maior a afluência na implementação de robôs, devido ao grande aumento de produção e ao retorno a nível financeiro que os mesmos trazem.

De acordo com o estudo efetuado pela HowToRobot, os pontos mais comuns de implementação de robôs são no manuseamento, no chamado pick and place (apanhar e colocar), no embalamento, na paletização, na montagem, na soldadura entre inúmeras outras aplicações, como é possível verificar na Figura 27 [53] .

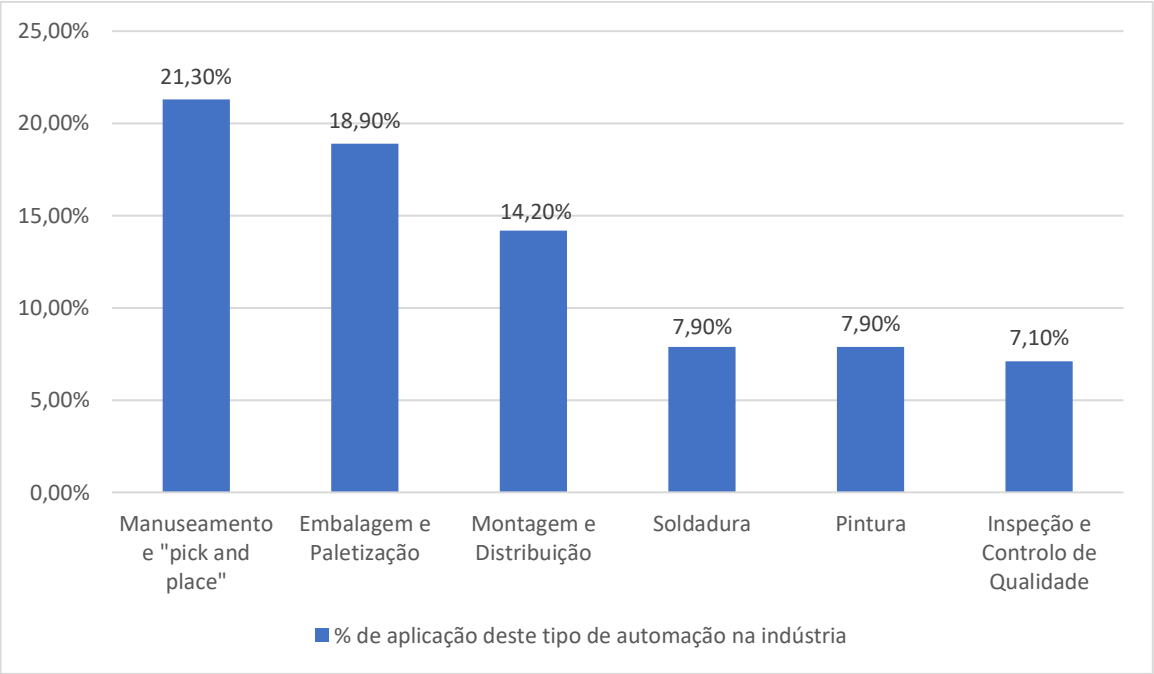


Figura 27 - Estudo efetuado pela *HowToRobot* [54]

A robótica tem vindo a ampliar-se pelo mundo todo com novos desenvolvimentos e criações para inúmeras áreas da indústria, sendo cada vez mais nítida a ampliação para outras áreas de aplicação destes robôs com o aparecimento da IA. Devido a esta ampliação e ao aparecimento de várias novas empresas de desenvolvimento e produção destes robôs, estes também se têm vindo a tornar cada vez mais acessíveis para pequenas e médias empresas. Dentro das inúmeras marcas de robôs existentes, as que mais se têm vindo a destacar até à atualidade são [55] :

1. **FANUC** – Uma empresa sediada no Japão que é conhecida pelos seus robôs industriais amarelos, esta produz atualmente 5000 robôs por mês, sendo que tem uma gama de mais de 100 modelos diferentes para várias aplicações;
2. **ABB** – Esta, já criada na Suíça, conta com mais de 300000 sistemas de automação industriais instalados em inúmeras empresas, sem dúvida também umas das maiores empresas neste ramo da robótica;
3. **KUKA** – Juntamente com a FANUC e a ABB, a KUKA é também uma das empresas mais conhecidas no mundo da robótica industrial. Com a sua sede na Alemanha, esta já está neste ramo desde 1898. Para além das suas aplicações nos ramos automóvel e na eletrónica, tem também aplicações na área da saúde;
4. **Stäubli** – Também com a sua sede principal na Suíça, é conhecida pelas suas aplicações industriais e na mecatrónica, fornecendo para quatro divisões dedicadas, conectores elétricos e de fluidos, na robótica e no têxtil. No seu portfólio têm robôs industriais de quatro a seis eixos, robôs colaborativos, bem como robôs móveis e AGVs;

5. **Denso** – Começou o desenvolvimento dos seus primeiros robôs em 1967, e veio implementá-los 3 anos depois na sua própria fábrica. Atualmente é uma das principais empresas no fornecimento de tecnologia automóvel para grandes empresas como a Toyota, a Honda, entre outras. Em 2014 estenderam-se para a área da farmacêutica, desenvolvendo uma gama de robôs farmacêuticos, a série VS, com uma ampla aplicação na área da saúde. Para além da indústria automóvel e da saúde, têm também implementações nos ramos aeroespacial, biomédica, ciências biológicas, etc.

Figura 28 - Logos das respetivas empresas mencionadas em cima [56-60]

Existem inúmeras empresas que desenvolvem este tipo de robôs industriais sendo estas as mais conhecidas e relevantes.

Falando agora da programação, esta pode ser classificada em dois tipos: programação online e programação offline. A programação online envolve contacto direto com o robô, o que leva a que o profissional tenha de estar junto da máquina. Esta programação é efetuada por operadores qualificados, recorrendo ao *teach pendant*, onde se realiza a programação que será posteriormente utilizada na execução da tarefa atribuída ao robô na fábrica. Para isto, o robô é guiado pelo caminho que é desejado, guardando pontos estratégicos da sua rota, para posterior movimento do mesmo, a chamada trajetória. Ao guardar este conjunto de coordenadas na área de trabalho do robô, este já será capaz de efetuar uma trajetória, que existirá devido à movimentação de ponto a ponto[61] . Este tipo de programação é normalmente utilizado para processos de fabrico que contenham uma trajetória definida num plano e pouco complexa, para que a existência de erros seja minimizada, pois será o próprio robô a escolher a melhor trajetória entre cada ponto, o que poderá levar a que a trajetória não seja a mais otimizada. A grande desvantagem deste tipo de programação é que sempre que é necessário programar o robô para executar uma tarefa, é necessário o próprio robô, isto é, terá de se imobilizar o robô da tarefa que está a realizar, para fazer uma nova programação, o que acaba por ser pouco produtivo.

Relativamente à programação offline, esta já não necessita de contacto direto com o robô, podendo ser feita enquanto o mesmo está a realizar a tarefa para o qual foi programado. Este tipo de programação está a ser cada vez mais utilizada, principalmente para aplicações que exigem um longo período de desenvolvimento, diminuindo o tempo improdutivo do robô, sendo esta a principal vantagem relativa à programação online. A programação offline é realizada com o auxílio de programas de simulação, normalmente disponíveis pela marca, onde é possível a importação de ficheiros CADs (Computer-aided design), ficheiros estes que contém os desenhos quer seja dos *grippers* necessários para a aplicação, quer seja da estrutura onde ficará o robô, de forma geral, todos os desenhos necessários para a realização da simulação da aplicação pretendida para uma melhor comparação com a aplicação real. Seguidamente é efetuada a simulação em ambiente virtual da aplicação que se pretende com o robô, onde é possível verificar possíveis colisões e trajetórias mais otimizadas, tudo feito sem a necessidade do fator de proximidade, o que leva a uma segurança acrescida. No fim da simulação basta fazer a importação do código para o robô e fazer os testes necessários.

Como mencionado acima, normalmente os fabricantes dos robôs têm o seu próprio software de simulação e linguagem de programação. Na Tabela 1 podemos ver alguns dos fabricantes de robôs industriais e os seus respetivos software e linguagens de programação.

Tabela 1 - Software dos fabricantes de robôs Industriais

Fabricantes	Software
FANUC	ROBOGUIDE [62]
ABB	RobotStudio [63]
KUKA	KUKA.Sim [64]
Stäubli	Stäubli Robotics Suite [65]
Denso	WINCAPS III [66]

Por sua vez, falando de segurança, este é um tema muito importante quando se aborda a robótica industrial, uma vez que falamos de máquinas potencialmente perigosas e danosas para os seres humanos, devido a serem máquinas bastante grandes, com muita força e que não têm meios para deteção própria sobre a presença de operadores na proximidade.

Para isto existem dois termos muito importantes para a segurança dos trabalhadores sobre as máquinas, a segurança passiva e ativa. Uma das principais regras que os trabalhadores devem ter em consideração é não entrar na zona de trabalho do robô enquanto este estiver com alimentação no seu braço robótico, mesmo que esteja parado, pois não implica que não esteja a meio de uma tarefa, ou que por alguma razão efetue um movimento e fira o operador. Assim como, mesmo sabendo qual a

trajetória que o robô esteja programado para fazer, nunca se deve admitir que o robô não saia dessa trajetória devido a algum erro.

Para além destes dois pontos mencionados que se devem ter sempre em atenção existe, a mencionada, segurança passiva, que consiste em sistemas projetados para reduzir a probabilidade e gravidade de possíveis acidentes que possam acontecer na área de trabalho do robô. Alguns exemplos de segurança passiva para aplicações de robótica industrial são [67] :

1. Barreiras, normalmente são cercas ou as chamadas jaulas, que são colocadas em volta da área de trabalho do robô para limitar a zona que o operador pode aceder em segurança;
2. Sensores ou cortinas de luz que detetam a presença ou movimentação de um operador na área de trabalho do robô e enviam essa informação ao robô, que por sua vez interrompe imediatamente a tarefa em execução;
3. Usar sempre o equipamento de segurança;
4. Alarmes para que seja possível os operadores terem um tipo de alerta sonoro para o conhecimento de possível perigo na zona;
5. Botão de emergência que permita os trabalhadores pararem o robô imediatamente em casos de emergência.

Figura 29 - Exemplos de segurança passiva [68-72]

Já a segurança ativa é projetada para detetar e funcionar aquando de situações perigosas, algo que é realizado no momento que ocorre o perigo. Alguns exemplos desta segurança são:

1. Ciclos que monitorizam o comportamento do robô e fornecem o respetivo feedback, permitindo realizar os ajustes necessários sempre que aplicável;

2. Sistemas *human-in-the-loop* que para determinadas ações e decisões do robô necessitam de confirmação e supervisão de um operador;
3. Sistemas *human-out-the-loop* que consistem em permitir que os trabalhadores deleguem autoridade e responsabilidade aos robôs;
4. Sistemas human-on-the-loop que são sistemas que proporcionam aos humanos a possibilidade de intervir e anular as ações e decisões que o robô irá tomar.

A segurança é uma das principais vertentes que se deve ter em atenção no trabalho junto a robôs industriais para evitar qualquer perigo e acidentes que possam vir a acontecer e para proteger os humanos dos robôs. Atualmente existem inúmeros meios de segurança para evitar este tipo de acidentes e cada vez é mais notável a evolução nesta área.

2.4 – Robótica colaborativa

Apesar da grande evolução da robótica industrial e por consequente a integração de sistemas de automação nos processos de produção, existem ainda certas tarefas que requerem a flexibilidade do operador, o que torna o humano num ponto vital em algumas cadeias de produção. Para além deste ponto, existem ainda trabalhos em que não é viável a introdução de uma máquina industrial para a sua automação, seja pelo seu custo elevado para pequenas/médias empresas, seja pelo seu tamanho, ou pela sua falta de flexibilidade, a nível de movimentos ou a nível de trabalho cooperativo com os operadores, deixando preocupações a nível da segurança [73] .

Com a ideia de superar estes tipos de limitações, foi explorada a possibilidade de reduzir a distância entre o robô e o operador, procurando colocá-los a trabalhar em conjunto num espaço colaborativo, onde se pode juntar a consistência, a precisão, a força e a eficiência do robô com a flexibilidade, o conhecimento e a possibilidade de tomar decisões do operador [74] , o que leva à chamada colaboração humano-robô, ou em inglês *Human-Robot Colaboration* (HRC) [75] . Com a implementação da HRC, a qualidade, a eficiência e por sua vez a produtividade são pontos que irão sofrer um aumento significativo, por outro lado, também vai reduzir os custos operacionais, devido aos pontos fracos de um dos lados ser compensado pelos pontos fortes do outro [76] .

Na Figura 30 podemos verificar um exemplo da divisão das áreas de trabalho num ambiente colaborativo, onde, tanto o robô como o trabalhador, têm a sua área de trabalho individual, mas, em simultâneo, têm uma área específica de trabalho compartilhada entre ambos.

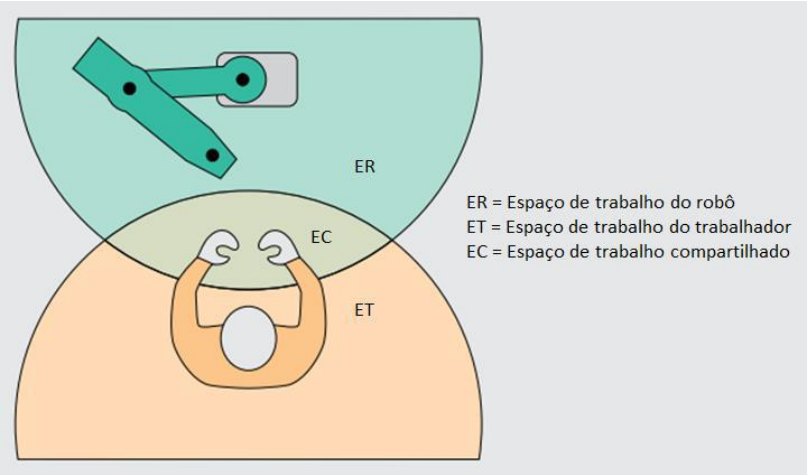


Figura 30 - Espaços de trabalho em ambiente colaborativo [77]

Os *cobots*, de acordo com a ISO 10218–2, são robôs projetados para a colaboração direta com o operador no espaço de trabalho, definido sem qualquer tipo de barreiras ou limitações, o que leva a que esta projeção tenha de garantir a interação humano-robô segura. O espaço de trabalho que vai ser partilhado por ambos, durante a realização das tarefas é denominado por espaço de trabalho colaborativo, que vai incluir toda a área de trabalho.

Dentro desta interação existem fatores e medidas que devem ser tomadas para garantir a segurança e produtividade no ambiente de trabalho, tais como, o envolvimento do operador na implementação do *cobot*, bem como providenciar todas as formações e informações necessárias para a boa compreensão e interação entre os dois. A ausência destas medidas, pode originar falta de confiança por parte do trabalhador com o robô, o que leva à diminuição da eficiência da HRC, para além de originar lesões e acidentes no local de trabalho.

A interação humano-robô, relativamente à sua área de trabalho, pode ser classificada de diferentes formas, nomeadamente [78] :

1. **Célula** – O robô trabalha numa gaiola ou jaula que consiste numa caixa em volta do robô com barreiras limitando o seu acesso. Este não é considerado um cenário de cooperação genuíno;
2. **Coexistência** – O robô e o operador realizam o seu trabalho lado a lado, mas não compartilham o espaço de trabalho;
3. **Sincronização** – O robô e o operador compartilham o espaço de trabalho, mas apenas uma das partes está presente no espaço de trabalho, não podendo estar ambas as partes na área de trabalho em simultâneo;
4. **Cooperação** - O robô e o operador compartilham o espaço de trabalho, sendo que ambas as partes podem efetuar tarefas simultaneamente, no espaço compartilhado, contudo, não trabalham no mesmo produto ou componente simultaneamente;

5. **Colaboração** - O robô e o operador compartilham o espaço de trabalho podendo efetuar tarefas ao mesmo tempo, e no mesmo produto ou componente no espaço compartilhado.

Para uma melhor elucidação destas diferentes formas de interação humano-robô, podemos verificar um exemplo prático na Figura 31.

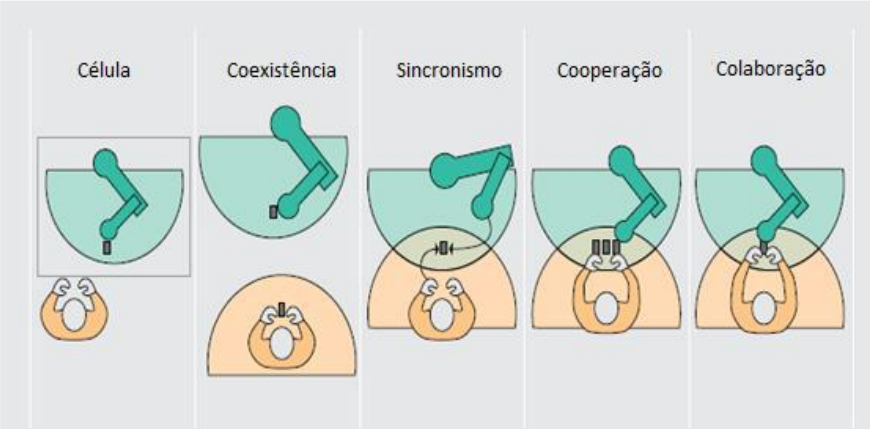


Figura 31 - Exemplo das diferentes formas de HRC [77]

Os *cobots* são, normalmente, robôs articulados com 6 ou mais eixos com a forma de um braço articulado, leves e de pequenas dimensões. Nestes podem ser integrados sistemas de visão, com câmaras e sensores que são capazes de transmitir a informação sobre o meio envolvente para o robô, detetando objetos ou os próprios operadores, evitando possíveis colisões. Têm ainda a capacidade de serem ligados a PLC's ou estarem interligados a outros *cobots*, podendo transmitir informação de uns para os outros, trabalhando em conjunto. Uma vez que o seu trabalho é colaborativo, muitas das vezes em conjunto com os trabalhadores, são programados para trabalhar a velocidades mais reduzidas, tendo ainda a mais-valia de pararem o seu movimento se sofrerem algum tipo de resistência, por exemplo, um toque com a mão de um operador.

Possuem ainda a capacidade de ser acoplado no fim do seu braço uma garra (*gripper*) que pode ter inúmeras funções, dependendo da tarefa que irá realizar, como é possível verificar na Figura 32.

Figura 32 - Gripper de vácuo, de precisão e "Soft hand" respetivamente [79-81]

O braço robótico vem, normalmente, junto com um controlador e um *teach pendant*, onde o controlador é dotado de entradas e saídas (inputs e outputs) que podem ser utilizadas para a comunicação do robô com equipamentos externos, tais como, PLC, câmaras e sensores, para a interligação de todos os equipamentos num sistema único. O *teach pendant* é utilizado para a manipulação do robô em tempo real e para a sua programação.

Figura 33 - Controlador e Teach Pendant respetivamente [82]

Estes, sendo leves e pequenos, podem ainda ser implementados em cima de um carrinho, num AGV ou num tapete rolante, o que irá possibilitar a movimentação do *cobot* para efetuar tarefas móveis, eliminando a desvantagem de ser uma máquina estática como acontece nos robôs industriais (Figura 34).

Figura 34 - *Cobot* móvel [83]

A implementação deste tipo de *cobots* é aconselhada em [84] :

1. **Aplicações repetitivas e pouco ergonómicas:** Tarefas que não exijam velocidades altas e destreza humana são tarefas que beneficiam a utilização de *cobots* para a sua automatização, pois irá aumentar a produtividade da produção. Tarefas como o próprio pick-and-place, a fixação de uma peça numa determinada posição para que seja possível o manuseamento pelo operador, aparafusamento de peças, polimento e inspeções de qualidade são ótimas opções para a implementação de *cobots*;
2. **Linhas de produção que incluem operadores:** Existem tarefas que, apesar da gama de aplicação dos *cobots* ser grande, não são fáceis de automatizar com um bom custo-benefício, mas que por sua vez são de fácil execução para o humano. São estes tipos de aplicações que irão beneficiar da implementação de *cobots*, devido à sua capacidade de trabalho colaborativo, uma vez que cada um pode executar a tarefa que melhor se adequa lado a lado. Tarefas essas como o tratamento de peças irregulares ou flexíveis e tarefas que exijam a regulação de pressão constante;
3. **Tarefas variáveis e de curto prazo:** Uma vez que a programação de robôs colaborativos é normalmente de rápida e fácil execução, nas indústrias onde é casual a mudança dos produtos produzidos, devido a serem empresas que produzam uma grande variação de produtos ou que os produzam por poucos períodos de tempo, a implementação da automação destas tarefas torna-se uma mais-valia no uso destes robôs pois podem ser rapidamente movidos e reprogramados para a sua nova tarefa.

Atualmente, e cada vez mais, os cobots acrescentam valor aos mais diversos setores industriais, contribuindo para o aumento da eficiência e da produtividade. É notável a evolução que a robótica colaborativa tem vindo a ter nos últimos anos e o que tem garantindo a sua credibilidade é que, à medida que as empresas sofrem mudanças a nível da sua produção, o desenvolvimento destes robôs tem acompanhado à mesma velocidade estas mudanças o que os torna sempre uma opção muito viável de adquirir.

Estes robôs são perfeitos para aplicações em linhas de produção e montagem, como em tarefas de aparafusamento, soldadura, embalamento e polimento, tal como em tarefas colaborativas como aplicações de pick-and-place, em que, posteriormente, o operador possa usar a peça que foi movida pelo robô para efetuar algum trabalho ou trabalhos de suporte como, por exemplo, pegar numa peça e posicioná-la para que o operador consiga trabalhar nessa peça, não tendo de ser ele a ter o esforço de a posicionar no devido sítio. São inúmeras as tarefas possíveis para a sua utilização existindo atualmente *cobots* em diversos setores como a indústria automóvel, eletrónica, logística, agricultura e até mesmo na medicina.

Mencionarei em seguida alguns dos muitos casos de sucesso que a empresa Universal Robots teve ao longo dos anos:

1. A nível da indústria automóvel um dos grandes sucessos da UR foi a implementação dos seus robôs na empresa Continental em Espanha, onde o objetivo foi implementar os robôs para proceder a tarefas de manipulação e validação de placas PCB (placas de controlo) e componentes enquanto eram fabricados. Uma tarefa que era considerada monótona e repetitiva, mas que necessitava de uma grande precisão e, ao mesmo tempo, delicadeza na sua execução. Com a implementação destes *cobots*, a empresa Continental viu grandes melhorias nos resultados tanto a nível da flexibilidade como do controlo, observou uma grande redução tanto a níveis de custos como de tempos de operação e notou uma grande segurança relativa aos operadores [85] . Outro grande sucesso também relativo à indústria automóvel foi na empresa Nissan situada no Japão, empresa responsável pelo desenvolvimento de carros. Nesta implementação o objetivo era a relativo à montagem. Primeiramente foram instalados robôs para o aperto de parafusos no suporte da cabeça do motor, onde os seus requisitos seriam os robôs serem leves e fáceis de mover e poderem ser utilizados sem necessitarem de cercas de segurança para o trabalho em conjunto com os operadores. Outra aplicação foi a implementação destes *cobots* para o processo de instalação de coletores de admissão no bloco do motor. Sendo esta uma tarefa que precisaria de ser efetuada perto de operadores, os robôs teriam de ter essa segurança sendo que, em simultâneo, precisariam de ser capazes de movimentar peças com peso entre 4 a 6 quilos [86] .
2. No contexto da indústria farmacêutica, destaca-se o caso de sucesso ocorrido no Hospital de Gentofte, na Dinamarca. O principal objetivo desta iniciativa foi automatizar o processo de controlo de qualidade das amostras de sangue, que chegam através de um sistema de tapete rolante. A solução exigia que o robô fosse capaz de recolher, classificar e depositar corretamente as amostras para posterior análise laboratorial. Para tal, a empresa Universal Robots (UR) implementou um sistema composto por dois robôs colaborativos e uma câmara de visão. O primeiro robô é responsável por recolher cada amostra de sangue e posicioná-la num ponto de verificação, onde a câmara identifica a cor da tampa. Com base nessa informação, o robô direciona a amostra para o compartimento correspondente. Por sua vez, o segundo robô realiza o transporte das amostras já classificadas até à máquina encarregue da centrifugação e subsequente análise laboratorial.[87] .
3. A indústria alimentar e agrícola também não ficou para trás. A empresa Nordic Sugar na Suécia, uma das grandes empresas responsável pela produção de açúcar na Europa, após ver uma exibição dos robôs da UR numa feira optou por testar a sua implementação na sua linha de produção onde o objetivo seria testar o braço robótico no processo de análise do açúcar. Implementaram então três robôs na sua linha que eram responsáveis por fazer *scan* a códigos de

barras e coletar recipientes com açúcar para serem analisados os seus níveis da filtração e posterior devolução à linha de produção. Para a execução deste processo foi necessário implementar no *gripper* do *cobot* uma pinça pneumática e um leitor de códigos de barras [88] .

Estes oferecem vantagens significativas seja para pequenas, médias ou grandes empresas, tendo sobretudo impacto maior nas empresas de pequenas e médias dimensões, visto serem robôs consideravelmente mais baratos que os robôs industriais e apresentarem um ROI (retorno sobre o investimento) maior. São robôs com configuração e programação rápida e fácil, e tendem a ser de pequenas dimensões e bastante leves, o que faz com que estes robôs também não necessitem de um grande espaço para a sua instalação. Associada a estas características está também a vertente da segurança, uma vez que estes robôs são especificamente concebidos para ambientes colaborativos, como já referido anteriormente. Tal implica que estejam equipados com elevados padrões de segurança, de forma a evitar qualquer risco para os operadores. Esta capacidade de operar lado a lado com os trabalhadores, sem necessidade de barreiras físicas de proteção, contribui para uma solução mais compacta e eficiente no espaço de trabalho. Têm ainda a possibilidade, devido às suas dimensões reduzidas, de ser instalados sobre uma correia de transporte, o que lhes permite executar tarefas em diferentes locais ao longo do processo produtivo [89]

Por sua vez, apesar de eliminar a desvantagem da segurança e do seu custo inicial ser mais baixo comparativamente aos robôs industriais, os *cobots* têm uma baixa capacidade de carga útil, isto é, não conseguem carregar uma grande quantidade de peso comparativamente aos robôs industriais, tal como a sua velocidade de trabalho terá de ser também inferior para assegurar a segurança necessária para o trabalho colaborativo.

Uma vez que a robótica colaborativa está a ganhar cada vez mais importância na indústria devido às vantagens mencionadas em cima, começaram a aparecer no mercado empresas que desenvolvem estes *cobots*. Apesar de algumas das empresas mencionadas no capítulo da robótica industrial também oferecerem soluções de *cobots*, existem empresas especializadas no desenvolvimento destes, sendo de destacar as três que mencionarei de seguida:

1. **Universal Robots (UR)** – Com a sua sede na Dinamarca, esta foi fundada em 2005 tendo o objetivo de disponibilizar soluções robóticas para médias e pequenas empresas. É uma das principais empresas na fabricação de robôs colaborativos, tendo lançado o seu primeiro *cobot* em 2008;
2. **Omron** - Tem a sua sede localizada no Japão e entraram no mercado da automação em 2015 após a aquisição da empresa americana Adept. Em 2018, ao colaborar com a empresa Techman Robot alargou a sua área para robôs utilizados na montagem, servomotores e sistemas de segurança para os robôs colaborativos;

3. **Doosan Robotics** – Por fim, com a sua sede na Coreia de Sul, a Doosan é conhecida pela sua vasta gama de *cobots* tendo foco em diversos setores como a indústria automóvel, eletrónica, indústria farmacêutica e logística. É uma empresa que tem vindo a ter uma grande evolução e tem vindo a ganhar um grande renome no meio da robótica colaborativa sendo uma empresa nova neste ramo, tendo sido fundada em 2015.

Figura 35 - Logos das respetivas empresas mencionadas em cima [90-92]

No que diz respeito à programação, os *cobots* são amplamente reconhecidos pela sua facilidade de utilização. Estes oferecem a possibilidade de programação online através do guiamento direto, permitindo que o operador manipule o robô fisicamente para o posicionar conforme necessário. O software associado é geralmente muito intuitivo, baseado num sistema de construção por blocos. O utilizador seleciona os elementos com as funções desejadas e organiza-os na sequência pretendida, como ilustrado na Figura 36. Com o apoio do guiamento direto, o robô pode ser posicionado manualmente nas coordenadas desejadas, sendo essas posições depois registadas nos respetivos blocos. Este processo permite o desenvolvimento de programas simples e eficazes de forma rápida, adaptando-se facilmente à tarefa a realizar.

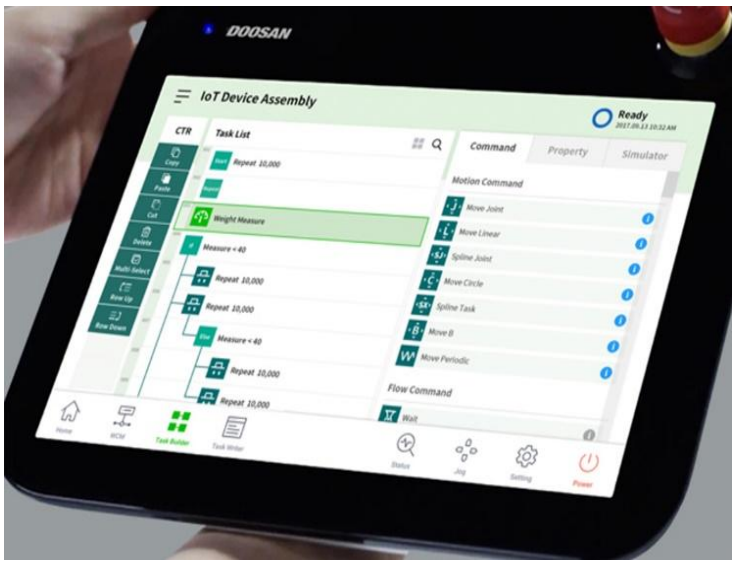


Figura 36 - Programação por Blocos [93]

Para além desta programação, é ainda possível efetuar uma programação offline recorrendo à utilização de simuladores, um dos mais conhecidos é o *software* RoboDK (Figura 37), onde é possível selecionar o robô desejado, pois este contém uma vasta

biblioteca de robôs. É ainda possível importar CADs, que consistem em desenhos da área, das peças e do próprio suporte onde o robô irá efetuar o seu trabalho e fazer uma simulação do que é pretendido para o trabalho do robô. Posteriormente é possível converter essa simulação no código para o determinado robô, onde o próprio programa é capaz de fazer essa conversão e importá-la para o *cobot*, para fazer os devidos testes em ambiente real.

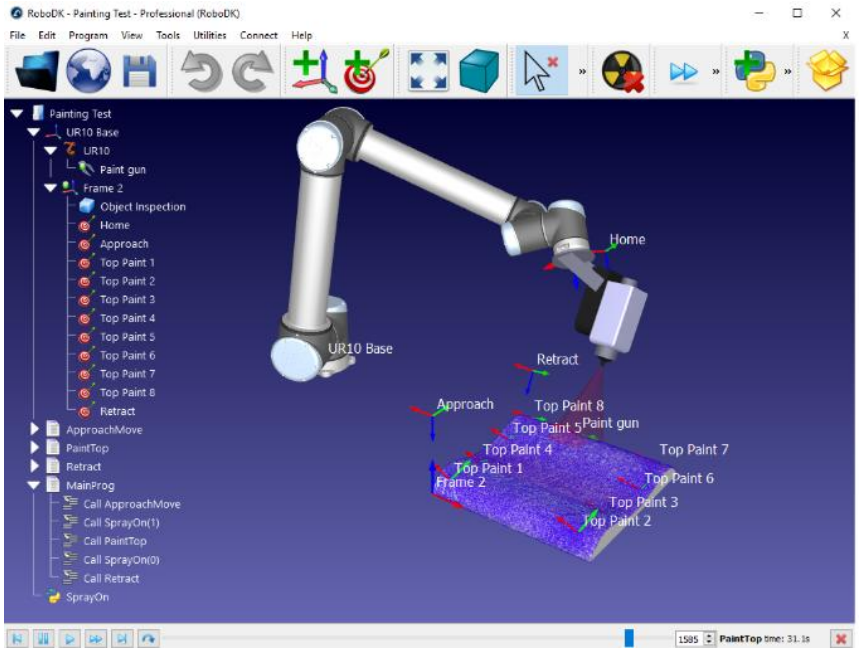


Figura 37 - Programação *offline* RoboDK [94]

É ainda possível fazer a programação destes robôs em formato de código tornando-a mais precisa, mas por sua vez, também mais complexa, necessitando de um trabalhador com formação para a elaboração do dito código. Este tipo de programação é utilizado em tarefas mais complexas, que exijam um nível de precisão bastante alto e que necessitem de comunicar com máquinas externas, sejam câmaras, PLCs ou até mesmo outros *cobots* visto que, através da programação, é possível estabelecer qualquer tipo de comunicação, algo que nem sempre é viável utilizando apenas o software nativo do robô.

Para a realização da programação do robô, seja de que modo for, é necessário realizar alguns procedimentos, nomeadamente a definição do TCP (*Tool Center Point*), o peso e o centro de gravidade da ferramenta que se está a utilizar na flange do robô. O TCP corresponde ao ponto central dos pontos de contacto do *gripper*, ou seja, num *gripper* de pinças o TCP será o ponto médio entre a extremidade de uma das pinças e a extremidade da segunda pinça, como é possível verificar na Figura 38.

Figura 38 - Exemplo de TCP num gripper de pinças (adaptado de [95])

Existem duas maneiras de definir o TCP de uma ferramenta, a primeira é feita através de uma medição direta que se retira da distância do ponto central da flange do robô até ao TCP pretendido no *gripper*. O segundo modo consiste no método dos quatro pontos em que é necessário posicionar o robô no mesmo ponto de quatro maneiras diferentes, isto é, alterando a orientação das juntas. Desta maneira o próprio software do robô irá utilizar os quatro pontos guardados para calcular o TCP da ferramenta. Este segundo método é o mais utilizado dos dois, pois se o robô se mantiver no mesmo ponto nas quatro posições diferentes, o cálculo do TCP será mais preciso. No que toca ao peso e o centro de gravidade, atualmente, os *cobots* já conseguem fazer o seu cálculo automaticamente com um programa pré-criado no software, é apenas necessário posicionar o robô em algumas posições diferentes e o próprio robô irá calibrar a força que necessita de manter para conseguir suportar a ferramenta, sem que o peso afete a sua trajetória.

Em relação à sua segurança, estes robôs, como mencionado anteriormente, são especialmente conhecidos pelo elevado nível de segurança, uma vez que foram desenvolvidos para o trabalho colaborativo, exigindo que estejam equipados com diversos sistemas de proteção. Estes robôs destacam-se pelo seu reduzido tamanho e peso, contando com sistemas de monitorização tanto da velocidade como da força exercida. Estão equipados com tecnologias avançadas de deteção de colisões, o que lhes permite interromper imediatamente o movimento e cessar a força aplicada ao identificar uma colisão, prevenindo assim impactos com operadores ou objetos. Têm ainda a possibilidade de ter interligação com tecnologias de segurança, incluindo sensores de proximidade, câmaras de visão ou botões de segurança que comunicam

com o *cobot* e, no caso de detetarem algum tipo de movimento, enviam um sinal ao robô, permitindo-lhe identificar que existe algo na sua proximidade.

Para um complemento a estas tecnologias os *cobots* possuem a possibilidade de criar zonas de trabalho como, por exemplo zonas colaborativas ou zonas de anticolisão onde a zona colaborativa é uma zona de trabalho do robô que irá limitar a sua velocidade e força que pode exercer para evitar possíveis ferimentos aos operários, já a zona de anticolisão é criada para que o robô tenha conhecimento que naquela zona não pode entrar, pois irá colidir com algum objetivo que está na sua área de trabalho.

Existem duas normas que são fundamentais para a segurança e desempenho de um robô conhecidas por ISO 10218 e ISO TS 15066.

A ISO 10218 refere-se aos perigos inerentes aos robôs industriais e aos seus sistemas. Esta norma foi apresentada em 2011 com incidência na robótica industrial, mas que se aplica também à robótica colaborativa, e é constituída por duas partes. A *ISO 10218-1:2011: Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots* que se refere aos perigos, situações perigosas e eventos com os robôs industriais e aos seus sistemas. A *ISO 10218-2:2011: Robots and robotic devices — Safety requirements for industrial robots — Part 2: Robot systems and integration* aborda os perigos relacionados com os sistemas robóticos industriais quando integrados e instalados em células e linhas de robôs industriais [96] .

Relativamente à segunda norma mencionada, a *ISO TS 15066:2016: Robots and robotic devices — Collaborative robot*, esta já relacionada aos *cobots*, foi apresentada em 2016 e especifica os requisitos de segurança para sistemas de robôs colaborativos e o ambiente de trabalho [97] . Esta segunda complementa a ISO 10218 e juntas são as normas mais importantes para a instalação destes robôs.

Para além destas duas, existe outra norma que os projetistas devem ter em atenção no desenvolvimento das máquinas. A ISO 12100:2010 é uma norma internacional que fornece os princípios e metodologias para garantir a segurança no desenvolvimento das máquinas. Esta norma abrange desde as mais simples ferramentas aos equipamentos industriais mais complexos [98] .

A ISO TS 15066 nomeia ainda quatro modos de funcionamento colaborativos para garantir uma maior segurança conforme a aplicação que se pretenda [99] :

1. **Paragem de segurança vigiada** (Figura 39): Neste modo o robô pára de efetuar o seu movimento quando o operador entra no espaço colaborativo e interage com o mesmo ou com o objeto em que o *cobot* está a efetuar o seu trabalho, voltando apenas à sua função quando o operador sai da zona de colaboração. Este estado pode ser monitorizado e o acionamento pode continuar ligado;

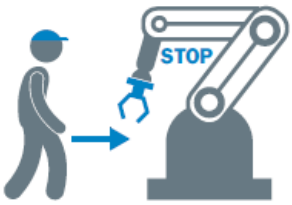


Figura 39 - Paragem de segurança vigiada [99]

2. **Guiamento manual** (Figura 40): Neste, a segurança da colaboração humano-robô é garantida pelo facto de o *cobot* poder ser guiado manualmente a uma velocidade reduzida e segura, estando este "consciente" de que está a ser movimentado por um operador;

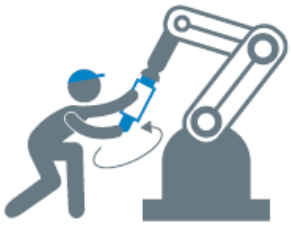


Figura 40 - Guiamento manual [99]

3. **Limitação da força e da potência** (Figura 41): Aqui a segurança é efetuada pela limitação da potência e da força que o robô vai exercer com valores considerados seguros, para evitar qualquer tipo de ferimento ou ameaça quando ocorre contacto físico entre o sistema robótico e o operador, sendo este intencional ou não intencional. Estes valores limites são especificados pela norma ISO TS 15066, que define os valores máximos que não podem ser excedidos na colisão do robô com um trabalhador;

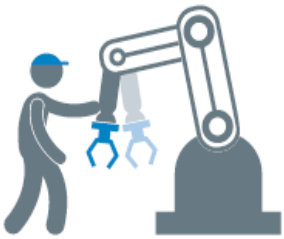


Figura 41 - Limitação da força e da potência [99]

4. **Monitorização da distância e da velocidade** (Figura 42): Por fim, neste quarto modo, a velocidade e a trajetória do robô são monitorizados e adaptados em função da velocidade e da posição do operador no espaço de colaboração.

Figura 42 - Monitorização da distância e da velocidade [99]

Para garantir uma boa proteção e segurança tanto a nível do robô como dos trabalhadores incluídos na aplicação de um *cobot*, devem ser escolhidos um ou vários métodos dos que foram apresentados anteriormente, dependendo da aplicação.

2.5 – Sistemas integrados

Atualmente é muito comum na aplicação de robôs colaborativos a implementação de sistemas integrados externos aos robôs, de forma a aumentar a sua produtividade, segurança e eficiência, isto é, são implementados juntamente com o *cobot*, sistemas de visão, sistemas de segurança e/ou sistemas de aparafusamento, que irão beneficiar e ajudar o *cobot* no seu trabalho, no que toca à segurança e produtividade.

A nível da segurança, é habitual a implementação de sensores ou de câmaras que são posicionadas em pontos estratégicos. Relativamente aos sensores, é comum a utilização de sensores de proximidade, cuja principal função é a deteção de movimento. Um exemplo prático consiste na sua aplicação para identificar a presença de um operador na zona de trabalho do robô. Ao detetar a entrada do operador nessa área, o sensor comunica essa informação ao *cobot*, indicando que existe um trabalhador próximo. Com base nessa informação, o *cobot* ajusta automaticamente o seu funcionamento, podendo reduzir os seus limites de velocidade e força ou, em alguns casos, interromper totalmente a tarefa até que o operador abandone a zona de trabalho. Outros sensores bastante utilizados são os sensores de força, que são implementados na ferramenta do robô e têm como função indicar em tempo real a força que está a ser exercida. Estes são exemplos de sensores que podem ser utilizados em complemento com o *cobot*.

Por outro lado, podem ainda ser utilizadas câmaras de visão que desempenharão o mesmo trabalho que os sensores. Estas terão a visão de toda a área de trabalho do robô e sempre que detetarem algo que não seja reconhecido irão comunicar com o robô para que este possa trabalhar de acordo com essa informação, evitando colisões [100] , [101] .

Em relação à produtividade e eficiência, existe também a possibilidade de recorrer a sistemas de visão, mas, neste caso, para a deteção de objetos, como por exemplo numa aplicação de pick and place, onde a câmara terá a tarefa de identificar a posição do objeto e comunicar ao robô para que o *cobot* o recolha e posicione no devido lugar.

Pode ainda ser utilizada numa aplicação de aparafusamento, na qual a câmara tem como função identificar a posição dos parafusos a apertar e, posteriormente, comunicar essa informação ao robô, permitindo-lhe deslocar-se com precisão até aos pontos onde deverá realizar os apertos [102] , [103] . Neste tipo de aplicação, é possível ainda implementar uma aparafusadora ao *gripper*, que terá um controlador responsável por fazer o aperto e de seguida comunicar ao robô toda a informação necessária para a sua aplicação [104] .

Para além dos sistemas mencionados, existem outros inúmeros sistemas que podem ser implementados e integrados nas aplicações com *cobots*, que aumentam a sua segurança, produtividade e eficiência.

Figura 43 - *Cobot* com sistema de visão e aparafusamento [105]

2.6 – Considerações finais

A elaboração da revisão da literatura foi uma mais-valia para a realização do projeto.

Através desta pesquisa, compreendeu-se que, relativamente à evolução da indústria, este projeto enquadra-se de forma coerente na transição da indústria 4.0 para a indústria 5.0, uma vez que consiste na implementação de robôs autónomos, os *cobots*, (indústria 4.0), que têm a capacidade de trabalhar juntamente com operadores em simultâneo, indústria centrada no humano (indústria 5.0).

No que diz respeito à robótica colaborativa, é possível compreender o funcionamento global de um sistema com *cobots*, tanto ao nível da interação humano-robô, como nas

diversas possibilidades de implementação destes robôs. Estas podem ocorrer em sistemas independentes ou colaborativos. Foram também abordados exemplos de diferentes tipos de *grippers* que podem ser integrados nos *cobots*, bem como a sua potencial aplicação em sistemas móveis, permitindo o seu deslocamento de forma totalmente autónoma. Esta mobilidade possibilita que o robô atue em diferentes áreas dentro de um determinado espaço, sem necessidade de estar fixo a um único ponto.

É mencionado também o tipo de programação que é utilizado nestes robôs, a segurança que é possível obter em aplicações de sistemas colaborativos e são ainda apontados alguns sistemas onde é aconselhada a implementação deste tipo de automação, tal como algumas implementações de sucesso.

Por fim temos um último ponto onde são referidos sistemas integrados, um tema bastante importante para este projeto uma vez que a sua implementação necessita de sistemas externos ao *cobot* para a sua aplicação, tais como sistemas de visão e de aparafusamento.

Em conclusão, a revisão da literatura revelou-se fundamental para o desenvolvimento do projeto, permitindo adquirir o conhecimento necessário sobre o estado atual da indústria e dos sistemas de automação, bem como sobre os robôs e tecnologias disponíveis. Esta base teórica facilitou a seleção dos periféricos mais adequados e dos sistemas mais vantajosos para a implementação da solução proposta.

CAPÍTULO 3 – Descrição do projeto

Este projeto consiste no aperto de duas partes distintas de uma carrinha de mercadorias, o Cloison e o Porteur, através de uma solução colaborativa. Para uma gestão mais eficiente do projeto, este foi dividido em duas fases, o aperto do Cloison e o aperto do Porteur, respetivamente, uma vez que se tratam de componentes distintos do veículo e são montados em etapas diferentes da linha de produção.

3.1 – Aperto Cloison

No caso do aperto do Cloison, este consiste no aparafusamento da placa de metal que se encontra a dividir a zona onde vão os passageiros da zona onde vai a mercadoria, como é possível verificar na Figura 44. Atualmente esse aparafusamento é efetuado manualmente pelos operadores, que realizam 12 apertos com uma aparafusadora a bateria com um binário de 8Nm +/- 15%.

Figura 44 – Chapa divisória da carrinha

O objetivo é o desenvolvimento de um sistema que seja capaz de fazer o aparafusamento de 13 porcas nessa placa de forma autónoma, com um limite máximo de 120 segundos, uma vez que é o tempo que a carrinha está parada na zona de aparafusamento. As porcas que serão necessárias apertar podem ser observadas na Figura 45 a vermelho, sendo que a que está a verde já vem previamente apertada para que a placa se encontre no sítio correto de instalação.

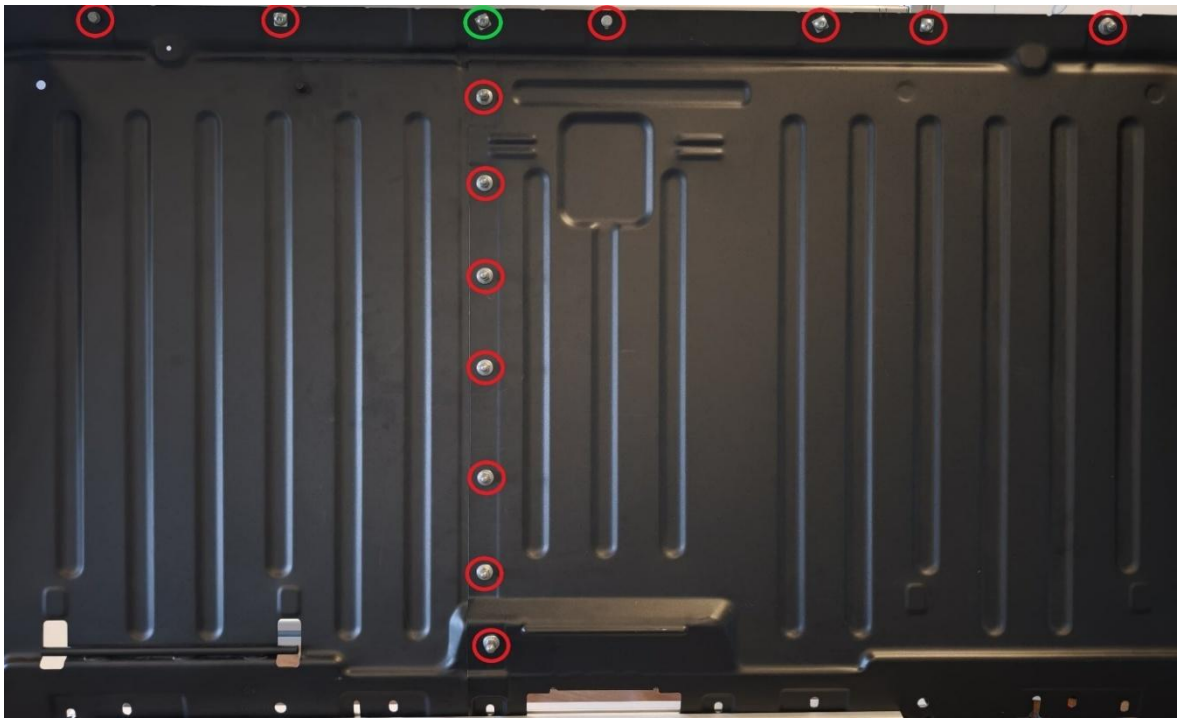


Figura 45 - Identificação das porcas

As porcas que farão parte deste aparafusamento podem ser vistas na Figura 46.

Figura 46 - Porcas de aperto

O objetivo deste projeto é a implementação de *cobot*s para a execução do processo de aparafusamento. Para tal, será necessário garantir que a alimentação das porcas seja realizada de forma autónoma para o robô, o qual deverá estar equipado com um sistema de aparafusamento automático e um sistema de visão capaz de identificar com precisão as posições onde os apertos deverão ser efetuados. Tanto a aparafusadora como a câmara de visão deverão ser acopladas no *gripper* do robô, o qual terá de ser projetado de forma a suportar e proteger adequadamente ambos os sistemas.

3.2 – Aperto Porteur

Relativamente ao aperto do Porteur, este processo envolve diversos elementos cruciais da roda. Conforme é possível observar na Figura 47, o Porteur é constituído por componentes como o disco e os travões, além de todos os elementos onde a roda é encaixada.

Figura 47 - Disco travão

A implementação desta automação terá de ser efetuada na zona que está indicada na Figura 48.

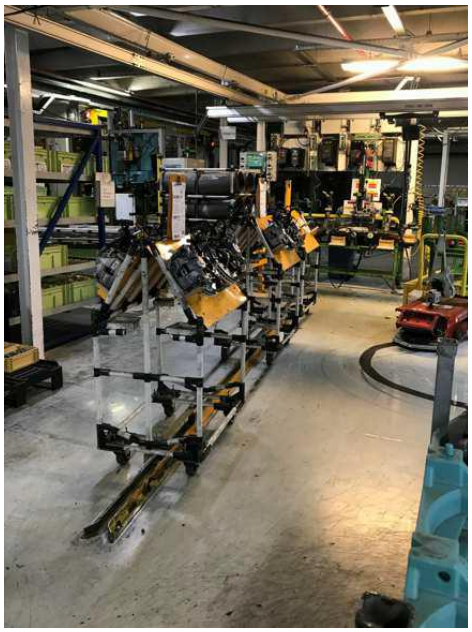


Figura 48 - Zona de implementação

Atualmente, todos os apertos são realizados manualmente pelo operador, utilizando duas aparafusadoras, sendo necessária a troca das chaves conforme o tipo de aperto a efetuar. O componente da roda, já com os parafusos previamente apontados, é

inserido numa maquete que serve de suporte para transporte. Cabe ao operador transportar essa maquete até ao posto de trabalho onde serão realizados os apertos. Na Figura 49, é possível observar a maquete tanto na sua versão vazia como com os componentes posicionados.

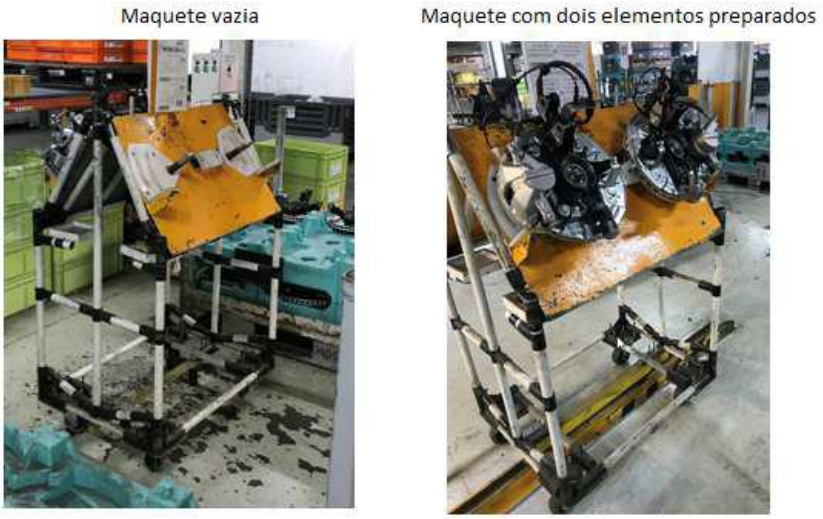


Figura 49 - Maquete de suporte dos elementos da roda

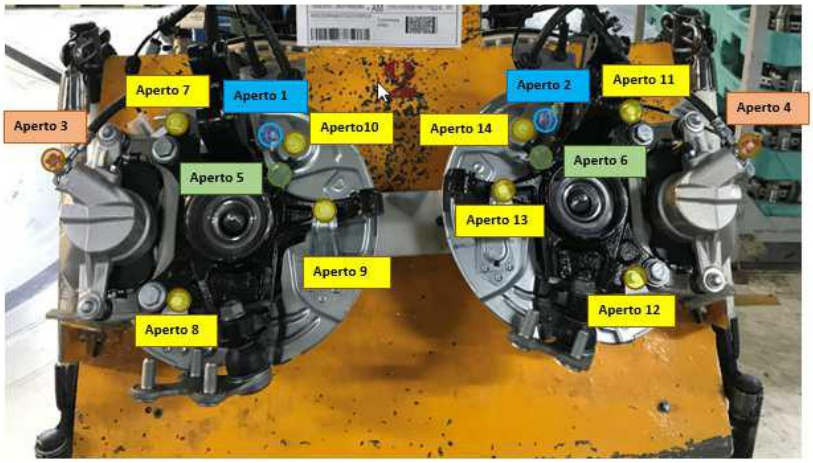
De seguida o operador transfere os dois discos para a mesa de trabalho onde serão realizados os apertos, como é possível observar na Figura 50.

Figura 50 - Elementos da roda posicionados para o aperto

O operador executa 14 apertos no total, 7 em cada um dos discos, utilizando duas aparafusadoras e quatro chaves diferentes.

O objetivo deste projeto é a implementação de um sistema colaborativo capaz de realizar os 14 apertos necessários num tempo máximo de 120 segundos. Estes apertos serão efetuados na maquete utilizada para a preparação e transferência dos componentes, recorrendo a duas aparafusadoras, sendo necessária a troca autónoma de chaves. Para além da troca de chaves, cada aperto requer um binário específico. A aparafusadora A ficará responsável pela utilização das chaves H13 e H11, com binários de 20 Nm e 40 Nm, respetivamente, enquanto a aparafusadora B utilizará uma chave TORX e uma H10, ambas com um binário de 8 Nm.

Na Figura 51 é possível verificar os apertos a realizar, tal como a indicação da chave e do binário necessário para cada aperto.



Apertos elemento de roda (Direito + Esquerdo)	Apertos	Binário (Nm)	Aparafusadora	Chave
Suporte flex travão ao pivot	1 e 2	20	A	H13
Flex travão estribo ao estribo	3 e 4	40	A	H11
Captor velocidade de roda	5 e 6	8	B	TORX
Protector disco	7, 8, 9, 10, 11, 12, 13 e 14	8	B	H10

Figura 51 - Informação dos apertos a realizar no elemento da roda

Para a execução desta tarefa, prevê-se a utilização de um ou dois *cobot*s, cada um equipado com um *gripper* ao qual estará acoplada com uma aparafusadora. Adicionalmente, será integrada uma câmara de visão responsável pela identificação das posições das porcas, permitindo a sua posterior fixação.

CAPÍTULO 4 – Desenvolvimento do projeto

Neste capítulo será apresentado todo o desenvolvimento do projeto, o qual será organizado em dois subcapítulos, uma vez que foi executado de forma separada, conforme referido anteriormente, com o intuito de facilitar a sua execução.

4.1 – Sistema de aparafusamento do Cloison

Inicialmente, foi realizado um estudo aprofundado com o intuito de identificar o método mais eficiente para a concretização deste projeto. Sabendo que o objetivo seria o aparafusamento de 13 porcas num tempo máximo de 120 segundos, deu-se início à análise da área disponível para a implementação dos *cobot*s. Após uma visita à fábrica, especificamente ao local onde o sistema seria instalado, verificou-se que existia uma área considerável para a sua integração. Constatou-se ainda a viabilidade de utilização de um ou dois *cobot*s para a realização dos apertos, sendo possível posicionar um robô de cada lado do veículo, acedendo ao seu interior pelas portas do condutor e do passageiro (Figura 52).

Figura 52 - Área lado esquerdo da carrinha

Procedeu-se à verificação de qual das soluções seria a mais adequada para esta implementação, se utilizando apenas um robô ou dois. Após verificar o local e retirar todas as medidas necessárias, chegou-se à conclusão de que a utilização de dois *cobot*s seria a mais eficiente, uma vez que o tempo em que o veículo está imóvel é de apenas 120 segundos e que, a utilização de apenas um *cobot* de um dos lados da carrinha não asseguraria totalmente a segurança do aperto da porca da extremidade oposta.

Concluiu-se que a utilização de um *cobot* de cada lado da carrinha seria a opção mais correta a implementar, tanto ao nível da eficiência, conseguindo concluir os aparafusamentos em menos de 120 segundos, como ao nível da segurança, uma vez que toda a área da placa iria estar coberta por um dos dois *cobot*s, garantindo assim

o aperto correto de todas as porcas. Estas conclusões foram confirmadas por simulações efetuadas, as quais irei mencionar mais à frente no capítulo.

4.1.1 – Escolha dos periféricos

No que diz respeito à seleção dos periféricos mais adequados para este projeto, iniciou-se o processo pela escolha do robô. A principal exigência nesta escolha foi a segurança, dado que o robô se encontra posicionado numa zona da linha de montagem onde operam trabalhadores, o que torna estritamente necessária a utilização de um *cobot*, e não de um robô industrial convencional. Para além do requisito de segurança, foi igualmente estabelecido pela empresa que os robôs a utilizar deveriam pertencer à marca Doosan Robotics ou Universal Robots.

Com base nestes dois pontos passou-se à pesquisa de qual seria a melhor solução entre estas duas marcas, de onde surgiram algumas opções de acordo com o que seria necessário para esta implementação. No processo de escolha de *cobot*s, existem quatro critérios fundamentais que orientam a decisão: o alcance, que determina a área de trabalho do robô; o *payload*, que corresponde à capacidade de carga que o robô consegue suportar na garra; a precisão do equipamento; e, por fim, o custo associado.

Após análise e estudo das especificações técnicas disponíveis, foram identificadas quatro opções viáveis para esta implementação: da Doosan Robotics, os modelos M1013 e M1509; e da Universal Robots, os modelos UR10e e UR16e, conforme ilustrado na Figura 53, respetivamente.

Figura 53 - Cobots selecionados para comparação [106-109]

De seguida foi efetuada uma tabela onde estão discriminadas as características mencionadas em cima para cada robô, para uma melhor comparação e posterior escolha da melhor opção para a execução da tarefa pretendida. É possível verificar na Tabela 2 - Principais características dos cobots da Doosan e Universal Robots [110] o *alcance*, o *payload*, a precisão e o preço de cada um dos *cobot*s.

Tabela 2 - Principais características dos cobots da Doosan e Universal Robots [110]

	M1013	M1509	UR10e	UR16e
Alcance (mm)	1300	900	1300	900
***Payload* (kg)**	10	15	12,5	16
Precisão (mm)	±0,05	±0,03	±0,03	±0,05
Preço (€)	32000-34200	36000-38200	47670	51900

Para realizar uma escolha acertada do robô a utilizar no desenvolvimento do projeto, foi necessário analisar todos os fatores anteriormente referidos, nomeadamente o alcance exigido, de forma a assegurar que o *cobot* consegue executar todos os aparafusamentos necessários, bem como o peso total do *gripper*, de modo a determinar o *payload* mínimo que o robô deverá suportar, visto que este integra a aparafusadora, o meio de alimentação das porcas e uma câmara de visão.

No que toca a determinar o *payload*, foi feito um estudo sobre os três componentes: a aparafusadora, a melhor maneira de fazer a alimentação das porcas e por fim qual a câmara mais adequada para o trabalho.

Relativamente à aparafusadora, tendo em conta as necessidades do projeto, foi essencial recorrer à utilização de um sistema de aparafusamento autónomo, capaz de operar mediante sinais provenientes de um PLC e de fornecer um relatório relativo ao aperto efetuado. A Stellantis indicou que a aparafusadora a utilizar deveria pertencer à marca Atlas Copco, uma vez que esta já é utilizada em toda a linha de montagem da empresa. Tendo isso em conta, concluiu-se que a opção mais adequada seria o modelo Atlas Copco ETD STR31-10-10-T25 (Figura 54). Este equipamento é capaz de exercer um binário compreendido entre 2 Nm e 10 Nm, o que satisfaz os requisitos definidos para o aperto das porcas, que exigem um binário de 8 Nm ± 15%. Adicionalmente, apresenta um peso de 1,3 kg e um comprimento de 436,5 mm.

Figura 54 - Atlas Copco ETD STR31-10-10-T25 [111]

Esta aparafusadora contém um controlador onde é possível criar o programa que irá executar e onde será possível configurar as entradas e saídas para uma boa comunicação com o *cobot* no momento do aperto.

Relativamente ao sistema de alimentação das porcas, entre as soluções existentes, recorreu-se a uma tecnologia inovadora baseada em cubas vibratórias. Estas cubas consistem em depósitos nos quais são colocadas as porcas, que, por meio de vibração, são orientadas, transportadas e, por fim, posicionadas de forma adequada, permitindo a alimentação eficiente do suporte para o posterior processo de aparafusamento.

São sistemas com uma grande durabilidade, alto rendimento, económicos e são dispositivos que não emitem muito ruído no local. Contêm vários sistemas e sensores para que seja garantida a posição correta da porca. É possível verificar a cuba vibratória que será usada na Figura 55.

Figura 55 - Cuba Vibratória

Como é possível verificar na Figura 52, a ideia será o robô entrar pela porta e fazer os aparafusamentos necessários. Para tal é necessário que se encontre instalado a uma altura superior do chão e bem fixo, para estar seguro durante todo o seu programa. Uma das opções que a empresa forneceu foi a utilização de pedestais já existentes noutros pontos da linha de montagem para a fixação do robô, podemos ver um exemplo desse pedestal na Figura 56.

Figura 56 - Exemplo de pedestal

Por fim, o último sistema necessário para esta implementação é o sistema de visão, cuja função principal é a deteção da posição dos parafusos na placa. Para tal, será necessário implementar uma câmara que permita identificar com precisão essas posições e transmitir essa informação ao robô, possibilitando-lhe a correta localização dos pontos de aparafusamento e, consequentemente, a execução precisa da fixação das porcas.

Após algum estudo e revisão do que existe no mercado chegou-se à conclusão que a câmara teria de ser de três dimensões, pois os parafusos também iriam ter profundidade diferentes. Para além disto, exigia uma grande precisão nos resultados obtidos, logo, necessitaria de uma boa captação e resolução de imagem, de forma a garantir que as posições passadas ao robô tivessem uma percentagem de erro muito pequena.

Com base nestes requisitos, procedeu-se à pesquisa de soluções adequadas, das quais resultaram duas opções viáveis, conforme se apresenta na Tabela 3.

Tabela 3 - Comparação entre PhoXi 3D Scanner S e Sick TriSpector 1030 [112] [113]

	PhoXi 3D Scanner S	**Sick TriSpector 1030**
Tecnologia	Luz estruturada	Triangulação 3D
Resolução	Até 3,2 milhões de pontos 2064 x 1544	2500 pontos por scan
Depth camera field of view	48º horizontal, 36º vertical	65º
Depth error	< 0,05 mm	< 280 µm
Nuvem de pontos	Grayscale	Grayscale
Min depth	384 mm	141 mm
Max depth	520 mm	541 mm
Frame rate	5 – 6,6 Hz	5000 3D profiles/s
Supported OS	Linux/Windows	Linux/Windows
Conectividade	Ethernet	Ethernet
Dimensões	77 x 68 x 296 mm	217 x 62 x 84 mm
Peso	900g	1,3kg
Preço	≈ 10000 €	≈ 5000 €

Fazendo alguns testes e trocando informações com ambos os fornecedores para saber qual a melhor câmara para a aplicação desejada, chegou-se à conclusão de que a Sick TriSpector 1030 V3T12S-MR32A7, Figura 57, seria a opção mais acertada, sendo suficiente para a tarefa pretendida e bastante mais barata, o que acabou também por ser um ponto muito importante na escolha do material.

Figura 57 - Sick TriSpector 1030 [114]

Uma vez escolhidas as tecnologias para o sistema de visão e de aparafusamento avançou-se para a fase de desenvolvimento do *gripper*, para o suporte destes periféricos. Por parte da Stellantis, esta impunha que todas as fixações teriam de ser encavilhadas e que deveria ser garantido que a aparafusadora fosse montada de modo a que fosse possível e fácil a sua troca ou remoção do *gripper*, e que permanecesse sempre na mesma posição. Tendo em conta os requisitos anteriormente definidos, procedeu-se ao desenvolvimento do *gripper*. Com base em modelos previamente utilizados em aplicações semelhantes, foi concebido um *gripper* em alumínio, no qual tanto a câmara como a aparafusadora se mantêm fixas, conforme ilustrado na Figura 58.

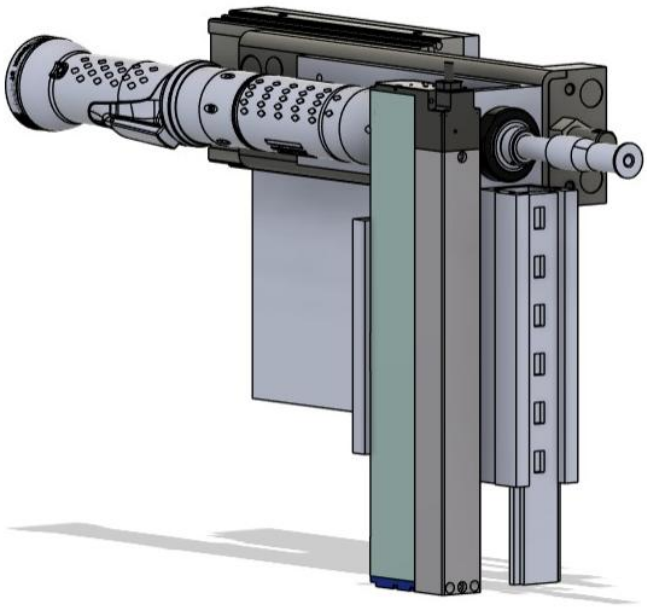


Figura 58 - CAD do Gripper desenvolvido

Foi identificado que, além da cuba vibratória utilizada para alimentar as porcas, seria vantajoso que o próprio *gripper* tivesse um pequeno *stock* de porcas. Desta forma, o robô não precisaria de se deslocar constantemente à cuba para recolher uma porca antes de cada aperto. Para isso, desenvolveu-se um sistema no *gripper* com capacidade para armazenar sete porcas: seis são mantidas numa calha e uma encontra-se já posicionada no bit (o bit é a peça metálica que se encontra na extremidade da aparafusadora e pode ser facilmente substituída).

Este sistema funciona com o auxílio de um mecanismo pneumático que, antes de o robô realizar o aparafusamento, coloca uma nova porca em frente ao bit da aparafusadora. Por meio de outro mecanismo pneumático a aparafusadora recua ligeiramente, a porca é posicionada em frente ao bit e, de seguida, a ferramenta retorna à sua posição inicial. Como o bit contém um íman, a porca fixa-se corretamente, ficando pronta para o próximo aperto. A Figura 59 mostra este sistema de armazenamento no *gripper*.

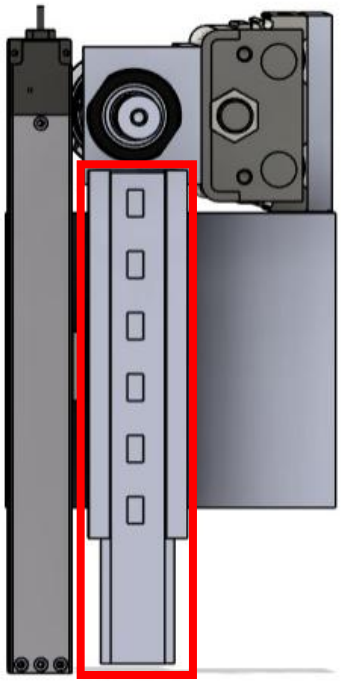


Figura 59 - Stock de 6 porcas

Como é possível observar nas figuras, todo o desenvolvimento foi inicialmente realizado em ambiente virtual, de forma a permitir a realização de testes antes da implementação em ambiente real. Este procedimento teve como objetivo verificar se o *gripper* garantiria o bom funcionamento dos dois periféricos envolvidos e se apresentaria resistência suficiente para os suportar.

Após a realização de vários testes e da seleção do material a utilizar na construção do *gripper*, concluiu-se que o seu peso final não ultrapassaria os 10 kg. Este valor é importante, uma vez que influencia diretamente a escolha do robô, sendo o *payload* um dos principais parâmetros a considerar.

Com o *payload* definido, foi necessário determinar o alcance mínimo que o robô deveria ter para conseguir executar a tarefa. Para isso, recorreu-se a um modelo CAD do interior da carrinha (Figura 60), onde foram feitas medições e simulações. Com base nesses testes, concluiu-se que o robô necessitaria de um alcance mínimo de 1000 mm para cumprir a sua função com eficácia.

Figura 60 - CAD do interior da carrinha

Com o *payload* e o alcance necessários já definidos, passou-se à fase de seleção do robô mais adequado para a tarefa. Após análise da Tabela 2 - Principais características dos cobots da Doosan e Universal Robots [110] , verificou-se que tanto o M1013 da Doosan como o UR10e da Universal Robots satisfaziam os requisitos técnicos do projeto.

Optou-se pelo modelo M1013, representado na Figura 61, principalmente devido ao seu custo mais reduzido, fator relevante tendo em conta que foi decidido utilizar dois robôs em vez de apenas um.

Figura 61 - Doosan M1013 [115]

4.1.2 – Programas utilizados para testes e simulações

Para a elaboração dos testes mencionados em cima e de futuros testes tal como das simulações necessárias e programação dos referidos periféricos foram utilizados os seguintes programas:

O SolidWorks, Figura 62, é um software de CAD amplamente utilizado em várias áreas da engenharia, como por exemplo em mecânica e eletromecânica. Este programa oferece um ambiente intuitivo para modelação tridimensional (3D), permitindo a criação e desenvolvimento de peças, montagens e desenhos técnicos com elevada precisão e eficiência. Neste caso, foi neste que foi feita a elaboração do *gripper* e todos os testes necessários para assegurar a fixação da aparafusadora e da câmara, assim como do *stock* e alimentação das porcas.

Figura 62 – SolidWorks [116]

O RoboDK, Figura 63, é um programa que consiste num software de simulação e programação offline para a área da robótica, amplamente utilizado na indústria para otimizar processos automatizados. Este programa permite simular, programar e controlar uma vasta gama de robôs de diferentes marcas, mais de 1000 braços robóticos [117] , sem haver a necessidade de saber a linguagem de programação dos diferentes robôs, uma vez que a programação é feita numa linguagem universal e posteriormente o próprio programa tem a vantagem de ter a capacidade de gerar o código automaticamente para a linguagem correta de cada robô. Tendo a possibilidade de importar modelos CADs, o sistema que pretendemos simular pode ser exatamente igual ao real, o que torna este programa uma mais-valia para as simulações.

Figura 63 – RoboDK [118]

Normalmente, os fabricantes disponibilizam a opção de adquirir o *Teach Pendant* em conjunto com o robô ou de adquirir apenas o robô. No caso da Doosan, é fornecido um software denominado Dart-Platform Figura 64 que permite realizar todas as

configurações e programações que seriam normalmente efetuadas através do *Teach Pendant*.

Este software permite que toda a configuração e programação do robô seja realizada diretamente a partir de um computador, o que torna o processo mais simples e rápido. A plataforma oferece todas as funcionalidades necessárias para configurar o robô e programar as suas operações, recorrendo a uma abordagem de programação por blocos. Este tipo de programação é bastante intuitiva e de fácil aprendizagem, sendo especialmente indicado para operadores com pouca experiência ou iniciantes na área da robótica colaborativa.

A Dart-Platform permite, assim, desenvolver programas completos para aplicações industriais, garantindo a flexibilidade e funcionalidade necessárias sem comprometer a eficiência ou a fiabilidade do sistema

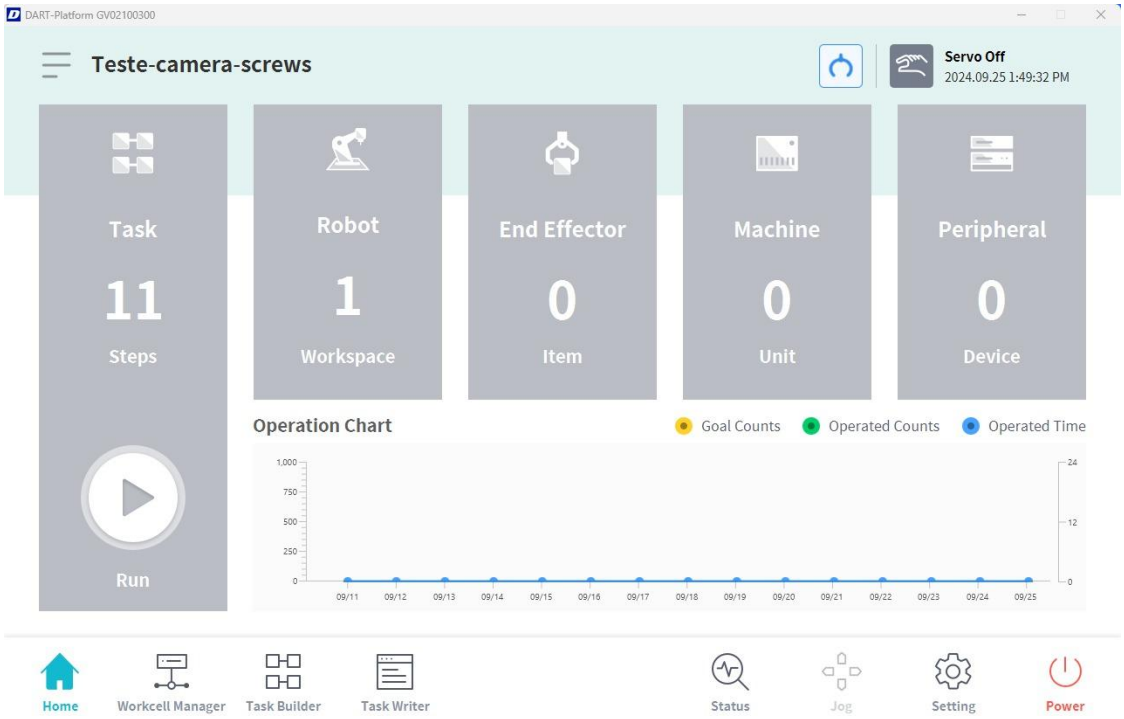


Figura 64 - Software Dart-Platform [106]

O último software que foi utilizado neste projeto foi o Dart-Studio (Figura 65). Este é um programa que necessita de licença, o que acarreta um custo extra para além da compra do robô, mas fornece um controlo total sobre a sua programação, uma vez que a programação é feita via código e não por blocos, como no Dart-Platform, possibilitando uma programação mais detalhada em vários níveis.

Uma das principais vantagens da utilização deste software está na integração com sistemas externos, como por exemplo sistemas de visão e de aparafusamento. Como a programação é feita por código, interligar os sistemas e permitir a comunicação entre eles torna-se possível, já que é suportada a comunicação por TCP/IP e por *sockets*.

Para além disso, o facto de a programação ser bastante detalhada e escrita linha a linha torna este software mais eficaz e preciso.

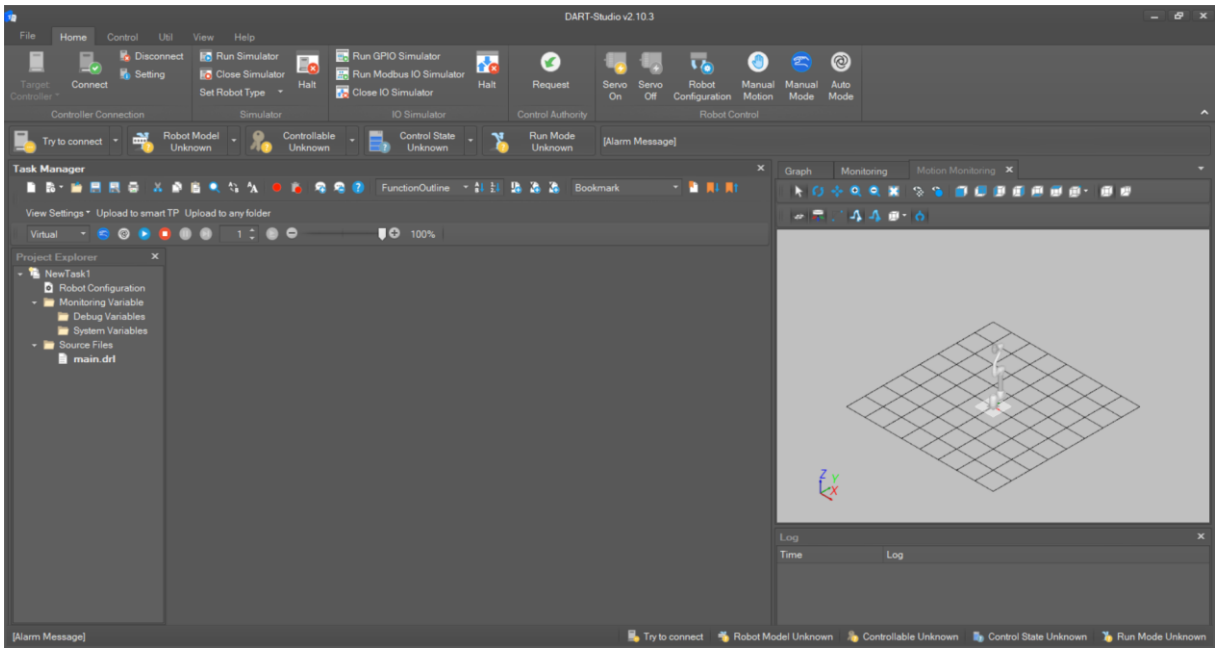


Figura 65 - Software Dart-Studio [106]

4.1.3 – Simulações em ambiente virtual

Escolhidos os periféricos mais adequados para a elaboração do projeto e com a elaboração do *gripper* que será utilizado, passou-se para a fase das simulações em ambiente virtual.

RoboDK

Para este efeito, foi utilizado o software RoboDK [117] , considerado um dos melhores softwares de simulação de automação robótica disponíveis no mercado, devido à possibilidade de importar ficheiros CAD, permitindo simular com elevada precisão o ambiente real. O programa conta com uma livraria de mais de 1000 robôs disponíveis para utilização, o que o tornou a escolha ideal para a realização de simulações e testes.

O primeiro passo consistiu na importação dos ficheiros CAD necessários para a simulação. Este processo é bastante simples no software, sendo apenas necessário clicar no ícone de importação, o que abre uma janela onde é possível selecionar os ficheiros a importar, como é apresentado na Figura 66.

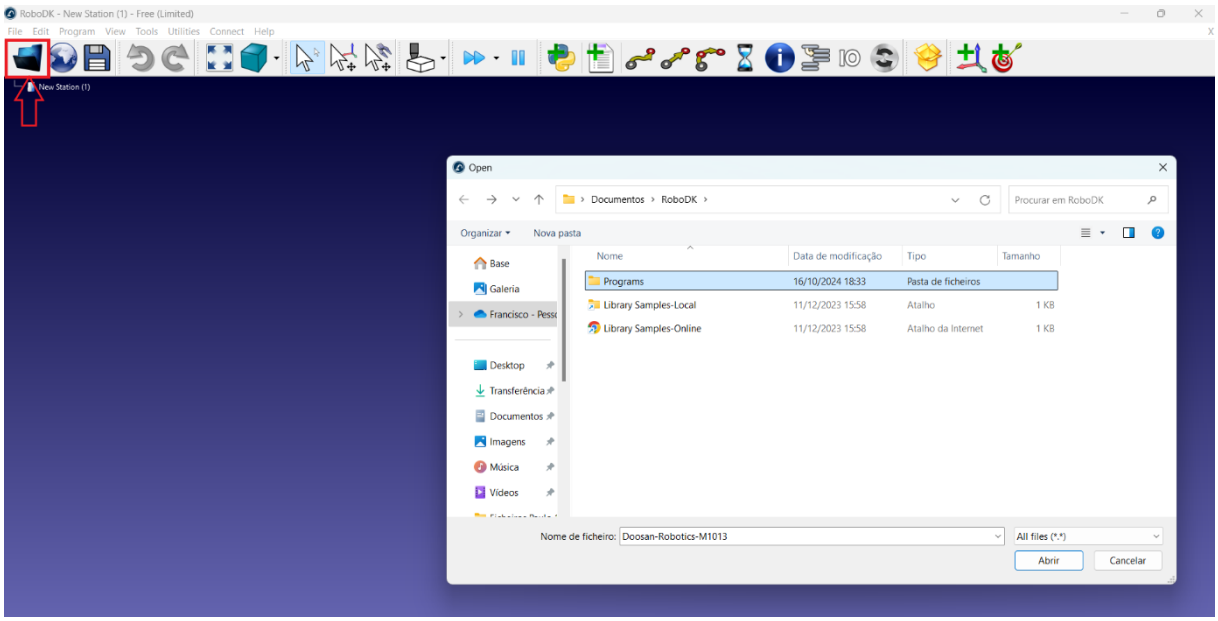


Figura 66 - Importar ficheiros [106]

Procedeu-se à importação de todos os objetos necessários para esta simulação, começando pela carrinha (Figura 67), fornecido pela empresa Stellantis.

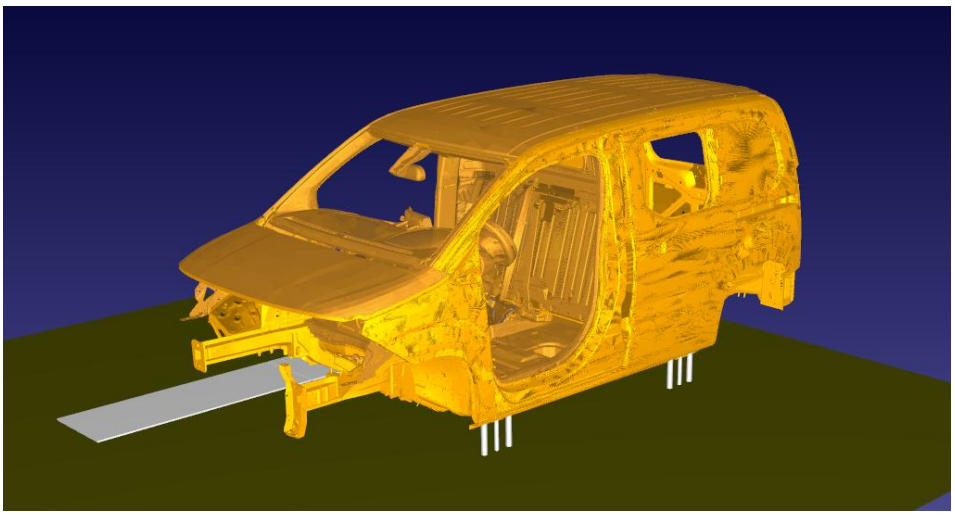


Figura 67 - CAD correspondente à carrinha

De seguida, procedeu-se à importação e posicionamento dos *cobot*s com os respetivos suportes, dos *grippers* desenvolvidos anteriormente e da cuba vibratória responsável pela alimentação das porcas, ficando assim concluído o *setup* necessário para a execução do projeto (Figura 68).

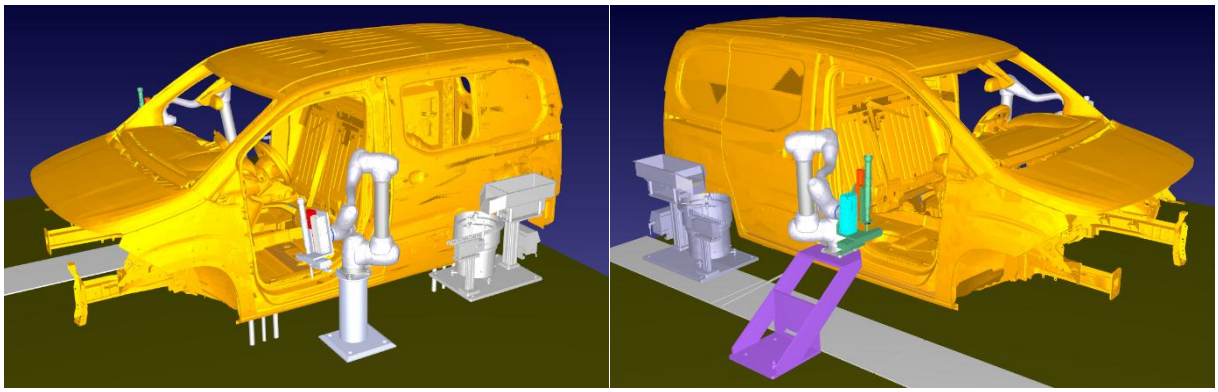


Figura 68 - Primeiro *setup*

Para importar o robô, é necessário aceder à livraria disponibilizada pelo software. Para isso, deve-se clicar no ícone correspondente, o qual abre uma página web com todos os modelos de robôs disponíveis. Após selecionar o modelo pretendido, procede-se ao seu download, como é possível observar na figura seguinte.

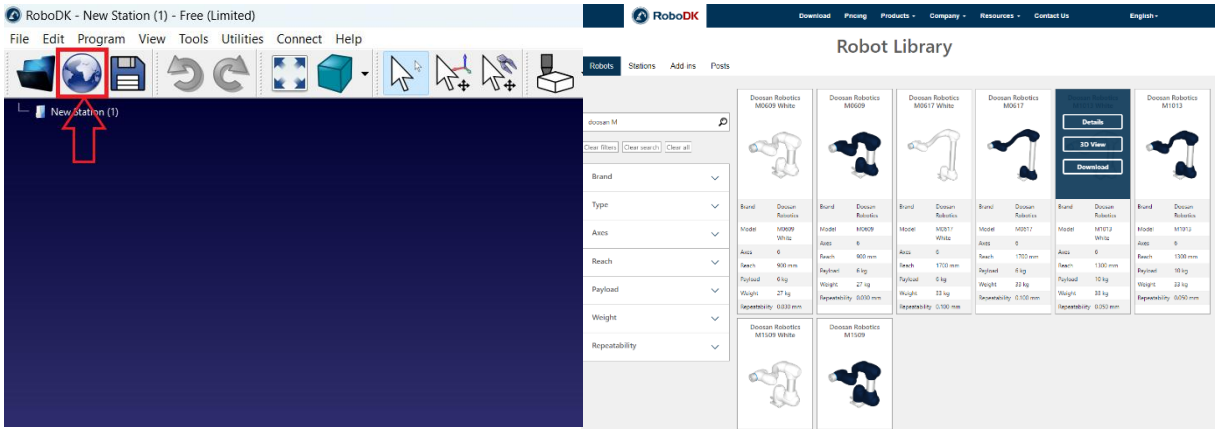


Figura 69 - Biblioteca de robôs disponíveis

De seguida, basta aceder novamente ao ícone de importação, e importar o robô, conforme é possível observar na Figura 70.

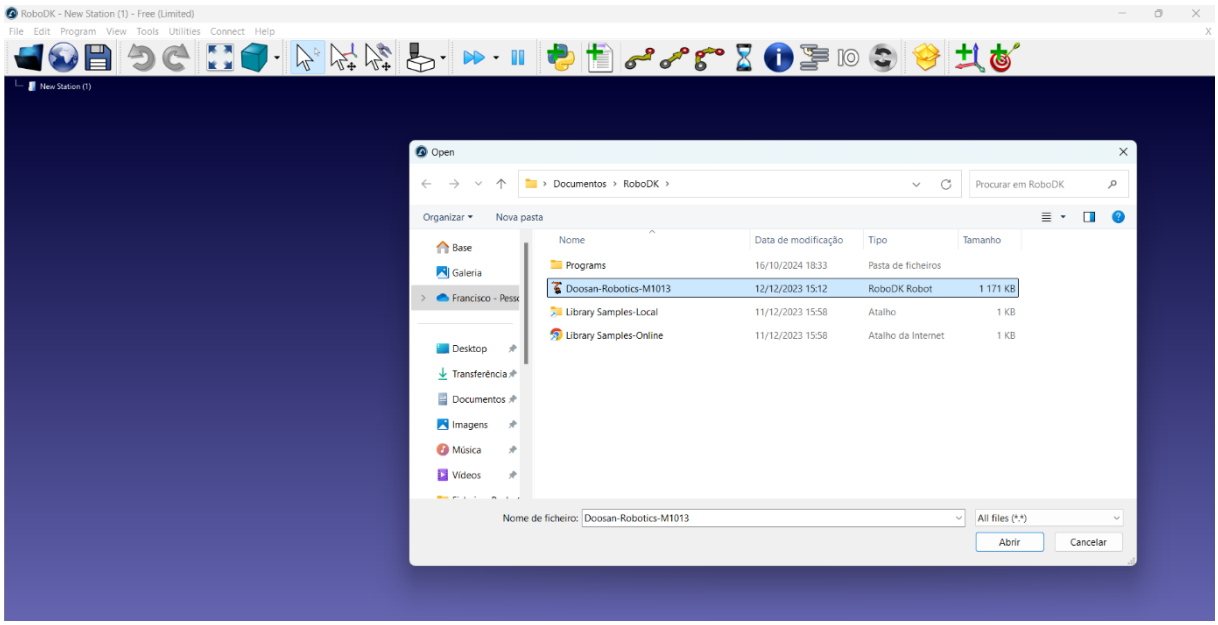


Figura 70 - Inserir robô

A Figura 68 ilustra o primeiro *setup* utilizado na execução da tarefa, com os periféricos devidamente posicionados. Foram utilizados dois pedestais diferentes para o suporte de cada robô, com o objetivo de avaliar qual seria a opção mais adequada.

Na Figura 71, é ainda possível observar o *gripper* desenvolvido, já com a aparafusadora e a câmara integradas.

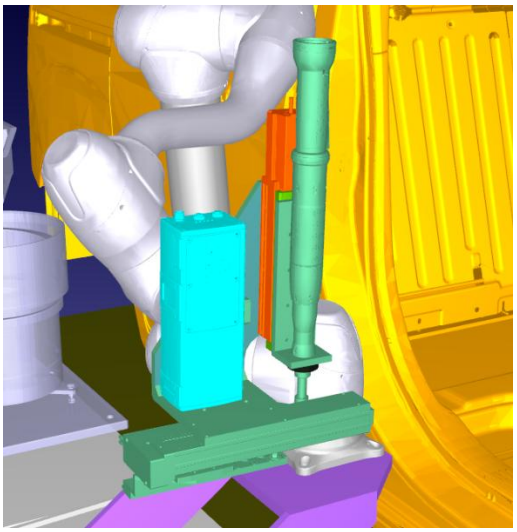


Figura 71 - CAD do gripper desenvolvido

Após algumas verificações, concluiu-se que a posição inicial das cubas vibratórias não era a mais eficiente para a inserção das porcas no *stock* existente no *gripper*. Por esse motivo, optou-se por reposicionar as cubas, colocando-as à frente dos robôs em vez de atrás, como estavam inicialmente. A nova disposição pode ser observada na figura seguinte, passando este a ser o *setup* final utilizado para a realização dos testes.

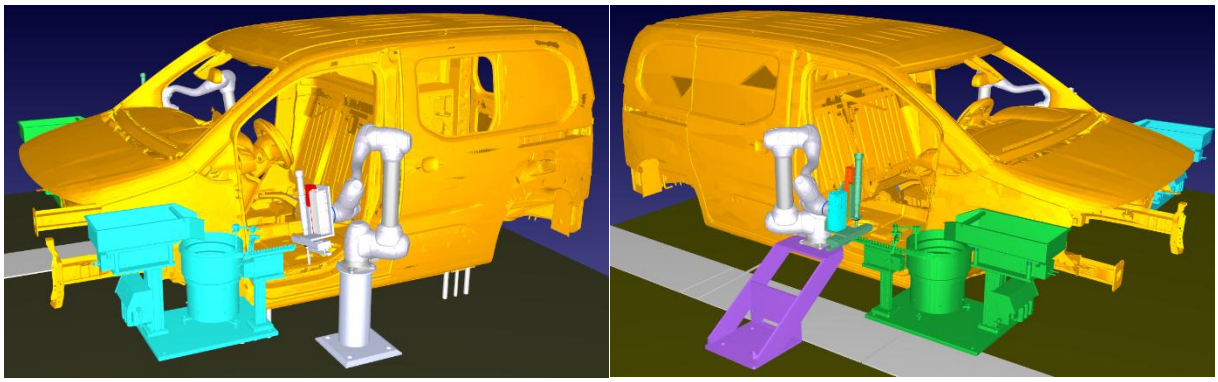


Figura 72 - Setup final para elaboração de testes

Relativamente à alimentação das porcas, foi utilizado o sistema de cubas vibratórias, responsável por transportar as porcas até ao bit do robô, bem como até ao *stock* existente no *gripper*. Tanto a cuba como o sistema de alimentação das porcas podem ser observados nas figuras seguintes.

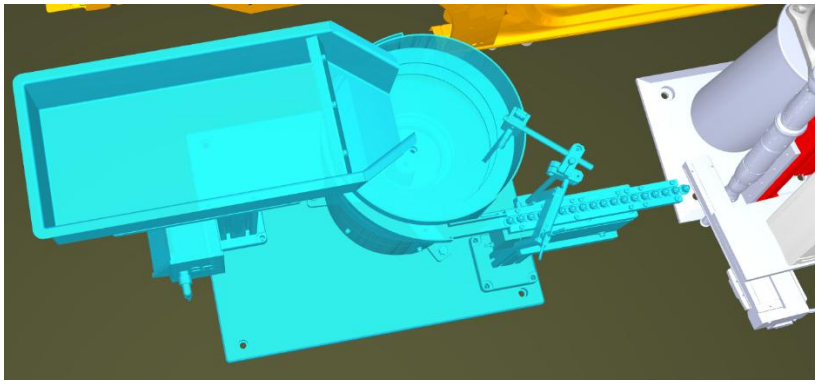


Figura 73 - Cuba vibratória

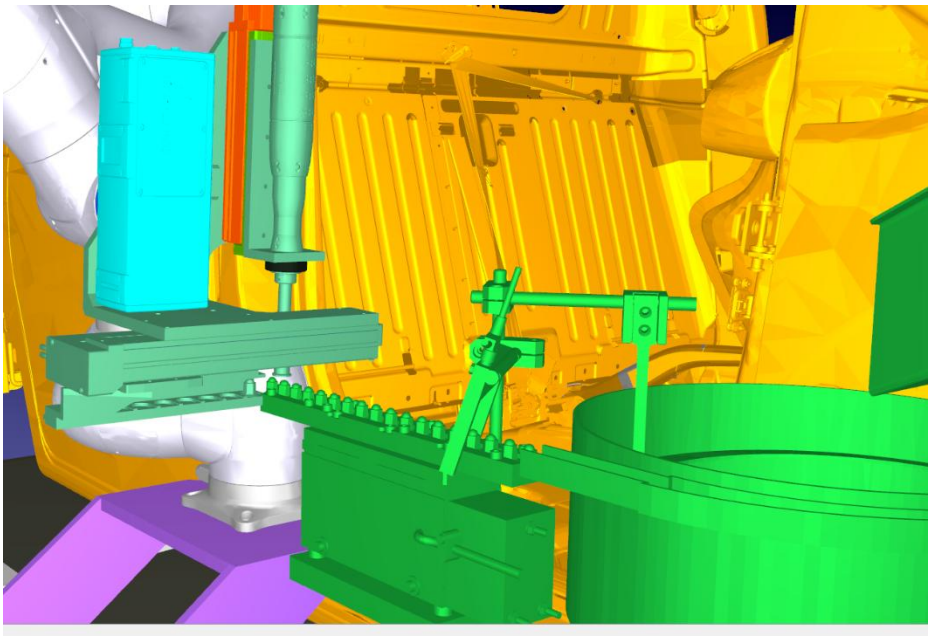


Figura 74 - Alimentação e stock das porcas

Como se pode constatar, o programa possui um sistema de simulação bastante próximo da realidade, permitindo a importação de todas as peças necessárias para a realização da simulação, sendo apenas limitado pelo próprio ficheiro CAD. Com a conclusão deste processo, considera-se finalizada a fase de importação, resultando num ambiente virtual (Figura 75).

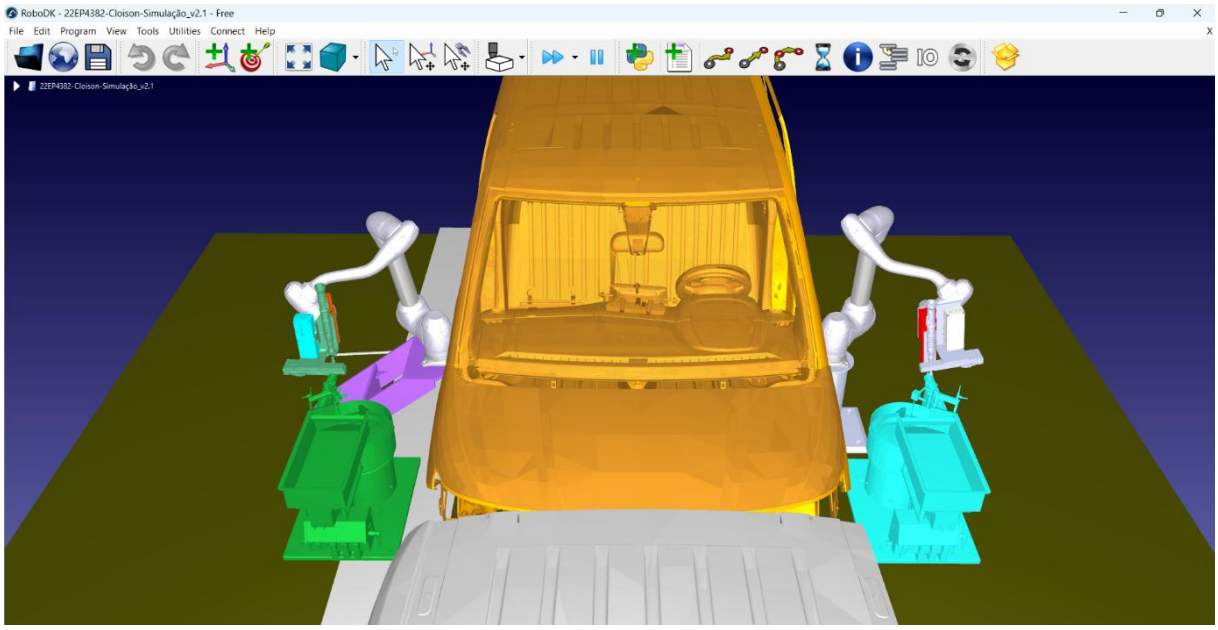


Figura 75 - Ambiente virtual

A partir deste ponto, foi possível avançar para a realização dos testes em ambiente virtual. O principal objetivo das simulações consistiu em validar as posições dos *cobots* e garantir que ambos conseguiam alcançar todos os pontos necessários para o aperto das porcas.

Inicialmente, foi necessário identificar quais as porcas que cada robô iria apertar, de forma a evitar sobreposições nas suas áreas de atuação. Definiu-se que o robô do lado direito ficaria responsável pelas porcas localizadas do lado direito da chapa, enquanto o robô do lado esquerdo apertaria as porcas do lado esquerdo.

Relativamente às porcas localizadas na zona central da chapa, foi considerado que a porca na posição 1 (de cima para baixo) já se encontra previamente apertada, garantindo o correto posicionamento da chapa. Assim, ficou definido que as porcas nas posições 2, 3, 6, 7 e 8 seriam apertadas pelo robô do lado esquerdo, além das restantes do lado esquerdo da chapa. As porcas nas posições 4 e 5, bem como as restantes do lado direito da chapa, seriam apertadas pelo robô da direita, estando de frente para a carrinha.

Esta divisão justifica-se pelo facto de o lado direito da chapa conter mais porcas para apertar (quatro) do que o lado esquerdo (duas), equilibrando assim o número de operações atribuídas a cada robô. Esta distribuição pode ser visualizada de forma

mais clara na Figura 76, onde a porca previamente apertada está representada a verde, as porcas atribuídas ao robô da esquerda a vermelho, e as do robô da direita a azul.

Figura 76 - Identificação das porcas

Definidas as porcas que cada robô teria de apertar, foi necessário testar as posições dessas porcas para garantir que ambos seriam capazes de realizar todas as operações atribuídas. Com esse objetivo, recorreu-se ao software para efetuar os testes, começando por posicionar o robô em cada um dos pontos correspondentes aos parafusos que lhe estavam atribuídos, assegurando que seria possível executar o respetivo aperto.

Para isso, abriu-se o painel de controlo do robô em questão, através de duplo clique sobre o modelo no ambiente virtual, o que permitiu aceder ao seu menu de controlo. A partir deste painel, foi possível movimentar o robô, iniciando-se o processo pelo posicionamento na primeira porca, ajustando os valores dos eixos X, Y, Z, Rx, Ry e Rz (Figura 77).

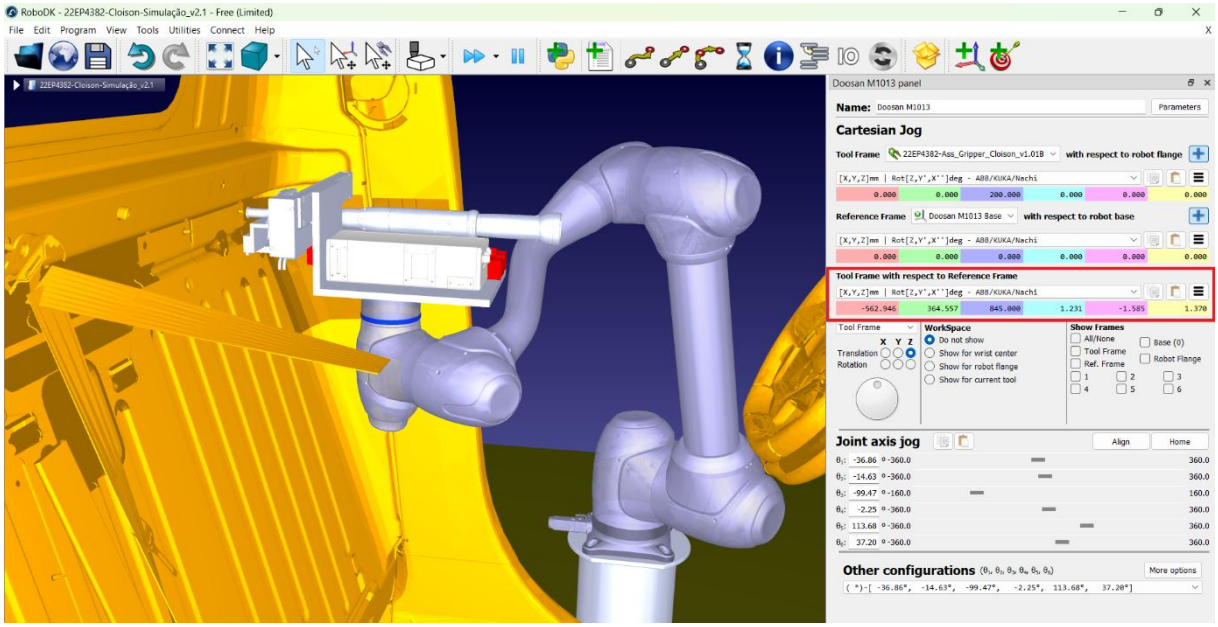


Figura 77 - Movimentar o robô

Para guardar a posição do robô, é possível criar um "*target*", que regista a posição atual do robô, permitindo que este seja movimentado e posteriormente retorne à posição guardada. Este método foi utilizado para registar todas as posições associadas às operações de aperto das porcas atribuídas ao robô.

As posições registadas são armazenadas na *Station Tree*, uma estrutura em árvore que organiza os diversos elementos do projeto, incluindo as peças importadas, os *targets* definidos e, quando aplicável, o programa desenvolvido. Como ilustrado na figura abaixo, foi adicionado o "*Target* 6", conforme indicado na *Station Tree*. A partir desse momento, a posição ocupada pelo *cobot* fica associada a esse *target*, podendo ser acedida sempre que necessário.

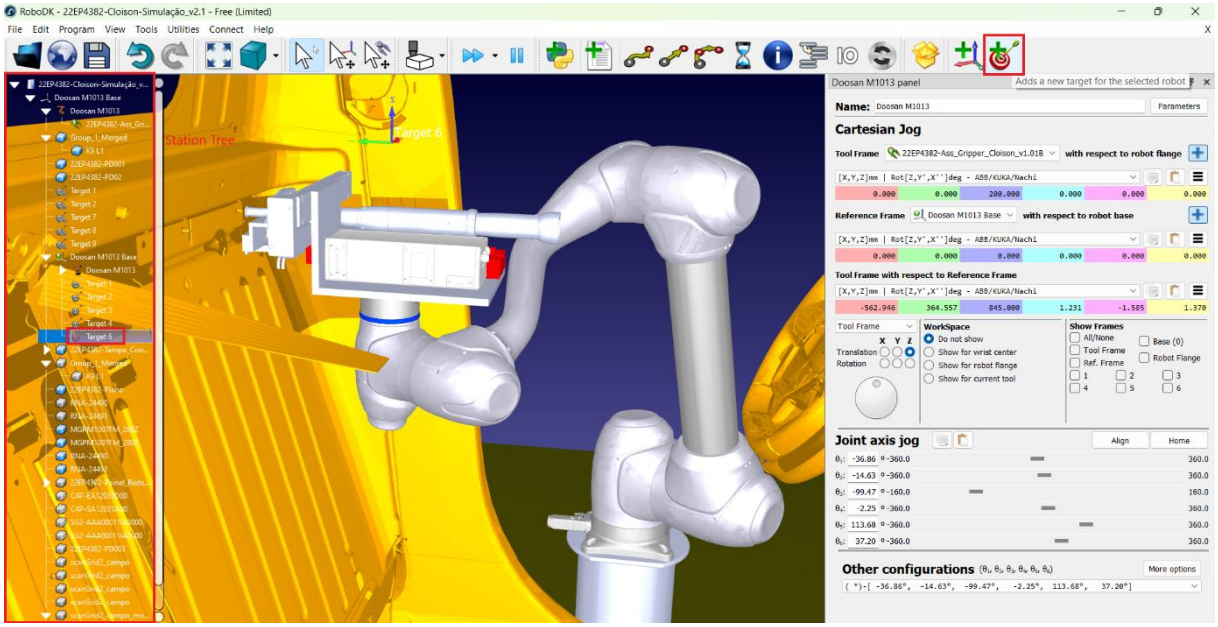


Figura 78 - Guardar uma posição

Existe ainda a possibilidade de aceder às informações do *target*, onde é possível consultar a posição guardada, bem como alterar os respetivos parâmetros. Estas modificações podem ser feitas tanto em coordenadas cartesianas como em coordenadas relativas às juntas do robô, conforme se pode observar na Figura 79.

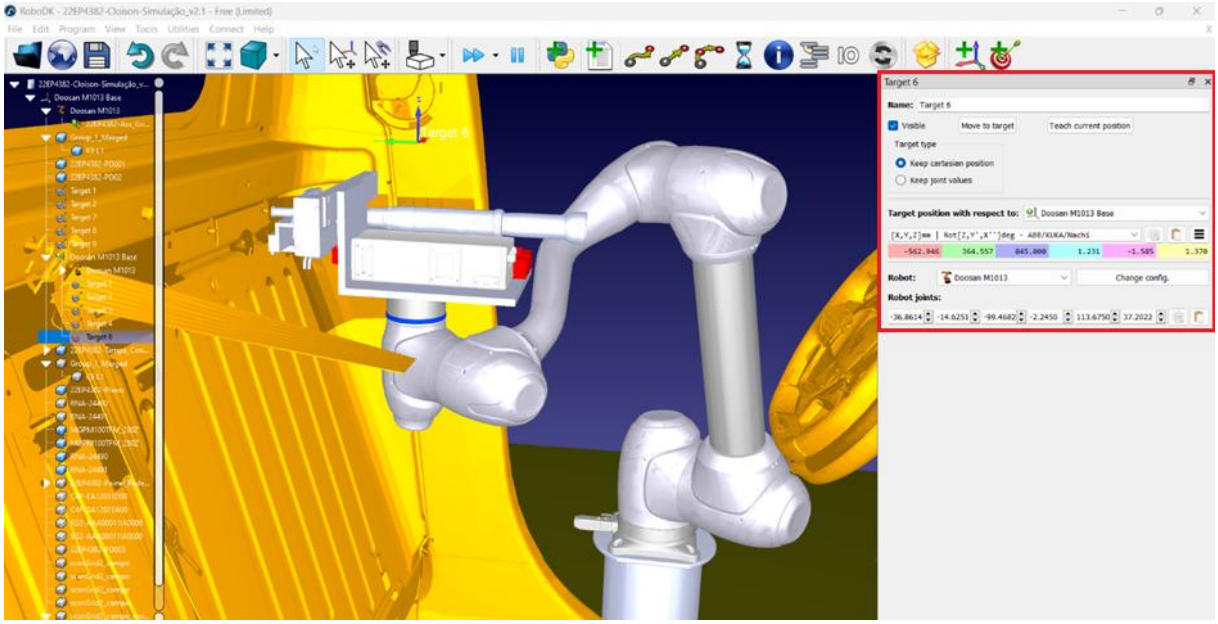


Figura 79 - Informações do target inserido

De seguida, o mesmo procedimento foi repetido para as restantes posições, tanto no robô da direita como no da esquerda, tendo sido todas essas posições devidamente guardadas. Desta forma, foram realizados os testes necessários para assegurar que ambos os *cobot*s conseguiam realizar corretamente o aperto de todas as porcas, sem qualquer impedimento. Nas figuras seguintes é possível observar alguns dos pontos registados durante os testes efetuados.

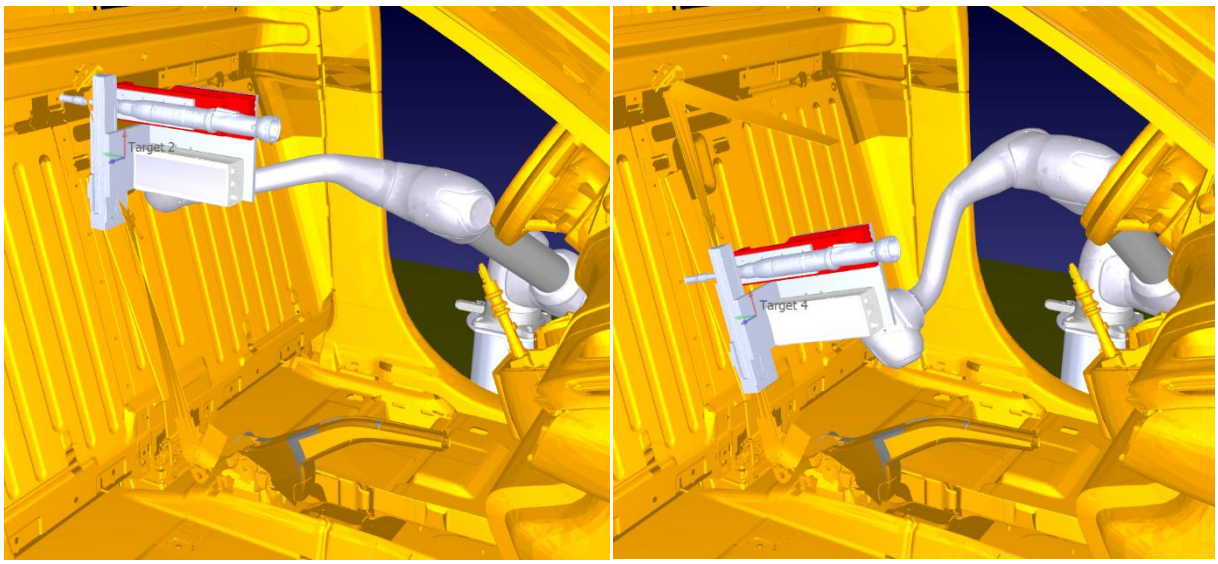


Figura 80 - Posições de aperto do robô da direita

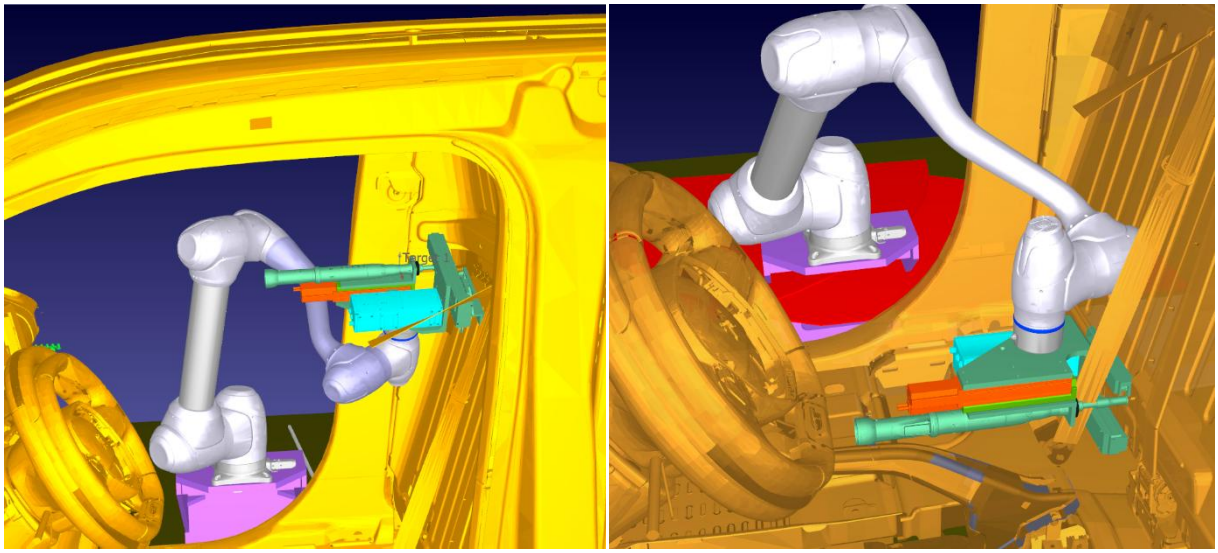


Figura 81 - Posições de aperto do robô da esquerda

Como se pode observar, confirmou-se que ambos os robôs conseguiam alcançar as posições exigidas para executar essas operações de forma adequada.

4.1.4 – Montagem em ambiente real na Europneumaq

Com as simulações concluídas, avançou-se para a realização de testes em ambiente real. Para tal, foi necessário proceder à aquisição dos periféricos, nomeadamente os robôs, as câmaras e as aparafusadoras. Os primeiros testes foram realizados nas instalações da Europneumaq, antes da sua implementação em chão de fábrica, uma vez que não era viável interromper a produção da fábrica para a realização destes testes.

O objetivo foi realizar todos os testes e programações dos periféricos na Europneumaq, de forma a garantir que, no momento da montagem em chão de fábrica, a maioria da programação já estivesse finalizada. Isto deve-se ao facto de que, após a instalação na Stellantis, qualquer ajuste ou intervenção só poderia ser feito durante os períodos em que a produção estivesse parada, o que limitava significativamente o tempo disponível (apenas aos sábados à tarde).

Neste sentido, a Stellantis disponibilizou uma carrinha de testes à Europneumaq, iniciando-se a fase da automação do processo em ambiente real.

Elaboração do *gripper*

Foi fabricado o *gripper* responsável por suportar a câmara e a aparafusadora. Este dispositivo é composto por dois sistemas pneumáticos: o primeiro permite o recuo e avanço da aparafusadora, sendo utilizado para inserir a porca no respetivo bit; o segundo é responsável pelo posicionamento das porcas armazenadas no *stock* incorporado no sistema. Na figura seguinte, é possível observar o *gripper* desenvolvido, bem como o movimento realizado pelo sistema pneumático associado à aparafusadora, indicado pela seta vermelha.

Figura 82 - Gripper

Na Figura 83 é apresentado o *stock* de porcas que o *gripper* é capaz de transportar, assim como o movimento do sistema pneumático correspondente, responsável por posicionar a porca no bit da aparafusadora.

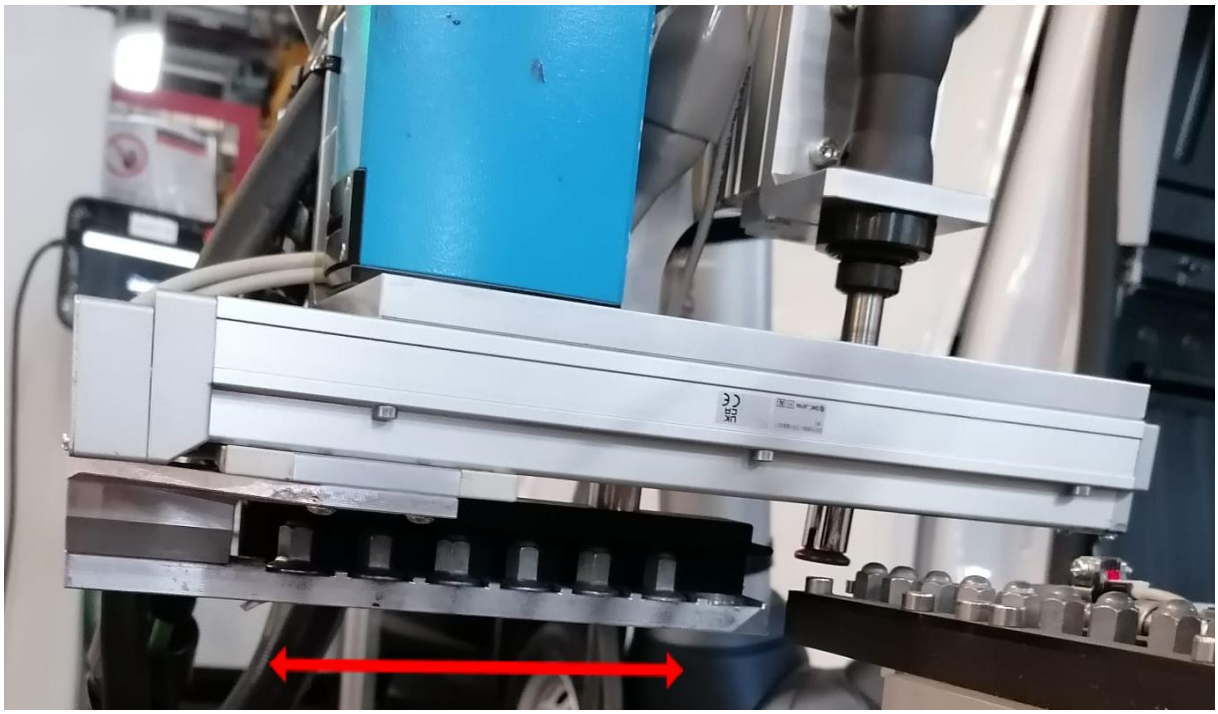


Figura 83 - Stock de porcas existente no gripper

Dart-Platform

Com a chegada dos periféricos e dos materiais necessários à execução do projeto, procedeu-se à configuração do *cobot*. Para este fim, recorreu-se ao software Dart-Platform, que emula as funcionalidades disponíveis no *teach pendant*, mas em ambiente computacional, tornando o processo mais eficiente.

Inicialmente, foram realizadas todas as configurações possíveis através deste programa, antes de se avançar para a fase de programação do robô. A primeira etapa consistiu na definição dos endereços de IP do controlador do robô, permitindo a futura comunicação com os periféricos. A configuração realizada pode ser visualizada na figura abaixo.

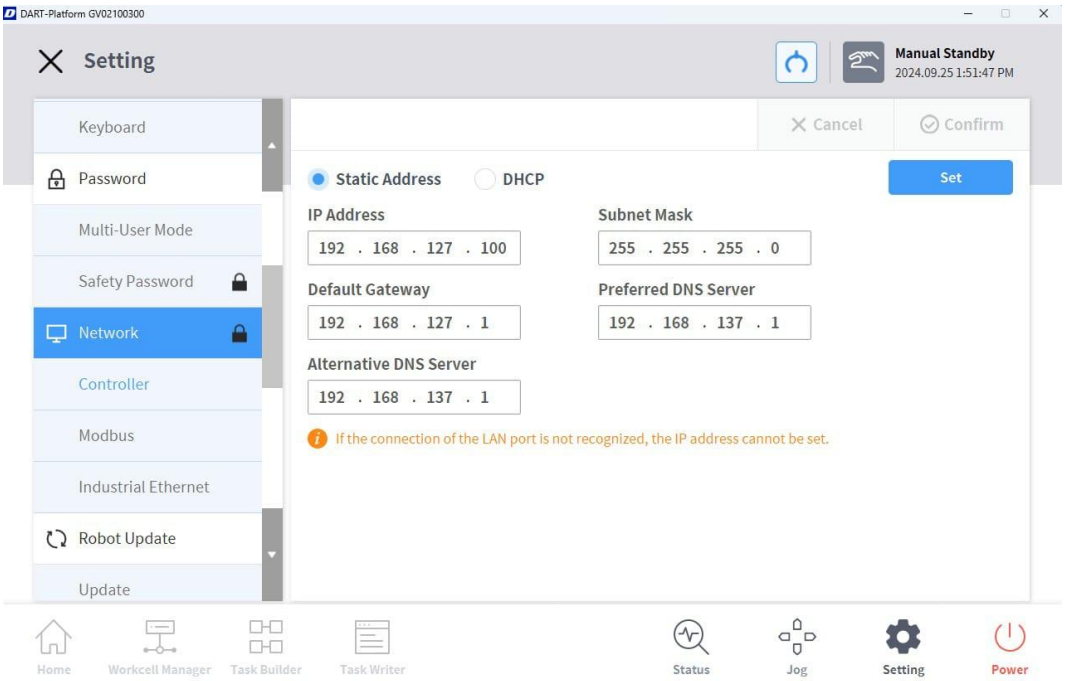


Figura 84 - Configuração do IP do Cobot

De seguida, procedeu-se à definição do *Tool Center Point* (TCP) e do peso do *gripper*, permitindo que o robô reconheça corretamente a presença e as características do dispositivo acoplado à sua extremidade. Esta etapa é fundamental, uma vez que tanto o TCP como o peso do *gripper* influenciam diretamente o comportamento e a precisão dos movimentos do robô.

Para a definição do TCP, recorreu-se à funcionalidade do robô que permite o cálculo automático a partir de quatro pontos. Neste processo, o *cobot* é posicionado exatamente no mesmo ponto, mas com quatro orientações diferentes. Com base nesses dados, o próprio programa calcula o TCP do *gripper* (Figura 85).

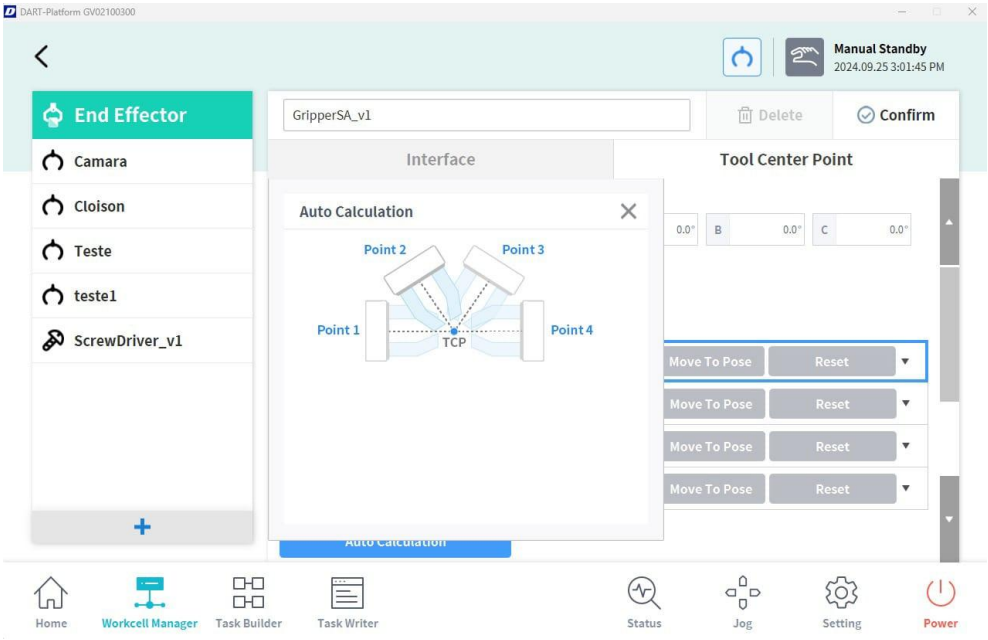


Figura 85 - Exemplo de cálculo do TCP

Foram registados os quatro pontos necessários para o cálculo do TCP do *gripper*, tendo-se procedido de seguida à sua determinação.

Relativamente ao peso do *gripper* e ao respetivo centro de gravidade, este processo é realizado de forma totalmente automática pelo robô. O utilizador pode optar entre dois modos: um em que são movimentadas todas as juntas ou outro em que apenas são movimentadas as juntas 3, 4 e 5, sendo esta última opção menos precisa.

Após garantir que o *cobot* dispunha do espaço necessário para a realização do procedimento, deu-se início à sua execução. Concluídos os movimentos requeridos, o sistema determina automaticamente o valor do peso e do centro de gravidade, conforme ilustrado na figura seguinte.

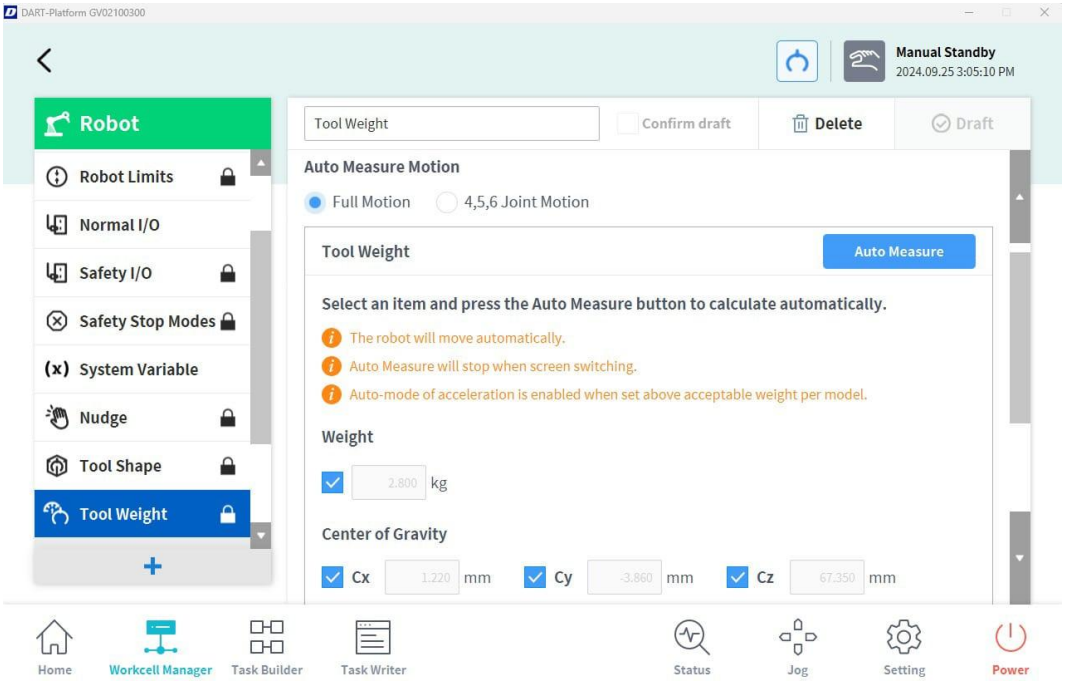


Figura 86 - Exemplo de cálculo do peso e do centro de gravidade do gripper

De seguida, procedeu-se à montagem de todos os periféricos nos locais pretendidos (Figura 87), uma vez que as configurações seguintes requerem que estes se encontrem posicionados no local onde irão desempenhar a sua função.

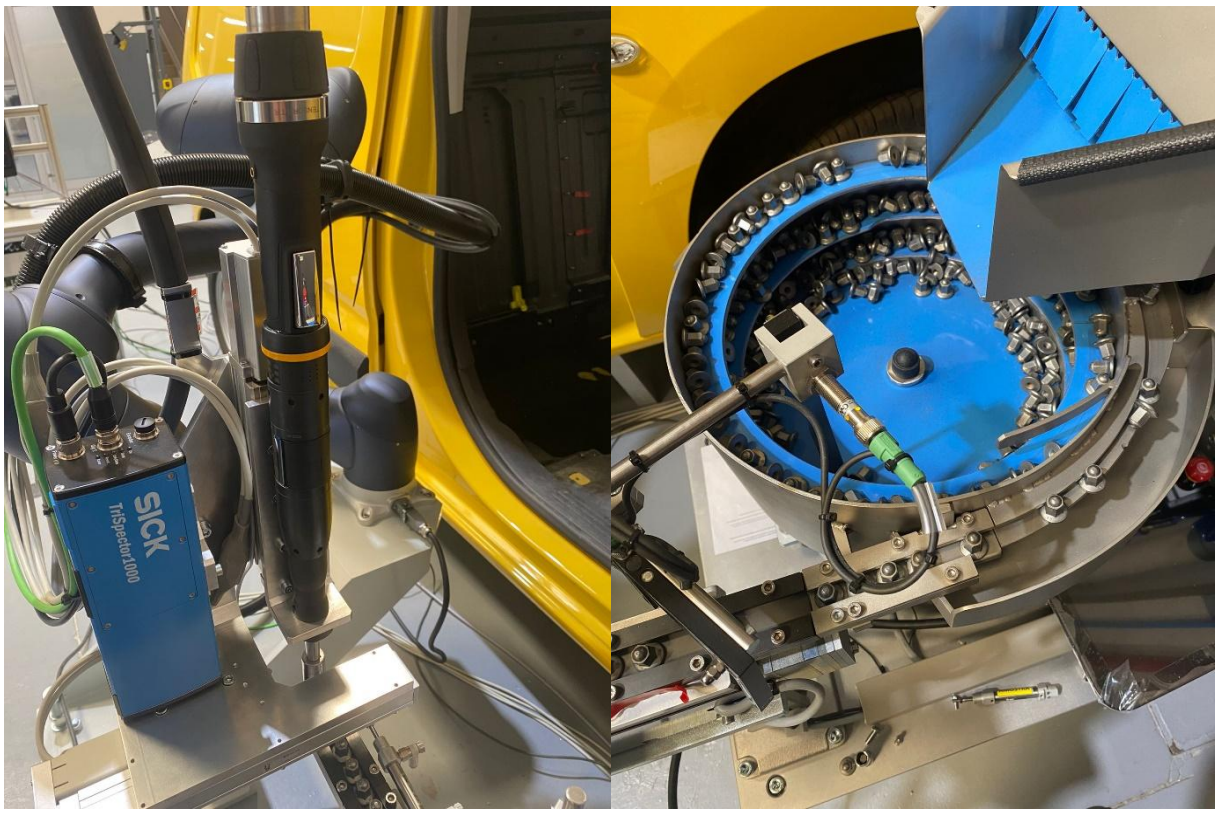


Figura 87 - Periféricos montados

Concluída a montagem, tornou-se possível proceder às configurações restantes do robô. A primeira etapa consistiu na definição da "*Home Position*", que corresponde à posição em que o robô permanece quando não está a executar nenhuma tarefa, sendo assim a posição inicial e final associada ao ciclo de trabalho. O sistema permite a definição de duas posições: uma posição predefinida (*default*) e uma segunda posição, configurável pelo utilizador (Figura 88).

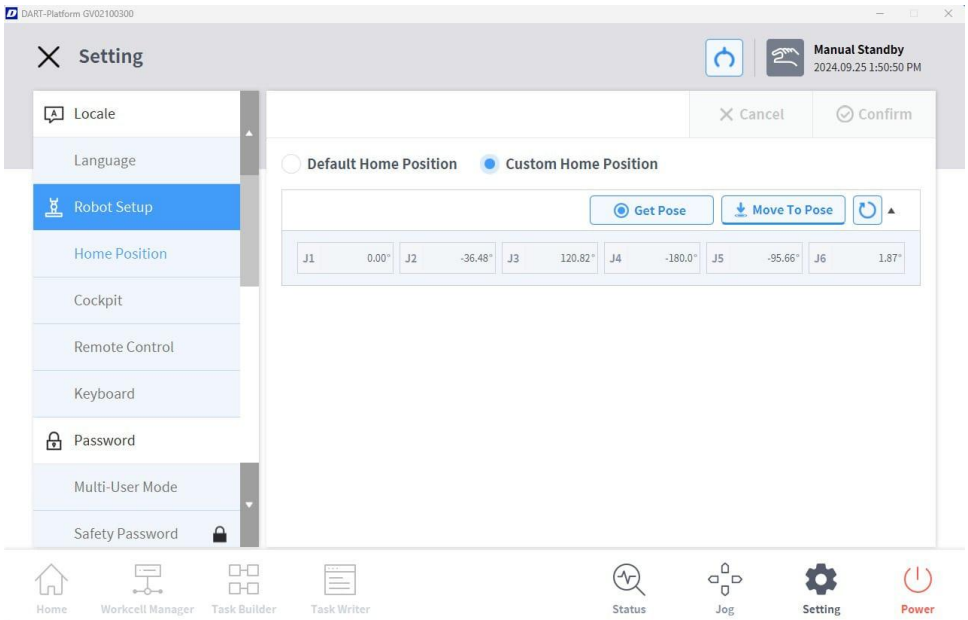


Figura 88 - Exemplo de uma *Home Position*

A zona colaborativa de um *cobot* constitui uma das configurações mais relevantes a definir no processo de implementação. Esta funcionalidade, é responsável pela crescente preferência pelos *cobots* em relação aos robôs industriais, uma vez que permite a sua integração em ambientes partilhados com operadores.

Através da definição desta zona, o *cobot* é capaz de detetar a presença de pessoas na sua proximidade e, automaticamente, ajustar os seus parâmetros operacionais, como a velocidade, a força e a aceleração, para garantir níveis de segurança compatíveis com a interação humana. Com base nestes princípios, procedeu-se à configuração da zona colaborativa, sendo possível selecionar o seu formato e dimensões de acordo com os requisitos específicos da aplicação em causa, conforme se pode observar na Figura 89.

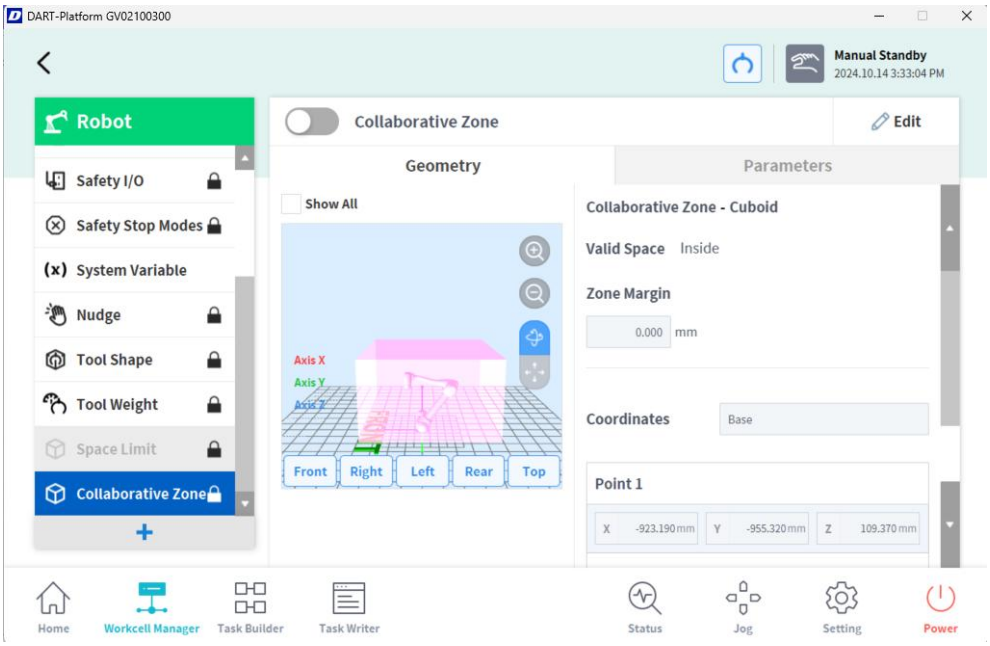


Figura 89 - Definição da zona colaborativa

Após a definição da zona colaborativa, é também possível configurar uma zona designada por *space limit* (Figura 90). Esta funcionalidade permite ao robô reconhecer determinadas áreas como zonas restritas, impedindo-o de se movimentar para dentro dessas regiões, mesmo que as instruções programadas assim o indiquem.

Adicionalmente, existe a possibilidade de definir o *space limit* de forma inversa, isto é, restringindo o movimento do *cobot* exclusivamente ao interior da zona definida. Esta opção é particularmente útil em contextos onde se pretende prevenir colisões com equipamentos, estruturas ou operadores, contribuindo assim para um ambiente de trabalho mais seguro.

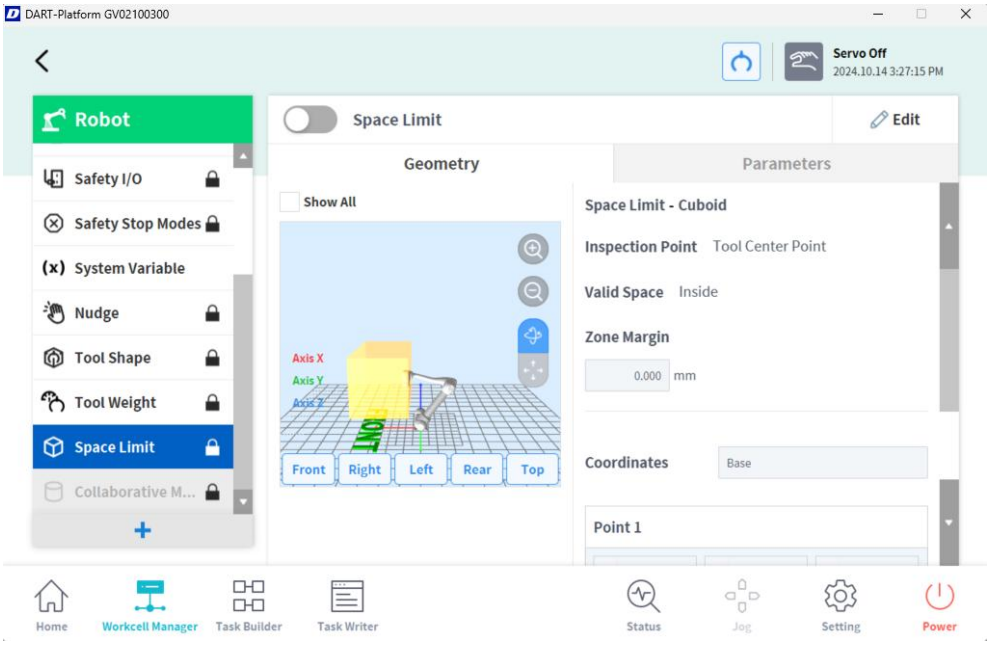


Figura 90 - Definição de space limit

Para concluir as configurações disponíveis neste software, recorreu-se à funcionalidade de criação de um *frame* de coordenadas. Este *frame* corresponde a um plano de referência que será utilizado posteriormente para registar os pontos relativos às posições de aparafusamento. A explicação detalhada sobre a sua aplicação prática será abordada numa secção posterior.

A criação deste *frame* requer a definição de três pontos distintos. Com base nestes pontos, o sistema constrói automaticamente o plano de referência, como demonstrado nas figuras seguintes.

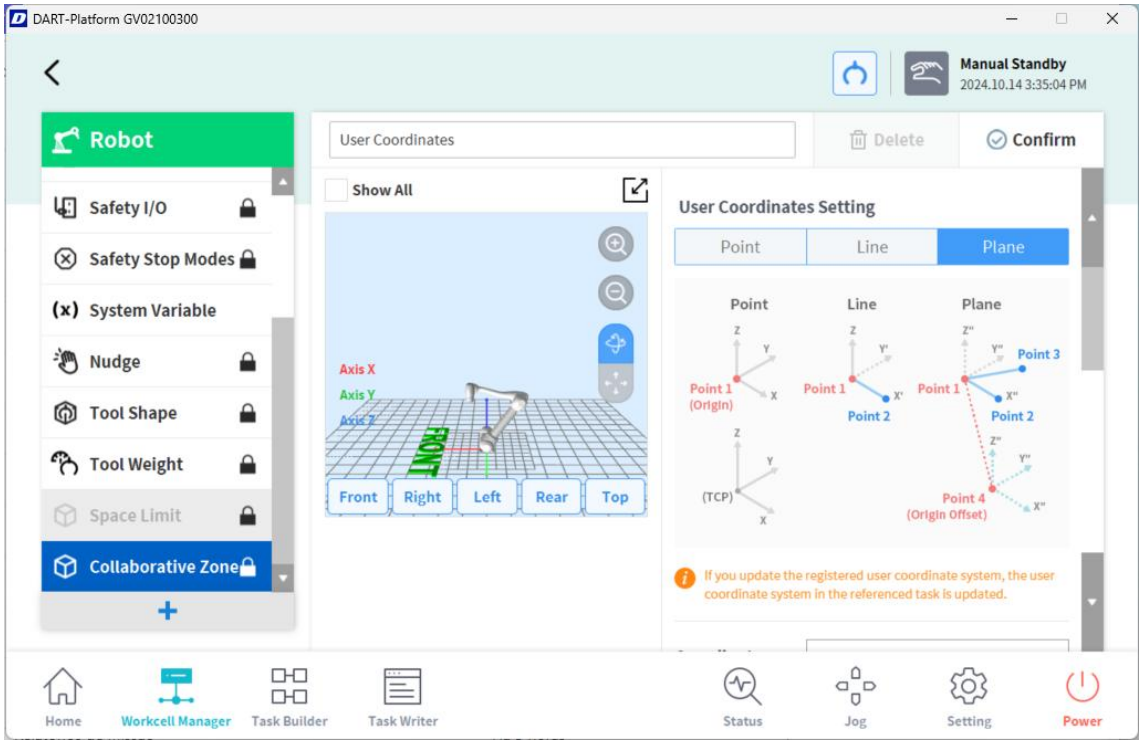


Figura 91 - Exemplo de como criar um *frame*

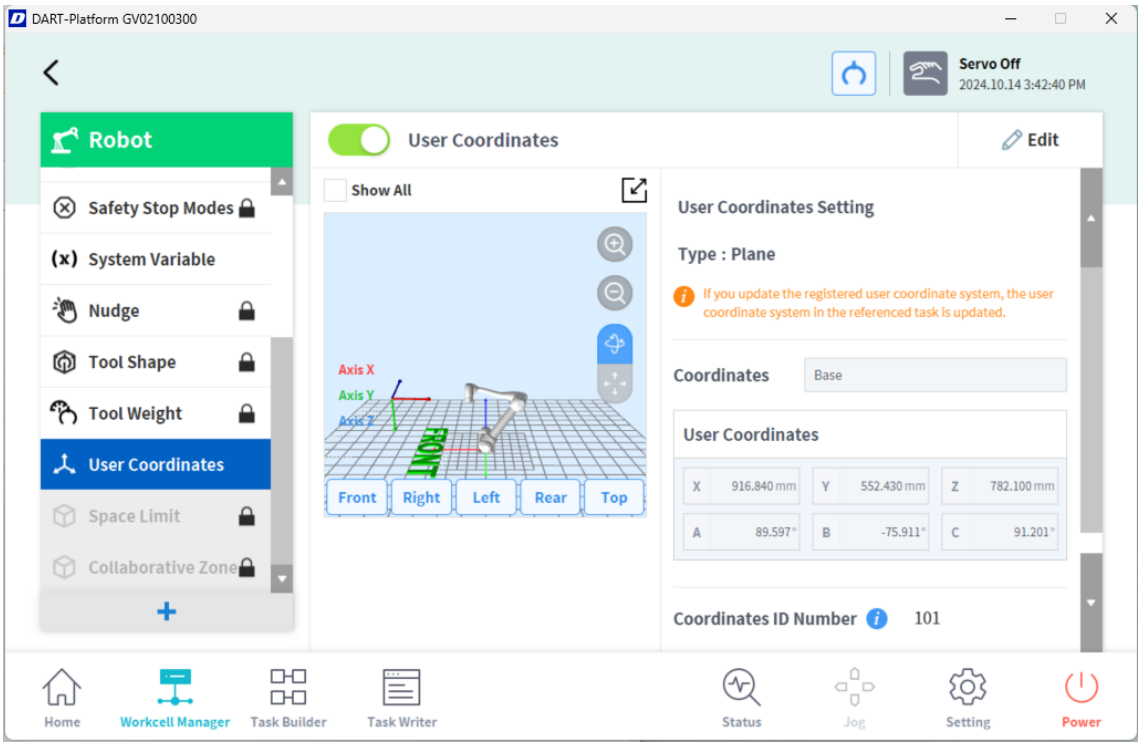


Figura 92 - *Frame* criado com as coordenadas relativas

Como se pode observar na Figura 92, a partir dos três pontos previamente registados foi criado o *frame* 101.

4.1.5 – Programação dos periféricos

Após efetuar todas as configurações necessárias do robô, passou-se à programação dos periféricos que seriam utilizados para a realização do projeto, sendo estes: a câmara, a aparafusadora e o robô.

Sistema de visão

Relativamente ao sistema de visão, foi utilizado o software SOPAS, desenvolvido pela própria SICK, para proceder à configuração e programação dos periféricos da marca. Este software disponibiliza uma *interface* gráfica intuitiva, que permite configurar os parâmetros dos dispositivos, visualizar dados em tempo real e elaborar o respetivo programa.

Ao iniciar o programa e selecionar a câmara à qual se está conectado, é apresentada a página *image*, onde são realizadas as configurações relacionadas com o tratamento da imagem adquirida. Nesta secção, é possível ajustar diversos parâmetros, de forma a garantir que a imagem captada apresenta a melhor qualidade possível, conforme ilustrado na Figura 93.

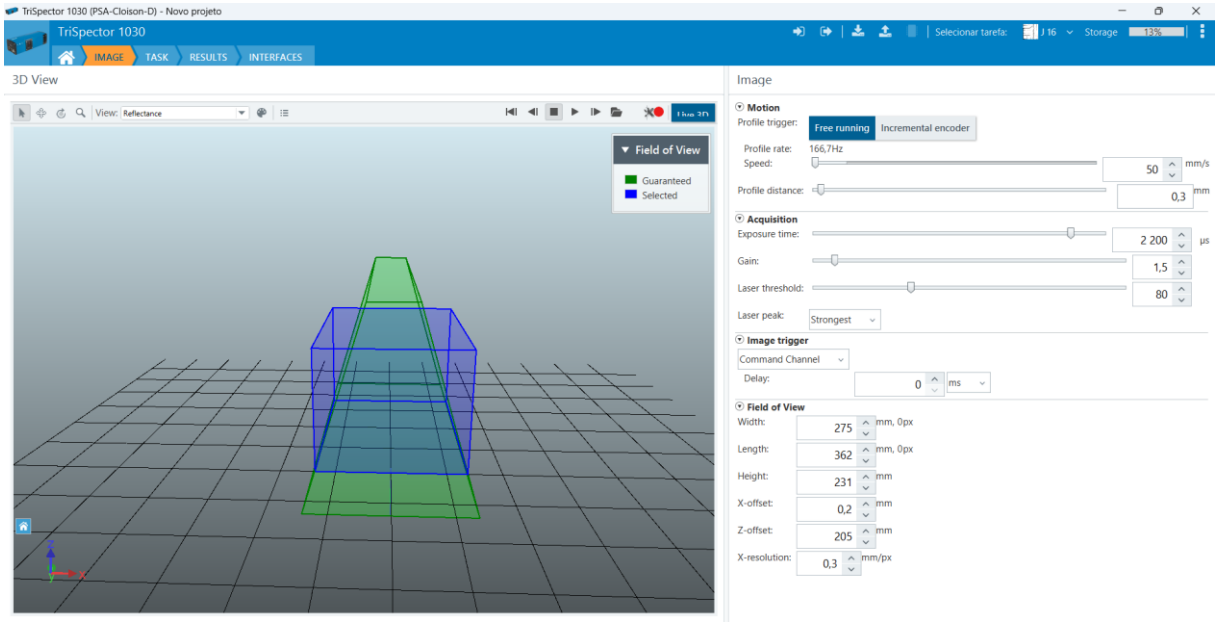


Figura 93 - Página para o tratamento da imagem

Esta página é responsável pelo processamento da imagem. Um dos parâmetros a configurar é o *speed*, o qual deve, idealmente, corresponder à velocidade do robô no momento da aquisição da imagem. O parâmetro *profile distance* está diretamente relacionado com o *speed* e com o *x-resolution*; ou seja, ao aumentar ou diminuir o valor do *speed*, o *profile distance* deverá ser ajustado de forma proporcional. Por sua vez, quanto mais próximo o valor do *x-resolution* estiver do *profile distance*, maior será a qualidade da imagem adquirida.

Outros parâmetros fundamentais incluem o *exposure time*, o *gain* e o *laser threshold*, que dizem respeito à iluminação da imagem. Estes devem ser ajustados consoante as condições de luminosidade, de modo a garantir uma imagem com contraste e nitidez adequados. Por fim, o *field of view* define a área que será captada pela câmara para a aquisição da imagem.

Após a realização de vários testes e ajustes, foi possível obter uma imagem de qualidade adequada (Figura 94).

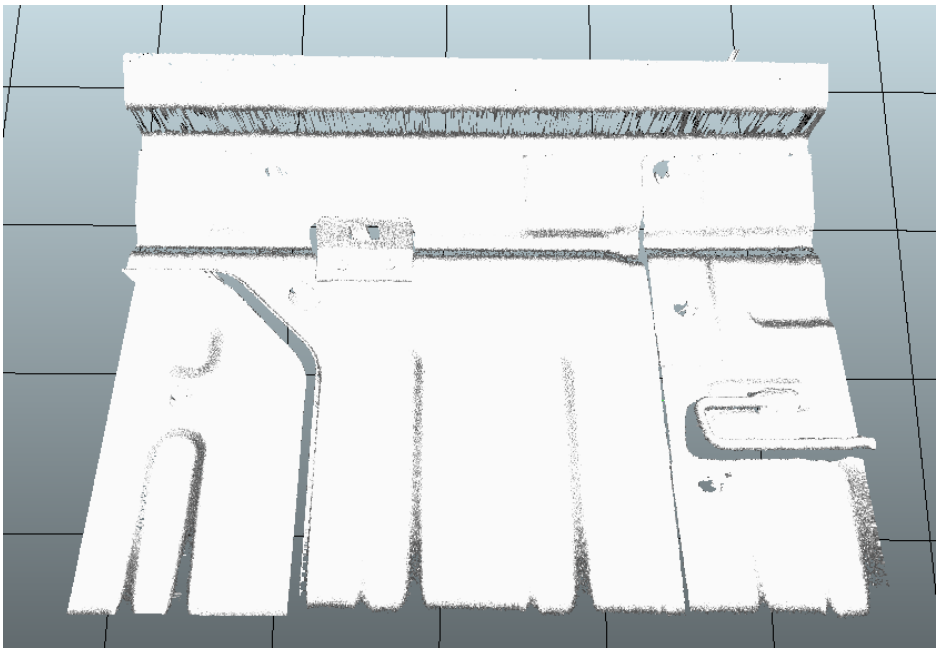


Figura 94 - Imagem guardada para o seu tratamento

De seguida, procedeu-se à elaboração do programa destinado à aquisição das informações pretendidas a partir da câmara (Figura 95). Relativamente aos dados que se pretendem obter, incluem-se a verificação da validade da imagem adquirida, isto é, se esta apresenta qualidade suficiente para ser processada, e a identificação das posições dos parafusos onde serão apertadas as porcas. Para tal, é necessário desenvolver um programa no qual se definem os critérios e funcionalidades desejados, de acordo com os objetivos da aplicação.

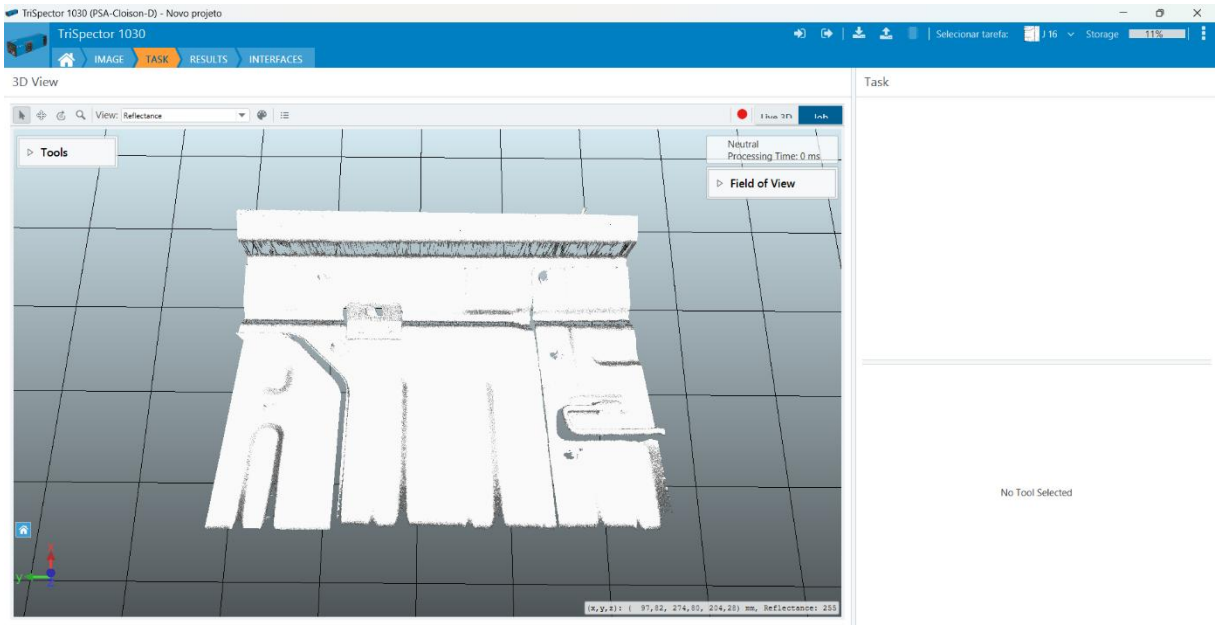


Figura 95 - Página Task para aquisição de informações da imagem

Como se pode observar na Figura 96, o software disponibiliza três opções principais para a criação do programa: Find, Inspect e Measure. A funcionalidade Find permite a seleção de planos, formas, *blobs*, entre outras opções, proporcionando a capacidade de identificar planos em que os objetos se encontram, bem como para adquirir as coordenadas dos pontos de interesse.

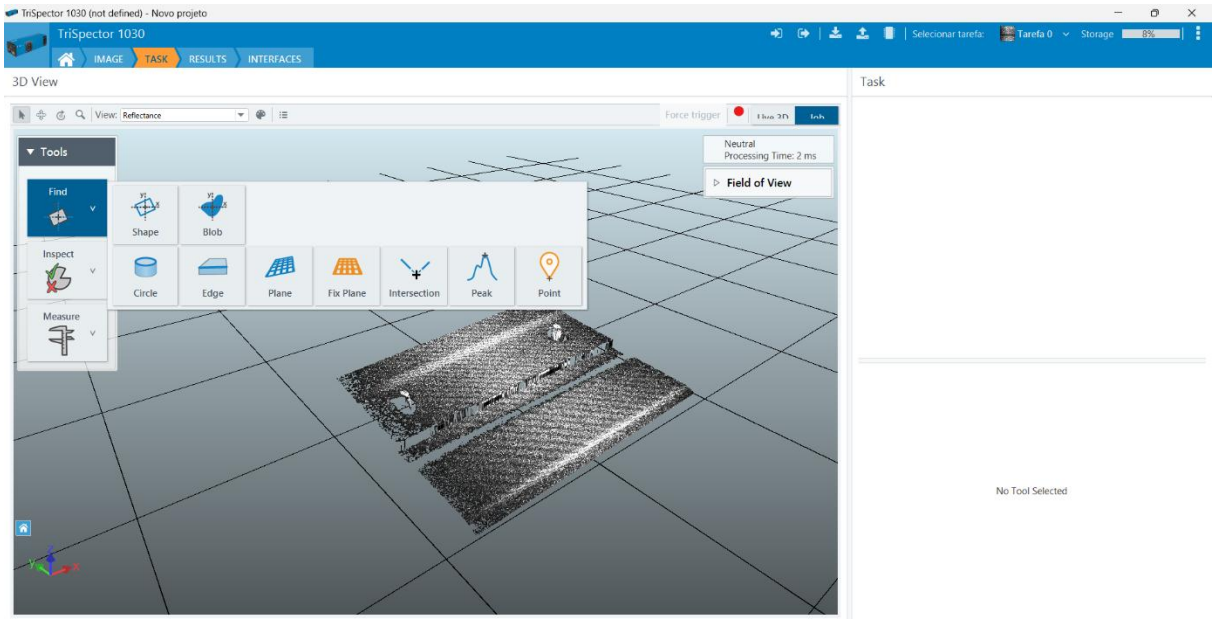


Figura 96 - Opções do *Find*

A funcionalidade *inspect* está relacionada com a análise da área adquirida da imagem (Figura 97). Nesta secção, é possível definir a percentagem da área que se pretende inspecionar, sendo que, com base nesse parâmetro, o sistema atribui um resultado OK ou NOK, consoante a conformidade da área adquirida face ao valor previamente estipulado.

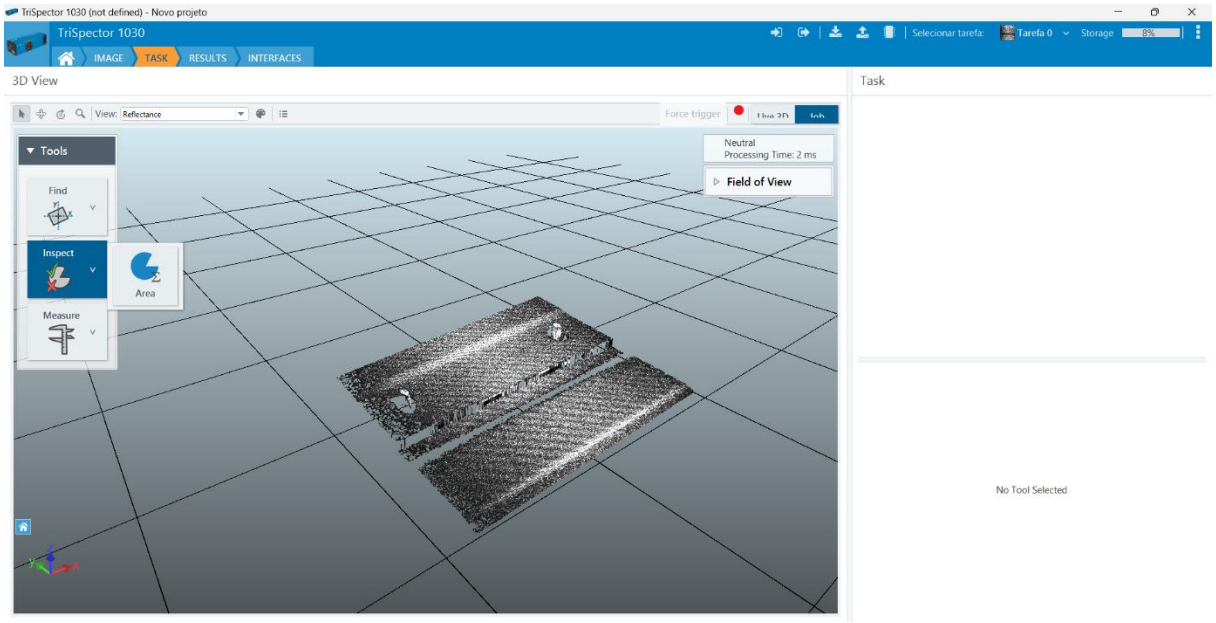


Figura 97 - Área adquirida da imagem

Por fim, a funcionalidade Measure permite obter medições de distâncias entre dois pontos, bem como calcular ângulos entre elementos selecionados, conforme ilustrado na Figura 98.

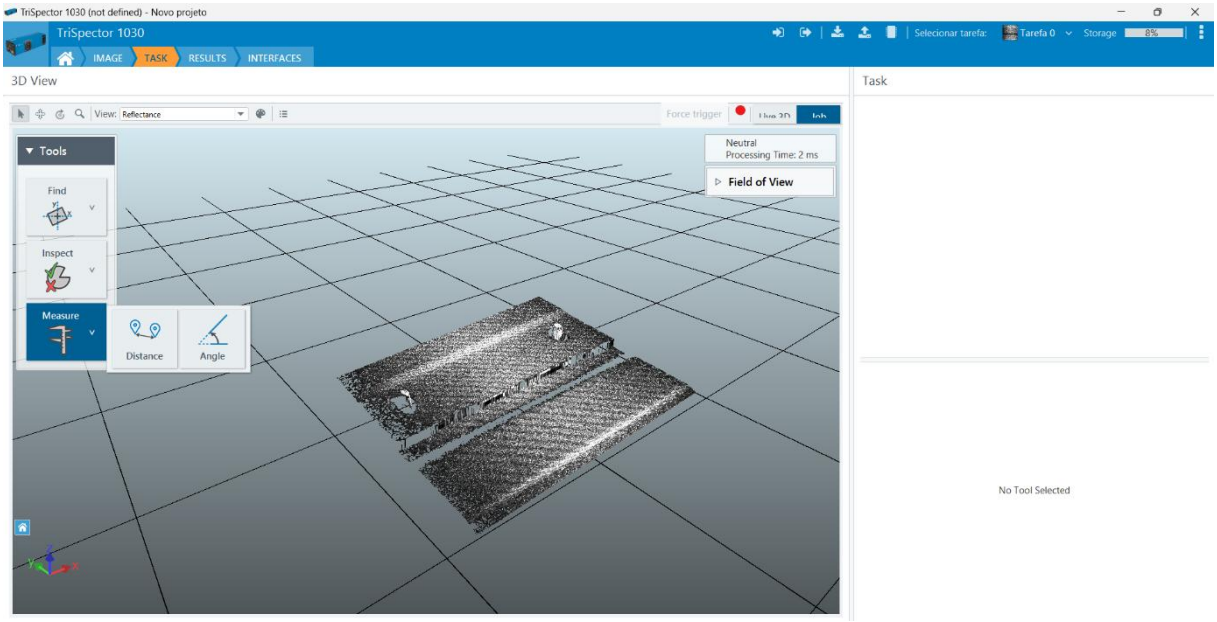


Figura 98 - Opções do *Measure*

O programa foi elaborado com base nas informações pretendidas da imagem adquirida, nomeadamente a área da chapa a ser inspecionada e as posições dos três parafusos visíveis na Figura 99.

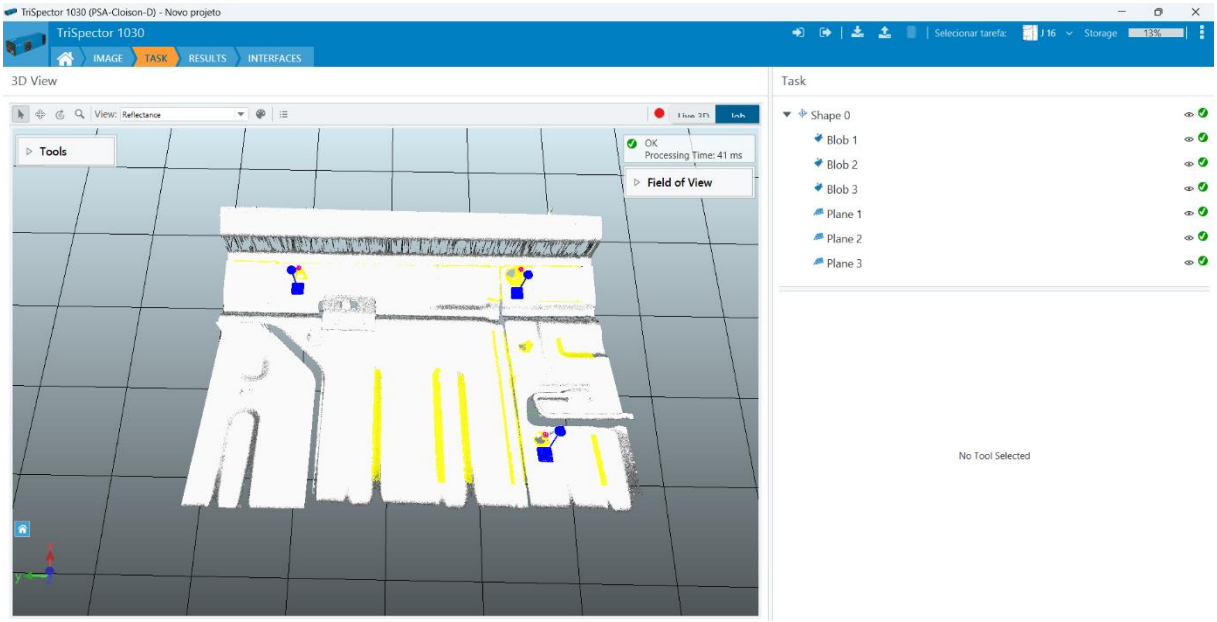


Figura 99 - Programa elaborado para aquisição da imagem

Para a aquisição dessas informações, foi utilizada a ferramenta *shape*. Nesta ferramenta, define-se a região de interesse da chapa, especificando as dimensões necessárias, assim como algumas exceções devido à presença de placas distintas. Este procedimento permite verificar se a chapa está corretamente posicionada e orientada, em termos de rotação, relativamente ao manipulador.

É através desta ferramenta que o programa avalia a validade da chapa para a realização do aparafusamento, conforme ilustrado na Figura 100.

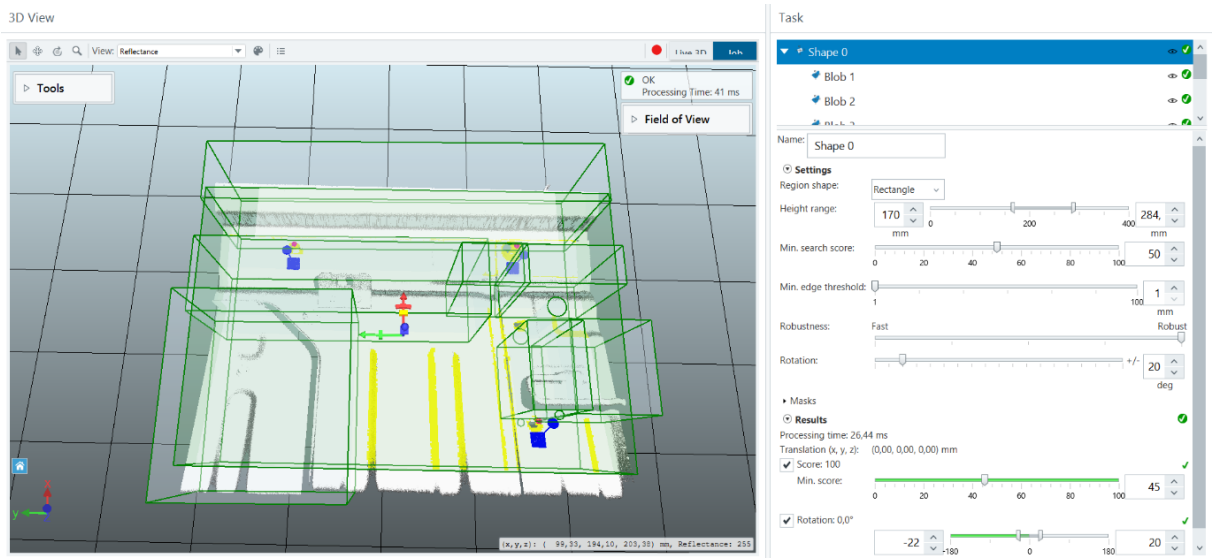


Figura 100 - Tool Shape

De seguida, utilizou-se a ferramenta *blob*, inserida no interior do *shape*, dado que a validade da chapa constitui uma condição prévia.

Esta ferramenta permite a deteção de agrupamentos de pontos dentro de intervalos previamente definidos de tamanho e altura, calculando o tamanho de cada agrupamento. Cada agrupamento corresponde à posição de um parafuso, sendo assim possível determinar as suas coordenadas (X e Y) relativamente ao eixo da câmara, bem como a sua área (Figura 101).

Para este propósito, foi necessário recorrer a três funções *blob* distintas, uma vez que se pretende obter a posição de três parafusos diferentes.

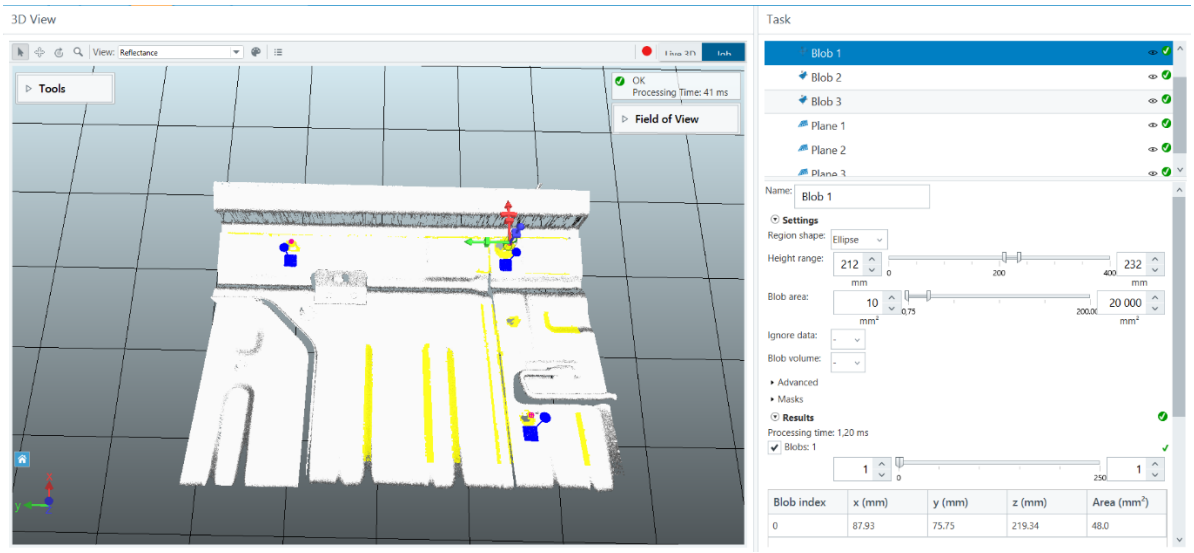


Figura 101 - Tool Blob

Para a aquisição da coordenada Z, dado que os pontos de aperto se encontram em alturas distintas e a ferramenta *blob* apenas fornece as coordenadas X e Y, recorreu-se à ferramenta *plane*. Esta permite definir um plano que fornece a coordenada Z correspondente (Figura 102), completando assim o conjunto de coordenadas necessárias para determinar as posições de cada parafuso (X, Y e Z).

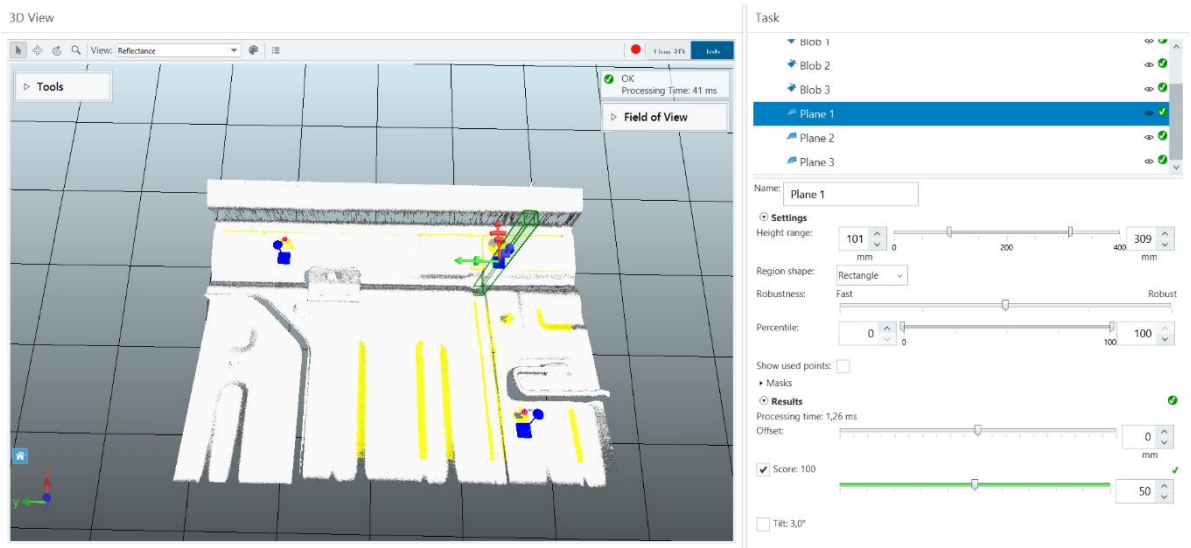


Figura 102 – Tool Plane

A página *result* é responsável pela seleção das informações específicas a serem obtidas, com base nas configurações e no processamento efetuados previamente na página *task*. Neste ambiente, definem-se as condições e critérios para a análise e apresentação dos dados extraídos das imagens. Como ilustrado na figura abaixo, foi incluída a condição relativa à validade da imagem adquirida (OK ou NOK) e foram recolhidas as coordenadas X e Y através das ferramentas *blob*, bem como a coordenada Z obtida pelos planos.

Figura 103 - Página Result

Por fim, a página *interfaces* destina-se à configuração dos parâmetros de comunicação e ao envio dos resultados. Nesta secção, é possível definir os endereços IP e as portas através das quais os dados serão transmitidos, nomeadamente via *command channel* e *output string*, entre outras opções disponíveis (Figura 104).

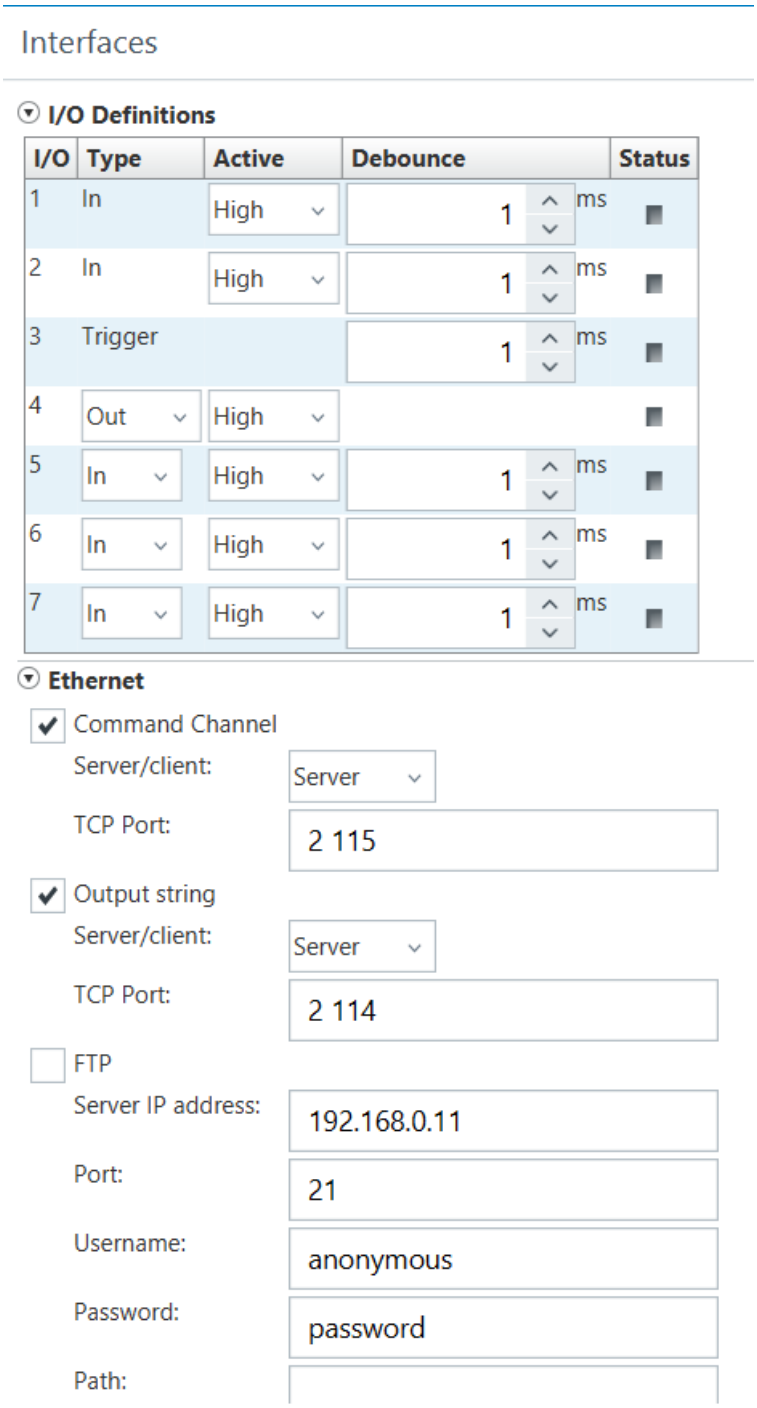


Figura 104 - Página Interfaces

Neste momento, encontravam-se concluídas todas as configurações e programações necessárias relativas ao sistema de visão.

Sistema de aparafusamento

No que diz respeito à programação das duas aparafusadoras, esta foi desenvolvida pela Stellantis, ficando sob a sua responsabilidade, sem qualquer intervenção da nossa parte. A informação que nos foi fornecida consistiu unicamente nos sinais de Output e nos respetivos valores a utilizar para enviar, através do PLC, o comando que inicia o programa das aparafusadoras e, consequentemente, o processo de aperto. Também nos foi indicado o Input responsável por receber o relatório do aperto, indicando se o processo foi realizado corretamente (relatório OK) ou incorretamente (relatório NOK).

Tendo conhecimento dos Inputs e Outputs a utilizar, procedeu-se então à programação dos robôs.

Programação do robô

Concluída a programação da câmara e da aparafusadora, avançou-se para a fase final da programação dos periféricos: a programação do robô, que constitui o código principal deste projeto, uma vez que é o robô que coordenará toda a execução da tarefa. Será este a definir o momento de atuação de cada periférico, de acordo com as necessidades detetadas durante o processo.

Para garantir a comunicação entre os periféricos, recorreu-se à utilização de um PLC, responsável por gerir todos os sinais trocados entre os três componentes principais (robô, aparafusadora e câmara), bem como os valores transmitidos através dos respetivos inputs e outputs.

Foram desenvolvidos dois programas distintos, dado que o projeto recorre à utilização de dois *cobots*, sendo que a lógica de funcionamento de ambos os códigos é semelhante, variando apenas as posições de aperto e a aquisição da imagem da chapa pela câmara 3D. Os respetivos códigos encontram-se disponíveis no capítulo dos Anexos, devidamente comentados para facilitar a sua compreensão (Anexo 1 – Código Cloison do robô do lado direito e Anexo 2 – Código Cloison do robô do lado esquerdo).

De seguida, será apresentado um fluxograma (Figura 105) que ilustra o funcionamento geral do código, com o objetivo de facilitar a sua interpretação. Posteriormente, será feita uma análise mais detalhada a algumas secções do código relativas ao robô posicionado no lado direito da carrinha (Anexo 1 – Código Cloison do robô do lado direito), destacando os aspetos considerados mais relevantes para a sua compreensão.

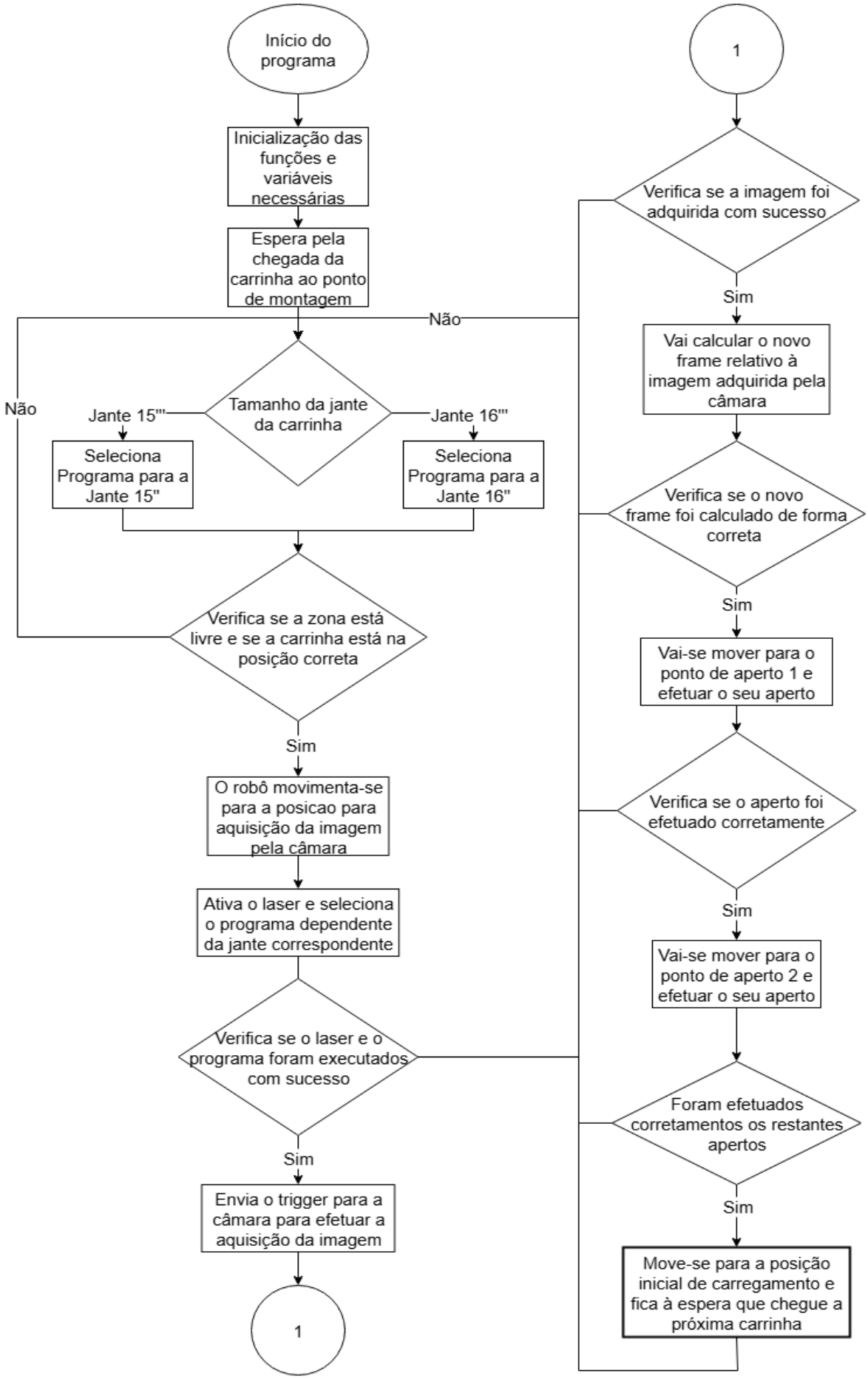


Figura 105 - Fluxograma relativo ao código do robô

É possível identificar as diversas etapas que compõem o código desenvolvido. O penúltimo bloco do fluxograma corresponde a uma condição que determina o funcionamento do restante código. Esta abordagem visa evitar repetições desnecessárias no fluxograma, uma vez que, a partir desse ponto, o robô executará ciclicamente o mesmo conjunto de operações até ao final do processo. Em concreto, o sistema realiza o aparafusamento e, de seguida, verifica se este foi executado corretamente. Caso o processo decorra sem erros, o robô prossegue para o parafuso seguinte, repetindo esta sequência até completar os sete apertos previstos. No entanto, se o sistema detetar que o aperto não foi corretamente realizado, o código é interrompido e o robô retorna à sua posição inicial, garantindo a segurança e a integridade da operação.

Avançando agora para uma análise mais detalhada de algumas secções específicas do código, o primeiro passo consistiu na definição das funções e variáveis essenciais à execução do programa.

Conforme ilustrado na Figura 106 a), foi criada uma função responsável por inicializar e estabelecer a ligação com a câmara, de forma a possibilitar a comunicação entre esta e o robô. Nesta função, são definidos o endereço IP da câmara e as respetivas portas de entrada e saída (input/output). Importa referir que esta função tem como única finalidade a inicialização e conexão inicial da câmara

Na Figura 106 b), é apresentada a função encarregue de escrever no *socket*, ou seja, será através desta que são enviadas para a câmara as instruções de operação pretendidas, como, por exemplo, ativar o laser ou iniciar a aquisição da imagem.

```python
#Função que inicializa e faz a conexão da câmara
def __init__(self, ip="192.168.1.61", portIN=2115, portOUT=2114):

    #Guarda os parâmetros
    self.ip = ip
    self.portIN = portIN
    self.portOUT = portOUT

    #Cria um socket TCP básico e tenta conectar-se à câmara para testar disponibilidade
    client = socket.socket()
    client.settimeout(3) #timeout de 3 segundos para nao parar o programa
    socket.setdefaulttimeout(3)
    client.connect((self.ip, self.portIN))

    #Usa a função oficial da Doosan para abrir um canal socket de comunicação de entrada
    #(Receber dados da câmara)
    try:
        self._socketIN = client_socket_open(self.ip, self.portIN)
    except Exception as e: #Caso falhe irá alertar
        tp_popup("Socket INPUT falha de ligação. Error: {0}".format(
            str(e)), DR_PM_ALARM)
        raise e

    time.sleep(0.1)

    #Vai abrir outro canal socket de comunicação mas neste caso de saída
    #(enviar comandos para a câmara)
    try:
        self._socketOUT = client_socket_open(self.ip, self.portOUT)
    except Exception as e: #Caso falhe irá alertar
        tp_popup("Socket OUTPUT falha de ligação. Error: {0}".format(
            str(e)), DR_PM_ALARM)
        raise e
```

a)

```python
#Função usada para escrever no socket (Comunicação Robô para Câmara)
def write(self, cmd, socket=None):

    #Vai usar o socket padrão caso o mesmo não seja especificado
    if socket == None:
        socket = self._socketIN

    #Converte o comando de string para bytes em ASCII, necessário para transmissão binária via socket
    cmd = bytes(cmd, encoding="ascii")

    #A informação é enviada da seguinte forma (padrão deste dispositivo) "STX - res - ETX"
    STX = client_socket_write(socket, b"\x02") #STX (start of text)
    res = client_socket_write(socket, cmd)     #Envia o comando pretendido
    ETX = client_socket_write(socket, b"\x03") ##ETX (End of Text)
    time.sleep(0.1)
    res, rx_data = client_socket_read(socket, 10, 1) #Vai ler até 10 bytes com um timeout de 1 seg
    #Caso haja uma resposta válida, vai converter os dados recebidos de bytes para string
    rxd=None
    if res > 1:
        rx_msg = rx_data.decode() #Converte os dados
        rxd =(rx_msg[1:3]) # Extrai 2 caracteres do meio da resposta
    #Vai registar logs de possiveis erros
    if res == -1:
        tp_log("Erro durante a escrita no socket : Server não conectado")
    elif res == -2:
        tp_log("Erro durante a escrita no socket: Erro de Socket")
    return rxd
```

b)

Figura 106 - Função de inicialização e de escrita da câmara, respetivamente

Seguidamente, foram definidas as funções necessárias para a leitura da informação enviada pela câmara ao robô, através do *command channel* e do *data channel* (Figura 107). A primeira função implementada tem como objetivo ler as respostas relativas aos comandos transmitidos para a câmara, sendo utilizada para detetar eventuais erros no envio desses comandos via *command channel*.

Por sua vez, a função "*readata*" é responsável por ler e recolher a informação transmitida pela câmara após a aquisição e processamento da imagem. Será esta função que irá transmitir para o *cobot* os dados obtidos pela câmara, tais como a confirmação da correta aquisição da imagem e as coordenadas dos pontos de aperto, permitindo assim a correção e ajuste dos movimentos do robô de acordo com a posição real dos parafusos.

```python
#Função usada para ler a informação do command channel da câmara (Mensagens de verificação)
def readcmd(self, length, timeout, socket=None ):

    #Vai usar o socket padrão caso o mesmo não seja especificado
    if socket == None:
        socket = self._socketIN

    #Lê os dados do socket
    res, rx_data = client_socket_read(socket, length, timeout)
    rx_msg = rx_data.decode() #Converte de bytes para string
    rxd =(rx_msg[1:3])        #Extrai a informação da posicao 1 e 2 (Código do estado)

    #Vai registar logs de possiveis erros
    if res == -1:
        tp_log("Erro durante a leitura do socket : Server não conectado")
    elif res == -2:
        tp_log("Erro durante a leitura do socket: Erro de Socket")
    elif res == -3:
        tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
    return rxd
```

```python
#Função usada para ler do socket a informação enviada pela câmara (posições dos parafusos)
def readata(self, length, timeout, socket=None ):

    #Vai usar o socket padrão caso o mesmo não seja especificado
    if socket == None:
        socket = self._socketOUT

    #Inicializa variáveis e lê os dados do socket
    res = 0
    rx_data = 0
    res, rx_data = client_socket_read(socket, length, timeout)

    #Vai registar logs de possiveis erros
    if res == -1:
        tp_log("Erro durante a leitura do socket : Server não conectado")
    elif res == -2:
        tp_log("Erro durante a leitura do socket: Erro de Socket")
    elif res == -3:
        tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
    else:
        rx_msg = rx_data.decode() #Converte os dados de bytes para string
        return rx_msg #Retorna a informação enviada pela câmara
```

Figura 107 - Funções utilizadas para ler o command channel e o data channel

Para finalizar a fase de definição das funções e variáveis, antes de avançar para o código principal, foi implementada mais uma função de particular importância, desta vez associada ao sistema de aparafusamento, bem como uma variável global e alguns outputs essenciais (Figura 108).

Relativamente à função, esta será responsável por aguardar pelo relatório de aperto emitido pela aparafusadora, enquanto esta executa o processo de aperto da porca. Assim que a operação de aparafusamento é concluída, a aparafusadora envia para o robô um relatório que indica se o aperto foi efetuado corretamente ou incorretamente. Esta função tem como objetivo receber essa informação e desativar a aparafusadora, interrompendo o seu funcionamento. Posteriormente, esse relatório será analisado e o sistema atuará em conformidade com o resultado obtido.

Adicionalmente, foi definida a variável global "*line*", que será responsável por comandar o fluxo de execução do código, funcionando como uma espécie de guia que determina a sequência de operações a seguir durante todo o programa.

Por fim, foram configurados alguns outputs relevantes, fundamentais para a comunicação com o PLC e para o controlo do funcionamento geral do robô. Entre eles, destaca-se o output responsável pela ativação e desativação da aparafusadora "set_output_register_int(2, 0) para OFF ou 1 para ON", e ainda a posição de carregamento, que, neste projeto, funcionará como a posição inicial ou "*Home Position*" do *cobot*, sendo o local onde este realiza o carregamento de porcas para posterior utilização durante o processo de montagem.

```
#Função que irá esperar que o torque pretendido seja atingido para desativar a aparafusadora
def wait_torch() :
                yy = True
                zz = 0
                while yy:
                    step_ack=get_input_register_int(1) #Guarda na variavel step_ack o valor do relatório
                    if(step_ack==1000) or (step_ack==2000): #Só entra neste if quando recebe o relatório, seja OK ou NOK
                        set_output_register_int(2, 0) #Output para desativar aparafusadora
                        time.sleep(0.1)
                        yy = False

                    #TimeOut de 35 segundos para caso não receber resposta da aparafusadora, o loop termina e a ferramenta é desligada por segurança.
                    if zz >= 35:
                        yy = False
                    time.sleep(0.1)
                    zz = zz + 0.1

#Definição de uma variável global que será usada posteriormente para indicar para onde irá a execução do programa
def goto(linenum):
    global line
    line = linenum
line = 1

#Dá set a 0 os seguintes outputs
set_output_register_int(0, 0) #Info para PLC etapa do Robô
set_output_register_int(1, 0) #Info para PLC resultado Visão
set_output_register_int(2, 0) #Pedido aperto
set_output_register_int(3, 0) #Info para PLC decisão recebida
set_output_register_int(4, 0) #Info para PLC zona ocupada

#Posição de carregamento de porcas
carregamento = posj(39.8, -15.16, -101.75, -72.25, 53.21, -121.47)
```

Figura 108 - Definição de funções, variáveis e *outputs*

Uma vez definidas todas as funções e variáveis necessárias para a implementação do código, avançou-se para o desenvolvimento do código corrente, que estará em execução contínua na linha de montagem. Tendo em consideração que, neste ponto do processo, podem chegar carrinhas equipadas com jantes de 15 ou de 16 polegadas, foi necessário desenvolver dois blocos de código distintos, um para cada tipologia de veículo. Isto porque uma carrinha com jantes de 16 polegadas apresenta uma altura superior relativamente à versão com jantes de 15 polegadas, o que influencia diretamente a posição dos pontos de aperto.

Dado que o PLC é responsável por enviar um sinal específico para identificar qual o tipo de carrinha que chegou ao ponto de montagem, a primeira etapa do programa consiste na leitura e interpretação desse sinal. Com base nessa informação, procede-se à conexão com a câmara e à seleção do programa adequado a ser utilizado. Esta distinção justifica-se pelo facto de terem sido desenvolvidos dois programas diferentes para a câmara, adaptados às características de cada tipo de carrinha, bem como dois fluxos distintos dentro do próprio código do robô.

É neste momento que se determina qual a secção do código que o robô deverá executar, de acordo com a informação recebida. Na Figura 109 é possível observar as três condições previstas, onde se evidência que tanto o programa da câmara (*job*) como a secção de código correspondente (*goto*) diferem consoante o tipo de carrinha detetada. É desta forma que o robô identifica corretamente qual o veículo presente na estação de trabalho.

O programa permanecerá em modo de espera (*loop*) nesta fase sempre que algum periférico se encontre fora de serviço e até que uma nova carrinha chegue ao ponto de montagem, conforme ilustrado na última parte do excerto de código apresentado.

```
#Uma vez que existem duas carrinhas diferentes, uma com jante 15 e outra com jante 16 isso irá interferir nas posição das posições de aperto,
#logo aqui é feita a validação de qual das carrinhas chegou à linha e o programa irá correr de acordo com a carrinha respetiva
#get_input_register_int(0) == 10 = Jante 15
if line == 1 and get_input_register_int(0) == 10 and get_digital_input(16)==1 : #Jante 15
    set_output_register_int(0, 0) #Info para PLC etapa do Robô
    set_output_register_int(1, 0) #Info para PLC resultado Visão
    set_output_register_int(2, 0) #Pedido aperto
    set_output_register_int(3, 0) #Info para PLC que a decisão foi recebida (ACK)
    set_output_register_int(4, 0) #Info para PLC zona ocupada

    #Define o IP da câmara e estabelece comunicação com a mesma
    camera_ip = "192.168.1.61"
    trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114)
    time.sleep(0.5)
    job = trispector.write("set job \"J 15\"") #Envia comando para a câmara para seleciona o programa relativo à jante 15
    set_output_register_int(0, 100) #Define a etapa do robô para 100 (Programa 15 selecionado)

    goto(1002) #Faz com que o robô avance para a line 1002, é com base neste função que correrá o código
    continue

if line == 1 and get_input_register_int(0) == 11 and get_digital_input(16)==1 : #get_input_register_int(0) == 11 = Jante 16
    set_output_register_int(0, 0) #Info para PLC etapa do Robô
    set_output_register_int(1, 0) #Info para PLC resultado Visão
    set_output_register_int(2, 0) #Pedido aperto
    set_output_register_int(3, 0) #Info para PLC decisão recebida (ACK)
    set_output_register_int(4, 0) #Info para PLC zona ocupada

    #Define o IP da câmara e estabelece comunicação com a mesma
    camera_ip = "192.168.1.61"
    trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114) #Faz a conecção com a câmara
    time.sleep(0.5)
    job = trispector.write("set job \"J 16\"") #Envia comando para a câmara para seleciona o programa relativo à jante 16
    set_output_register_int(0, 101) #Define a etapa do robô para 100 (Programa 16 selecionado)

    goto(1102) #Faz com que o robô avance para a line 1102
    continue

#Vai verificar se o sistema está fora de serviço (algum periférico não está pronto, ou carrinha não chegou ao local)
if line == 1 and get_input_register_int(0) == 99 and get_digital_input(16)==1 :
    set_output_register_int(0, 0) #Info para PLC etapa do Robô
    set_output_register_int(1, 0) #Info para PLC resultado Visão
    time.sleep(0.5)
    set_output_register_int(0, 999) #Indica que está fora de serviço
    goto(99)
    continue

if line == 99:
    set_output_register_int(0, 999)
    #Irá verificar se ainda permanece fora de serviço, se sim fica aqui até deixar de estar
    outserv=get_input_register_int(0)
    if(outserv==0): #Se não tiver fora de serviço volta às condições de cima
        time.sleep(0.2)
        set_output_register_int(0, 400)
        time.sleep(1)
        set_output_register_int(0, 0)
        goto(1)
        continue
    time.sleep(0.1)
    goto(99)
    continue
```

Figura 109 - Identificação da carrinha que chegou ao ponto de montagem

Após a identificação do tipo de carrinha, procede-se à aquisição da imagem por parte da câmara. Para isso, o primeiro passo consiste em mover o robô para a posição adequada de captura. Esta movimentação é realizada através da função “movj”, responsável pelo movimento do robô, em conjunto com a função “posj”, que define as coordenadas de destino.

Conforme ilustrado nas imagens seguintes, as coordenadas de movimento incluem os componentes de posição X, Y, Z e as rotações Rx, Ry, Rz que definem a orientação do movimento. A estas coordenadas associam-se também parâmetros de velocidade “v”, aceleração “a” e raio de curvatura “r”, os quais determinam a dinâmica do movimento.

Desta forma, garante-se que o *cobot* se posiciona corretamente para permitir uma aquisição de imagem eficaz por parte do sistema de visão 3D.

```
#Função usada para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a velocidade "v", a aceleracao "a" e o radius "r"
movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=200, a=120, r=50)
set_output_register_int(0, 105)
movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=200, a=120, r=50)
movej(posj(93.1, 37.24, -131.84, -4.77, 97.09, -94.91), v=200, a=100, r=10)
goto(1003)
continue

if line == 1044: #Se a carrinha estiver na posição correta o programa prossegue para a aquisição da imagem pelo sistema de visão

    #Entrada no carro
    movej(posj(163.68, 29.35, -129.86, -12.54, 103.8, -69.28), v=220, a=180, r=50)
    movej(posj(184.55, -13.96, -105.01, -17.47, 116.57, -95.72), v=220, a=180, r=100)

    #Posição para se dar ao inicio da aquisição da imagem pelo sistema de visão
    movej(posj(189.46, -53.72, -35.29, 0, 89.01, -9.46), v=220, a=120, r=0)
    mwait(0)
    set_tcp("U_Tool_1")
    movel(posx(1050.00, 140.0, 650.0, 90, 0, 90), v=120, a=150)
    time.sleep(0.05)
    goto(10444)
    continue
```

Figura 110 - Movimentar o *cobot*

Estando o robô posicionado na coordenada definida para a aquisição da imagem, procede-se ao envio dos comandos necessários à câmara. Nesta fase, é enviado um sinal para ativar o laser, seguido da seleção do programa correspondente (job), de forma a garantir que está carregado o programa correto para a carrinha identificada. De seguida, é emitido o sinal de "*trigger*", que dará início à aquisição da imagem por parte do sistema de visão.

Tanto a ativação do laser como a seleção do programa são operações sujeitas a verificação. Caso alguma destas etapas não seja executada corretamente, o robô interromperá a sequência e regressará à posição inicial, reiniciando o programa de forma controlada.

Um dos aspetos cruciais deste projeto é o controlo rigoroso de erros. O código foi desenvolvido com mecanismos de verificação contínua, permitindo que o sistema esteja permanentemente preparado para detetar e reagir a eventuais falhas, garantindo, assim, a robustez e fiabilidade do processo. Todos estes passos são ilustrados na figura abaixo.

```
#Vai tentar ativar o laser e selecionar o job pretendido, caso falhe em algumas das etapas o robô feha os sockets, informa o PLC e volta à posicao inicial
if line == 10444:
    set_output_register_int(3, 0) #Info para PLC decisão recebida (reset)
    laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da imagem
    #Verifica se o laser foi ativado, caso não tenha sido o robô abortará a tarefa de aquisição da imagem e voltará para a posição inicial
    if (laser_on != 'OK'):
        trispector.closeIN(None) #Fecha os sockets
        trispector.closeOUT(None)
        set_output_register_int(1, 11) #Info para PLC sobre erro ocorrido no sistema de visão
        #Vai se mover para a posição de carregamento (posição incial)
        movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
        movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
        movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
        movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
        movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
        movej(carregamento, v=20, a=10)
        set_tcp("U_Tool_2")
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
        goto(1)
        continue

    job = trispector.write("set job \"J 15\"") #Seleciona o programa "J 15" para ser utilizado na câmara
    #Verifica se o programa foi selecionado corretamente caso contrário aborta e voltará ao inicio
    if (job != 'OK'):
        trispector.closeIN(None) #Fecha os sockets
        trispector.closeOUT(None)
        set_output_register_int(1, 11) #Info para PLC sobre erro ocorrido no sistema de visão
        #Vai se mover para a posição de carregamento (posição incial)
        movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
        movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
        movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
        movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
        movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
        movej(carregamento, v=20, a=10)
        set_tcp("U_Tool_2")
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
        goto(1)
        continue

    #Caso o laser a seleção do programa tenham sido executados com sucesso irá passar-se à aquisição da imagem
    time.sleep(0.3)
    amovel(posx(688.00, 140.0, 650.0, 90.00, 00.00, 90.00), v=50, a=200) #Posição de inicio de aquisicao da imagem
    trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio à aquisição da imagem
    mwait(0) #Aguarda o fim completo do movimento (amovel) antes de prosseguir
```

Figura 111 - Envio de comandos para a câmara

Após a aquisição da imagem, a câmara envia ao robô os dados obtidos, nomeadamente a validação da imagem capturada e as coordenadas dos três pontos identificados. Esta informação é recebida e processada pela função "*readata*", sendo armazenada numa variável para posterior utilização. Caso a imagem seja considerada válida, o programa prossegue com a execução; caso contrário, o robô regressa à posição inicial e o processo é reiniciado.

Com os pontos fornecidos pela câmara e tendo como referência um *frame* pré-definido (que serve de base para o posicionamento relativo dos restantes pontos de aperto), procede-se ao cálculo do desvio da carrinha. Este cálculo baseia-se na diferença entre as coordenadas dos pontos obtidos pela câmara e os pontos correspondentes do *frame* de referência. Através dessa diferença, é possível determinar o deslocamento da carrinha relativamente à posição esperada, como demonstrado na Figura 112.

```
rx_msg = trispector.readata(128, 6) #Lê e guarda a informação enviada pela câmara relativa à imagem adquirida (128 caracteres com timeout de 6 seg)
rxd =(rx_msg[0:128])
data = rx_msg.split(', ') #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui irá dividir-se a informação pelas virgulas criando uma array
valid_locate =(rx_msg[0:1]) #Vai guardar o primeiro caracter que contem a informação se a imagem foi adquirida corretamente

#Valores imagem referência esperada para comparar com a adquirida (valores do frame de referência)
refx1 = 87.656254
refy1 = 72.368235
refz1 = 207.349081
refx2 = 82.712757
refy2 = 285.062823
refz2 = 207.748271
refx3 = -76.414542
refy3 = 65.350911
refz3 = 246.823760

if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

    #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência para saber quanto se moveu a chapa da posição de referência
    x1 =  float(data[1]) - refx1
    y1 =  float(data[2]) - refy1
    z1 =  float(data[3]) - refz1
    x2 =  float(data[4]) - refx2
    y2 =  float(data[5]) - refy2
    z2 =  float(data[6]) - refz2
    x3 =  float(data[7]) - refx3
    y3 =  float(data[8]) - refy3
    z3 =  float(data[9]) - refz3
    #Vai arredondar os valores para 4 casas decimais
    x1 = round(x1, 4)
    y1 = round(y1, 4)
    z1 = round(z1, 4)
    x2 = round(x2, 4)
    y2 = round(y2, 4)
    z2 = round(z2, 4)
    x3 = round(x3, 4)
    y3 = round(y3, 4)
    z3 = round(z3, 4)


    #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência (inverto o eixo X para alinhar os eixos de coordenadas - ferramenta - câmara)
    visp1 = [x1*-1, y1, z1]
    visp2 = [x2*-1, y2, z2]
    visp3 = [x3*-1, y3, z1]

    laser_off = trispector.write("set laser off") #Desliga o laser

if valid_locate!="1": #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
    tp_log("Erro a Localizar!")
    trispector.closeIN(None)
    trispector.closeOUT(None)
    set_output_register_int(1, 33) #Info para PLC imagem nao adquirida corretamente
    #Vai se mover para a posição de carregamento (posição incial)
    movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
    movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
    movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
    movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
    movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
    movej(carregamento, v=20, a=10)
    set_tcp("U_Tool_2")
    set_output_register_int(4, 0) #Info para PLC Zona Livre
    set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
    goto(1)
    continue

set_output_register_int(0, 130) #Avança
goto(1005)
continue
```

Figura 112 - Leitura dos dados fornecidos pela câmara

De seguida, procedeu-se ao cálculo de um novo *frame* de referência com base na posição real da chapa obtida através do sistema de visão (Figura 113). Para esse efeito, os valores de deslocamento previamente calculados foram subtraídos às coordenadas dos pontos de referência originais, permitindo assim a criação de um novo *frame* ajustado à posição atual da chapa. Com este novo referencial definido, passam a estar disponíveis as coordenadas corrigidas de todos os pontos de aperto, possibilitando a execução precisa dos respetivos aparafusamentos.

```
#Define os pontos de referência das três posições de aperto usadas para gerar o frame de referência
PF1 = posx(916.84, 552.43, 782.1, 90.84, -93.16, -0.04)
PF2 = posx(704.89, 552.84, 777.79, 90.84, -93.16, -0.04)
PF3 = posx(922.67, 511.58, 619.51, 90.84, -93.16, -0.04)
Frame1 = posx(916.84, 552.43, 782.1, 89.597, -75.911, 91.201) #Frame Referência (informativo) - +X direção porta, +Y direção travão, +Z direção vidro

#Vai calcular os novos pontos de referência de acordo com os desvios obtidos com o sistema de visão relativos às coordenadas X, Y e Z
P1vis = [PF1[0]-visp1[1], PF1[1]-visp1[2], PF1[2]-visp1[0]]
P2vis = [PF2[0]-visp2[1], PF2[1]-visp2[2], PF2[2]-visp2[0]]
P3vis = [PF3[0]-visp3[1], PF3[1]-visp3[2], PF3[2]-visp3[0]]

#Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem p1 p2 e p3 que são os novos pontos de referência
p1 = P1vis[0:3]+PF1[3:]
p2 = P2vis[0:3]+PF2[3:]
p3 = P3vis[0:3]+PF3[3:]

#Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
pose_user102 = calc_coord(p1, p2, p3, ref=DR_BASE, mod=0)

#Vai substituir o frame existente 102 pelo novo frame calculado de forma permanente
overwrite_user_cart_coord(102, pose_user102, ref=DR_BASE, apply_mod=DR_PERMANENT)

mwait(0)
set_ref_coord(102) #Seleciona o novo frame criado com os valores do sistema de visão, que irá corrigir os novos pontos de aperto das restantes porcas
set_tcp("U_Tool_4")

time.sleep(0.3)
```

Figura 113 - Novo *frame* obtido pelo sistema de visão

Com a posição dos novos pontos devidamente calculada, procedeu-se à execução da operação de aperto das porcas. Para tal, o *cobot* foi inicialmente movimentado para as posições de aproximação e, em seguida, para as posições de aperto correspondentes. Uma vez posicionado, foi enviado um sinal através do *output* 2 para ativar a aparafusadora e iniciar o processo de aperto. Durante essa operação, é chamada a função "*wait_torch*", previamente definida, cuja função consiste em aguardar a receção do relatório de aperto por parte da aparafusadora. Após a receção do relatório, o sistema procede à desativação da aparafusadora e o código avança para a etapa seguinte, conforme ilustrado na figura seguinte.

```
#A partir deste ponto irá passar-se para a fase de aperto das porcas
set_output_register_int(3, 0) #Info para PLC decisão recebida (reset)
#O robô vai se mover para posteriormente começar as posições de aperto
movej(posj(189.49, -18.66, -98.65, -16.87, 113.72, -43.64), v=220, a=250, r=50)
movej(posj(186.95, -22.47, -87.38, 179.29, 70.16, -259.86), v=220, a=250, r=40)

#Pontos de aproximação do primeiro parafuso (respetivos ao novo frame calculado)
movej(posj(185.2, -32.41, -89.96, 181.81, 57.23, -264.52), v=100, a=100, r=40) #Posição aproximação
movel(posx(-18.287, 670.634, 51.632, 92.739, 15.020, 176.622), v=150, a=250, r=20) # Posição perto do parafuso respetivo ao frame novo (Tool 4 em U_Frame_1 (101))
movel(posx(-17.895, 662.461, 21.130, 92.737, 15.020, 176.623), v=50, a=50) # Posição relativa ao toque no parafuso (Tool 4 em U_Frame_1 (101))
set_output_register_int(4, 0) #Info para PLC Zona Livre
set_output_register_int(0, 140) #Info para PLC etapa aperto P1

#Aperto P1
set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
time.sleep(0.2)
wait_torch() #Espera pelo torque pretendido da aparafusadora
goto(1007)
continue
```

Figura 114 - Aperto 1

Após a execução do primeiro aperto, o robô desloca-se para a posição de aproximação relativa ao aperto da segunda porca. Quando chega a esta posição, antes de proceder ao aperto, é realizada uma verificação de eventuais erros associados ao processo de aparafusamento, tais como a confirmação da correta posição do veículo, possíveis deslocações do mesmo, bem como falhas de comunicação com o PLC, onde esta verificação está assinalada pela condição "line == 1008" (Figura 115). Caso o aperto tenha sido efetuado com sucesso, o programa

prossegue para o aperto da segunda porca, repetindo-se este procedimento para as porcas subsequentes até ao aperto da última. Caso seja detetada alguma falha durante o aparafusamento, o robô regressa à sua posição inicial.

```
if line == 1007:#Depois de fazer o aperto da primeira porca, passoa para as posições relativas ao aperto 2

    movel(posx(-18.287, 670.634, 51.632, 92.739, 15.020, 176.622), v=150, a=250, r=20) #Volta à posição perto do parafuso para depois recuar em segurança
    movej(posj(191.63, -27.95, -80.99, 162.81, 71.99, -252.02), v=120, a=120, r=10) #Posição de Aproximação do parafuso 2
    set_output_register_int(0, 150) #Info para PLC etapa robô
    movel(posx(-10.386, 590.062, 40.613, 67.625, -2.053, -155.496), v=50, a=50)  # Posição perto do parafuso respetivo ao frame novo (Tool 4 em U_Frame_1 (101))
    goto(1008)
    continue

if line == 1008: #Verificação de possiveis erros
        step_ack=get_input_register_int(2)
        if(step_ack==5555): #Caso não haja erros
            set_output_register_int(3, 777) #Info para PLC ACK
            #time.sleep(0.1)
            goto(1009)
            continue

        error=get_input_register_int(2)
        if(error==9999): #Caso haja erros
            set_output_register_int(3, 777) #Info para PLC ACK
            #Vai se mover para a posição de carregamento (posição incial)
            movej(posj(190.3, -19.14, -88.61, 81.84, 78.11, -160.8), v=100, a=100, r=50)
            movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
            movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
            movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_output_register_int(3, 0) #Dá reset no ACK do PLC
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
            time.sleep(0.1)
            goto(1)
            continue
```

```
if line == 1009:  #Aperto P2
    set_output_register_int(3, 0) #Dá reset no ACK do PLC
    movel(posx(-9.913, 590.705, 6.316, 67.622, -2.053, -155.494), v=50, a=50)# Posição relativa ao toque no parafuso (Tool 4 em U_Frame_1 (101))
    set_output_register_int(0, 160)  #Info para PLC etapa aperto P2
    time.sleep(0.1)
    set_output_register_int(2, 55) #Sinal para a aparafusadora para iniciar o aperto
    time.sleep(0.2)
    wait_torch() #Espera pelo torque pretendido da aparafusadora
    movel(posx(-8.386, 590.062, 40.613, 67.625, -2.053, -155.496), v=50, a=50) #Volta à posição perto do parafuso para depois recuar em segurança
    goto(1010)
    continue
```

Figura 115 - Aperto 2

Após a conclusão de todos os apertos, o *cobot* é deslocado para a posição inicial, onde procede ao carregamento das porcas, preparando-se para executar o aperto da carrinha seguinte que chegará ao posto. O sistema mantém-se neste ciclo contínuo durante todo o período de produção (Figura 116).

```
if line == 1027:
    #Saida para a posição inicial depois de efetuar todos os apertos
    mwait(0)
    set_ref_coord(DR_BASE) #Define o sistema de referencia de coordenadas
    set_output_register_int(0, 270) #Info para PLC etapa robô (Retorno à posição inicial)
    #Vai se mover para a posições intermédias de retorno
    movej(posj(178.5, -11.38, -95.73, -1.56, 109.1, -118.76), v=150, a=150, r=50)
    movej(posj(170.2, 27.74, -128.61, -1.43, 102.92, -116.27), v=150, a=150, r=0)
    set_output_register_int(4, 0) #Info para PLC Zona Livre
    movej(posj(115.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=120, a=150, r=50)
    movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=120, a=150, r=50)
    movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=100, a=100, r=50)
    movej(carregamento, v=20, a=10) #(posição incial)

    set_tcp("U_Tool_2")
    set_ref_coord(DR_BASE)
    set_output_register_int(3, 0) #Dá reset no ACK do PLC
    set_tcp("U_Tool_2")
    set_output_register_int(4, 0) #Info para PLC Zona Livre
    set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
    goto(1)
    continue
```

Figura 116 – Retorno à posição inicial

Relativamente ao segundo *cobot* implementado na linha, o seu código apresenta-se bastante semelhante ao referido anteriormente, alterando-se apenas as coordenadas dos pontos de aperto, bem como os pontos de referência. Contudo, o método utilizado mantém-se idêntico ao do robô mencionado anteriormente. O código correspondente pode ser consultado no Anexo 2 – Código Cloison do robô do lado esquerdo.

4.1.6 – Implementação em chão de fábrica

Tendo os periféricos programados, passou-se à fase final da implementação do sistema de aparafusamento do Cloison, realizada em ambiente de fábrica na Stellantis. Durante esta fase, foram identificados alguns constrangimentos.

Um dos principais desafios incidiu na disponibilidade de tempo para a montagem do sistema, uma vez que a empresa opera de forma contínua 6 dias por semana, com interrupções apenas aos sábados à tarde e aos domingos. Este fator limitou significativamente as janelas disponíveis para intervenção, tornando possível realizar trabalhos de instalação apenas aos sábados e domingos.

Durante a montagem, foram também detetadas algumas discrepâncias nas posições dos pontos de aperto inicialmente definidas na fase de preparação na Europneumaq. Estas inconsistências obrigaram à realização de ajustes e correções nos pontos de aperto pré-definidos, de modo a garantir o correto funcionamento do sistema.

Após efetuadas as correções necessárias, o sistema foi totalmente implementado e encontra-se atualmente em funcionamento em chão de fábrica. Na sequência, são

apresentadas algumas imagens ilustrativas da instalação: as figuras a), b) e c) referem-se ao robô implementado no lado direito da carrinha, enquanto as figuras d), e) e f) dizem respeito ao robô localizado no lado esquerdo.

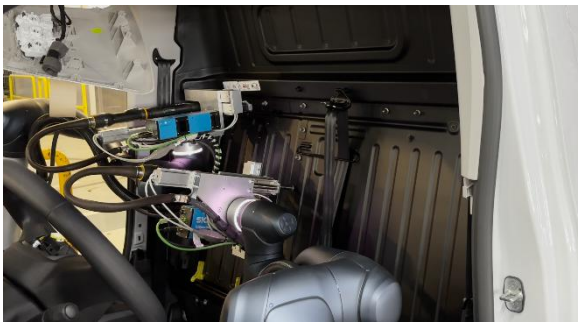
a)

d)

b)

e)

c)



f)

Figura 117 - Imagens da implementação do Cloison em chão de fábrica

Além das imagens, é também possível visualizar alguns vídeos do sistema em funcionamento através dos links indicados abaixo. Estes vídeos permitem observar, de forma mais detalhada, a operação dos robôs durante o processo de aparafusamento, bem como a interação entre os diferentes periféricos integrados no sistema.

Vídeo 1 – Aparafusamento lado direito: https://youtu.be/fkjfhaJC-9c

Vídeo 2 – Aparafusamento lado esquerdo: https://youtu.be/1HchMpNi3lM

4.2 – Sistema de aparafusamento do Porteur

Relativamente ao aparafusamento do Porteur, este encontra-se ainda em fase de desenvolvimento no que diz respeito à sua implementação em chão de fábrica. Com base na experiência adquirida no projeto do Cloison, a metodologia aplicada será similar. Considerando que deverão ser efetuados 14 apertos num intervalo de 120 segundos, será necessário utilizar quatro chaves de aperto distintas, adequadas aos diferentes tipos de parafusos.

Para cumprir os requisitos de tempo e complexidade, optou-se também pela utilização de dois *cobots* em vez de um único, atendendo tanto ao número de chaves necessárias quanto ao tempo disponível para a execução dos apertos.

4.2.1 – Escolha dos periféricos

No que se refere aos periféricos, o sistema de visão adotado será igual ao utilizado no projeto Cloison, uma câmara Sick TriSpector 1030 para a deteção precisa das posições dos pontos de aperto. Relativamente aos *cobots*, verificou-se que os modelos M1013, utilizados no Cloison, não seriam adequados para esta aplicação. Após análise, constatou-se que a necessidade de utilizar duas chaves diferentes para o aparafusamento implicaria um aumento significativo do peso do *gripper*, resultando num *payload* insuficiente para esses robôs. Além disso, o espaço disponível para a sua implementação é mais reduzido, o que, aliado ao comprimento dos braços do modelo M1013, comprometeria o seu funcionamento. Deste modo, foram realizados

testes com o robô Doosan M1509, que dispõe de um *payload* de 15 kg e alcance de 900 mm, apresentando-se mais adequado às exigências do projeto.

No que diz respeito à aparafusadora, optou-se por manter a marca Atlas Copco, porém desta vez foi escolhida uma ferramenta angular em substituição à reta anteriormente utilizada. Esta decisão resultou de testes práticos que indicaram ser esta a melhor solução, devido às posições específicas dos parafusos. Assim, foi selecionada a aparafusadora Atlas Copco ETV ST61-30-10 (Figura 118) para equipar ambos os *cobots*.

Figura 118 - Atlas Copco ETV ST61-30-10 [119]

4.2.2 – Simulações em ambiente virtual

Com os periféricos selecionados para a execução do projeto, avançou-se para a etapa de simulação, para garantir que os componentes escolhidos seriam os mais adequados para a tarefa.

RoboDK

Iniciou-se o processo pela importação dos modelos CAD para o software de simulação, seguida da definição do posicionamento dos periféricos. Na figura seguinte, é possível observar a área de aparafusamento, com os robôs devidamente posicionados. Quanto ao *gripper*, o apresentado não corresponde ao modelo final, tendo sido utilizado apenas para testes durante a fase de desenvolvimento.

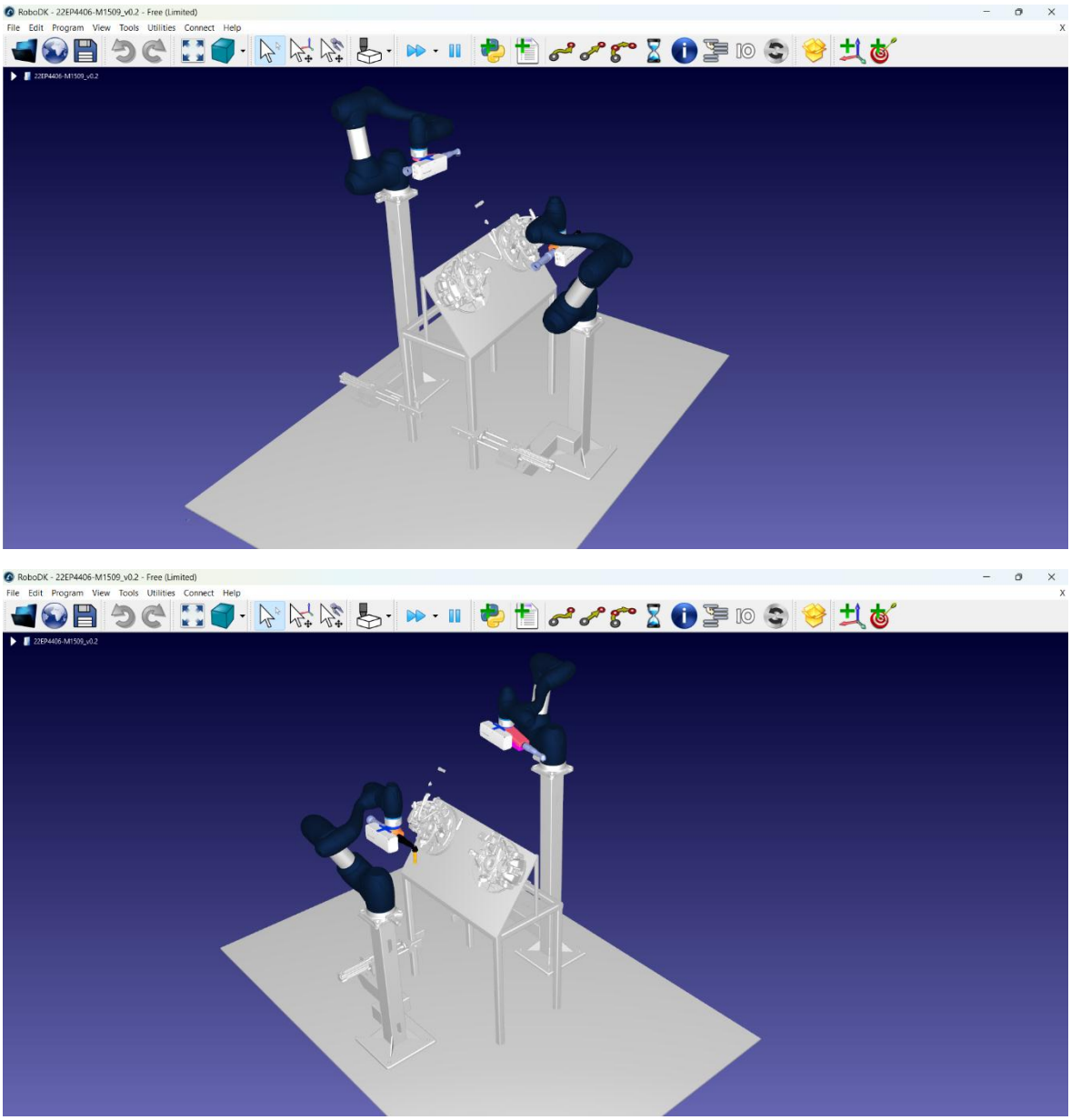


Figura 119 - Importação dos CADs no RoboDK

Após a importação dos modelos CAD, foram definidos alguns pontos de aperto em cada robô, com o objetivo de assegurar que estes seriam capazes de realizar os apertos de forma correta e sem dificuldades (Figura 120). Não foi necessário inserir todos os 14 pontos de aperto, uma vez que alguns parafusos se encontravam no mesmo plano. Assim, foi suficiente verificar que o *cobot* conseguiria alcançar pelo menos um dos pontos desse plano, garantindo, a capacidade de alcançar os restantes pontos localizados na mesma superfície.

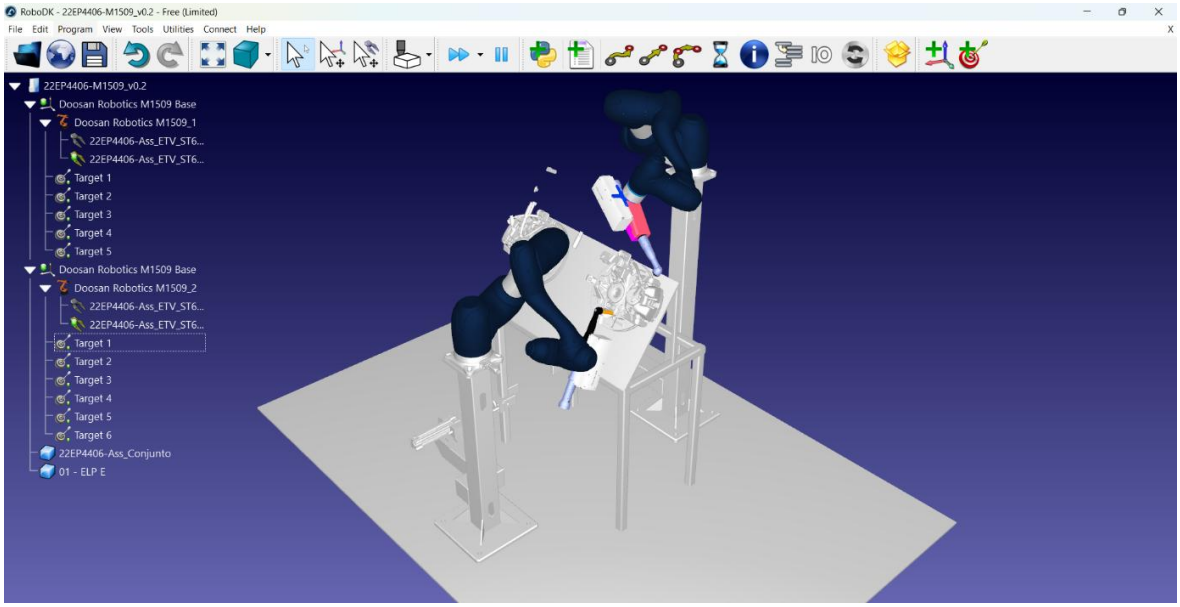


Figura 120 - Inserção e teste de alguns pontos de aperto

Como é possível verificar na figura acima, foram inseridos vários "*targets*" que correspondem a diferentes pontos de aperto. Com base nesses pontos, foram realizados testes em ambiente virtual para garantir que os periféricos escolhidos seriam os mais adequados para a tarefa.

4.2.3 – Montagem em ambiente real na Europneumaq

Finalizadas as simulações necessárias, deu-se início à execução do projeto em ambiente real, começando pela aquisição dos materiais e realização dos testes preliminares, primeiramente na Europneumaq, para posterior implementação em chão de fábrica.

Foram adquiridos dois robôs Doosan M1509, uma câmara Sick, igual às utilizadas no Cloison, e as duas aparafusadoras mencionadas anteriormente. Diferente do que ocorreu no Cloison, no projeto do Porteur a Stellantis concedeu maior flexibilidade quanto aos horários de trabalho na fábrica. Com isso, foram realizadas apenas pequenas configurações dos periféricos e testes na Europneumaq (Figura 121), enquanto se aguardava a chegada do material, sendo que o trabalho principal foi desenvolvido diretamente em chão de fábrica.

Figura 121 - Pequenos testes na Europneumaq

Dart-Platform

Com a chegada dos robôs, procedeu-se à sua configuração. Não me irei alongar neste tema, visto que já foi abordado de forma detalhada no capítulo **4.1.4 – Montagem em ambiente real na Europneumaq**, relativo ao Cloison, e o processo aplicado foi o mesmo. Foram configurados os IPs dos robôs, definidos os seus TCPs correspondentes ao novo *gripper*, incluindo o peso, e, por fim, estabelecidas as *Home Position* e as zonas colaborativas, as quais foram posteriormente ajustadas durante a implementação na fábrica. Todos estes aspetos encontram-se descritos com maior pormenor no capítulo referido.

4.2.4 – Programação dos periféricos

Após a realização das configurações necessárias, bem como o desenvolvimento do *gripper*, procedeu-se à fase de programação dos periféricos, nomeadamente a câmara de visão, as aparafusadoras e, posteriormente, os robôs.

Sistema de visão

Relativamente ao sistema de visão, após análise do local onde será implementada a automação e com o apoio dos testes realizados em ambiente de simulação, concluiu-se que seria viável automatizar todo o processo com recurso a uma única câmara de visão, instalada no robô posicionado de frente para os discos. A opção por utilizar apenas uma câmara revela-se vantajosa tanto em termos de tempo de execução do processo de aparafusamento, dado que apenas esse robô necessita de realizar o varrimento dos pontos, como também do ponto de vista económico, uma vez que reduz a necessidade de aquisição de equipamento adicional.

Com base nesta decisão, procedeu-se à programação do sistema de visão. Foi utilizado o mesmo software aplicado no projeto Cloison, o SOPAS, dado que a câmara é igual. Iniciou-se a configuração da ligação à câmara, seguida da parametrização do tratamento de imagem (Figura 122), onde é possível ajustar diversos parâmetros (detalhados na Secção **4.1.5 – Programação dos periféricos**) de forma a otimizar a captação da imagem.

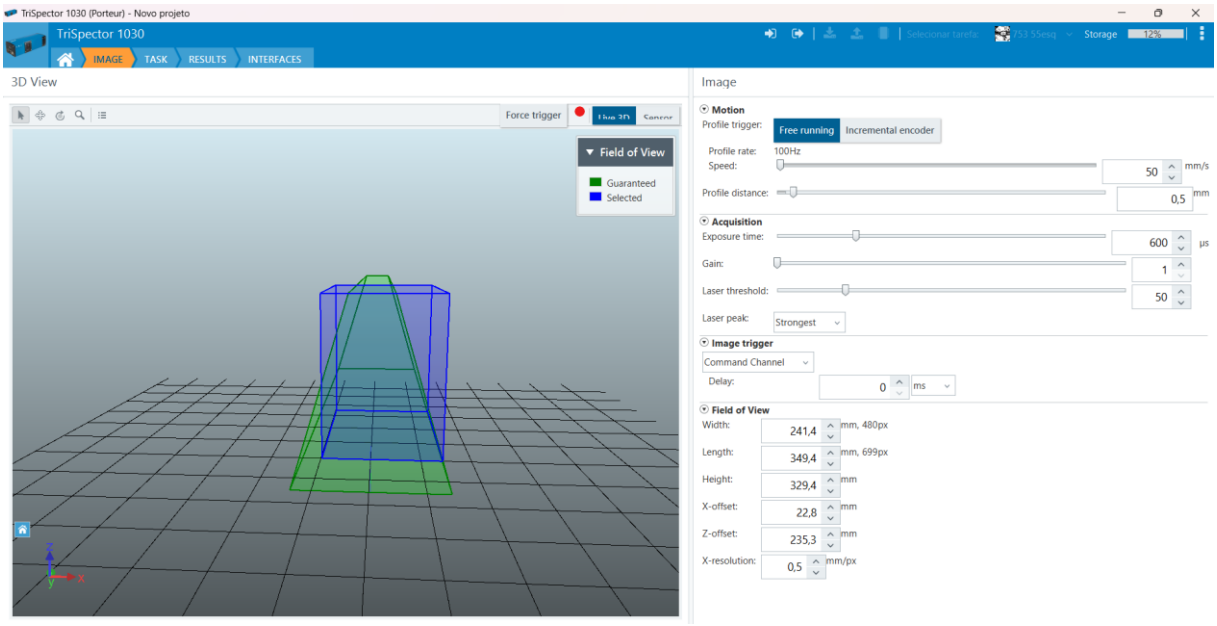


Figura 122 - Configuração da câmara para aquisição da imagem

Concluída a configuração, foi possível obter uma imagem com a nitidez evidenciada na Figura 123, a qual permite a deteção dos parafusos necessários, possibilitando assim a posterior correção da posição de cada ponto de aperto.

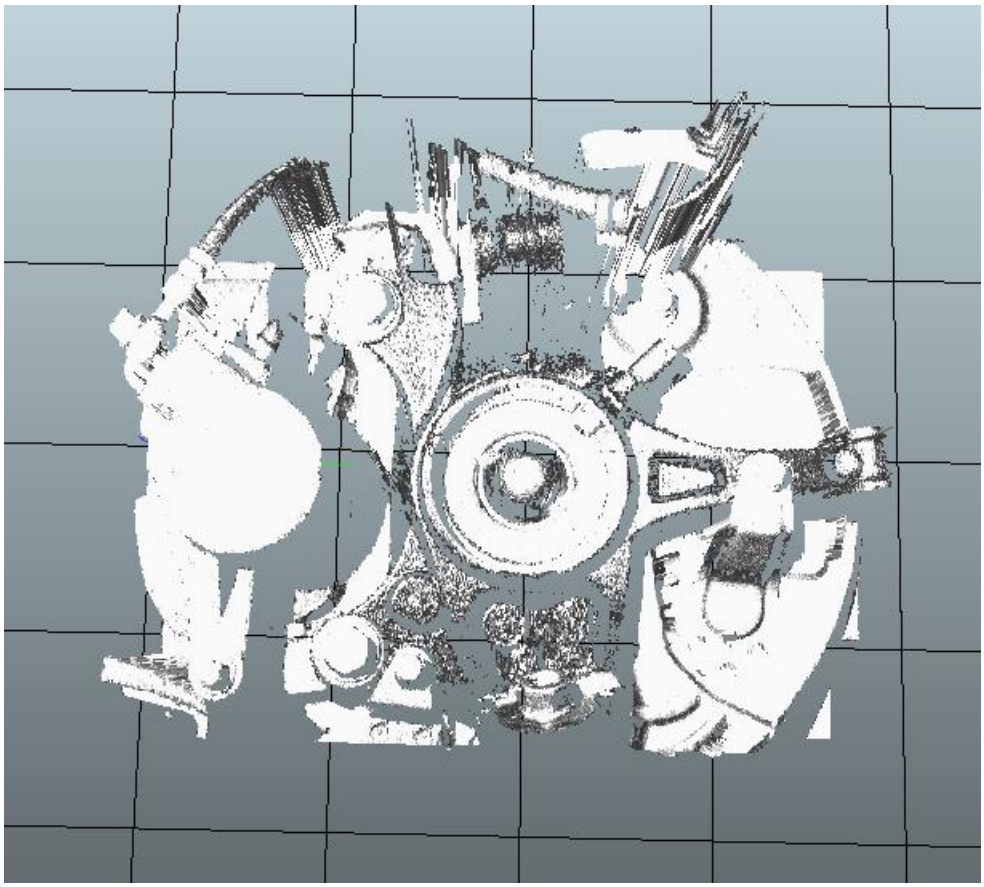


Figura 123 - Imagem adquirida para o seu tratamento

Com a imagem suficientemente nítida para permitir a deteção dos parafusos, procedeu-se ao desenvolvimento do programa da câmara destinado à aquisição das coordenadas X, Y e Z dos parafusos, com vista à correção das respetivas posições. Foi elaborado um programa que permite a captação dessas coordenadas em três parafusos de referência, os quais serão posteriormente utilizados para corrigir o *frame* pré-definido no robô. A Figura 124 apresenta o programa desenvolvido para esse efeito.

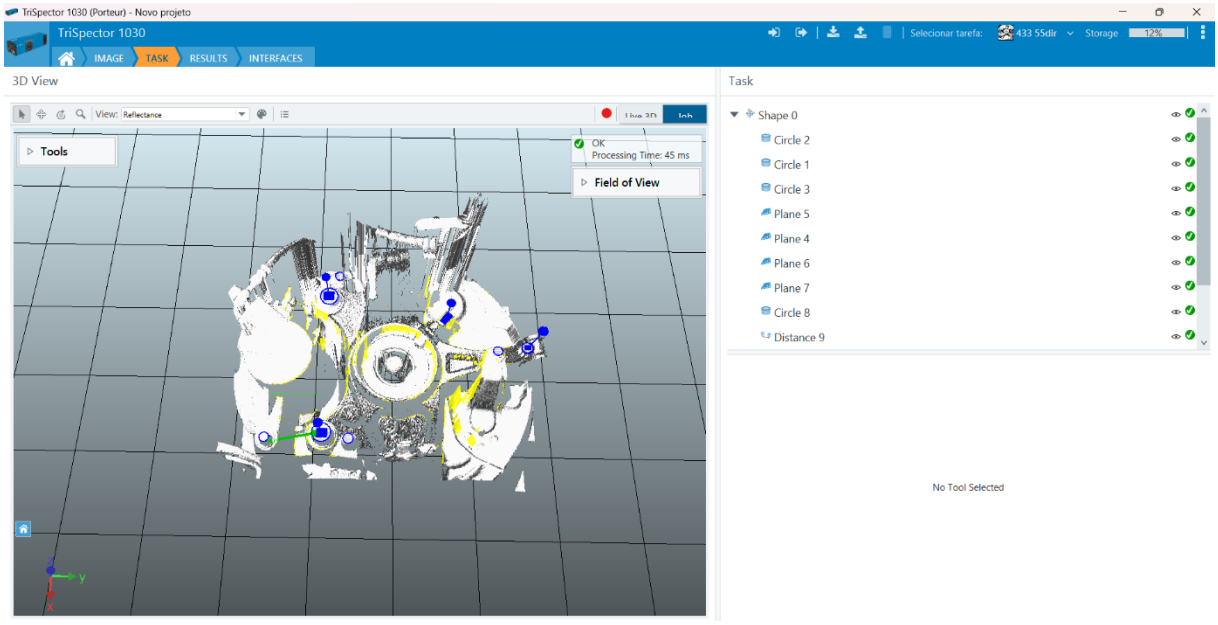


Figura 124 - Programa para aquisição das informações dos parafusos

Recorreu-se novamente à ferramenta *shape* para verificar a validade da imagem adquirida. Esta ferramenta permite a seleção da área de interesse, excluindo as regiões que não são relevantes para a captação da imagem, como zonas sem utilidade específica ou de difícil visualização devido à profundidade. Para esse fim, foram aplicadas máscaras que delimitam as áreas a ignorar, uma vez que estas não influenciam o resultado final pretendido.

Adicionalmente, é atribuída uma pontuação à imagem com o objetivo de avaliar a sua qualidade e fiabilidade. No presente caso, a ferramenta foi configurada com os parâmetros e máscaras apresentados na figura abaixo. Após a realização de vários testes, concluiu-se que uma pontuação de 75% constituía o valor mais adequado para validar a imagem adquirida. Este valor foi definido tendo em conta possíveis interferências, nomeadamente variações nas condições de iluminação e eventuais deformações mínimas. Estas pequenas variações não comprometem o objetivo principal da aquisição, que consiste na identificação das posições dos parafusos.

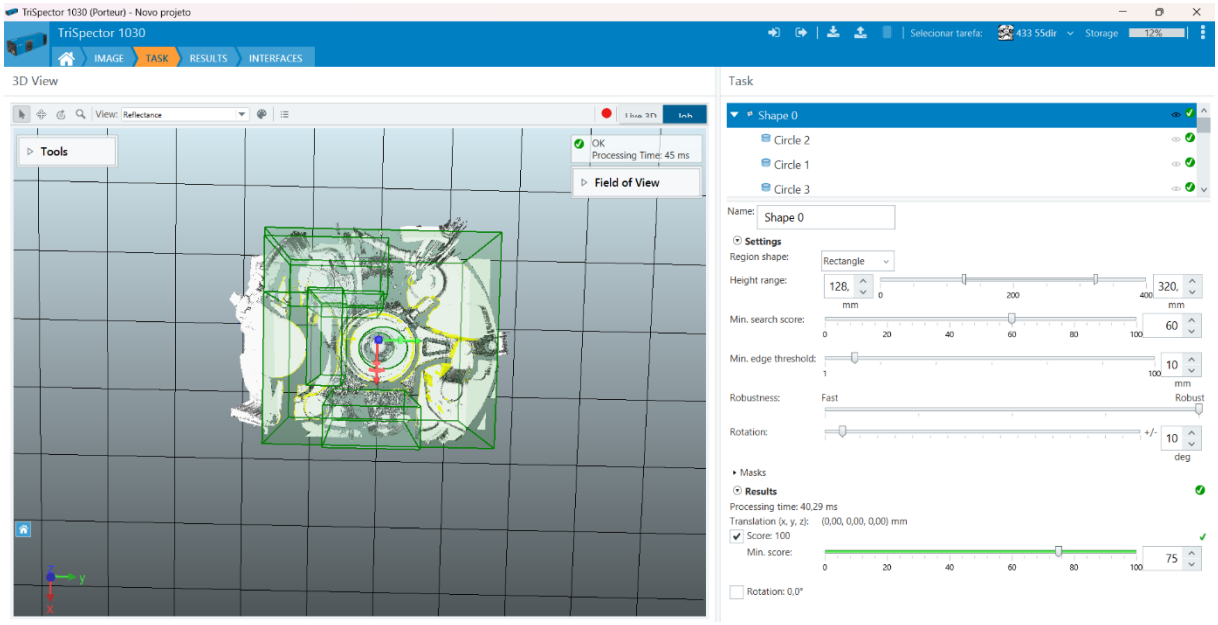


Figura 125 - Tool Shape

Na figura apresentada, é possível identificar vários retângulos a verde. O retângulo de maiores dimensões delimita a área de interesse, enquanto os retângulos e a elipse localizados no seu interior correspondem às máscaras previamente referidas.

Para a aquisição das coordenadas X e Y dos parafusos, foi utilizada uma ferramenta diferente da utilizada no projeto Cloison. Se anteriormente se recorreu à ferramenta *blob* para essa finalidade, neste caso, devido à menor nitidez da imagem, causada por variações acentuadas no relevo e nas condições de iluminação, a utilização da ferramenta *blob* não produzia resultados suficientemente fiáveis. Por esse motivo, optou-se pela ferramenta *circle*, mais adequada ao contexto.

A *tool circle* é uma ferramenta destinada à deteção e verificação de círculos, buracos ou arestas com forma circular. Considerando que os parafusos podem ser identificados visualmente como círculos, os testes realizados demonstraram que esta seria a abordagem mais eficaz para a aquisição das suas coordenadas.

Procedeu-se, assim, à configuração dos círculos correspondentes aos diferentes parafusos a apertar, tendo sido selecionados os três que apresentaram resultados mais consistentes. A figura seguinte ilustra a configuração de um desses círculos, sendo que os restantes apresentam configurações semelhantes, variando apenas na sua posição. São igualmente visíveis, com círculos azuis, os três parafusos selecionados para obtenção das respetivas coordenadas.

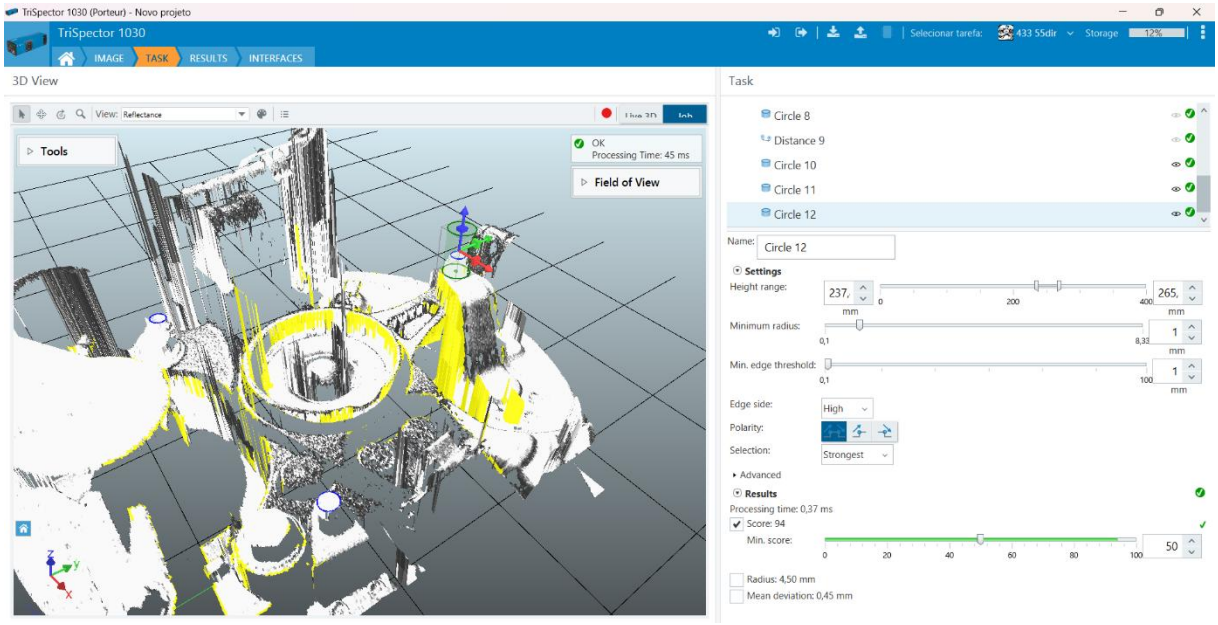


Figura 126 - Tool Circle

Para obter a coordenada Z dos parafusos referidos, recorreu-se à mesma ferramenta utilizada no projeto Cloison, a *tool plane*. Esta ferramenta permite a seleção de um plano, retornando a coordenada correspondente à sua posição no espaço. Na figura seguinte é possível observar a aplicação desta ferramenta, bem como as configurações associadas à sua parametrização.

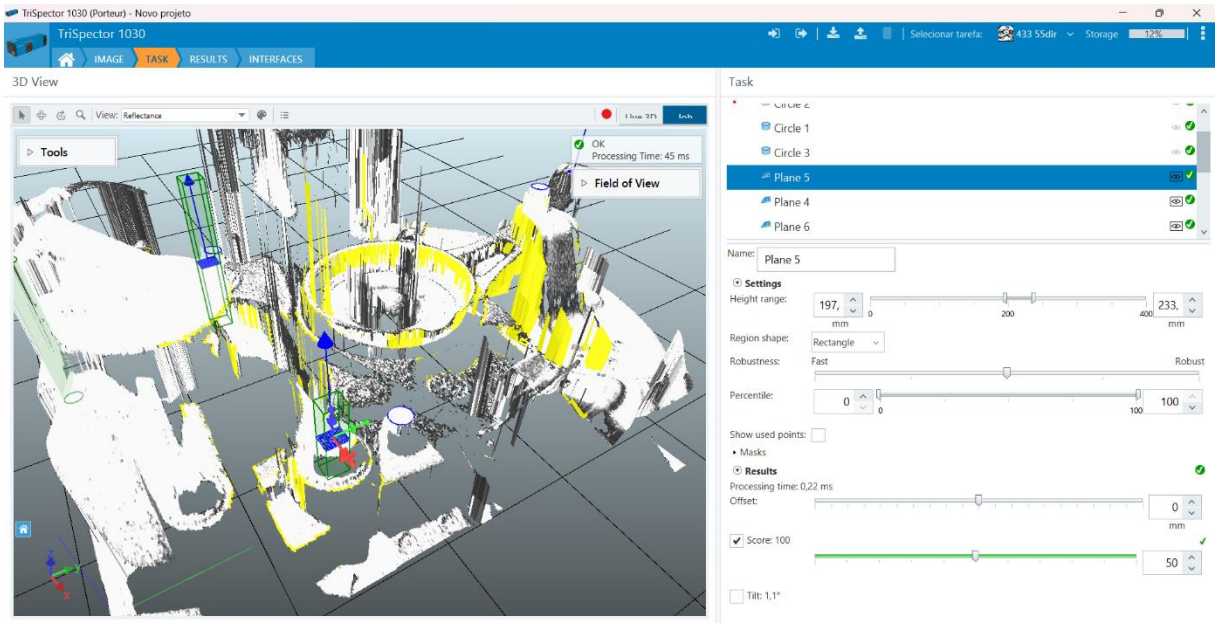


Figura 127 - Tool Plane

A partir deste momento, o programa encontrava-se finalizado, fornecendo as coordenadas X, Y e Z dos três parafusos selecionados. Nesta fase, restava apenas proceder à configuração da página *results*. Esta configuração foi realizada de forma semelhante à adotada no projeto Cloison, conforme ilustrado na Figura 128.

Figura 128 - Página Results

Por fim, procedeu-se à configuração da página de *interface*, responsável pela gestão da comunicação com a câmara. Nesta secção, é possível configurar o endereço IP da câmara, de forma a estabelecer a comunicação, bem como definir o *command channel* e a *output string*. A figura seguinte apresenta a configuração efetuada.

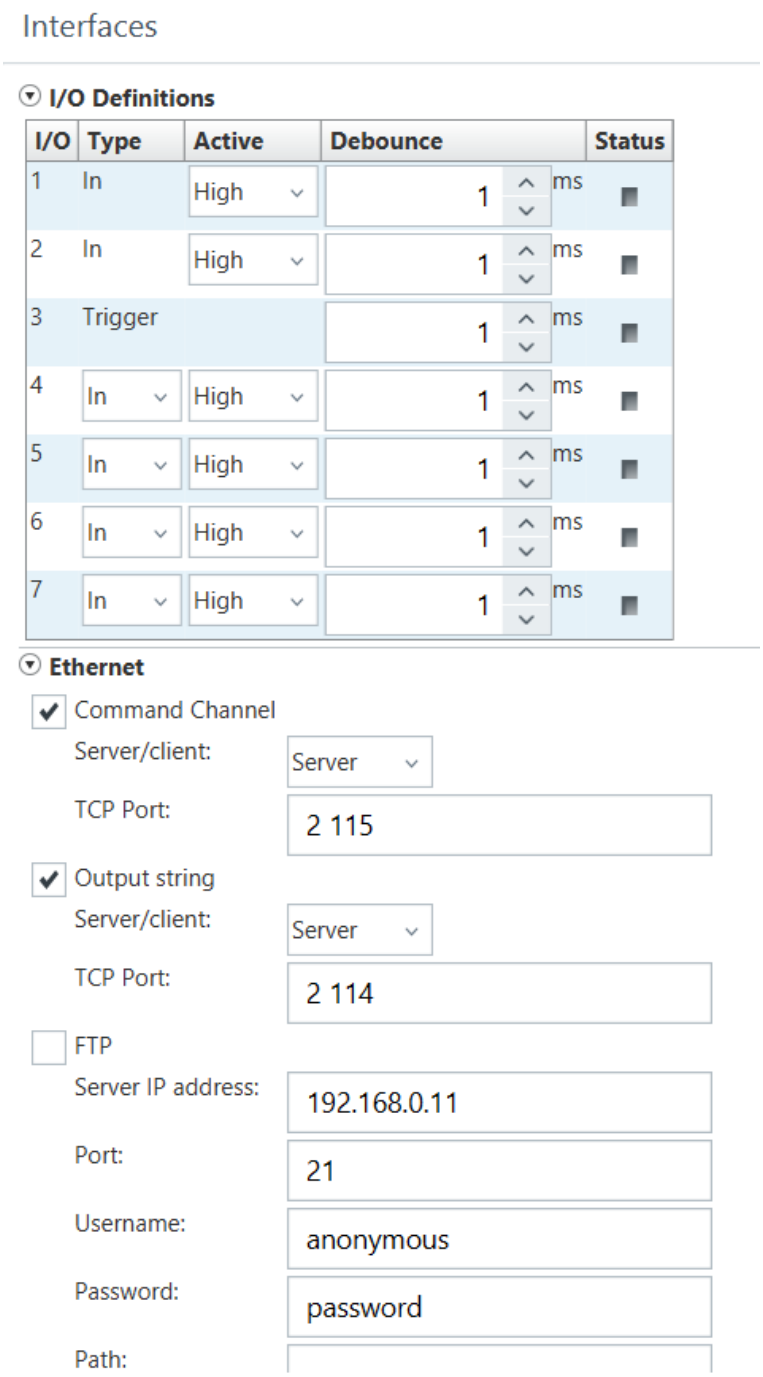


Figura 129 - Página Interface

Dado que o processo de aperto é realizado em dois conjuntos do disco, foi necessário replicar todo o procedimento para o disco do lado direito. Tal como se pode observar na maquete de simulação representada na Figura 119, a estrutura integra dois discos, um à esquerda e outro à direita, sendo o objetivo determinar as coordenadas de três parafusos em cada um. Por este motivo, foi desenvolvido um segundo programa, desta vez destinado ao disco do lado direito.

Este novo programa é, em grande parte, idêntico ao anterior, tanto ao nível das configurações como da lógica implementada, diferindo apenas na localização das

ferramentas utilizadas. Com esta etapa, ficou concluída a programação do sistema de visão

Sistema de aparafusamento

No que diz respeito ao sistema de aparafusamento, foi desenvolvido um *gripper* capaz de suportar o peso da aparafusadora angular (Figura 118) e da câmara de visão (Figura 57). Apesar de apenas um dos robôs utilizar uma câmara, os *grippers* de ambos são muito semelhantes, diferenciando-se apenas pela presença do suporte para a câmara.

O *gripper* é composto por duas partes distintas que se encaixam entre si, sendo responsável por suportar a aparafusadora na zona central e a câmara lateralmente. As duas componentes do *gripper* mencionadas podem ser observadas na figura abaixo.

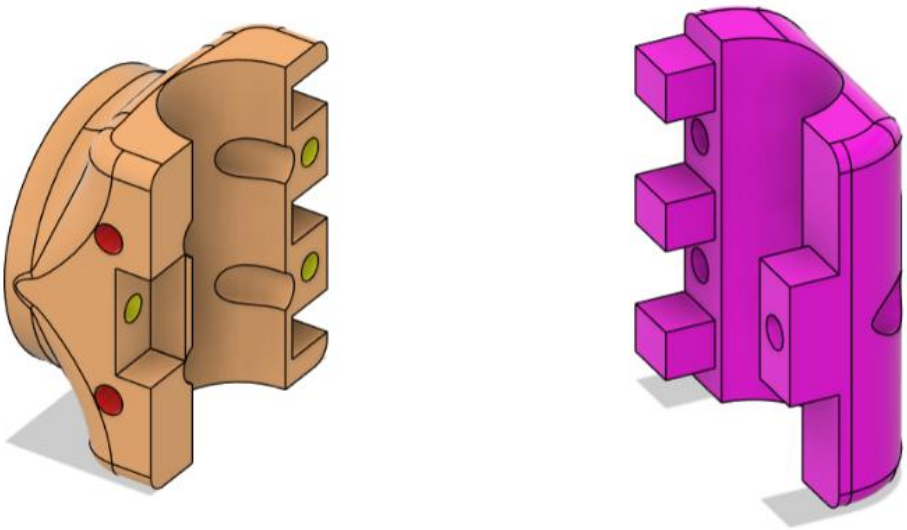


Figura 130 - Partes separadas do gripper

Segue-se a apresentação da versão final do *gripper*, com a aparafusadora e a câmara instaladas.

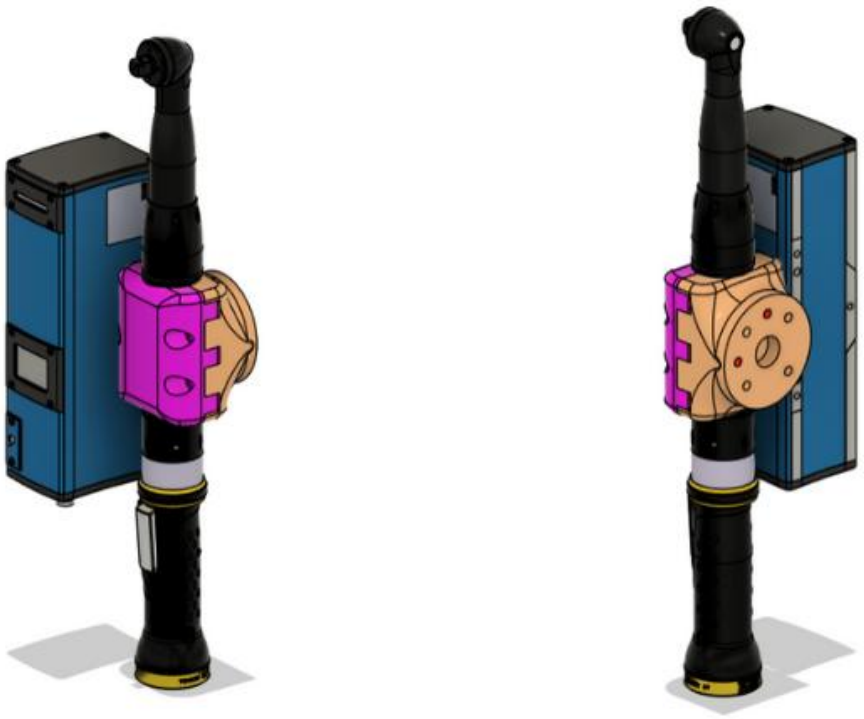


Figura 131 - Gripper completo

Os bits utilizados pelas aparafusadoras do Porteur serão diferentes em cada um dos robôs, dado que cada um deverá operar com dois bits distintos para o aperto. A aparafusadora A estará equipada com as chaves H13 e H11, enquanto a aparafusadora B utilizará as chaves TORX e H10.

Relativamente ao bit da aparafusadora A, este permitirá aparafusar os parafusos H13 e H11 sem necessidade de troca de bit, uma vez que o bit incorpora a chave H11 no interior da chave H13, ou seja, a primeira é menor que a segunda, possibilitando assim o desenvolvimento de um único bit que integra ambas as chaves. Na figura abaixo, pode-se observar o bit completo em a) e a diferença entre as duas chaves em b).

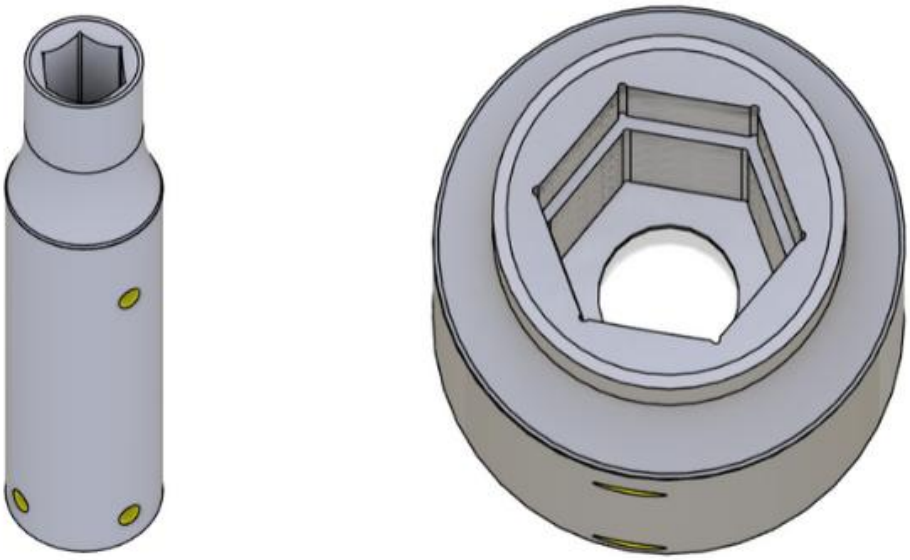


Figura 132 - Bit para aparafusadora A (Chave H13 - Chave H11)

Quanto ao bit da aparafusadora B, este deverá ser capaz de apertar parafusos TORX e H10. Para tal, foi desenvolvido um bit que permite a transição entre a chave H10 e a TORX através da simples adição ou remoção de uma peça, procedimento que pode ser efetuado em poucos segundos por um operador. Este mesmo operador será responsável pelo posicionamento e remoção da maquete no início e no final do processo de aparafusamento. As duas chaves encontram-se ilustradas na figura abaixo.

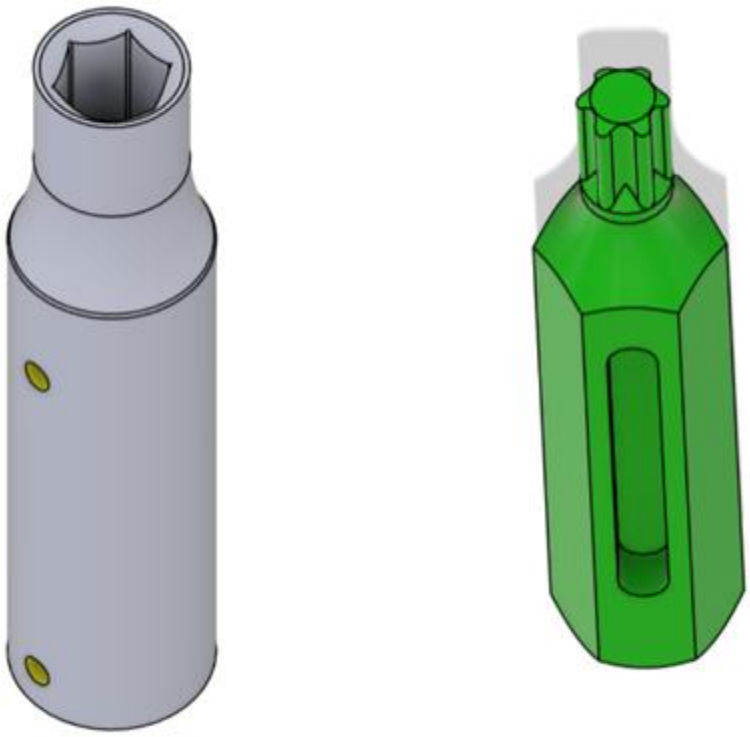


Figura 133 - Chave H10 - Chave TORX

Com as duas chaves disponíveis, é possível inserir a chave TORX no interior da chave H10, conforme ilustrado na Figura 134.

Figura 134 – Chave TORX aplicada na chave H10

A programação das duas aparafusadoras seguiu o mesmo procedimento utilizado no projeto Cloison. Esta programação foi integralmente da responsabilidade da Stellantis, que nos forneceu apenas as informações relativas aos inputs e outputs necessários para a programação dos robôs, nomeadamente o sinal de início do programa da aparafusadora (início do aperto) e o input que indica o relatório do aperto (OK ou NOK), sendo que toda a comunicação ocorre através do PLC.

Com o conhecimento dos *inputs* e *outputs* a utilizar, procedeu-se à implementação da programação dos robôs.

Programação do robô

Concluída a programação dos periféricos restantes, permaneceu apenas a tarefa de programar os robôs. Como referido anteriormente, foram utilizados dois robôs devido à necessidade de operar com chaves diferentes. Os códigos apresentam algumas diferenças, sobretudo na componente do sistema de visão, dado que apenas uma câmara será utilizada. Esta estará instalada no robô 1 e será responsável pela aquisição das imagens necessárias, enquanto o robô 2 apenas comunicará com a câmara para obter os respetivos resultados.

Os códigos de ambos os robôs encontram-se disponíveis no capítulo dos Anexos, devidamente comentados: o Anexo 3 – Código Porteur do robô 1 contém o código do robô 1, e o Anexo 4 – Código Porteur do robô 2 apresenta o código do robô 2.

Para facilitar a compreensão da aplicação em questão e do seu funcionamento, foi elaborado um fluxograma (Figura 135). Seguidamente, serão explicados, de forma detalhada, algumas secções do código referentes apenas ao robô 1, uma vez que o robô 2 segue um raciocínio semelhante.

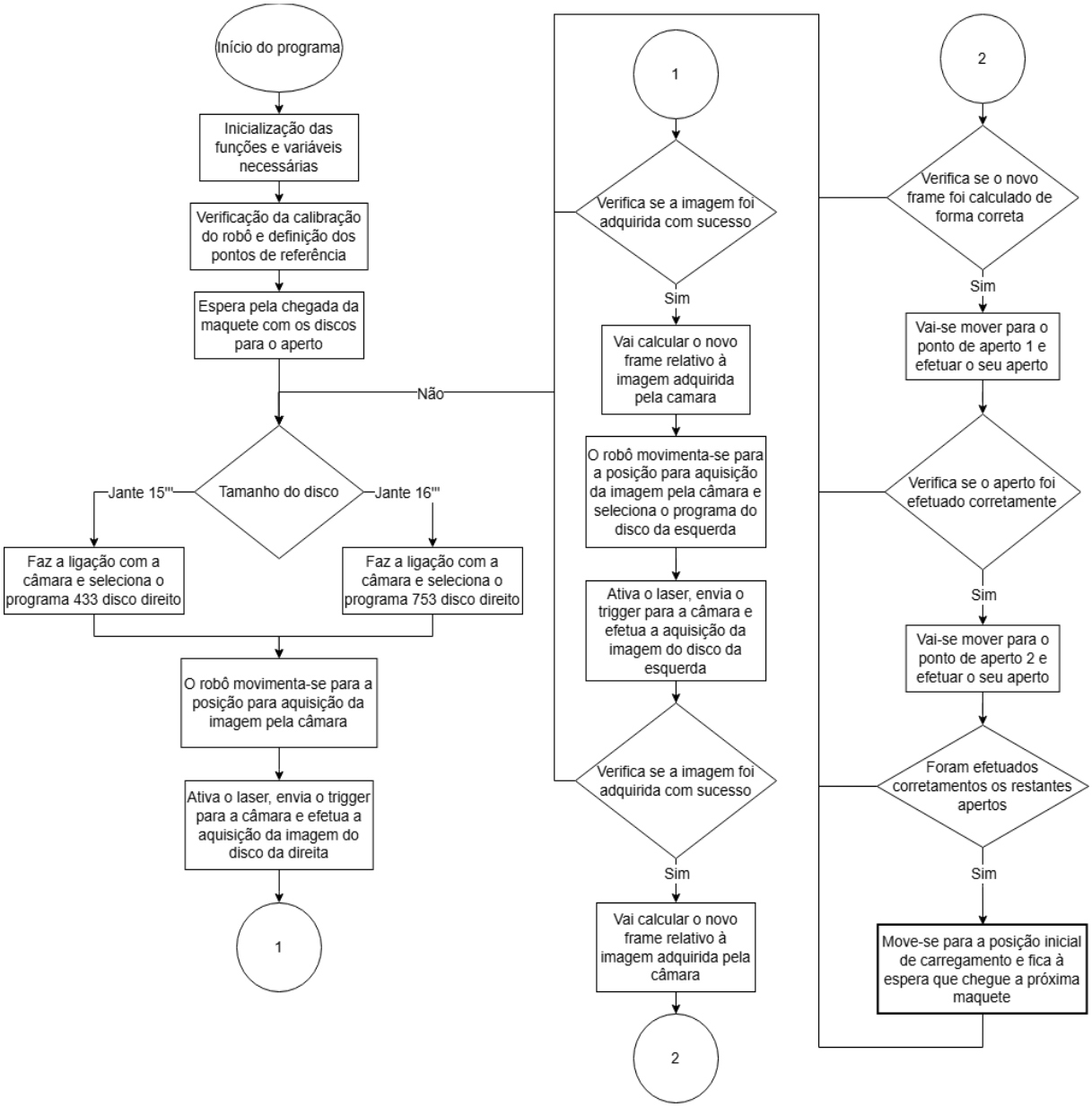


Figura 135 - Fluxograma relativo à aplicação

No fluxograma apresentado, é possível verificar as diversas etapas que compõem o código de forma simplificada. O penúltimo bloco corresponde à verificação do aperto dos restantes parafusos, criado para reduzir a complexidade do fluxograma e evitar a repetição desnecessária. A partir desse ponto, o código limita-se a executar o aperto,

confirmar se este foi realizado com sucesso e avançar para o seguinte. Caso o aperto não seja efetuado corretamente, o robô regressa à sua posição inicial.

Relativamente ao código (Anexo 3 – Código Porteur do robô 1), no início são definidas as funções relacionadas com o sistema de visão. Destacam-se a função “def init(self, ip, portIN, portOUT, state=0)”, utilizada para estabelecer a ligação com a câmara; a função “def write(self, cmd, socket=None)”, que permite enviar comandos ao *socket*; a função “def readcmd(self, length, timeout, socket=None)”, responsável pela leitura da informação recebida pelo *command channel*; e a função “def readata(self, length, timeout, socket=None)”, que obtém os dados provenientes do sistema de visão. Não será aprofundada a descrição destas funções, dado que já foram explicadas detalhadamente no capítulo 4.1.5 **– Programação dos periféricos**, relativo à programação do robô no projeto Cloison, sendo a abordagem idêntica.

Seguem-se as funções responsáveis pelo sistema de aparafusamento, com poucas alterações em relação às utilizadas no Cloison. Inicialmente, destaca-se a função “wait_clear”, que aguarda o relatório da aparafusadora durante o processo de aperto, ou seja, quando o aperto é concluído, a aparafusadora envia um relatório que indica se o procedimento foi ou não bem-sucedido. Esta função é requisitada na função seguinte, “wait_torque”, que controla todo o processo de aperto. Quando requisitada, “wait_torque” envia o sinal para iniciar o aperto, espera pela sua conclusão e pelo relatório emitido (utilizando “wait_clear”) e informa o robô para agir consoante o resultado do aperto.

Por fim, destaca-se a função “wait_touch”, que, como o nome indica, tem por objetivo aguardar o contacto da aparafusadora com o parafuso. Este procedimento consiste em o robô exercer uma pressão na direção do aperto, garantindo que seja aplicada uma força aproximada de 20 N antes do início do aperto propriamente dito, assegurando assim uma inserção eficaz do bit no parafuso.

Estas funções podem ser observadas na figura seguinte.

```
#Função que vai aguardar que a aparafusadora esteja pronta novamente (que terminou o ciclo)
def wait_clear() :
        tt = True
        while tt:
            aperto=get_input_register_int(1)
            if(aperto==0):
                time.sleep(0.2)
                tt = False
            time.sleep(0.2)

#Função que irá esperar que o torque pretendido seja atingido para desativar a aparafusadora
def wait_torque() :
            yy = True
            zz = 0
            set_output_register_int(2, 55) #Manda sinal para o PLC para ativar a aparafusadora
            while yy:
                step_ack=get_input_register_int(1) #Guarda na variavel step_ack o valor do relatório
                if(step_ack==1111) or (step_ack==2222) or (step_ack==4444): #Só entra neste if quando recebe o relatório, seja OK ou NOK
                    time.sleep(0.1)
                    set_output_register_int(2, 0) #Output para desativar aparafusadora
                    result=get_input_register_int(1) #Guarda o relatório do aperto em result
                    yy = False
                    set_output_register_int(3, 777) #Info para PLC ACK (Informa o PLC que recebeu a tomou a decisao relativa ao relatorio)
                    wait_clear() #Aguarda pelo termino do ciclo da aparafusadora
                    #Vai verificar o relatório enviado pela aparafusadora
                    if result==4444: #Valor caso o relatório seja OK
                        set_output_register_bit(10, 1)
                    if result!=4444: #Valor caso o relatório seja NOK
                        set_output_register_bit(10, 0)

                #TimeOut de 50 segundos para caso nao receber resposta da aparafusadora, o loop termina e a ferramenta é desligada por segurança.
                if zz >= 50:
                    yy = False
                    set_output_register_int(2, 0)
                time.sleep(0.1)
                zz = zz + 0.1

#Função utilizada para confirmar que a aparafusadora está a fazer pressão no parafuso (20N)
def wait_touch() :
            xx = True
            while xx:
                xx = check_force_condition(axis=DR_AXIS_Z, max=20, ref=DR_TOOL)
                wait(0.1)
```

Figura 136 - Funções wait_clear, wait_torque e wait_touch

Foram igualmente definidos vários outputs e variáveis essenciais para o correto funcionamento do código, destacando-se a seleção da ferramenta a utilizar, realizada pela função “set_tool”, a definição do TCP correspondente, através da função “set_tcp”, bem como a determinação do referencial pretendido, estabelecida pela função “set_ref_coord”. Foi também declarada a variável global “*line*”, responsável por controlar a execução sequencial do código, conforme implementado no projeto Cloison.

De seguida, procedeu-se à ligação com a câmara de visão, seguida da verificação da validade da respetiva conexão. Por fim, verificou-se se o robô estava devidamente calibrado, sendo posteriormente enviado para a sua posição inicial (*Home Position*), conforme ilustrado na Figura 137.

```
#Definição de algumas variáveis globais e outputs importantes
set_motion_end(DR_CHECK_ON) #Funçãoo para que o robo espere terminar o movimento atual antes de executar o próximo comando
set_output_register_int(0, 0) # Info para PLC qual etapa está o Robô
set_output_register_int(1, 0) # Info para PLC resultado Visão
set_tool ("U_Tool_W1")  #Seleciona a tool que se pretende
set_tcp("U_Tool_0")     #Seleciona o TCP da respetiva tool
set_ref_coord(DR_BASE)  #Define o referencial de coordenadas para movimentar o robô
time.sleep(0.5)

#Definição de uma variável global que será usada posteriormente para indicar para onde irá a execução do programa
def goto(linenum):
    global line
    line = linenum
line = 1

#Vai fazer conecção do robô com a câmara de visão
camera_ip = "192.168.2.81"
trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114, state=0)
time.sleep(0.5)

#Verifica se a conecção foi estabelecida
if trispector.state != 1:
    set_output_register_int(1, 20) # Info para PLC resultado Visão (não estabelecida)
    set_output_register_int(0, 210) #Infor para PLC da etapa
    time.sleep(2)
    set_output_register_int(0, 401) #Indica que não foi efetuada corretamente a ligação com a camara ao PLC
    exit() #Termina o programa

laser_off = trispector.write("set laser off") #Desliga o lazer

#Vai fechar os sockets uma vez que terminou a comunicação com a câmara
trispector.closeIN(None)
trispector.closeOUT(None)

zero = check_robot_mastering() #Verifica se o robô está num estado que requer masterização ou não (calibração)

if zero == 1: #Se necessitar de recalibração
    move_home(DR_HOME_TARGET_USER) #Move-se para a posição Home definida pelo utilizador para fazer recalibração


movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
```

Figura 137 - Definição de outputs e variáveis globais

Antes de iniciar o ciclo de execução do código, foram também definidos os pontos de referência. Estes pontos constituem o sistema de coordenadas base, que servirá de comparação com o sistema gerado a partir dos pontos obtidos pelo sistema de visão, permitindo assim a criação de um novo referencial para o aperto dos parafusos. Esta correção é necessária devido a possíveis desvios na posição da maquete quando esta é colocada, que poderão comprometer a precisão do processo.

Para compensar esses desvios, recorre-se ao sistema de visão, conforme referido anteriormente, sendo que os pontos de referência desempenham um papel fundamental na correção durante a fase de aperto pelo robô.

Adicionalmente, na Figura 138, pode observar-se um conjunto de pontos identificados por 433, localizado no lado esquerdo da figura, e outro conjunto, identificado por 753, no lado direito. Esta diferenciação deve-se à existência de discos com diferentes diâmetros e pinças de tamanhos variados que podem ser posicionados na maquete, tornando imprescindível que o robô seja capaz de efetuar o aperto independentemente do disco que seja utilizado.

```
#**Pontos de aperto 433 DIREITO (disco 1)

#Ponto1 direito 433

System_433d_pa1 = posx(-0.810, 4.800, 18.750, 81.53, -168.44, -69.0)
System_433d_p1 = posx(-1.050, 0.600, -0.560, 81.53, -168.44, -69.0)

#Ponto2 direito 433

System_433d_pa2 = posx(185.550, 4.910, 22.320, 80.36, -168.15, -34.44)
System_433d_p2 = posx(184.700, -0.090, -1.840, 80.36, -168.15, -34.44)

#Ponto3 direito 433

System_433d_pa3 = posx(117.710, 158.820, 19.530, 87.68, -168.34, -10.71)
System_433d_p3 = posx(117.570, 155.160, 1.760, 87.68, -168.34, -10.71)

#TXDireito 433

System_433d_pa4 = posx(67.770, 103.190, -5.130, 128.49, -80.13, 14.12)
System_433d_p4 = posx(77.860, 90.420, -2.310, 127.09, -80.48, 14.36)

#*****************************************************************************************
#*****************************************************************************************

#Ponto1 esquerdo 433

System_433e_pa1 = posx(-0.250, 2.350, -15.190, 88.88, -11.01, 128.45)
System_433e_p1 = posx(-0.300, -0.610, 0.020, 88.88, -11.01, 128.45)

#Ponto2 esquerdo 433

System_433e_pa2 = posx(182.980, 4.150, -18.980, 73.75, -11.44, -129.07)
System_433e_p2 = posx(181.850, 0.260, 1.050, 73.75, -11.44, -129.07)

#Ponto3 esquerdo 433

System_433e_pa3 = posx(112.410, 158.810, -22.640, 79.96, -12.37, -102.83)
System_433e_p3 = posx(111.550, 153.960, -0.160, 79.96, -12.37, -102.83)

#TXEsquerdo 433

System_433e_pa4 = posx(62.010, 108.130, 5.510, 128.49, -97.63, 135.08)
System_433e_p4 = posx(76.300, 90.160, 2.450, 128.49, -97.63, 135.08)
```

```
#**Pontos de aperto 753 (disco 2)

#Ponto1 direito 753

System_753d_pa1 = posx(-2.510, 3.510, 12.180, 89.95, -168.71, -57.18)
System_753d_p1 = posx(-0.510, 0.630, -0.040, 89.95, -168.71, -57.18)

#Ponto2 direito 753

System_753d_pa2 = posx(184.260, 5.720, 26.220, 85.9, -168.03, -47.72)
System_753d_p2 = posx(183.870, 0.320, 0.630, 85.9, -168.03, -47.72)

#Ponto3 direito 511

System_753d_pa3 = posx(117.370, 160.010, 26.230, 87.24, -169.03, -31.32)
System_753d_p3 = posx(117.120, 154.930, 0.000, 87.24, -169.03, -31.32)

#TXDireito 753

System_753d_pa4 = posx(55.700, 117.840, -8.320, 130.78, -80.41, 10.23)
System_753d_p4 = posx(77.340, 92.280, -3.360, 130.4, -80.39, 10.31)

#*****************************************************************************************
#*****************************************************************************************

#Ponto1 esquerdo 753

System_753e_pa1 = posx(0.660, 4.780, -16.170, 91.5, -11.48, 123.66)
System_753e_p1 = posx(1.070, 1.600, 0.680, 91.5, -11.48, 123.66)

#Ponto2 esquerdo 753

System_753e_pa2 = posx(184.200, 2.520, -18.150, 90.68, -11.57, -121.91)
System_753e_p2 = posx(182.340, -0.140, 0.730, 61.37, -11.5, -147.72)

#Ponto3 esquerdo 753

System_753e_pa3 = posx(118.400, 158.970, -21.400, 74.06, -11.3, -168.32)
System_753e_p3 = posx(116.950, 155.550, 0.350, 74.06, -11.3, -168.32)

#TXEsquerdo 753

System_753e_pa4 = posx(56.840, 121.350, 8.590, 128.25, -97.81, 130.47)
System_753e_p4 = posx(80.290, 92.860, 3.620, 128.25, -97.81, 130.47)
```

Figura 138 - Pontos de referência relativos a dois discos distintos

Com todas as variáveis e funções devidamente definidas, iniciou-se o desenvolvimento do código que será executado em ciclo contínuo ao longo de toda a aplicação. O primeiro passo consistiu na identificação do conjunto de discos presentes, uma vez que, como referido anteriormente, existem dois tamanhos diferentes de discos.

No código apresentado abaixo, verifica-se que, caso o conjunto de discos 433 tenha chegado ao ponto de montagem, o programa entra no primeiro "*while*". Por outro lado, se o conjunto 753 estiver presente, o código passa a executar o segundo "*while*". Desta forma, realiza-se a distinção no código sobre qual conjunto de discos foi recebido, indicando qual a parte do programa que o robô deverá executar: o segmento correspondente ao conjunto 433 ou o relativo ao conjunto 753.

```
#Vai ser feita a validação de qual tipo de disco é que chegou ao ponto de aperto para a continuação do código de acordo com o dito disco
while line == 1 and get_input_register_int(0) == 11 and get_digital_input(16)==0 :  # Disco 433
    set_output_register_int(0, 0) #Info para PLC etapa robo
    set_output_register_int(1, 0) #Info para PLC resultado visao
    set_output_register_int(2, 0) #Pedido aperto
    set_output_register_int(3, 777) # Info para PLC decisão recebida
    set_output_register_int(4, 0) # Info para PLC zona ocupada
    set_output_register_int(0, 111)
    traj=get_input_register_int(0)
    goto(1002) #Define a variável line a 1002 e será com esta variável que se irá controlar qual parte do codigo correr a seguir
    break


while line == 1 and get_input_register_int(0) == 12 and get_digital_input(16)==0 :  # Disco 753
    set_output_register_int(0, 0)   #Info para PLC etapa robo
    set_output_register_int(1, 0)   #Info para PLC resultado visao
    set_output_register_int(2, 0)   #Pedido aperto
    set_output_register_int(3, 777) #Info para PLC decisão recebida
    set_output_register_int(4, 0)   #Info para PLC zona ocupada
    set_output_register_int(0, 112)
    traj=get_input_register_int(0)
    goto(1102) #Define a variável line a 1102 e será com esta variável que se irá controlar qual parte do codigo correr a seguir
    break
```

Figura 139 - Identificação dos discos no ponto de montagem

Após identificar o conjunto de discos presente, o passo seguinte consiste em estabelecer a ligação com a câmara de visão e posicionar o *cobot* nas posições necessárias para iniciar a aquisição de imagens. Na Figura 140, é possível observar a tentativa de conexão com a câmara, bem como a verificação do sucesso dessa ligação.

Caso a conexão seja estabelecida com êxito, é selecionado o programa da câmara correspondente ao disco direito do conjunto identificado, conforme indicado na linha de código “job = trispector.write("set job \"433 55dir\"")”.

Posteriormente, o robô é movimentado utilizando os comandos “movj” e “movel”, previamente explicados na seção referente ao código do Cloison. Com o robô posicionado para a aquisição das imagens, é ativado o laser, através do comando “laser_on = trispector.write("set laser on")”, e é enviado um *trigger* para a câmara, “trispector.write("trigger")”, iniciando assim o processo de aquisição da imagem. Por fim, os dados adquiridos pela câmara são armazenados para posterior tratamento.

```
#Faz a conecção com a câmara de visão
camera_ip = "192.168.2.81"
trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114, state=0)
time.sleep(0.5)
if trispector.state != 1: #Verifica se a conecção foi estabelecida corretamente
    set_output_register_int(1, 20)  #Info para PLC resultado Visão (não estabelecida)
    set_output_register_int(0, 210) #Infor para PLC da etapa
    time.sleep(3)
    set_output_register_int(0, 401) #Indica que não foi efetuada corretamente a ligação com a camara ao PLC
    goto(1)
    continue

job = trispector.write("set job \"433 55dir\"") #Seleciona o programa da câmara referente ao disco 433 direito
set_output_register_int(4, 888) # Info para PLC zona ocupada

#Vai movimentar o robô para a posição para se dar ao inicio da aquisição da imagem pelo sistema de visão
movej(posj(53.85, 56.67, 59.84, -0.01, 63.49, 143.81), v=100, a=100, r=100)
#Função usada para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a velocidade "v", a aceleracao "a" e o radius "r"
movel(posx(350.0, 552.8, 91.96, 0, 180, 90), v=100, a=100)

set_output_register_int(3, 0)# Limpa a informação anterior sobre ACK
set_output_register_int(0, 210) #Etapa do robô

laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da imagem
time.sleep(0.3)
amovel(posx(350.0, 164.03, 91.96, 0, -180, 90), v=50, a=200)
trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio à aquisição da imagem
mwait(0)
set_digital_output(15, ON) #Refere que se trata de valores do disco direito
rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela câmara relativa à imagem adquirida
rxd =(rx_msg[0:128])
#Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui irá dividir-se a informação pelas virgulas criando uma array
data = rx_msg.split(', ')
valid_locate =(rx_msg[0:1]) #Vai guardar a informação se a imagem foi adquirida corretamente
time.sleep(0.2)
```

Figura 140 - Conexão do robô com a câmara e aquisição da imagem

Após a aquisição das informações pela câmara, procedeu-se ao tratamento dos dados para a criação do novo referencial (*frame*). Inicialmente, foram definidos os valores correspondentes aos pontos de referência. Caso a imagem adquirida tenha sido validada, calcula-se a diferença entre os pontos de referência pré-definidos e os pontos obtidos pela câmara.

Esta diferença é armazenada em variáveis específicas, que são posteriormente subtraídas aos valores de referência dos parafusos, permitindo assim a obtenção das novas coordenadas corrigidas dos pontos de aperto, os quais poderão ter sofrido pequenos desvios devido ao posicionamento da maquete.

Por fim, com os novos pontos calculados, é criado o novo *frame*, ajustado com base na informação recolhida pelo sistema de visão, corrigindo qualquer desvio ocorrido na colocação da maquete. Todo este processo é ilustrado na Figura 141.

```
#Valores imagem referência (valores do frame de referência criado)
refx1 = -74.745480
refy1 = 134.420716
refz1 = 213.746662
refx2 = 110.374540
refy2 = 147.874876
refz2 = 214.483098
refx3 = 28.898749
refy3 = 292.389366
refz3 = 249.882767

if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

    #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência para saber quanto se moveu a chapa da posição de referência
    x1 =  float(data[1]) - refx1
    y1 =  float(data[2]) - refy1
    z1 =  float(data[3]) - refz1
    x2 =  float(data[4]) - refx2
    y2 =  float(data[5]) - refy2
    z2 =  float(data[6]) - refz2
    x3 =  float(data[7]) - refx3
    y3 =  float(data[8]) - refy3
    z3 =  float(data[9]) - refz3
    #Arredonda o valor da variável para 6 casas decimais.
    x1 = round(x1, 6)
    y1 = round(y1, 6)
    z1 = round(z1, 6)
    x2 = round(x2, 6)
    y2 = round(y2, 6)
    z2 = round(z2, 6)
    x3 = round(x3, 6)
    y3 = round(y3, 6)
    z3 = round(z3, 6)

    #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência em arrays
    d433visao1 = [x1, y1, -z1]
    d433visao2 = [x2, y2, -z2]
    d433visao3 = [x3, y3, -z3]

#Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência em arrays
d433visao1 = [x1, y1, -z1]
d433visao2 = [x2, y2, -z2]
d433visao3 = [x3, y3, -z3]

#Define a posição exata dos 3 parafusos de referência
d433pf1=posx(352.640, 332.070, -360.540, 29.17, -177.38, 52.84)
d433pf2=posx(168.460, 317.250, -352.440, 66.04, -179.50, 126.72)
d433pf3=posx(249.22, 172.71, -322.02, 15.22, -178.21, 91.68)

#Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
d433visao1[0] = d433pf1[0] - d433visao1[0]
d433visao1[1] = d433pf1[1] - d433visao1[1]
d433visao1[2] = d433pf1[2] - d433visao1[2]
d433visao2[0] = d433pf2[0] - d433visao2[0]
d433visao2[1] = d433pf2[1] - d433visao2[1]
d433visao2[2] = d433pf2[2] - d433visao2[2]
d433visao3[0] = d433pf3[0] - d433visao3[0]
d433visao3[1] = d433pf3[1] - d433visao3[1]
d433visao3[2] = d433pf3[2] - d433visao3[2]

#Acrescenta às novas posições o Rx, Ry e Rz das posições de referência pois vão ser iguais
d433p1 = d433visao1[0:3]+d433pf1[3:]
d433p2 = d433visao2[0:3]+d433pf2[3:]
d433p3 = d433visao3[0:3]+d433pf3[3:]

#Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
pose_user106 = calc_coord(d433p1, d433p2, d433p3, ref=DR_BASE, mod=0)

#Vai alterar a pose (posição e orientação no espaço 3D) e o sistema de coordenadas de referência para o novo sistema de coordenadas calculado
overwrite_user_cart_coord(106, pose_user106, ref=DR_BASE, apply_mod=DR_TEMPORARY)
```

Figura 141 - Tratamento dos dados obtidos pelo sistema de visão

De seguida, procede-se de forma idêntica para o disco esquerdo. O robô é posicionado para a aquisição da imagem relativa a este disco, é selecionado o programa correspondente e dá-se início ao processo de aquisição da imagem.

Após a aquisição, realiza-se o tratamento dos dados obtidos pela câmara utilizando o mesmo método aplicado ao disco direito, com a única diferença na posição dos pontos de referência, uma vez que os dados agora dizem respeito ao disco esquerdo. Com este tratamento, calcula-se o novo *frame* relativo a este disco.

Desta forma, obtêm-se os novos *frames* corrigidos, tanto para o disco direito como para o disco esquerdo, que fornecerão as coordenadas exatas de cada parafuso, permitindo assim o início do processo de aparafusamento em ambos os discos.

Segue-se a seleção do *frame* correspondente e o robô é movimentado para o primeiro ponto de aproximação, relativo ao parafuso 1. Ao chegar a esta posição, são ativadas duas funções que auxiliam na inserção do bit no parafuso: a função "*compliance*", que permite ao robô sofrer pequenos desvios no movimento, comportando-se como uma mola para facilitar a inserção do bit, e a função "*force*", que aplica uma força na intensidade, eixo e direção desejados, garantindo a pressão necessária do bit sobre o parafuso para uma inserção eficaz.

Posteriormente, é chamada a função "wait_touch", que, como referido anteriormente, aguarda o contacto da aparafusadora com o parafuso (isto é, a pressão aplicada pelo "*force*" até atingir o valor definido). Em seguida, é ativada a função "wait_torque", que inicia o aperto do parafuso. A aparafusadora começa por rotacionar lentamente para garantir a correta inserção do bit no parafuso, procedendo depois ao aperto propriamente dito, até que seja enviado o relatório do processo para o robô.

Por fim, são desativadas as funções "*compliance*" e "*force*", e o robô movimenta-se para o ponto de aproximação do parafuso 2 (Figura 142).

```
set_ref_coord(108) #Seleciona o novo frame criado com os valores do sistema de visão, que irá corrigir os novos pontos de aperto das restantes porcas
set_output_register_int(3, 0) # Info para PLC decisão recebida
set_output_register_int(4, 888) # Info para PLC Zona Ocupada

#A partir deste ponto irá passar-se para a fase de aperto das porcas
movej(posj(-41.6, 64.43, 55.36, 7.32, 60.42, -69.23), v=250, a=250, r=50)
movej(posj(-43.71, 74.91, 47.92, -0.81, 56.03, -183.91), v=150, a=150, r=50)

movel(System_433e_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
mwait(0)
set_output_register_int(0, 230) #Info PLC etapa do robô
set_digital_output(13, ON)

#Define o referêncial para o referêncial da tool pois para os apertos vai se usar 3 funções para fazer o aperto
#task_compliance_ctrl - O compliance permite que o robô possa sofrer pequenos desvios no seu movimento para permitir a inserção no parafuso mais facilmente
#set_stiffnessx - Com esta função vai se definir o valor da rigidez referente ao compliance durante um determinado tempo
#set_desired_force - Com a função force, aplica-se uma força no robo com a direção pretendida
set_ref_coord(DR_TOOL)
#Aperto 1
task_compliance_ctrl() #Ativa o compliance
set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da rigidez do robô com os indicados durante 1 segundo
fd = [0, 0, 50, 0, 0, 0] #Define o valor da força que se pretende aplicar no Z
fctrl_dir= [0, 0, 1, 0, 0, 0] #Define o sentido da força, nesta caso no eixo Z positivo
set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Aplica a força no robô

delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
wait_touch() #Vai esperar pela pressão do parafuso
stop(DR_QSTOP) #Paragem rápida e controlada do movimento do robô
wait(0.1)

amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto e espera pelo relatório de final de aperto
stop(DR_QSTOP) #Paragem rápida e controlada do movimento do robô
wait(0.5)
release_force() #Desativa a força exercida no robô
wait(0.2)
delta_retract = [0.0, 0.0, -40.0, 0.0, 0.0, 0.0]

movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
release_compliance_ctrl() #Desativa o compliance
mwait(0)
set_ref_coord(108) #Seleciona de novo o frame calculado
```

Figura 142 - Aperto 1

O robô repetirá o mesmo procedimento para os restantes parafusos referentes ao disco esquerdo. Após concluir o aperto de todos os parafusos deste disco, o sistema verificará se está autorizado a iniciar os apertos relativos ao disco direito. Caso a condição seja satisfeita, o robô selecionará o *frame* corrigido correspondente ao disco do lado direito e procederá ao aperto dos seus parafusos, como pode ser observado na Figura 143.

```
#Vai verificar se pode passar ao aperto dos parafusos do disco direito
while line == 1055:
    decision=get_input_register_int(2) #Lê o sinal enviado pelo PLC sobre a possibilidade de avançar
    set_output_register_int(0, 270) #Info para PLC sobre etapa do robô
    set_output_register_int(3, 0)# Limpa a informação anterior sobre ACK

    #Vai verificar a resposta enviada pelo PLC
    if(decision==555): # 555 = Pode avançar
        set_output_register_int(3, 777) # Info para PLC decisão recebida
        time.sleep(0.1)
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        goto(1005)
        break
    if(decision==999): # 999 = Não Pode Avançar
        set_output_register_int(3, 777) # Info para PLC decisão recebida
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        movej(posj(-39.09, 32.33, 116.07, 3.3, 39.01, -70.12), v=50, a=50, r=50)
        movej(posj(-23.48, 36.31, 113.85, 16.04, 15.59, -34.75), v=80, a=50, r=50)
        movej(posj(-12.82, 38.79, 113.27, 38.46, 12.65, 34.61), v=100, a=80, r=50)
        movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        set_output_register_int(0, 402) # Info para PLC etapa robô
        time.sleep(1)
        set_digital_output(15, OFF)
        set_digital_output(16, OFF)
        set_output_register_int(1, 0) # Info para PLC resultado visao
        set_output_register_int(3, 0) # Limpa a informação anterior sobre ACK
        goto(1)
        break

while line == 1005:

    #Código relativo ao aperto do disco direito
    mwait(0)
    #Seleciona o novo frame criado com os valores do sistema de visão relativo ao disco direito, que irá corrigir os novos pontos de aperto das restantes porcas
    set_ref_coord(106)
    set_output_register_int(0, 300) # Info PLC sobre etapa robô
    set_output_register_int(3, 0)   # Limpa a informação anterior sobre ACK
    mwait(0)
    #Posições de aperto do primeiro parafuso do disco direito
    movej(posj(-39.09, 32.33, 116.07, 3.3, 39.01, -70.12), v=100, a=80, r=50)
    movej(posj(-23.48, 36.31, 113.85, 16.04, 15.59, -34.75), v=150, a=80, r=50)
    movej(posj(-12.82, 38.79, 113.27, 38.46, 12.65, 34.61), v=250, a=150, r=50)
    movej(posj(55.35, 47.13, 95.36, 6.41, 42.32, 100.11), v=150, a=150, r=50)
    movej(posj(62.2, 56.69, 105.32, 0.39, 19.34, 86.48), v=100, a=100)

    movel(System_433d_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
```

Figura 143 - Verificação da possível passagem para o disco seguinte

Os restantes apertos serão realizados de forma idêntica aos anteriormente descritos. Relativamente ao segundo robô, a lógica de funcionamento será exatamente a mesma, com a diferença de que este não estará equipado com a câmara de visão. Contudo, estará conectado à câmara e comunicará com esta para obter os dados recolhidos pelo primeiro robô, permitindo assim calcular os novos *frames* de forma idêntica ao robô 1.

Ambos os robôs manterão uma comunicação constante através de um PLC, assegurando que os apertos sejam efetuados sem risco de colisão. A estratégia consiste em que, enquanto um robô realiza os apertos do disco esquerdo, o segundo procederá aos apertos do disco direito, alternando dessa forma de modo coordenado.

O código referente ao robô 2 pode ser consultado no Anexo 4 – Código Porteur do robô 2.

4.2.5 – Implementação em chão de fábrica

Como mencionado anteriormente, ao contrário do projeto Cloison, no Porteur a disponibilidade para trabalhar na fábrica foi bastante ampla. Por esse motivo, grande parte do desenvolvimento e implementação da automação ocorreu diretamente na fábrica, o que permitiu eliminar erros de posicionamento e imprecisões que poderiam surgir ao transferir o sistema da Europneumaq para a Stellantis.

O sistema foi implementado na Stellantis, onde foram realizadas todas as programações necessárias para que, futuramente, seja possível integrá-lo plenamente na linha de montagem.

De seguida, serão apresentadas algumas imagens do sistema implementado no chão de fábrica, atualmente em funcionamento parcial, dado que ainda se encontra em fase de desenvolvimento.

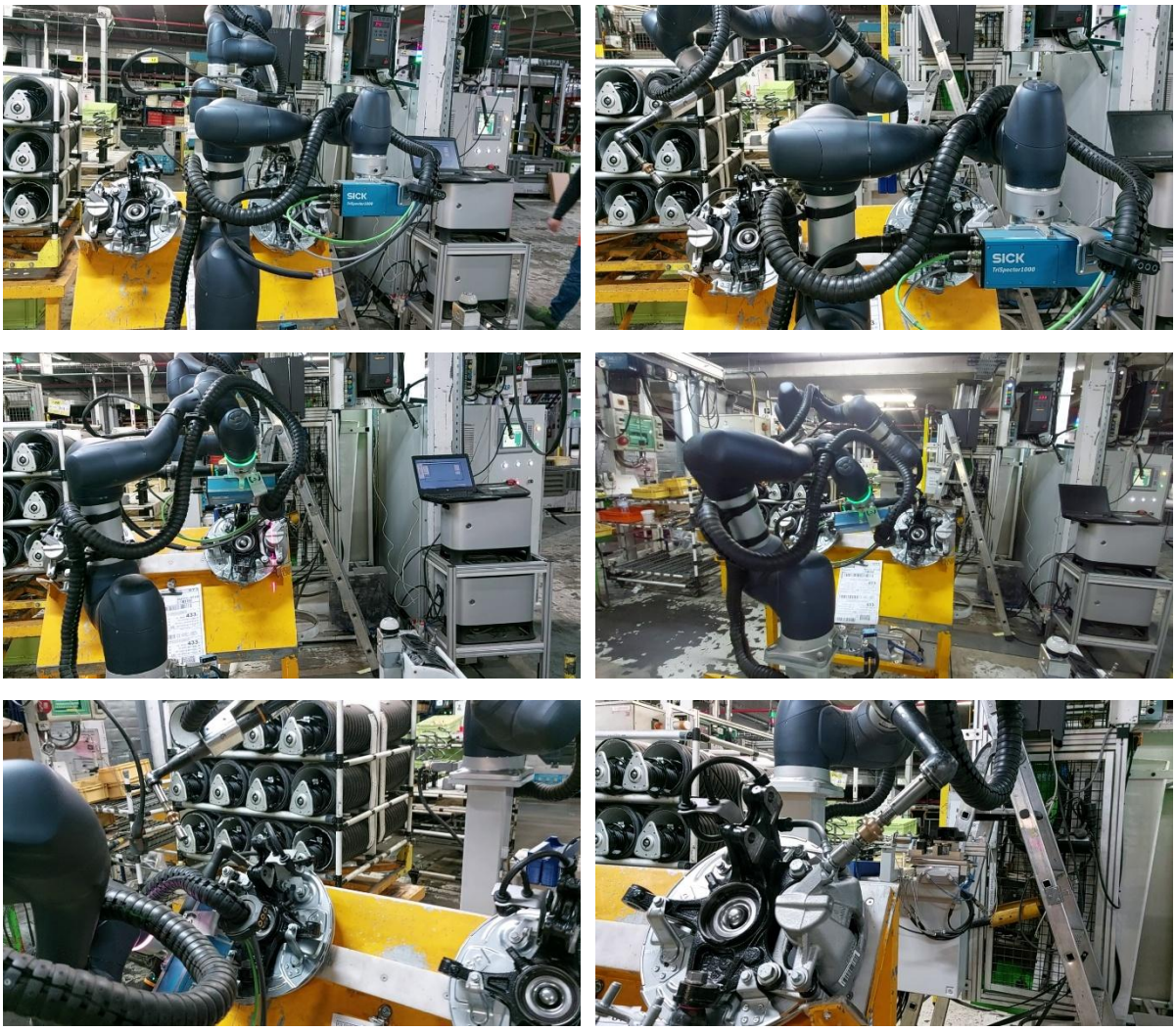


Figura 144 - Imagens da implementação do Porteur no chão de fábrica

Para complementar as imagens, disponibilizam-se dois vídeos que mostram a implementação do sistema no chão de fábrica, acessíveis através dos links a seguir:

Vídeo 1 – Robô 1: https://youtube.com/shorts/OPlr8umbcYg?feature=share

Vídeo 1 – Robô 2: https://youtu.be/I9CHxyGMG2k

CAPÍTULO 5 – CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo, serão apresentadas as conclusões finais do projeto, destacando os objetivos alcançados com a implementação da automação em dois pontos de aperto numa linha de montagem, bem como os trabalhos futuros previstos.

5.1 – Conclusões

Concluímos que a automação dos pontos de aperto numa linha de montagem, através da implementação de *cobots* para realizar tarefas tradicionalmente executadas por operadores, traz vantagens significativas em termos de eficiência, precisão e, a longo prazo, redução de custos. Os robôs conseguem executar repetidamente a mesma tarefa com uma probabilidade de erro muito reduzida, além de poderem trabalhar sem a necessidade de pausas. Para além de terem a capacidade de colaborar com humanos, potencializando ainda mais a eficiência, o que torna a sua implementação uma opção bastante atrativa.

Com este projeto, podemos afirmar que a implementação dos *cobots* no ponto de aperto Cloison garantiu uma melhoria significativa, automatizando completamente a montagem com alta precisão e uma probabilidade de erro quase nula. Além disso, a longo prazo, essa automação representa uma vantagem financeira, pois os *cobots* podem operar continuamente, sem necessidade de pausas ou de supervisão constante por parte de um operador.

No que diz respeito ao aperto do Porteur, a automação encontra-se em desenvolvimento, com avanços notáveis. Esta automação trará melhorias relevantes não só na precisão, mas também na redução do tempo e na redistribuição do trabalho, permitindo que o operador realize outras tarefas, limitando-se apenas à troca do bit quando necessário e ao posicionamento e remoção da maquete, sem precisar executar diretamente os aparafusamentos.

Quanto aos objetivos do projeto, eles estão bem encaminhados para o sucesso. Atualmente, no Cloison, é necessário apenas o supervisionamento para garantir o correto funcionamento, além de ajustes pontuais para reduzir o tempo de aperto para menos de 120 segundos, conforme o previsto. Já no Porteur, embora ainda em fase de desenvolvimento, o progresso está alinhado com os prazos estabelecidos e a conclusão está prevista para breve, uma vez que as etapas mais complexas já foram implementadas.

5.2 – Trabalhos futuros

Para o Cloison, está prevista a realização de melhorias focadas na redução do tempo de aparafusamento, com o objetivo de otimizar ainda mais a eficiência do sistema. Já para o Porteur, o principal desafio é a finalização e implementação completa do sistema de aparafusamento, que, embora esteja em fase de desenvolvimento, encontra-se dentro dos prazos estabelecidos.

Além disso, há diversas possibilidades para expandir o projeto, incluindo a automação dos restantes pontos de montagem da linha de produção que ainda operam manualmente. Outra linha de trabalho futura envolve o desenvolvimento de uma solução para a troca automática entre as chaves H10 e TORX, criando uma ferramenta capaz de realizar ambos os tipos de aperto sem intervenção manual.

Por fim, destaca-se a necessidade de desenvolver um sistema automatizado para o manuseamento da maquete, capaz de posicioná-la e retirá-la automaticamente no início e no fim dos aparafusamentos. Esta solução reduziria ainda mais a dependência da presença de um operador, tornando o processo mais eficiente e autónomo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. Benešová and J. Tupa, “Requirements for Education and Qualification of People in Industry 4.0,” Procedia Manufacturing, vol. 11, pp. 2195–2202, Jan. 2017, doi: 10.1016/j.promfg.2017.07.366.

[2] J. Kildal, A. Tellaeche, I. Fernández, and I. Maurtua, “Potential users’ key concerns and expectations for the adoption of cobots,” Procedia CIRP, vol. 72, pp. 21–26, Jan. 2018, doi: 10.1016/j.procir.2018.03.104.

[3] E. Makrini et al., “Working with Walt: How a Cobot Was Developed and Inserted on an Auto Assembly Line,” IEEE Robotics & Automation Magazine, vol. 25, no. 2, pp. 51–58, May 2018, doi: 10.1109/mra.2018.2815947.

[4] Mathur, A. Dabas, and N. Sharma, “Evolution From Industry 1.0 to Industry 5.0,” 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Dec. 2022, doi: 10.1109/icac3n56670.2022.10074274.

[5] W. Colombo et al., “A 70-Year Industrial Electronics Society Evolution Through Industrial Revolutions: The Rise and Flourishing of Information and Communication Technologies,” IEEE Industrial Electronics Magazine, vol. 15, no. 1, pp. 115–126, Jan. 2021, doi: 10.1109/mie.2020.3028058.

[6] M. Bosovska, M. Boiko, L. Bovsh, and A. Okhrimenko, “Models of the Industrial Revolution 5.0,” 2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES), pp. 1–4, Oct. 2022, doi: 10.1109/mees58014.2022.10005761.

[7] J. Leng et al., “Industry 5.0: Prospect and retrospect,” Journal of Manufacturing Systems, vol. 65, pp. 279–295, Oct. 2022, doi: 10.1016/j.jmsy.2022.09.017.

[8] R. Sakurai and J. D. Zuchi, “REVOLUÇÕES INDUSTRIAIS ATÉ A INDUSTRIA 4.0,” Revista Interface Tecnológica, vol. 15, no. 2, pp. 480–491, Dec. 2018, doi: 10.31510/infa.v15i2.386.

[9] T. S. Chu, A. B. Culaba, and J. A. C. Jose, “Robotics in the Fifth Industrial Revolution,” 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), pp. 1–6, Dec. 2022, doi: 10.1109/hnicem57413.2022.10109473.

[10] Silva, “Tudo mudando na Saúde e Segurança do Trabalho. E agora?,” Oct. 13, 2020. https://www.linkedin.com/pulse/tudo-mudando-na-sa%C3%BAde-e-seguran%C3%A7a-do-trabalho-agora-santos-silva/

[11] Mendes, J. A. & Faculdade de Letras de Coimbra. (n.d.). Industrialização e Património Industrial: Desenvolvimento e Cultura. https://www.icea.pt/wp-content/uploads/2023/07/21_10h30m_Jose_A_Mendes.pdf

[12] M. N. H. Reza, C. A. N. Malarvizhi, S. Jayashree, and M. Mohiuddin, “Industry 4.0–Technological Revolution and Sustainable Firm Performance,” 2021 Emerging Trends in Industry 4.0 (ETI 4.0), pp. 1–6, May 2021, doi: 10.1109/eti4.051663.2021.9619363.

[13] V. Alcácer and V. Cruz-Machado, “Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems,” Engineering Science and Technology an International Journal, vol. 22, no. 3, pp. 899–919, Feb. 2019, doi: 10.1016/j.jestch.2019.01.006.

[14] J. Wan, J. Yang, S. Wang, D. Li, P. Li, and M. Xia, “Cross-Network Fusion and Scheduling for Heterogeneous Networks in Smart Factory,” IEEE Transactions on

Industrial Informatics, vol. 16, no. 9, pp. 6059–6068, Nov. 2019, doi: 10.1109/tii.2019.2952669.

[15] D. P. F. Moller, H. Vakilzadian, and R. E. Haas, “From Industry 4.0 towards Industry 5.0,” 2022 IEEE International Conference on Electro Information Technology (eIT), May 2022, doi: 10.1109/eit53891.2022.9813831.

[16] S. Nahavandi, “Industry 5.0—A Human-Centric Solution,” Sustainability, vol. 11, no. 16, p. 4371, Aug. 2019, doi: 10.3390/su11164371.

[17] Roberts, Verdadeira História Da Ficção Científica, 1st ed. Editora Seoman, 2018.

[18] R. Murphy and D. D. Woods, “Beyond Asimov: The Three Laws of Responsible Robotics,” IEEE Intelligent Systems, vol. 24, no. 4, pp. 14–20, Jul. 2009, doi: 10.1109/mis.2009.69.

[19] Wikipedia contributors, “Laws of robotics,” Wikipedia, Nov. 25, 2024. https://en.wikipedia.org/wiki/Laws_of_robotics

[20] E. Apriaskar, N. Fahmizal, and M. R. Fauzi, “Robotic technology towards industry 4.0: Automatic object sorting robot arm using kinect sensor,” Journal of Physics Conference Series, vol. 1444, no. 1, p. 012030, Jan. 2020, doi: 10.1088/1742-6596/1444/1/012030.

[21] Marsh, “In 1961, the First Robot Arm Punched In,” IEEE Spectrum, Mar. 29, 2023. [Online]. Available: https://spectrum.ieee.org/unimation-robot

[22] “Unimate - The First Industrial Robot,” Automate. https://www.automate.org/robotics/engelberger/joseph-engelberger-unimate (accessed Apr. 15, 2024).

[23] Ethw, “Milestones:SHAKEY: The World’s First Mobile Intelligent Robot, 1972 - Engineering and Technology History Wiki,” ETHW, Feb. 12, 2024. https://ethw.org/Milestones:SHAKEY:_The_World%E2%80%99s_First_Mobile_Intelligent_Robot,_1972

[24] KUKA AG, “A história da KUKA,” KUKA AG, Nov. 09, 2023. https://www.kuka.com/pt-br/empresa/sobre-a-kuka/hist%C3%B3ria

[25] IFR International Federation of Robotics, “International Federation of Robotics,” IFR International Federation of Robotics. https://ifr.org/robot-history (accessed Mar. 28, 2024).

[26] L. Yao and L. Doetsch, “The Rover Challenge: Mars Rover Evolution 1970-2020,” Accu. https://www.accu.co.uk/p/145-rover-challenge-19702020 (accessed Apr. 15, 2024).

[27] “La robótica avanzada busca el mercado de masas,” Interempresas. https://www.interempresas.net/Componentes_Mecanicos/Articulos/29907-La-robotica-avanzada-busca-el-mercado-de-masas.html (accessed April. 29, 2024).

[28] Oliveira and C. Oliveira, “aeroporto de Houston,” AstroPT - Informação E Educação Científica, Nov. 20, 2010. https://www.astropt.org/2007/12/16/aeroporto-de-houston/

[29] “MECA500 | Six-Axis Industrial Robot ArM | Electromate Inc,” Electromate Inc. https://www.electromate.com/meca500/ (accessed May. 03, 2024).

[30] “Doosan Robotics M1013 - Unchained Robotics,” Unchained Robotics. https://unchainedrobotics.de/en/products/robot/cobot/doosan-robotics-m1013 (accessed May. 03, 2024).

[31] “Atlas | Boston Dynamics,” Boston Dynamics, May. 10, 2024. https://bostondynamics.com/atlas/

[32] jeffbullas.com, “7 of the World’s Most Famous AI Robots – A Glimpse into the Not-so-Distant Future,” jeffbullas.com, May. 11, 2024. https://www.jeffbullas.com/ai-robot/

[33] S. R. A. Inc, “Pepper.” https://us.softbankrobotics.com/pepper (accessed May. 11, 2024).

[34] “Moxi — Diligent Robotics,” Diligent Robotics. https://www.diligentrobots.com/moxi (accessed May. 11, 2024).

[35] R. Team, “Da Vinci,” ROBOTS: Your Guide to the World of Robotics, May 31, 2024. https://robotsguide.com/robots/davinci

[36] ROBOTS: Your Guide to the World of Robotics, “ROBOTS: Your Guide to the World of Robotics,” ROBOTS: Your Guide to the World of Robotics. https://robotsguide.com/ (accessed May. 11, 2024).

[37] IFR International Federation of Robotics, “International Federation of Robotics,” IFR International Federation of Robotics. https://ifr.org/standardisation (accessed May. 19, 2024).

[38] X. Gong, M. Li, Z. Zhao, and D. Cui, “Research on industrial Robot system security based on Industrial Internet Platform,” 2022 7th IEEE International Conference on Data Science in Cyberspace (DSC), vol. 116, pp. 214–218, Jul. 2022, doi: 10.1109/dsc55868.2022.00035.

[39] “Safety | MMCI Robotic Palletizers.” https://www.mmci-automation.com/palletizing-safety.html (accessed May. 25, 2024).

[40] OSHA, “Industrial Robot Systems and Industrial Robot System Safety,” Occupational Safety and Health Administration. https://www.osha.gov/otm/section-4-safety-hazards/chapter-4 (accessed Jun. 05, 2024).

[41] MFGRobots, “Preço do braço robótico industrial:Quanto custa um braço robótico?,” MFGRobots. https://pt.mfgrobots.com/equipment/robot/1004016467.html (accessed Jun. 06, 2024).

[42] KUKA AG, “KUKA KR C4: The Power of Control,” KUKA AG, Jun. 08, 2021. https://www.kuka.com/pt-br/produtos-servi%C3%A7os/sistemas-de-rob%C3%B4/unidades-de-comando-de-rob%C3%B4/kr-c4

[43] R. Dot, “Red Dot Design Award: KUKA smartPAD-2,” Red Dot, Dec. 19, 2024. https://www.red-dot.org/project/kuka-smartpad-2-24200-24199

[44] IFR International Federation of Robotics, “International Federation of Robotics,” IFR International Federation of Robotics. https://ifr.org/industrial-robots (accessed Jun. 20, 2024).

[45] “Auto Robotic System Service,” indiamart.com. https://www.indiamart.com/proddetail/auto-robotic-system-service-2851827077797.html (accessed Jun. 21, 2024).

[46] “Robô SCARA SR-6iA/C - Fanuc.” https://www.fanuc.eu/pt/pt/rob%c3%b4s/p%c3%a1gina-filtro-rob%c3%b4s/scara-series/sr-6ia-c (accessed Jun. 21, 2024).

[47] “Robôs automatizam a produção de compota - Fanuc.” https://www.fanuc.eu/pt/pt/casos-de-clientes/hero (accessed Jun. 21, 2024).

[48] “MPP3H,” Roboplan - Robotics Experts. https://www.roboplan.pt/pt/produtos/robot-industrial-yaskawa/robot-yaskawa-mpp3 (accessed Jun. 22, 2024).

[49] Omron Europe B.V., “Quattro,” Omron Europe B.V. https://industrial.omron.pt/pt/products/quattro (accessed Jun. 22, 2024).

[50] Evs, “Robôs Industriais – O guia definitivo,” EVS Robot, Dec. 27, 2023. https://www.evsint.com/pt/guide-to-industrial-robots/

[51] Innovationsystems, “Προϊόντα | Innovation Systems,” Innovation Systems, Mar. 20, 2023. https://innovationsystems.gr/en/products/

[52] K. T. Saharia, “BENEFITS AND PROBLEMS OF INDUSTRIAL ROBOTICS: A CASE STUDY,” Экономика И Социум, 2023, [Online]. Available: https://www.researchgate.net/publication/368690868

[53] “Top 12 Industrial Robot Applications and Uses,” HowToRobot. https://howtorobot.com/expert-insight/industrial-robot-applications (accessed Jul. 11, 2024).

[54] “Connecting Industrial Robot Suppliers & Buyers – HowToRobot.” https://howtorobot.com/ (accessed Jul. 14, 2024).

[55] Akyüz and A. Akyüz, “Top Robotic Manufacturing Companies 2024 - ESSERT Robotics,” ESSERT Robotics - The future of robot automation., Mar. 20, 2024. https://www.essert.com/blog/robotics/robotic-manufacturing-companies-2024/

[56] Machine Tools Network, “Fanuc - Machine Tools Network,” Machine Tools Network, Sep. 11, 2024. https://www.machinetoolsnetwork.co.za/directory/fanuc/

[57] “ABB Logo - Unlimited Download. cleanpng.com.,” cleanpng.com. https://www.cleanpng.com/png-abb-group-baldor-electric-company-manufacturing-lo-1854236/ (accessed Jul. 15, 2024).

[58] “KUKA Brand Logo - Unlimited Download. cleanpng.com.,” cleanpng.com. https://www.cleanpng.com/png-kuka-systems-robot-logo-industry-fast-robotics-fas-6340745/ (accessed Jul. 15, 2024).

[59] File:Stäubli International logo.svg - Wikimedia Commons. [Online]. Available: https://commons.wikimedia.org/wiki/File:St%C3%A4ubli_International_logo.svg (accessed Jul. 15, 2024).

[60] Wikipedia contributors, “File:Denso logo.svg - Wikipedia.” https://en.m.wikipedia.org/wiki/File:Denso_logo.svg (accessed Jul. 15, 2024).

[61] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, “Recent progress on programming methods for industrial robots,” Robotics and Computer-Integrated Manufacturing, vol. 28, no. 2, pp. 87–94, Aug. 2011, doi: 10.1016/j.rcim.2011.08.004.

[62] “Simulação de robô 3D inteligente offline com o ROBOGUIDE - Fanuc.” https://www.fanuc.eu/pt/pt/rob%c3%b4s/acess%c3%b3rios/roboguide (accessed Jul. 21, 2024).

[63] “RobotStudio.” https://webshop.robotics.abb.com/br/catalog/product/view/id/24/s/robot-studio/ (accessed Jul. 22, 2024).

[64] KUKA AG, “KUKA.Sim,” KUKA AG, Nov. 21, 2023. https://www.kuka.com/pt-pt/produtos-servi%C3%A7os/sistemas-de-rob%C3%B4/software/planejamento-proje%C3%A7%C3%A3o-service-seguran%C3%A7a/kuka,-d-,sim

[65] “Stäubli Robotics Suite.” https://www.staubli.com/hk/en/robotics/products/robot-software/staeubli-robotics-suite.html (accessed Jul. 29, 2024).

[66] DENSO Robotics, “WINCAPS III Offline Software | DENSO Robotics,” DENSO Robotics, May 07, 2021. https://www.densorobotics.com/products/software/wincaps/

[67] “Safety Features of Robotic Automation,” Midwest Engineered Systems, Jun. 27, 2023. https://www.mwes.com/safety-features-of-robotic-automation/

[68] F. Dozio, “Barreiras de protecção para robôs industriais,” Satech Machine Guards, Sep. 26, 2023. https://www.satech.it/pt/barreiras-proteccao-robos-industriais/

[69] "Cortinas de Luz de Segurança," Weg. https://www.weg.net/catalog/weg/PT/pt/Seguran%C3%A7a-de-M%C3%A1quinas%2C-Sensores-Industriais-e-Fontes-de-Alimenta%C3%A7%C3%A3o/Seguran%C3%A7a-de-M%C3%A1quinas/Cortinas-de-Luz-de-Seguran%C3%A7a/Cortinas-de-Luz-de-Seguran%C3%A7a/p/MKT_WDC_GLOBAL_SAFETY_LINE_LIGHT_CURTAINS (accessed Jul. 25, 2024).

[70] Alexi, "Robotics in Manufacturing: Benefits Include Enhanced Safety and Profits," Jun. 17, 2024. https://www.linkedin.com/pulse/robotics-manufacturing-benefits-include-enhanced-safety-charles-alexi-tofoe/

[71] ODBO design+web, "MavSyrena Sinalizadores," MavSyrena Sinalizadores. https://mavsyrena.com.br/ (accessed Jul. 28, 2024).

[72] "Schneider Electric XALK178G interruptor elétrico Pushbutton switch Vermelho, Amarelo," Fnac. https://www.fnac.pt/mp17720877/Schneider-Electric-XALK178G-interruptor-eletrico-Pushbutton-switch-Vermelho-Amarelo (accessed Jun. 28, 2024).

[73] H. Afsarmanesh, L. M. Camarinha-Matos, and A. L. Soares, Collaboration in a Hyperconnected World. 2016. doi: 10.1007/978-3-319-45390-3.

[74] G. J. De Vries, E. Gentile, S. Miroudot, and K. M. Wacker, "The rise of robots and the fall of routine jobs," Labour Economics, vol. 66, p. 101885, Jul. 2020, doi: 10.1016/j.labeco.2020.101885.

[75] J. T. C. Tan and T. Inamura, "Integration of work sequence and embodied interaction for collaborative work based human-robot interaction," 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 239–240, Mar. 2013, doi: 10.1109/hri.2013.6483590.

[76] G. Charalambous, S. Fletcher, and P. Webb, "Development of a Human Factors Roadmap for the Successful Implementation of Industrial Human-Robot Collaboration," in Advances in intelligent systems and computing, 2016, pp. 195–206. doi: 10.1007/978-3-319-41697-7_18.

[77] L. Marques, R. S. Carrijo, A. S. D. Morais, and FEELT – Universidade Federal de Uberlândia, "ROBÓTICA COLABORATIVA: IMPORTÂNCIA E DESAFIOS," FEELT – Universidade Federal De Uberlândia. (accessed Sep. 10, 2024).

[78] M. Bender, M. Braun, P. Rally, and Fraunhofer Institute for Industrial Engineering IAO, Lightweight robots in manual assembly – best to start simply! Examining companies' initial experiences with lightweight robots. Fraunhofer Institute for Industrial Engineering IAO, 2016, pp. 3–7. (accessed Sep. 11, 2024).

[79] S. Stone and S. Stone, "Cobot Palletizers: Vacuum Grippers vs. Others | Cisco-Eagle," Warehousing Insights | Material Handling Systems, Jun. 09, 2023. https://www.cisco-eagle.com/blog/2022/08/18/vacuum-grippers-and-lightweight-palletizers/

[80] Cobots, "Precision Gripper/Caliper - Cobots," Cobots - Collaborative Robots, Feb. 10, 2023. https://www.cobots.co.za/product/precision-gripper-caliper/

[81] "qb SoftHand Research." https://www.universal-robots.com/marketplace/products/01tP40000071NYaIAM/ (accessed Sep. 17, 2024).

[82] Iker, "Serie H: el cobot más potente y seguro del mercado," Duplostock, Jul. 27, 2023. https://duplostock.com/serie-h-cobot-mas-potente-seguro-mercado/

[83] Heggem, N. M. Wahl, and L. Tingelstad, "Configuration and Control of KMR iiwa Mobile Robots using ROS2," 2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS), pp. 1–6, Jun. 2020, doi: 10.1109/sims49386.2020.9121554.

[84] IFR International Federation of Robotics, “IFR publishes collaborative industrial robot definition and estimates supply,” IFR International Federation of Robotics. https://ifr.org/post/international-federation-of-robotics-publishes-collaborative-industrial-rob (accessed Sep. 17, 2024).

[85] “Continental ES la Empresa de Automoción más Robotizada | UR.” https://www.universal-robots.com/es/casos-practicos/continental/ (accessed Sep. 20, 2024).

[86] “Cobot UR10 para la Industria de Automoción | Universal Robots.” https://www.universal-robots.com/es/casos-practicos/nissan-motor-company/ (accessed Sep. 20, 2024).

[87] “Robots para la Clasificación de las Muestras de Sangre | UR.” https://www.universal-robots.com/es/casos-practicos/hospital-de-gentofte/ (accessed Sep. 25, 2024).

[88] “Brazo Robótico UR5 Fácil de Manejar y Programar | U. Robots.” https://www.universal-robots.com/es/casos-practicos/nordic-sugar/ (accessed Sep. 25, 2024).

[89] “What Are Collaborative Robots?,” Automate. https://www.automate.org/robotics/cobots/what-are-collaborative-robots (accessed Sep. 25, 2024).

[90] “Universal Robots Company Summary.” https://www.abiresearch.com/companies/universal-robots/ (accessed Sep. 17, 2024).

[91] Contribuidores da Wikipédia, “Ficheiro:OMRON Logo.svg – Wikipédia, a enciclopédia livre.” https://pt.m.wikipedia.org/wiki/Ficheiro:OMRON_Logo.svg (accessed Oct. 7, 2024).

[92] “File:Doosan Group and Corporation - Logo.svg - Wikipedia,” Feb. 27, 2023. https://so.wikipedia.org/wiki/File:Doosan_Group_and_Corporation_-_Logo.svg

[93] “DOOSAN ROBOTICS COBOT FOR MANUFACTURING | Ellison Technologies.” https://www.ellisontechnologies.com/doosan-robotics-cobot (accessed Oct. 13, 2024).

[94] “Basic Guide - RoboDK Documentation.” https://robodk.com/doc/en/Basic-Guide.html (accessed Oct. 13, 2024).

[95] Bresimar Automação, S.A., “Gripper de Pinças RG6 - Bresimar Automação, S.A.,” Bresimar Automação, S.A., Jan. 05, 2024. https://bresimar.pt/produto/gripper-de-pincas-rg6/

[96] “ISO 10218-1:2011,” ISO. https://www.iso.org/standard/51330.html (accessed Oct. 18, 2024).

[97] “ISO/TS 15066:2016,” ISO. https://www.iso.org/standard/62996.html (accessed Oct. 18, 2024).

[98] “ISO 12100:2010,” ISO. https://www.iso.org/standard/51528.html (accessed Oct. 18, 2024).

[99] Platbrood, O. Görnemann, and SICK AG, “SAFE ROBOTICS – A SEGURANÇA EM SISTEMAS ROBÓTICOS COLABORATIVOS,” Jun. 2018.

[100] “Vision-Based Mobile Collaborative Robot Incorporating a Multicamera Localization System,” IEEE Journals & Magazine | IEEE Xplore, Sep. 15, 2023. https://ieeexplore.ieee.org/document/10210569

[101] R. Nogueira, J. Reis, R. Pinto, and G. Goncalves, “Self-adaptive Cobots in Cyber-Physical Production Systems,” 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 521–528, Sep. 2019, doi: 10.1109/etfa.2019.8869165.

[102] M. U. Anjum, U. S. Khan, W. S. Qureshi, A. Hamza, and W. A. Khan, “Vision-Based Hybrid Detection For Pick And Place Application In Robotic Manipulators,” 2023 International Conference on Robotics and Automation in Industry (ICRAI), Mar. 2023, doi: 10.1109/icrai57502.2023.10089602.

[103] X. Yang, Z. Zhou, J. H. Sørensen, C. B. Christensen, M. Ünalan, and X. Zhang, “Automation of SME production with a Cobot system powered by learning-based vision,” Robotics and Computer-Integrated Manufacturing, vol. 83, p. 102564, Mar. 2023, doi: 10.1016/j.rcim.2023.102564.

[104] M. Gautam, H. Fagerlund, B. Greicevci, F. Christophe, and J. Havula, “Collaborative Robotics in Construction: A Test Case on Screwing Gypsum Boards on Ceiling,” 2020 5th International Conference on Green Technology and Sustainable Development (GTSD), Nov. 2020, doi: 10.1109/gtsd50082.2020.9303061.

[105] Omron Europe B.V., “TM,” Omron Europe B.V. https://industrial.omron.eu/en/products/tm-cobot (accessed Nov. 17, 2024).

[106] “두산로보틱스.” https://www.doosanrobotics.com/de/products/series/m1013 (accessed Nov. 19, 2024).

[107] “두산로보틱스.” https://www.doosanrobotics.com/de/products/series/m1509 (accessed Nov. 19, 2024).

[108] “UR10e Medium-sized, versatile cobot.” https://www.universal-robots.com/products/ur10e/ (accessed Nov. 19, 2024).

[109] “UR16e Heavy-duty, compact cobot.” https://www.universal-robots.com/products/ur16e/ (accessed Nov. 19, 2024).

[110] Qviro, “Leading Robotics Marketplace and Community,” QVIRO, Nov. 21, 2024. https://qviro.com/

[111] “ETD STR31-10-10-T25 | Atlas Copco.” https://www.atlascopco.com/pt-pt/itba/products/assembly-solutions/electric-assembly-tools/etd-str31-10-10-t25-sku8436623011 (accessed Nov. 20, 2024).

[112] Photoneo, “PhoXi 3D Scanner S,” Photoneo Focused on 3D, May 02, 2024. https://www.photoneo.com/products/phoxi-scan-s/

[113] SICK, “Product data sheet,” Nov. 2024.

[114] “SICK | Sensor Intelligence,” SICK. https://www.sick.com/us/en/catalog/products/machine-vision-and-identification/machine-vision/trispector1000/v3t12s-mr32a7/p/p448147?tab=detail (accessed Nov. 25, 2024).

[115] “M1013,” DoosanRobots. https://doosanrobots.ru/catalog/m-series/m1013.html (accessed Nov. 25, 2024).

[116] “Solidworks Logo Vector Logo - Download Free SVG Icon | Worldvectorlogo.” https://worldvectorlogo.com/pt/logo/solidworks-logo-1 (accessed Nov. 27, 2024).

[117] RoboDK, “Simulator for industrial robots and offline programming - RoboDK,” RoboDK. https://robodk.com/ (accessed Nov. 27, 2024).

[118] Owen-Hill, “RoboDK blog,” RoboDK Blog, Dec. 16, 2024. https://robodk.com/blog/

[119] “8433202392 ETV ST61-30-10.” https://servaid.atlascopco.com/AssertWeb/en-US/AtlasCopco/Catalogue/4386 (accessed Mar. 11, 2025).

ANEXOS

Anexo 1 – Código Cloison do robô do lado direito

```
#Importa as bibliotecas necessárias e utilizadas pelo código
from DRCF import*
import numpy as np
import sys
import os
import time
import socket
import math



class DoosanSick:

#*********************************

    #Função que inicializa e faz a conexão da câmara
    def __init__(self, ip="192.168.1.61", portIN=2115, portOUT=2114):

        #Guarda os parâmetros
        self.ip = ip
        self.portIN = portIN
        self.portOUT = portOUT

        #Cria um socket TCP básico e tenta conectar-se à câmara para testar disponibilidade
        client = socket.socket()
        client.settimeout(3) #timeout de 3 segundos para nao parar o programa
        socket.setdefaulttimeout(3)
        client.connect((self.ip, self.portIN))

        #Usa a função oficial da Doosan para abrir um canal socket de comunicação de entrada
        #(Receber dados da câmara)
        try:
            self._socketIN = client_socket_open(self.ip, self.portIN)
        except Exception as e: #Caso falhe irá alertar
            tp_popup("Socket INPUT falha de ligação. Error: {0}".format(
                str(e)), DR_PM_ALARM)
            raise e

        time.sleep(0.1)

        #Vai abrir outro canal socket de comunicação mas neste caso de saída
        #(enviar comandos para a câmara)
        try:
            self._socketOUT = client_socket_open(self.ip, self.portOUT)
        except Exception as e: #Caso falhe irá alertar
            tp_popup("Socket OUTPUT falha de ligação. Error: {0}".format(
                str(e)), DR_PM_ALARM)
            raise e

#*********************************

    #Função usada para escrever no socket (Comunicação Robô para Câmara)
    def write(self, cmd, socket=None):

        #Vai usar o socket padrão caso o mesmo não seja especificado
        if socket == None:
            socket = self._socketIN

        #Converte o comando de string para bytes em ASCII, necessário para transmissão binária via
socket
        cmd = bytes(cmd, encoding="ascii")

        #A informação é enviada da seguinte forma (padrão deste dispositivo) "STX - res - ETX"
        STX = client_socket_write(socket, b"\x02") #STX (start of text)
        res = client_socket_write(socket, cmd)     #Envia o comando pretendido
```

```
        ETX = client_socket_write(socket, b"\x03") ##TX (End of Text)
        time.sleep(0.1)
        res, rx_data = client_socket_read(socket, 10, 1) #Vai ler até 10 bytes com um timeout de
1 seg
        #Caso haja uma resposta válida, vai converter os dados recebidos de bytes para string
        rxd=None
        if res > 1:
            rx_msg = rx_data.decode() #Converte os dados
            rxd =(rx_msg[1:3]) # Extrai 2 caracteres do meio da resposta
        #Vai registar logs de possiveis erros
        if res == -1:
            tp_log("Erro durante a escrita no socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a escrita no socket: Erro de Socket")
        return rxd

#*********************************

    #Função usada para ler a informação do command channel da câmara (Mensagens de verificação)
    def readcmd(self, length, timeout, socket=None ):

        #Vai usar o socket padrão caso o mesmo não seja especificado
        if socket == None:
           socket = self._socketIN

        #Lê os dados do socket
        res, rx_data = client_socket_read(socket, length, timeout)
        rx_msg = rx_data.decode() #Converte de bytes para string
        rxd =(rx_msg[1:3])        #Extrai a informação da posicao 1 e 2 (Código do estado)

        #Vai registar logs de possiveis erros
        if res == -1:
            tp_log("Erro durante a leitura do socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a leitura do socket: Erro de Socket")
        elif res == -3:
            tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
        return rxd

#*********************************

    #Função usada para ler do socket a informação enviada pela câmara (posições dos parafusos)
    def readata(self, length, timeout, socket=None ):

        #Vai usar o socket padrão caso o mesmo não seja especificado
        if socket == None:
           socket = self._socketOUT

        #Inicializa variáveis e lê os dados do socket
        res = 0
        rx_data = 0
        res, rx_data = client_socket_read(socket, length, timeout)

        #Vai registar logs de possiveis erros
        if res == -1:
            tp_log("Erro durante a leitura do socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a leitura do socket: Erro de Socket")
        elif res == -3:
            tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
        else:
            rx_msg = rx_data.decode() #Converte os dados de bytes para string
            return rx_msg #Retorna a informação enviada pela câmara

#*********************************

    #Função que irá fechar as conexões dos socket IN e OUT utilizados para efetuar a comunicação
com a câmara
    def closeIN(self, socket=None):

        if socket == None:
            socket = self._socketIN
```

```
        client_socket_close(socket)

    def closeOUT(self, socket=None):

        if socket == None:
            socket = self._socketOUT

        client_socket_close(socket)

#Função que irá esperar que o torque pretendido seja atingido para desativar a aparafusadora
def wait_torch() :
                yy = True
                zz = 0
                while yy:
                    step_ack=get_input_register_int(1) #Guarda na variavel step_ack o valor do
relatório
                    if(step_ack==1000) or (step_ack==2000): #Só entra neste if quando recebe o
relatório, seja OK ou NOK
                        set_output_register_int(2, 0) #Output para desativar aparafusadora
                        time.sleep(0.1)
                        yy = False

                    #TimeOut de 35 segundos para caso não receber resposta da aparafusadora, o
loop termina e a ferramenta é desligada por segurança.
                    if zz >= 35:
                        yy = False
                    time.sleep(0.1)
                    zz = zz + 0.1

#*************************************************************************

#Definição de algumas variáveis globais e outputs importantes
set_motion_end(DR_CHECK_ON)
set_output_register_int(0, 0)
set_output_register_int(1, 0)
set_digital_output(6, OFF)
set_digital_output(8, OFF)
set_tool ("U_Tool_W1")  #Seleciona a tool que se pretende
set_tcp("U_Tool_2")     #Seleciona o TCP da respetiva tool
set_ref_coord(DR_BASE)  #Define o referêncial de coordenadas para movimentar o robô
time.sleep(0.1)

if get_digital_input(16)==0: #Segurança
 exit()

#Definição de uma variável global que será usada posteriormente para indicar para onde irá a
execução do programa
def goto(linenum):
    global line
    line = linenum
line = 1

#Dá set a 0 os seguintes outputs
set_output_register_int(0, 0) #Info para PLC etapa do Robô
set_output_register_int(1, 0) #Info para PLC resultado Visão
set_output_register_int(2, 0) #Pedido aperto
set_output_register_int(3, 0) #Info para PLC decisão recebida
set_output_register_int(4, 0) #Info para PLC zona ocupada

time.sleep(0.1)

if get_digital_input(16)==0: #Segurança
 exit()

time.sleep(0.1)
move_home(DR_HOME_TARGET_USER)

set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)

#Posição de carregamento de porcas
carregamento = posj(39.8, -15.16, -101.75, -72.25, 53.21, -121.47)
```

```
#Inicio do programa de aperto
while True:

    time.sleep(0.1)

    #Uma vez que existem duas carrinhas diferentes, uma com jante 15 e outra com jante 16 isso irá interferir nas posição das posições de aperto,
    #logo aqui é feita a validação de qual das carrinhas chegou à linha e o programa irá correr de acordo com a carrinha respetiva
    #get_input_register_int(0) == 10 = Jante 15
    if line == 1 and get_input_register_int(0) == 10 and get_digital_input(16)==1 : #Jante 15
        set_output_register_int(0, 0) #Info para PLC etapa do Robô
        set_output_register_int(1, 0) #Info para PLC resultado Visão
        set_output_register_int(2, 0) #Pedido aperto
        set_output_register_int(3, 0) #Info para PLC que a decisão foi recebida (ACK)
        set_output_register_int(4, 0) #Info para PLC zona ocupada

        #Define o IP da câmara e estabelece comunicação com a mesma
        camera_ip = "192.168.1.61"
        trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114)
        time.sleep(0.5)
        job = trispector.write("set job \"J 15\"") #Envia comando para a câmara para seleciona o programa relativo à jante 15
        set_output_register_int(0, 100) #Define a etapa do robô para 100 (Programa 15 selecionado)

        goto(1002) #Faz com que o robô avance para a line 1002, é com base neste função que correrá o código
        continue

    if line == 1 and get_input_register_int(0) == 11 and get_digital_input(16)==1 : #get_input_register_int(0) == 11 = Jante 16
        set_output_register_int(0, 0) #Info para PLC etapa do Robô
        set_output_register_int(1, 0) #Info para PLC resultado Visão
        set_output_register_int(2, 0) #Pedido aperto
        set_output_register_int(3, 0) #Info para PLC decisão recebida (ACK)
        set_output_register_int(4, 0) #Info para PLC zona ocupada

        #Define o IP da câmara e estabelece comunicação com a mesma
        camera_ip = "192.168.1.61"
        trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114) #Faz a conecção com a câmara
        time.sleep(0.5)
        job = trispector.write("set job \"J 16\"") #Envia comando para a câmara para seleciona o programa relativo à jante 16
        set_output_register_int(0, 101) #Define a etapa do robô para 100 (Programa 16 selecionado)

        goto(1102) #Faz com que o robô avance para a line 1102
        continue

    #Vai verificar se o sistema está fora de serviço (algum periférico não está pronto, ou carrinha não chegou ao local)
    if line == 1 and get_input_register_int(0) == 99 and get_digital_input(16)==1 :
        set_output_register_int(0, 0) #Info para PLC etapa do Robô
        set_output_register_int(1, 0) #Info para PLC resultado Visão
        time.sleep(0.5)
        set_output_register_int(0, 999) #Indica que está fora de serviço
        goto(99)
        continue

    if line == 99:
        set_output_register_int(0, 999)
        #Irá verificar se ainda permanece fora de serviço, se sim fica aqui até deixar de estar
        outserv=get_input_register_int(0)
        if(outserv==0): #Se não tiver fora de serviço volta às condições de cima
            time.sleep(0.2)
            set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
            time.sleep(1)
            set_output_register_int(0, 0)
            goto(1)
            continue
        time.sleep(0.1)
```

```
        goto(99)
        continue

#########################################################################################################################################################
    #Como dito existem duas carrinhas diferentes então o código está dividido em dois sendo uma parte para uma carrinha e outra para a outra carrinha
    #Aqui está o código para a carrinha com a jante 15, caso a carrinha que chegue ao ponto de montagem contenha a jante relativa à jante 15 correrá esta parte do código caso contrario irá saltar esta parte e executar a parte relativa
    #à segunda parte do código

    if line == 1002:

        set_tcp("U_Tool_1") #Seleciona o TCP da tool
        set_ref_coord(DR_BASE) #Define o eixo de coordenadas do robô

        set_output_register_int(0, 100) #Info PLC etapa do robô

        #Movimento de Saída Cuba
        #Função usada para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a velocidade "v", a aceleracao "a" e o radius "r"
        movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=200, a=120, r=50)
        set_output_register_int(0, 105)  #Info PLC etapa do robô
        movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=200, a=120, r=50)
        movej(posj(93.1, 37.24, -131.84, -4.77, 97.09, -94.91), v=200, a=100, r=10)
        goto(1003)
        continue

    if line == 1003:
        set_output_register_int(0, 110)
        free_zone=get_input_register_int(4) #Verifica se a zona de aperto está livre, caso esteja livre avança senão espera que a mesma esteja
        if(free_zone==222):
            #time.sleep(0.1)
            set_output_register_int(4, 888) #Envia a informacao para o PLC que a zona irá ficar ocupada pelo robô
            goto(1004)
            continue
        else:
            time.sleep(0.1)
            goto(1003)
            continue

    if line == 1004:
        set_output_register_int(0, 120)
        car_pos=get_input_register_int(2)
        if(car_pos==5555): #Verifica se a carrinha está na posição correta
            set_output_register_int(3, 777)
            time.sleep(0.01)
            goto(1044)
            continue

        #Caso a informação que chega ao robô indique que a carrinha não está na posição correta o robô voltará à posição inicial (carrrgamento) e o código voltará à parte da verificação inicial da posição da carrinha
        if(car_pos==9999):
            set_output_register_int(3, 777)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50) #Porta
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(posj(37.8, -15.37, -103.01, -70.17, 59.06, -123.26), v=20, a=10)
            movej(carregamento, v=20, a=10)
            set_output_register_int(3, 777)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
            time.sleep(2)
            goto(1)
            continue
```

```
    if line == 1044: #Se a carrinha estiver na posição correta o programa prossegue para a aquisição da imagem pelo sistema de visão

        #Entrada no carro
        movej(posj(163.68, 29.35, -129.86, -12.54, 103.8, -69.28), v=220, a=180, r=50)
        movej(posj(184.55, -13.96, -105.01, -17.47, 116.57, -95.72), v=220, a=180, r=100)

        #Posição para se dar ao inicio da aquisição da imagem pelo sistema de visão
        movej(posj(189.46, -53.72, -35.29, 0, 89.01, -9.46), v=220, a=120, r=0)
        mwait(0)
        set_tcp("U_Tool_1") #Seleciona o TCP da tool
        movel(posx(1050.00, 140.0, 650.0, 90, 0, 90), v=120, a=150)
        time.sleep(0.05)
        goto(10444)
        continue

    #Vai tentar ativar o laser e selecionar o job pretendido, caso falhe em algumas das etapas o robô feha os sockets, informa o PLC e volta à posicao inicial
    if line == 10444:
        set_output_register_int(3, 0) #Info para PLC decisão recebida (reset)
        laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da imagem
        #Verifica se o laser foi ativado, caso não tenha sido o robô abortará a tarefa de aquisição da imagem e voltará para a posição inicial
        if (laser_on != 'OK'):
            trispector.closeIN(None) #Fecha os sockets
            trispector.closeOUT(None)
            set_output_register_int(1, 11) #Info para PLC sobre erro ocorrido no sistema de visão
            #Vai se mover para a posição de carregamento (posição incial)
            movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
            movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
            movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
            goto(1)
            continue

        job = trispector.write("set job \"J 15\"") #Seleciona o programa "J 15" para ser utilizado na câmara
        #Verifica se o programa foi selecionado corretamente caso contrário aborta e voltará ao inicio
        if (job != 'OK'):
            trispector.closeIN(None) #Fecha os sockets
            trispector.closeOUT(None)
            set_output_register_int(1, 11) #Info para PLC sobre erro ocorrido no sistema de visão
            #Vai se mover para a posição de carregamento (posição incial)
            movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
            movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
            movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
            goto(1)
            continue

        #Caso o laser a seleção do programa tenham sido executados com sucesso irá passar-se à aquisição da imagem
        time.sleep(0.3)
        amovel(posx(688.00, 140.0, 650.0, 90.00, 00.00, 90.00), v=50, a=200) #Posição de inicio de aquisicao da imagem
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio à aquisição da imagem
        mwait(0) #Aguarda o fim completo do movimento (amovel) antes de prosseguir
```

```
        rx_msg = trispector.readata(128, 6) #Lê e guarda a informação enviada pela câmara relativa à imagem adquirida (128 caracteres com timeout de 6 seg)
        rxd =(rx_msg[0:128])
        data = rx_msg.split(', ') #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui irá dividir-se a informação pelas virgulas criando uma array
        valid_locate =(rx_msg[0:1]) #Vai guardar o primeiro caracter que contem a informação se a imagem foi adquirida corretamente

        #Valores imagem referência esperada para comparar com a adquirida (valores do frame de referência)
        refx1 = 87.656254
        refy1 = 72.368235
        refz1 = 207.349081
        refx2 = 82.712757
        refy2 = 285.062823
        refz2 = 207.748271
        refx3 = -76.414542
        refy3 = 65.350911
        refz3 = 246.823760

        if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência para saber quanto se moveu a chapa da posição de referência
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x3 =  float(data[7]) - refx3
            y3 =  float(data[8]) - refy3
            z3 =  float(data[9]) - refz3
            #Vai arredondar os valores para 4 casas decimais
            x1 = round(x1, 4)
            y1 = round(y1, 4)
            z1 = round(z1, 4)
            x2 = round(x2, 4)
            y2 = round(y2, 4)
            z2 = round(z2, 4)
            x3 = round(x3, 4)
            y3 = round(y3, 4)
            z3 = round(z3, 4)

            #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência (inverto o eixo X para alinhar os eixos de coordenadas - ferramenta - câmara)
            visp1 = [x1*-1, y1, z1]
            visp2 = [x2*-1, y2, z2]
            visp3 = [x3*-1, y3, z1]

            laser_off = trispector.write("set laser off") #Desliga o laser

        if valid_locate!="1": #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
            tp_log("Erro a Localizar!")
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 33) #Info para PLC imagem nao adquirida corretamente
            #Vai se mover para a posição de carregamento (posição incial)
            movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
            movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
            movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
            goto(1)
            continue

        set_output_register_int(0, 130) #Avança
```

```
        goto(1005)
        continue

    if line == 1005: #Irá verificar possiveis erros para garantir que está tudo correto para posteriormente iniciar o aperto
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1006)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                trispector.closeIN(None)
                trispector.closeOUT(None)
                #Vai se mover para a posição de carregamento (posição incial)
                movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1006:

        trispector.closeIN(None)
        trispector.closeOUT(None)

        mwait(0)
        set_tcp("U_Tool_2")

        #Define os pontos de referência das três posições de aperto usadas para gerar o frame de referência
        PF1 = posx(916.84, 552.43, 782.1, 90.84, -93.16, -0.04)
        PF2 = posx(704.89, 552.84, 777.79, 90.84, -93.16, -0.04)
        PF3 = posx(922.67, 511.58, 619.51, 90.84, -93.16, -0.04)
        Frame1  =  posx(916.84,  552.43,  782.1,  89.597,  -75.911,  91.201)  #Frame  Referência (informativo) - +X direção porta, +Y direção travão, +Z direção vidro

        #Vai calcular os novos pontos de referência de acordo com os desvios obtidos com o sistema de visão relativos às coordenadas X, Y e Z
        P1vis = [PF1[0]-visp1[1], PF1[1]-visp1[2], PF1[2]-visp1[0]]
        P2vis = [PF2[0]-visp2[1], PF2[1]-visp2[2], PF2[2]-visp2[0]]
        P3vis = [PF3[0]-visp3[1], PF3[1]-visp3[2], PF3[2]-visp3[0]]

        #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem p1 p2 e p3 que são os novos pontos de referência
        p1 = P1vis[0:3]+PF1[3:]
        p2 = P2vis[0:3]+PF2[3:]
        p3 = P3vis[0:3]+PF3[3:]

        #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
        pose_user102 = calc_coord(p1, p2, p3, ref=DR_BASE, mod=0)

        #Vai substituir o frame existente 102 pelo novo frame calculado de forma permanente
        overwrite_user_cart_coord(102, pose_user102, ref=DR_BASE, apply_mod=DR_PERMANENT)

        mwait(0)
        set_ref_coord(102) #Seleciona o novo frame criado com os valores do sistema de visão, que irá corrigir os novos pontos de aperto das restantes porcas
        set_tcp("U_Tool_4")

        time.sleep(0.3)
```

```
        #A partir deste ponto irá passar-se para a fase de aperto das porcas
        set_output_register_int(3, 0) #Info para PLC decisão recebida (reset)
        #O robô vai se mover para posteriormente começar as posições de aperto
        movej(posj(189.49, -18.66, -98.65, -16.87, 113.72, -43.64), v=220, a=250, r=50)
        movej(posj(186.95, -22.47, -87.38, 179.29, 70.16, -259.86), v=220, a=250, r=40)

        #Pontos de aproximação do primeiro parafuso (respetivos ao novo frame calculado)
        movej(posj(185.2, -32.41, -89.96, 181.81, 57.23, -264.52), v=100, a=100, r=40) #Posição
aproximação
        movel(posx(-18.287, 670.634, 51.632, 92.739, 15.020, 176.622), v=150, a=250, r=20) #
Posição perto do parafuso respetivo ao frame novo (Tool 4 em U_Frame_1 (101))
        movel(posx(-17.895, 662.461, 21.130, 92.737, 15.020, 176.623), v=50, a=50) # Posição
relativa ao toque no parafuso (Tool 4 em U_Frame_1 (101))
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 140) #Info para PLC etapa aperto P1

        #Aperto P1
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        goto(1007)
        continue

    if line == 1007:#Depois de fazer o aperto da primeira porca, passoa para as posições relativas
ao aperto 2

        movel(posx(-18.287, 670.634, 51.632, 92.739, 15.020, 176.622), v=150, a=250, r=20) #Volta
à posição perto do parafuso para depois recuar em segurança
        movej(posj(191.63, -27.95, -80.99, 162.81, 71.99, -252.02), v=120, a=120, r=10) #Posição
de Aproximação do parafuso 2
        set_output_register_int(0, 150) #Info para PLC etapa robô
        movel(posx(-10.386, 590.062, 40.613, 67.625, -2.053, -155.496), v=50, a=50)  # Posição
perto do parafuso respetivo ao frame novo (Tool 4 em U_Frame_1 (101))
        goto(1008)
        continue

    if line == 1008: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555): #Caso não haja erros
                set_output_register_int(3, 777) #Info para PLC ACK
                #time.sleep(0.1)
                goto(1009)
                continue

            error=get_input_register_int(2)
            if(error==9999): #Caso haja erros
                set_output_register_int(3, 777) #Info para PLC ACK
                #Vai se mover para a posição de carregamento (posição incial)
                movej(posj(190.3, -19.14, -88.61, 81.84, 78.11, -160.8), v=100, a=100, r=50)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0) #Dá reset no ACK do PLC
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1009:  #Aperto P2
        set_output_register_int(3, 0) #Dá reset no ACK do PLC
        movel(posx(-9.913, 590.705, 6.316, 67.622, -2.053, -155.494), v=50, a=50)# Posição relativa
ao toque no parafuso (Tool 4 em U_Frame_1 (101))
        set_output_register_int(0, 160)  #Info para PLC etapa aperto P2
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para a aparafusadora para iniciar o aperto
        time.sleep(0.2)
```

```
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        movel(posx(-8.386, 590.062, 40.613, 67.625, -2.053, -155.496), v=50, a=50) #Volta à posição
perto do parafuso para depois recuar em segurança
        goto(1010)
        continue

    if line == 1010: #Passagem do P2 para o P3
        mwait(0)
        set_tcp("U_Tool_3")
        set_output_register_int(0, 170)
        movej(posj(192.74, -22.35, -108.64, 19.48, 45.34, -120), v=120, a=120, r=10) #PA3
        movel(posx(-7.255, 489.025, 36.901, 75.350, -1.414, -164.210), v=100, a=100, r=10) #PA3
Tool 3 em U_Frame_1 (101)
        goto(1011)
        continue

    if line == 1011: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1012)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue


    if line == 1012: #Aperto P3
        set_output_register_int(3, 0)
        movel(posx(-7.041, 489.848, 2.411, 75.347, -1.413, -164.207), v=50, a=50) #PE3 Tool 3 em
U_Frame_1 (101)
        set_output_register_int(0, 180)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(-7.255, 489.025, 36.901, 75.350, -1.414, -164.210), v=100, a=100, r=10) #PA3
Tool 3 em U_Frame_1 (101)
        goto(1013)
        continue

    if line == 1013:
        free_zone=get_input_register_int(4) #Verifica se a zona está livre
        if(free_zone==222):
            time.sleep(0.01)
            set_output_register_int(4, 888) #Info para PLC Zona Ocupada
            goto(1014)
            continue
        else:
            goto(1013)
            continue

    if line == 1014: #Passagem do P3 para o P4
        set_output_register_int(0, 190)
        set_output_register_int(4, 888) #Info para PLC Zona Ocupada
        movej(posj(198.13, -18.33, -80.28, 65.44, 20.01, -170.06), v=100, a=100, r=10) #PA4
```

```
        movel(posx(-6.277, 168.134, 28.475, 68.046, -0.980, -156.908), v=100, a=100, r=0) # PA4
Tool 3 em U_Frame_1 (101)
        goto(1015)
        continue

    if line == 1015: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1016)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1016: #Aperto P4
        set_output_register_int(3, 0)
        movel(posx(-6.068, 168.647, -4.121, 68.042, -0.981, -156.903), v=50, a=50) #PE4 Tool 3 em
U_Frame_1 (101)
        set_output_register_int(0, 200)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(-6.277, 168.134, 28.475, 68.046, -0.980, -156.908), v=100, a=100, r=0) #PA4
Tool 3 em U_Frame_1 (101)
        goto(1017)
        continue

    if line == 1017: #Passagem do P4 para o P5
        set_output_register_int(0, 210)
        movej(posj(198.71, -18.99, -70.41, 91.79, 18.72, -197.39), v=120, a=120, r=10) #PA5
        movel(posx(-4.591, 76.259, 32.663, 68.049, -0.980, -156.911), v=100, a=100) #PA5 Tool 3 em
U_Frame_1 (101)
        goto(1018)
        continue

    if line == 1018: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1019)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
```

```
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1019: #Aperto P5
        set_output_register_int(3, 0)
        movel(posx(-4.340, 76.879, -6.631, 68.047, -0.980, -156.909), v=80, a=80) #PE5 Tool 3 em
U_Frame_1 (101)
        set_output_register_int(0, 220)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(-4.591, 76.259, 32.663, 68.049, -0.980, -156.911), v=100, a=100) #PA5 Tool 3 em
U_Frame_1 (101)
        goto(1020)
        continue

    if line == 1020: #Passagem do P5 para o P6
        mwait(0)
        set_output_register_int(0, 230)
        set_tcp("U_Tool_2")
        movej(posj(197.98, -25.16, -77.05, 0.01, 102.22, -107.97), v=150, a=150, r=10) #PA6
        movel(posx(212.535, 13.357, 42.676, 88.276, 16.171, -178.048), v=150, a=150) #PA6 Tool 2
em U_Frame_1 (101)
        goto(1021)
        continue

    if line == 1021: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1022)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1022: #Aperto P6
        set_output_register_int(3, 0)
        movel(posx(212.182, 1.532, 1.877, 88.277, 16.171, -178.049), v=50, a=50) #PE6 Tool 2 em
U_Frame_1 (101)
        set_output_register_int(0, 240)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(212.535, 13.357, 42.676, 88.276, 16.171, -178.048), v=150, a=150) #PA6 Tool 2
em U_Frame_1 (101)
        goto(1023)
        continue
```

```
    if line == 1023: #Passagem do P6 para o P7
        set_output_register_int(0, 250)
        time.sleep(1)
        goto(1024)
        continue

    if line == 1024: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1025)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1025: #Aperto P7
        set_output_register_int(3, 0)
        movej(posj(204.2, -10.57, -96.5, 0.01, 107.08, -114.2), v=220, a=120, r=0) #PA7
        set_output_register_int(0, 260)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        movel(posx(409.056, 11.468, 32.503, 88.277, 16.171, 179.930), v=150, a=150, r=10) #PA7
Tool 2 em U_Frame_1 (101)
        movel(posx(408.785, 2.535, 1.673, 88.277, 16.171, 179.930), v=100, a=150) #PE7 Tool 2 em
U_Frame_1 (101)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(409.056, 11.468, 32.503, 88.277, 16.171, 179.930), v=150, a=150, r=10) #PA7
Tool 2 em U_Frame_1 (101)
        goto(1026)
        continue

    if line == 1026: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1027)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1027)
                continue

    if line == 1027:
        #Saida para a posição inicial depois de efetuar todos os apertos
        mwait(0)
        set_ref_coord(DR_BASE) #Define o sistema de referencia de coordenadas
        set_output_register_int(0, 270) #Info para PLC etapa robô (Retorno à posição inicial)
        #Vai se mover para a posições intermédias de retorno
        movej(posj(178.5, -11.38, -95.73, -1.56, 109.1, -118.76), v=150, a=150, r=50)
```

```
        movej(posj(170.2, 27.74, -128.61, -1.43, 102.92, -116.27), v=150, a=150, r=0)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        movej(posj(115.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=120, a=150, r=50)
        movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=120, a=150, r=50)
        movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=100, a=100, r=50)
        movej(carregamento, v=20, a=10) #(posição incial)

        set_tcp("U_Tool_2")
        set_ref_coord(DR_BASE)
        set_output_register_int(3, 0) #Dá reset no ACK do PLC
        set_tcp("U_Tool_2")
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
        goto(1)
        continue



################################################################################################################################
    #Segunda parte do codigo relativo à jante 16, a ideologia desta segunta parte do código é igual à primeira apenas com as correções das posições devido à diferença de alturas

    if line == 1102: #Jante 16

        set_tcp("U_Tool_1")
        set_ref_coord(DR_BASE)

        #Movimento de Saida Cuba
        set_output_register_int(0, 101)
        movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=200, a=120, r=50) #Função usada para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a velocidade "v", a aceleracao "a" e o radius "r"
        set_output_register_int(0, 105)
        movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=200, a=120, r=50)
        movej(posj(93.1, 37.24, -131.84, -4.77, 97.09, -94.91), v=200, a=100, r=10)
        goto(1103)
        continue

    if line == 1103:
        set_output_register_int(0, 110)
        free_zone=get_input_register_int(4)
        if(free_zone==222): #Verifica se a zona de aperto está livre, caso esteja livre avança senão espera que a mesma esteja
            #time.sleep(0.1)
            set_output_register_int(4, 888) #Envia a informacao para o PLC que a zona irá ficar ocupada pelo robô
            goto(1104)
            continue
        else:
            time.sleep(0.1)
            goto(1103)
            continue

    if line == 1104:
        set_output_register_int(0, 120)
        car_pos=get_input_register_int(2)
        if(car_pos==5555): #Verifica se a carrinha está na posição correta
            set_output_register_int(3, 777)
            time.sleep(0.01)
            goto(1144)
            continue

        if(car_pos==9999): #Caso a informação que chega ao robô indique que a carrinha não está na posição correta o robô voltará à posição inicial (carrrgamento) e o código voltará à parte da verificação inicial da posição da carrinha
            set_output_register_int(3, 777)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50) #Porta
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(posj(37.8, -15.37, -103.01, -70.17, 59.06, -123.26), v=20, a=10)
            movej(carregamento, v=20, a=10)
```

```
            set_output_register_int(3, 777)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
            time.sleep(2)
            goto(1)
            continue

    if line == 1144: #Se a carrinha estiver na posição correta o programa prossegue para a aquisição
da imagem pelo sistema de visão

        #Entrada Carro
        movej(posj(163.68, 29.35, -129.86, -12.54, 103.8, -69.28), v=220, a=180, r=50)
        movej(posj(184.55, -13.96, -105.01, -17.47, 116.57, -95.72), v=220, a=180, r=100)

        #Posição para se dar ao inicio da aquisição da imagem pelo sistema de visão
        movej(posj(189.46, -53.84, -33.01, 0, 86.85, -9.46), v=220, a=120, r=10)
        mwait(0)
        set_tcp("U_Tool_1")
        movel(posx(1050.00, 140.0, 670.0, 90, 0, 90), v=120, a=150)
        time.sleep(0.05)
        goto(11444)
        continue

    if line == 11444:
        set_output_register_int(3, 0)
        laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da
imagem
        if (laser_on != 'OK'): #Verifica se o laser foi ativado, caso não tenha sido o robô abortará
a tarefa de aquisição da imagem e voltará para a posição inicial
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 11)
            movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
            movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
            movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
            goto(1)
            continue

        job = trispector.write("set job \"J 16\"") #Seleciona o programa "J 16" para ser utilizado
na câmara
        if (job != 'OK'): #Verifica se o programa foi selecionado corretamente caso contrário
aborta e voltará ao inicio
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 11)
            movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
            movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
            movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
            goto(1)
            continue

        #Caso o laser a seleção do programa tenham sido executados com sucesso irá passar-se à
aquisição da imagem
        time.sleep(0.3)
        amovel(posx(688.00, 140.0, 670.0, 90.00, 00.00, 90.00), v=50, a=200)
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio
à aquisição da imagem
```

```
        mwait(0)
        rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida

        rxd =(rx_msg[0:128])
        data = rx_msg.split(', ') #Uma vez que a informação enviada pela câmara vem toda seguida
separada por virgulas, aqui irá dividir-se a informação pelas virgulas criando uma array
        valid_locate  =(rx_msg[0:1])  #Vai  guardar  a  informação  se  a  imagem  foi  adquirida
corretamente

        #Valores imagem referência (valores do frame de referência criado)
        refx1 = 87.932299
        refy1 = 75.752267
        refz1 = 201.480092
        refx2 = 87.556817
        refy2 = 288.538625
        refz2 = 203.379294
        refx3 = -76.332865
        refy3 = 72.511114
        refz3 = 240.720468

        if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x3 =  float(data[7]) - refx3
            y3 =  float(data[8]) - refy3
            z3 =  float(data[9]) - refz3
            x1 = round(x1, 4)
            y1 = round(y1, 4)
            z1 = round(z1, 4)
            x2 = round(x2, 4)
            y2 = round(y2, 4)
            z2 = round(z2, 4)
            x3 = round(x3, 4)
            y3 = round(y3, 4)
            z3 = round(z3, 4)

            #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência
            visp1 = [x1*-1, y1, z1]
            visp2 = [x2*-1, y2, z2]
            visp3 = [x3*-1, y3, z1]

            laser_off = trispector.write("set laser off") #Desativa o laser

        if valid_locate!="1": #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta
e volta ao inicio
            tp_log("Erro a Localizar!")
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 33)
            movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
            movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
            movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
            movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
            movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400) #Info para PLC etapa do robô (erro)
            goto(1)
            continue

        set_output_register_int(0, 130)
        goto(1105)
        continue
```

```
    if line == 1105: #Irá verificar possiveis erros para garantir que está tudo correto para posteriormente iniciar o aperto
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1106)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                trispector.closeIN(None)
                trispector.closeOUT(None)
                movej(posj(190.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1106:

        trispector.closeIN(None)
        trispector.closeOUT(None)

        mwait(0)
        set_tcp("U_Tool_2")

        #Define os pontos de referência das três posições de aperto usadas para gerar o frame de referência
        PF1 = posx(914.6, 558.66, 803.56, 89.79, -93.32, 0.83)
        PF2 = posx(701.63, 555.43, 802.09, 89.78, -93.04, 0.83)
        PF3 = posx(917.05, 523.030, 636.84, 89.51, -74.25, 0.86)
        Frame3 = posx(914.60, 558.66, 803.56, 90.784, -77.927, 90.404) #Frame Referência - +X direção porta, +Y direção travão, +Z direção vidro

        #Vai calcular os novos pontos de referência de acordo com os valores obtidos com o sistema de visão relativos às coordenadas X, Y e Z
        P1vis = [PF1[0]-visp1[1], PF1[1]-visp1[2], PF1[2]-visp1[0]]

        P2vis = [PF2[0]-visp2[1], PF2[1]-visp2[2], PF2[2]-visp2[0]]

        P3vis = [PF3[0]-visp3[1], PF3[1]-visp3[2], PF3[2]-visp3[0]]

        #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem p1 p2 e p3 que são os novos pontos de referência
        p1 = P1vis[0:3]+PF1[3:]
        p2 = P2vis[0:3]+PF2[3:]
        p3 = P3vis[0:3]+PF3[3:]

        pose_user104 = calc_coord(p1, p2, p3, ref=DR_BASE, mod=0) #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão

        overwrite_user_cart_coord(104, pose_user104, ref=DR_BASE, apply_mod=DR_PERMANENT)

        mwait(0)
        set_ref_coord(104) #Seleciona o novo frame criado com os valores do sistema de visão, que irá corrigir os novos pontos de aperto das restantes porcas
        set_tcp("U_Tool_4")

        time.sleep(0.3)

        #A partir deste ponto irá passar-se para a fase de aperto das porcas
```

```
        set_output_register_int(3, 0)
        movej(posj(189.49, -18.66, -98.65, -16.87, 113.72, -43.64), v=220, a=250, r=50)
        movej(posj(186.95, -22.47, -87.38, 179.29, 70.16, -259.86), v=220, a=250, r=40)

        #Pontos de aproximação do primeiro parafuso (respetivos ao novo frame calculado)
        movej(posj(185.2, -32.41, -89.96, 181.81, 57.23, -264.52), v=100, a=100, r=40) #PA1
        movel(posx(-9.452, 667.151, 66.480, 90.412, 12.855, 179.482), v=150, a=250, r=20) #PA1
Tool 4 em U_Frame_3
        movel(posx(-9.414, 663.146, 44.529, 90.413, 12.855, 179.481), v=50, a=50) #PE1 Tool 4 em
U_Frame_3 (103)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 140) #Info para PLC etapa aperto P1

        #Aperto P1
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        goto(1107)
        continue

    if line == 1107: #Depois de fazer o aperto da primeira porca, passoa para as posições relativas
ao aperto 2

        movel(posx(-9.452, 669.151, 66.480, 90.412, 12.855, 179.482), v=150, a=250, r=20) #PA1
Tool 4 em U_Frame_3
        movej(posj(191.63, -27.95, -80.99, 162.81, 71.99, -252.02), v=120, a=120, r=10) #PA2
        set_output_register_int(0, 150)
        movel(posx(-2.409, 589.185, 57.536, 85.112, -3.555, -172.641), v=50, a=50) #PA2 Tool 4 em
U_Frame_3 (103)
        goto(1108)
        continue

    if line == 1108: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                #time.sleep(0.1)
                goto(1109)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(190.3, -19.14, -88.61, 81.84, 78.11, -160.8), v=100, a=100, r=50)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1109: #Aperto P2
        set_output_register_int(3, 0)
        movel(posx(-2.250, 591.094, 26.739, 85.116, -3.555, -172.645), v=50, a=50) #PE2 Tool 4 em
U_Frame_13(103)
        set_output_register_int(0, 160)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(-2.409, 589.185, 57.536, 85.112, -3.555, -172.641), v=50, a=50) #PA2 Tool 4 em
U_Frame_3 (103)
        goto(1110)
        continue
```

```
    if line == 1110: #Passagem do P2 para o P3
        mwait(0)
        set_tcp("U_Tool_3")
        set_output_register_int(0, 170)
        movej(posj(192.74, -22.35, -108.64, 19.48, 45.34, -120), v=120, a=120, r=10) #PA3
        movel(posx(-0.927, 488.338, 55.204, 98.430, -3.543, 173.086), v=100, a=100, r=10) #PA3
Tool 3 em U_Frame_3 (103)
        goto(1111)
        continue

    if line == 1111: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1112)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1112:  #Aperto P3
        set_output_register_int(3, 0)
        movel(posx(-1.251, 490.534, 19.345, 98.428, -3.543, 173.088), v=50, a=50) #PE3 Tool 3 em
U_Frame_3 (103)
        set_output_register_int(0, 180)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(-0.927, 488.338, 55.204, 98.430, -3.543, 173.086), v=100, a=100, r=10) #PA3
Tool 3 em U_Frame_3 (103)
        goto(1113)
        continue

    if line == 1113:
        free_zone=get_input_register_int(4)
        if(free_zone==222): #Verifica se a zona está livre
            time.sleep(0.01)
            set_output_register_int(4, 888) #Info para PLC Zona Ocupada
            goto(1114)
            continue
        else:
            goto(1113)
            continue

    if line == 1114: #Passagem do P3 para o P4
        set_output_register_int(0, 190)
        set_output_register_int(4, 888) #Info para PLC Zona Ocupada
        movej(posj(198.13, -18.33, -80.28, 65.44, 20.01, -170.06), v=100, a=100, r=10) #PA4
        movel(posx(-2.868, 169.909, 32.424, 98.432, -3.540, 173.079), v=100, a=100, r=0) #PA4 Tool
3 em U_Frame_3 (103)
        goto(1115)
        continue

    if line == 1115: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
```

```
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1116)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1116: #Aperto P4
        set_output_register_int(3, 0)
        movel(posx(-3.150, 171.812, 1.384, 98.428, -3.540, 173.083), v=50, a=50) #PE4 Tool 3 em
U_Frame_3 (103)
        set_output_register_int(0, 200)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(-2.868, 169.909, 32.424, 98.432, -3.540, 173.079), v=100, a=100, r=0) #PA4 Tool
3 em U_Frame_3 (103)
        goto(1117)
        continue

    if line == 1117: #Passagem do P4 para o P5
        set_output_register_int(0, 210)
        movej(posj(198.71, -18.99, -70.41, 91.79, 18.72, -197.39), v=120, a=120, r=10) #PA5
        movel(posx(-2.696, 77.157, 23.876, 98.897, -3.544, 172.586), v=100, a=100) #PA5 Tool 3 em
U_Frame_3 (103)
        goto(1118)
        continue

    if line == 1118: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1119)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1119:  #Aperto P5
```

```
        set_output_register_int(3, 0)
        movel(posx(-2.963, 78.879, -4.370, 98.892, -3.544, 172.592), v=80, a=80) #PE5 Tool 3 em
U_Frame_3 (103)
        set_output_register_int(0, 220)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(-2.696, 77.157, 23.876, 98.897, -3.544, 172.586), v=100, a=100) #PA5 Tool 3 em
U_Frame_3 (103)
        goto(1120)
        continue

    if line == 1120: #Passagem do P5 para o P6
        mwait(0)
        set_output_register_int(0, 230)
        set_tcp("U_Tool_2")
        movej(posj(197.98, -25.16, -77.05, 0.01, 102.22, -107.97), v=150, a=150, r=10) #PA6
        movel(posx(214.615, 8.538, 31.867, 88.149, 15.110, -177.925), v=150, a=150) #PA6 Tool 2 em
U_Frame_3 (103)
        goto(1121)
        continue

    if line == 1121: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1122)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1122: #Aperto P6
        set_output_register_int(3, 0)
        movel(posx(214.338, 0.020, 0.273, 88.149, 15.110, -177.926), v=50, a=50) #PE6 Tool 2 em
U_Frame_3 (103)
        set_output_register_int(0, 240)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(214.615, 8.538, 31.867, 88.149, 15.110, -177.925), v=150, a=150) #PA6 Tool 2 em
U_Frame_3 (103)
        goto(1123)
        continue

    if line == 1123: #Passagem do P6 para o P7
        set_output_register_int(0, 250)
        time.sleep(1)
        goto(1124)
        continue

    if line == 1124: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
```

```
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.01)
                goto(1125)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(184.26, -14.36, -104.48, -17, 114.98, -100.95), v=100, a=100, r=50)
                movej(posj(171.14, 30.98, -129.09, -15.69, 100.18, -76.6), v=100, a=100, r=50)
                movej(posj(121.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=100, a=100, r=50)
                movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=80, a=80, r=50)
                movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1125: #Aperto P7
        set_output_register_int(3, 0)
        movej(posj(204.2, -10.57, -96.5, 0.01, 107.08, -114.2), v=220, a=120, r=0) #PA7
        set_output_register_int(0, 260)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        movel(posx(410.650, 9.070, 29.263, 88.148, 15.110, -177.923), v=150, a=150, r=10) #PA7
Tool 2 em U_Frame_3 (103)
        movel(posx(410.387, 0.866, -1.148, 88.149, 15.110, -177.925), v=100, a=150) #PE7 Tool 2 em
U_Frame_3 (103)
        time.sleep(0.1)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.1)
        movel(posx(410.650, 9.070, 29.263, 88.148, 15.110, -177.923), v=150, a=150, r=10) #PA7
Tool 2 em U_Frame_3 (103)
        goto(1126)
        continue

    if line == 1126: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1127)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)

                time.sleep(0.1)
                goto(1127)
                continue

    if line == 1127:
        #Saida para a posição inicial depois de efetuar todos os apertos
        mwait(0)
        set_ref_coord(DR_BASE)
        set_output_register_int(0, 270)
        movej(posj(178.5, -11.38, -95.73, -1.56, 109.1, -118.76), v=150, a=150, r=50)
        movej(posj(170.2, 27.74, -128.61, -1.43, 102.92, -116.27), v=150, a=150, r=0)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        movej(posj(115.25, 30.98, -129.09, -15.69, 100.18, -122.76), v=120, a=150, r=50)
        movej(posj(41.71, 1.69, -110.92, -65.41, 55.67, -119.93), v=120, a=150, r=50)

        movej(posj(36.62, -9.21, -110.03, -70.05, 58.63, -124.89), v=100, a=100, r=50)

        movej(carregamento, v=20, a=10)
```

```
        set_tcp("U_Tool_2")
        set_ref_coord(DR_BASE)
        set_output_register_int(3, 0)
        set_tcp("U_Tool_2")
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 410) #Info para PLC etapa do robô (Inicio)
        goto(1)
        continue

    #Fim do código
```

Anexo 2 – Código Cloison do robô do lado esquerdo

```
#Importa as bibliotecas necessárias e utilizadas pelo código
from DRCF import*
import numpy as np
import sys
import os
import time
import socket

class DoosanSick:

#*********************************

    #Função que inicializa e faz a conexão da câmara
    def __init__(self, ip="192.168.1.71", portIN=2115, portOUT=2114):

        self.ip = ip
        self.portIN = portIN
        self.portOUT = portOUT
        client = socket.socket()
        client.settimeout(3)
        socket.setdefaulttimeout(3)
        client.connect((self.ip, self.portIN))

        try:
            self._socketIN = client_socket_open(self.ip, self.portIN)
        except Exception as e:
            tp_popup("Socket INPUT falha de ligação. Error: {0}".format(
                str(e)), DR_PM_ALARM)
            raise e

        time.sleep(0.1)

        try:
            self._socketOUT = client_socket_open(self.ip, self.portOUT)
        except Exception as e:
            tp_popup("Socket OUTPUT falha de ligação. Error: {0}".format(
                str(e)), DR_PM_ALARM)
            raise e

#*********************************

    #Função usada para escrever no socket
    def write(self, cmd, socket=None):

        if socket == None:
            socket = self._socketIN

        cmd = bytes(cmd, encoding="ascii")

        STX = client_socket_write(socket, b"\x02")
        res = client_socket_write(socket, cmd)
        STX = client_socket_write(socket, b"\x03")
        time.sleep(0.1)
        res, rx_data = client_socket_read(socket, 10, 1)
        rxd=None
        if res > 1:
```

```
            rx_msg = rx_data.decode()
            rxd =(rx_msg[1:3])
        if res == -1:
            tp_log("Erro durante a escrita no socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a escrita no socket: Erro de Socket")
        return rxd

#*********************************

    #Função usada para ler a informação do command channel
    def readcmd(self, length, timeout, socket=None ):

        if socket == None:
           socket = self._socketIN

        res, rx_data = client_socket_read(socket, length, timeout)
        rx_msg = rx_data.decode()
        rxd =(rx_msg[1:3])
        if res == -1:
            tp_log("Erro durante a leitura do socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a leitura do socket: Erro de Socket")
        elif res == -3:
            tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
        return rxd

#*********************************

    #Função usada para ler do socket a informação recebida
    def readata(self, length, timeout, socket=None ):

        if socket == None:
           socket = self._socketOUT

        res = 0
        rx_data = 0

        res, rx_data = client_socket_read(socket, length, timeout)

        if res == -1:
            tp_log("Erro durante a leitura do socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a leitura do socket: Erro de Socket")
        elif res == -3:
            tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
        else:
            rx_msg = rx_data.decode()
            return rx_msg

#*********************************

    #Função que irá fechar as conexões dos socket IN e OUT utilizados para efetuar a comunicação
com a câmara
    def closeIN(self, socket=None):

        if socket == None:
            socket = self._socketIN

        client_socket_close(socket)

    def closeOUT(self, socket=None):

        if socket == None:
            socket = self._socketOUT

        client_socket_close(socket)

#Função que irá esperar que o torque pretendido seja atingido para desativar a aparafusadora
def wait_torch() :
            yy = True
            zz = 0
```

```
            while yy:
                step_ack=get_input_register_int(1)
                if(step_ack==1000) or (step_ack==2000):
                    set_output_register_int(2, 0) #Output para ativar e desativar aparafusadora
                    time.sleep(0.1)
                    yy = False
                if zz >= 35:
                    yy = False
                time.sleep(0.1)
                zz = zz + 0.1

#*************************************************************************

#Definição de algumas variáveis globais e outputs importantes
set_motion_end(DR_CHECK_ON)
set_output_register_int(0, 0)
set_output_register_int(1, 0)
set_digital_output(6, OFF)
set_digital_output(8, OFF)
set_tool ("U_Tool_W1")  #Seleciona a tool que se pretende
set_tcp("U_Tool_2")     #Seleciona o TCP da respetiva tool
set_ref_coord(DR_BASE)  #Define o referêncial de coordenadas para movimentar o robô
time.sleep(0.5)

if get_digital_input(16)==0: #Segurança
 exit()

#Definição de uma variável global que será usada posteriormente para indicar para onde irá a
execução do programa
def goto(linenum):
    global line
    line = linenum
line = 1

#Dá set a 0 os seguintes outputs
set_output_register_int(0, 0) #Info para PLC etapa do Robô
set_output_register_int(1, 0) #Info para PLC resultado Visão
set_output_register_int(2, 0) #Pedido aperto
set_output_register_int(3, 0) #Info para PLC decisão recebida
set_output_register_int(4, 0) #Info para PLC zona ocupada

time.sleep(0.5)

if get_digital_input(16)==0: #Segurança
 exit()

time.sleep(0.1)
move_home(DR_HOME_TARGET_USER)

def wait_zero() : #Função usava para esperar pela informacao "Zona Livre" do PLC
        zz = True
        while zz:
            traj=get_input_register_int(0)
            if(traj==0) or (traj==99):
                time.sleep(0.2)
                zz = False
            time.sleep(1.5)
wait_zero()
stop(DR_QSTOP)

set_output_register_int(0, 410)

#Posição de carregamento de porcas
carregamento = posj(139.76, 17.15, 98.56, 73.84, -52.07, 119.28)

#Inicio do programa de aperto
while True:

    time.sleep(0.2)

    #Uma vez que existem duas carrinhas diferentes, uma com jante 15 e outra com jante 16 isso irá
interferir nas posição das posições de aperto,
```

```
    #logo aqui é feita a validação de qual das carrinhas chegou à linha e o programa irá correr de
acordo com a carrinha respetiva
    if line == 1 and get_input_register_int(0) == 10 and get_digital_input(16)==1 : #Jante 15
        set_output_register_int(0, 0)
        set_output_register_int(1, 0)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(4, 0)
        camera_ip = "192.168.1.71"
         trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114) #Faz a conecção com a
câmara
        time.sleep(0.5)
        job = trispector.write("set job \"J 15\"") #Seleciona o programa da câmara
        set_output_register_int(0, 100)
         goto(1002) #Define a variável line a 1002 e será com esta variável que se irá controlar
qual parte do codigo correr a seguir
        continue

    if line == 1 and get_input_register_int(0) == 11 and get_digital_input(16)==1 : #Jante 16
        set_output_register_int(0, 0)
        set_output_register_int(1, 0)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(4, 0)
        camera_ip = "192.168.1.71"
         trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114) #Faz a conecção com a
câmara
        time.sleep(0.5)
        job = trispector.write("set job \"J 16\"") #Seleciona o programa da câmara
        set_output_register_int(0, 101)
         goto(1102) #Define a variável line a 1102 e será com esta variável que se irá controlar
qual parte do codigo correr a seguir
        continue

    #Função que fica à espera que a carrinha chegue ao ponto de montagem, enquanto a mesma não
chega o programa fica em loop aqui
    if line == 1 and get_input_register_int(0) == 99 and get_digital_input(16)==1 :
        set_output_register_int(0, 0)
        set_output_register_int(1, 0)
        time.sleep(0.5)
        set_output_register_int(0, 999) #Fora serviço
        goto(99)
        continue

    if line == 99:
        set_output_register_int(0, 999)
        outserv=get_input_register_int(0)
        if(outserv==0):
            time.sleep(0.2)
            set_output_register_int(0, 400)
            time.sleep(1)
            set_output_register_int(0, 0)
            goto(1)
            continue
        time.sleep(0.3)
        goto(99)
        continue

##########################################################################################################
#########################################################################
    #Como dito existem duas carrinhas diferentes então o código está dividido em dois sendo uma
parte para uma carrinha e outra para a outra carrinha
    #Aqui está o código para a carrinha com a jante 15, caso a carrinha que chegue ao ponto de
montagem contenha a jante relativa à jante 15 correrá esta parte do código caso contrario irá
saltar esta parte e executar a parte relativa
    #à segunda parte do código

    if line == 1002:

        set_tcp("U_Tool_1")
        set_ref_coord(DR_BASE)
```

```
        #Movimento de Saida Cuba
        set_output_register_int(0, 100)

        movej(carregamento, v=20, a=10)
        movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50) #Função usada
para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a velocidade "v", a aceleracao
"a" e o radius "r"
        set_output_register_int(0, 105)
        movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=80, a=80, r=50)
        movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=100, a=100, r=50)
        goto(1003)
        continue

    if line == 1003:
        set_output_register_int(0, 110)
        free_zone=get_input_register_int(4) #Verifica se a zona de aperto está livre, caso esteja
livre avança senão espera que a mesma esteja
        if(free_zone==222):
            time.sleep(0.1)
            set_output_register_int(4, 888) #Envia a informacao para o PLC que a zona irá ficar
ocupada pelo robô
            goto(1004)
            continue
        else:
            time.sleep(0.1)
            goto(1003)
            continue

    if line == 1004:
        set_output_register_int(0, 120)
        car_pos=get_input_register_int(2)
        if(car_pos==5555): #Verifica se a carrinha está na posição correta
            set_output_register_int(3, 777)
            time.sleep(0.1)
            goto(1044)
            continue

        if(car_pos==9999): #Caso a informação que chega ao robô indique que a carrinha não está na
posição correta o robô voltará à posição inicial (carrrgamento) e o código voltará à parte da
verificação inicial da posição da carrinha
            set_output_register_int(3, 777)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=0)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_output_register_int(3, 777)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 410)
            time.sleep(2)
            goto(1)
            continue

    if line == 1044: #Se a carrinha estiver na posição correta o programa prossegue para a aquisição
da imagem pelo sistema de visão

        #Entrada no carro
        movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=100, a=100, r=50)
        movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=100, a=100, r=50)

        #Posição para se dar ao inicio da aquisição da imagem pelo sistema de visão
        movej(posj(-11.26, 44.37, 44.27, 0, -88.64, 11.26), v=100, a=100)
        mwait(0)
        set_tcp("U_Tool_1")
        movel(posx(980.0, -160.0, 730.00, 162.59, 0, -162.59), v=100, a=150)
        time.sleep(1)
        goto(10444)
        continue

    if line == 10444:
```

```
        set_output_register_int(3, 0)
         laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da
imagem
       if (laser_on != 'OK'): #Verifica se o laser foi ativado, caso não tenha sido o robô abortará
a tarefa de aquisição da imagem e voltará para a posição inicial
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 11)
            movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
            movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400)
            time.sleep(0.1)
            goto(1)
            continue

       job = trispector.write("set job \"J 15\"") #Seleciona o programa "J 15" para ser utilizado
na câmara
        if (job != 'OK'): #Verifica se o programa foi selecionado corretamente caso contrário
aborta e voltará ao inicio
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 11)
            movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
            movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400)
            time.sleep(0.1)
            goto(1)
            continue

        #Caso o laser a seleção do programa tenham sido executados com sucesso irá passar-se à
aquisição da imagem
        time.sleep(0.3)
        amovel(posx(720.0, -160.0, 730.0, 162.59, 0, -162.59), v=50, a=200)
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio
à aquisição da imagem
        mwait(0)
        rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida

        rxd =(rx_msg[0:128])
        data = rx_msg.split(', ') #Uma vez que a informação enviada pela câmara vem toda seguida
separada por virgulas, aqui irá dividir-se a informação pelas virgulas criando uma array
       valid_locate =(rx_msg[0:1]) #Vai guardar a informação se a imagem foi adquirida corretamente

        #Valores imagem referência (valores do frame de referência criado)
        refx1 =  -30.475324
        refy1 = 65.602353
        refz1 = 219.600016
        refx2 = -31.856441
        refy2 = 180.180622
        refz2 = 218.980757
        refx3 = 36.503447
        refy3 = 162.041334
        refz3 = 232.769182


        if valid_locate=="1":#Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
```

```
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x1 = round(x1, 4)
            y1 = round(y1, 4)
            z1 = round(z1, 4)
            x2 = round(x2, 4)
            y2 = round(y2, 4)
            z2 = round(z2, 4)

            #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência
            visp1 = [x1, y1, z1*-1]
            visp2 = [x2, y2, z2*-1]

            laser_off = trispector.write("set laser off") #Desativa o laser

        if valid_locate!="1": #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta
e volta ao inicio
            tp_log("Erro a Localizar!")
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 33)
            movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
            movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400)
            time.sleep(0.1)
            goto(1)
            continue

        set_output_register_int(0, 130)
        time.sleep(0.1)
        goto(1005)
        continue

    if line == 1005: #Irá verificar possiveis erros para garantir que está tudo correto para
posteriormente iniciar o aperto
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1006)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1006:

        trispector.closeIN(None)
```

```
        trispector.closeOUT(None)

        mwait(0)
        set_tcp("U_Tool_2")

        #Define os pontos de referência das três posições de aperto usadas para gerar o frame de
referência
        PF1 = posx(851.89, -557.37, 811.410, 90.51, 90, 0.02)
        PF2 = posx(737.74, -559.82, 811.800, 90.51, 90, 0.02)
        PF3 = posx(756.18, -549.46, 740.200, 90.51, 90, 0.02)
        Frame1 = posx(851.89, -557.37, 811.410, 91.202, -97.928, 89.802)

       #Vai calcular os novos pontos de referência de acordo com os valores obtidos com o sistema
de visão relativos às coordenadas X, Y e Z
        P1vis = [PF1[0]-visp1[1], PF1[1]-visp1[2], PF1[2]-visp1[0]]

        P2vis = [PF2[0]-visp2[1], PF2[1]-visp2[2], PF2[2]-visp2[0]]

        P3vis = [PF3[0]-visp2[1], PF3[1]-visp2[2], PF3[2]-visp2[0]]

        #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem p1 p2 e p3 que são
os novos pontos de referência
        p1 = P1vis[0:3]+PF1[3:]
        p2 = P2vis[0:3]+PF2[3:]
        p3 = P3vis[0:3]+PF3[3:]

       pose_user102 = calc_coord(p1, p2, p3, ref=DR_BASE, mod=0) #Cria o novo user frame com base
na nova posição da chapa obtida pelo sistema de visão

        overwrite_user_cart_coord(102, pose_user102, ref=DR_BASE, apply_mod=DR_PERMANENT)

        pose, ref = get_user_cart_coord(102)

        set_ref_coord(102) #Seleciona o novo frame criado com os valores do sistema de visão, que
irá corrigir os novos pontos de aperto das restantes porcas
        set_tcp("U_Tool_3")

        time.sleep(0.3)

        #A partir deste ponto irá passar-se para a fase de aperto das porcas
        set_output_register_int(3, 0)
        movej(posj(-13.91, 21.98, 58.44, -92.76, -15.71, 135.3), v=100, a=100, r=50) #Ponto para
troca Tool

        set_output_register_int(0, 140)

        movej(posj(-14.28, 30.46, 52.96, -112.43, -15.44, 206.38), v=120, a=120, r=10) #PA1
        movel(posx(-228.824, 9.311, -23.269, 88.398, 169.046, 178.016), v=250, a=250, r=10) #PA1
Tool 3 em U_Frame_1
        movel(posx(-229.993, 4.277, 8.952, 88.398, 169.046, 178.016), v=50, a=50) #PE1 Tool 3 em
U_Frame_1


        #Aperto P1
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        time.sleep(0.5)
        goto(1007)
        continue


    if line == 1007: #Depois de fazer o aperto da primeira porca, passoa para as posições relativas
ao aperto 2

        movel(posx(-228.824, 9.311, -23.269, 88.398, 169.046, 178.016), v=250, a=250, r=5) #PA1
Tool 3 em U_Frame_1
        movej(posj(-14.74, 27.6, 57.44, -106.53, -15.31, 200.31), v=120, a=120, r=10)
        movej(posj(-15.85, 14.27, 76.05, -87.08, -15.79, 180.19), v=100, a=100, r=10)
        movej(posj(-22.25, 11.21, 81.74, -83.19, -21.91, 171.49), v=100, a=100) #PA2
        set_output_register_int(0, 150)
```

```
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        movel(posx(2.194, 10.063, -22.760, 88.398, 169.046, 178.016), v=120, a=120) #PA2 Tool 3 em
U_Frame_1
        goto(1008)
        continue

    if line == 1008: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1009)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1009: #Aperto P2
        set_output_register_int(3, 0)
         movel(posx(2.048, 4.869, 5.144, 88.398, 169.046, 178.016), v=50, a=50) #PE2 Tool 3 em
U_Frame_1
        set_output_register_int(0, 160)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        time.sleep(0.2)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para a aparafusadora para iniciar o aperto
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        time.sleep(0.5)
        movel(posx(2.194, 10.063, -22.760, 88.398, 169.046, 178.016), v=120, a=120) #PA2 Tool 3 em
U_Frame_1
        goto(1010)
        continue

    if line == 1010: #Passagem do P2 para o P3
        set_output_register_int(0, 170)
        movej(posj(-25.71, 7.89, 84.98, -84.12, -25.86, 174.79), v=80, a=100) #PA3
         movel(posx(117.021, 9.738, -32.435, 88.398, 169.046, 178.016), v=120, a=120) #PA3 Tool 3
em U_Frame_1
        goto(1011)
        continue

    if line == 1011: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1012)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
```

```
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1012: #Aperto P3
        set_output_register_int(3, 0)
         movel(posx(116.835, 4.040, 3.222, 88.398, 169.046, 178.016), v=50, a=50) #PE3 Tool3 em
U_Frame_1
        set_output_register_int(0, 180)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        time.sleep(0.2)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        time.sleep(0.5)
        movel(posx(117.021, 10.738, -32.435, 88.398, 169.046, 178.016), v=120, a=120) #PA3 Tool 3
em U_Frame_1
        goto(1013)
        continue

    if line == 1013: #Verificação de possiveis erros
        set_output_register_int(0, 190)
        time.sleep(0.1)
        goto(1014)
        continue

    if line == 1014:
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1015)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1015: #Aperto P4
        mwait(0)
        set_tcp("U_Tool_2")
        set_output_register_int(3, 0)
        movej(posj(-26.86, 13.83, 92.61, -5.15, -100.98, 115.4), v=80, a=100) #PA4
        movel(posx(310.860, 3.516, -38.607, 95.696, 168.722, -163.479), v=120, a=120) #PA4 Tool 2
em U_Frame_1
       movel(posx(311.546, -3.354, -3.927, 95.698, 168.723, -163.476), v=50, a=50) #PE4 Tool 2 em
U_Frame_1
        set_output_register_int(0, 200)
        time.sleep(0.2)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
```

```
        time.sleep(0.5)
        movel(posx(310.860, 3.516, -38.607, 95.696, 168.722, -163.479), v=120, a=120) #PA4 Tool 2
em U_Frame_1
        goto(1016)
        continue

    if line == 1016: #Passagem do P4 para o P5
        movej(posj(-17.87, 20.29, 82.96, -2.26, -103.59, 117.36), v=50, a=50)
        set_output_register_int(0, 210)
        time.sleep(1)
        goto(1017)
        continue

    if line == 1017: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1018)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) # Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1018:

        free_zone=get_input_register_int(4)
        if(free_zone==222): #Verifica se a zona está livre
            time.sleep(0.1)
            set_output_register_int(4, 888) #Info para PLC Zona Ocupada
            goto(1019)
            continue
        else:
            time.sleep(0.1)
            goto(1018)
            continue

    if line == 1019: #Aperto P5
        set_output_register_int(3, 0)
        set_output_register_int(4, 888) #Info para PLC Zona Ocupada
        set_tcp("U_Tool_3")
        movej(posj(-11.21, 46.73, 44.83, -79.67, -11.32, 183.7), v=220, a=150, r=20) #PA5
        movel(posx(-360.490, 272.324, -45.634, 77.683, -172.331, 166.323), v=120, a=120, r=5) #PA5
Tool 3
        movel(posx(-359.573, 275.526, -13.697, 77.683, -172.331, 166.324), v=60, a=60) #PE5 Tool
3
        set_output_register_int(0, 220)
        time.sleep(0.1)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.5)
        movel(posx(-360.490, 272.324, -45.634, 77.683, -172.331, 166.323), v=120, a=120, r=5) #PA5
Tool 3
        goto(1020)
        continue
```

```
    if line == 1020: #Passagem do P5 para o P6

        set_output_register_int(0, 230)
        movej(posj(-9.1, 47.33, 52.49, -41.58, -13.67, 146.02), v=150, a=100, r=10) #PA6
        movel(posx(-359.937, 369.865, -63.965, 77.689, -172.331, 166.327), v=120, a=120) #PA6 Tool
3
        goto(1021)
        continue

    if line == 1021: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1022)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1022: #Aperto P6
        set_output_register_int(3, 0)
        movel(posx(-358.876, 374.742, -27.101, 77.683, -172.331, 166.323), v=50, a=50) #PE6 Tool
3
        set_output_register_int(0, 240)
        time.sleep(0.1)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.5)
        movel(posx(-359.937, 369.865, -63.965, 77.689, -172.331, 166.327), v=120, a=120) #PA6 Tool
3
        goto(1023)
        continue

    if line == 1023: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1024)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)

                time.sleep(0.1)
                goto(1024)
                continue

    if line == 1024:
        #Saida para a posição inicial depois de efetuar todos os apertos
        set_ref_coord(DR_BASE)
        set_output_register_int(0, 250)
        movej(posj(-16.84, 21.39, 68.64, 0.01, -90.03, 109.38), v=150, a=120, r=50)
        movej(posj(2.56, -26.6, 116.57, 2.88, -88.76, 128.26), v=150, a=120, r=50)
```

```
        movej(posj(103.74, -25.78, 122.5, 2.94, -97.34, 72.17), v=150, a=120, r=50)

        movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=200, a=150, r=80)
        movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=200, a=150, r=50)
        movej(carregamento, v=20, a=10)
        set_output_register_int(4, 0) #Info para PLC Zona Livre

        set_tcp("U_Tool_2")
        set_ref_coord(DR_BASE)
        set_output_register_int(3, 0)
        set_tcp("U_Tool_2")
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 410)
        goto(1)
        continue



########################################################################################################################
    #Segunda parte do codigo relativo à jante 16, a ideologia desta segunta parte do código é igual à primeira apenas com as correções das posições devido à diferença de alturas

    if line == 1102: #Jante 16

        set_tcp("U_Tool_1")
        set_ref_coord(DR_BASE)

        #Movimento de Saida Cuba
        set_output_register_int(0, 101)
        movej(carregamento, v=20, a=10)
         movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50) #Função usada para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a velocidade "v", a aceleracao "a" e o radius "r"
        set_output_register_int(0, 105)
        movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=80, a=80, r=50)
        movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=100, a=100, r=50)
        goto(1103)
        continue

    if line == 1103:
        set_output_register_int(0, 110)
        free_zone=get_input_register_int(4)
         if(free_zone==222): #Verifica se a zona de aperto está livre, caso esteja livre avança senão espera que a mesma esteja
            time.sleep(0.1)
             set_output_register_int(4, 888) #Envia a informacao para o PLC que a zona irá ficar ocupada pelo robô
            goto(1104)
            continue
        else:
            goto(1103)
            continue

    if line == 1104:
        set_output_register_int(0, 120)
        car_pos=get_input_register_int(2)
        if(car_pos==5555): #Verifica se a carrinha está na posição correta
            set_output_register_int(3, 777)
            time.sleep(0.1)
            goto(1144)
            continue

        if(car_pos==9999): #Caso a informação que chega ao robô indique que a carrinha não está na posição correta o robô voltará à posição inicial (carrrgamento) e o código voltará à parte da verificação inicial da posição da carrinha
            set_output_register_int(3, 777)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=0)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_output_register_int(3, 777)
```

```
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 410)
            time.sleep(2)
            goto(1)
            continue

    if line == 1144: #Se a carrinha estiver na posição correta o programa prossegue para a aquisição
da imagem pelo sistema de visão

        #Entrada Carro
        movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=100, a=100, r=50)
        movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=100, a=100, r=50)

        #Posição para se dar ao inicio da aquisição da imagem pelo sistema de visão
        movej(posj(-11.26, 44.37, 44.27, 0, -88.64, 11.26), v=100, a=100)
        mwait(0)
        set_tcp("U_Tool_1")
        movel(posx(980.0, -160.0, 730.00, 162.59, 0, -162.59), v=100, a=150)
        time.sleep(1)
        goto(11444)
        continue

    if line == 11444:
        set_output_register_int(3, 0)
         laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da
imagem
       if (laser_on != 'OK'): #Verifica se o laser foi ativado, caso não tenha sido o robô abortará
a tarefa de aquisição da imagem e voltará para a posição inicial
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 11)
            movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
            movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400)
            time.sleep(0.1)
            goto(1)
            continue

        job = trispector.write("set job \"J 16\"") #Seleciona o programa "J 16" para ser utilizado
na câmara
         if (job != 'OK'): #Verifica se o programa foi selecionado corretamente caso contrário
aborta e voltará ao inicio
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 11)
            movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
            movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400)
            time.sleep(0.1)
            goto(1)
            continue

         #Caso o laser a seleção do programa tenham sido executados com sucesso irá passar-se à
aquisição da imagem
        time.sleep(0.3)
        amovel(posx(720.0, -160.0, 730.0, 162.59, 0, -162.59), v=50, a=200)
```

```
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio
à aquisição da imagem
        mwait(0)
        rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida

        rxd =(rx_msg[0:128])
        data = rx_msg.split(', ') #Uma vez que a informação enviada pela câmara vem toda seguida
separada por virgulas, aqui irá dividir-se a informação pelas virgulas criando uma array
       valid_locate =(rx_msg[0:1]) #Vai guardar a informação se a imagem foi adquirida corretamente

        #Valores imagem referência (valores do frame de referência criado)
        refx1 =  -43.885886
        refy1 = 57.404575
        refz1 = 216.409296
        refx2 = -43.439081
        refy2 = 172.254235
        refz2 = 215.272629
        refx3 = 36.503447
        refy3 = 162.041334
        refz3 = 232.769182


        if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x1 = round(x1, 4)
            y1 = round(y1, 4)
            z1 = round(z1, 4)
            x2 = round(x2, 4)
            y2 = round(y2, 4)
            z2 = round(z2, 4)

            #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência
            visp1 = [x1, y1, z1*-1]
            visp2 = [x2, y2, z2*-1]

            laser_off = trispector.write("set laser off") #Desativa o laser

        if valid_locate!="1": #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta
e volta ao inicio
            tp_log("Erro a Localizar!")
            trispector.closeIN(None)
            trispector.closeOUT(None)
            set_output_register_int(1, 33)
            movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
            movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
            movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
            movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
            movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
            movej(carregamento, v=20, a=10)
            set_tcp("U_Tool_2")
            set_output_register_int(4, 0) #Info para PLC Zona Livre
            set_output_register_int(0, 400)
            time.sleep(0.1)
            goto(1)
            continue

        set_output_register_int(0, 130)
        goto(1105)
        continue

    if line == 1105: #Irá verificar possiveis erros para garantir que está tudo correto para
posteriormente iniciar o aperto
            step_ack=get_input_register_int(2)
```

```
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1106)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1106:

        trispector.closeIN(None)
        trispector.closeOUT(None)

        mwait(0)
        set_tcp("U_Tool_2")

        #Define os pontos de referência das três posições de aperto usadas para gerar o frame de
referência
        PF1 = posx(861.03, -563.69, 825.11, 90.69, 94.02, -0.31)
        PF2 = posx(746.05, -563.4, 823.92, 90.69, 94.02, -0.31)
        PF3 = posx(765.24, -554.02, 750.5, 90.69, 94.02, -0.31)
         Frame1 = posx(861.03, -563.69, 825.11, 89.931, -97.298, 90.598) #Frame Referência - +X
direção porta, +Y direção travão, +Z direção mala

       #Vai calcular os novos pontos de referência de acordo com os valores obtidos com o sistema
de visão relativos às coordenadas X, Y e Z
        P1vis = [PF1[0]-visp1[1], PF1[1]-visp1[2], PF1[2]-visp1[0]]

        P2vis = [PF2[0]-visp2[1], PF2[1]-visp2[2], PF2[2]-visp2[0]]

        P3vis = [PF3[0]-visp2[1], PF3[1]-visp2[2], PF3[2]-visp2[0]]

        #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem p1 p2 e p3 que são
os novos pontos de referência
        p1 = P1vis[0:3]+PF1[3:]
        p2 = P2vis[0:3]+PF2[3:]
        p3 = P3vis[0:3]+PF3[3:]

       pose_user104 = calc_coord(p1, p2, p3, ref=DR_BASE, mod=0) #Cria o novo user frame com base
na nova posição da chapa obtida pelo sistema de visão

        overwrite_user_cart_coord(104, pose_user104, ref=DR_BASE, apply_mod=DR_PERMANENT)

        pose, ref = get_user_cart_coord(104)

        set_ref_coord(104) #Seleciona o novo frame criado com os valores do sistema de visão, que
irá corrigir os novos pontos de aperto das restantes porcas
        set_tcp("U_Tool_3")

        time.sleep(0.3)

        #A partir deste ponto irá passar-se para a fase de aperto das porcas
        set_output_register_int(3, 0)
         movej(posj(-13.91, 21.98, 58.44, -92.76, -15.71, 135.3), v=100, a=100, r=50) #Ponto para
troca Tool

        set_output_register_int(0, 140)
```

```
        movej(posj(-14.28, 30.46, 52.96, -112.43, -15.44, 206.38), v=120, a=120, r=10) #PA1
        movel(posx(-228.013, 9.899, -32.276, 90.917, 171.521, -178.713), v=250, a=250, r=10) #PA1
Tool 3 em U_Frame_3
        movel(posx(-227.937, 5.225, -0.902, 90.917, 171.522, -178.713), v=50, a=50) #PE1 Tool 3 em
U_Frame_3

        #Aperto P1
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        time.sleep(0.5)
        goto(1107)
        continue


    if line == 1107: #Depois de fazer o aperto da primeira porca, passoa para as posições relativas
ao aperto 2


        movel(posx(-228.013, 9.899, -32.276, 90.917, 171.521, -178.713), v=250, a=250, r=10) #PA1
Tool 3 em U_Frame_3
        movej(posj(-14.74, 27.6, 57.44, -106.53, -15.31, 200.31), v=120, a=120, r=10)
        movej(posj(-15.85, 14.27, 76.05, -87.08, -15.79, 180.19), v=100, a=100, r=10)
        movej(posj(-22.25, 11.21, 81.74, -83.19, -21.91, 171.49), v=100, a=100) #PA2
        set_output_register_int(0, 150)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        movel(posx(3.010, 10.789, -29.879, 90.918, 171.521, -178.712), v=120, a=120) #PA2 Tool 3
em U_Frame_3
        goto(1108)
        continue

    if line == 1108: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1109)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1109: #Aperto P2
        set_output_register_int(3, 0)
        movel(posx(3.078, 4.881, 2.605, 90.917, 171.521, -178.713), v=50, a=50) #PE2 Tool 3 em
U_Frame_3
        set_output_register_int(0, 160)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        time.sleep(0.2)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        time.sleep(0.5)
        movel(posx(3.010, 10.789, -29.879, 90.918, 171.521, -178.712), v=120, a=120) #PA2 Tool 3
em U_Frame_3
```

```
        goto(1110)
        continue

    if line == 1110: #Passagem do P2 para o P3
        set_output_register_int(0, 170)
        movej(posj(-25.71, 7.89, 84.98, -84.12, -25.86, 174.79), v=80, a=100) #PA3
        movel(posx(117.253, 10.634, -36.224, 90.782, 170.537, -178.846), v=120, a=120) #PA3 Tool
3 em U_Frame_3
        goto(1111)
        continue

    if line == 1111: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1112)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1112: #Aperto P3
        set_output_register_int(3, 0)
        movel(posx(117.357, 3.970, 3.760, 90.802, 170.536, -178.826), v=50, a=50) #PE3 Tool3 em
U_Frame_3
        set_output_register_int(0, 180)
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        time.sleep(0.2)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        time.sleep(0.5)
        movel(posx(117.253, 10.634, -36.224, 90.782, 170.537, -178.846), v=120, a=120) #PA3 Tool
3 em U_Frame_3
        goto(1113)
        continue

    if line == 1113:
        set_output_register_int(0, 190)
        time.sleep(0.1)
        goto(1114)
        continue

    if line == 1114: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1115)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
```

```
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1115: #Aperto P4
        mwait(0)
        set_tcp("U_Tool_2")
        set_output_register_int(3, 0)
        movej(posj(-26.99, 13.58, 93.02, -5.57, -100.31, 115.43), v=80, a=100) #PA4
        movel(posx(313.405, 4.783, -30.097, 85.089, 170.425, -176.229), v=120, a=120) #PA4 Tool 2
em U_Frame_3
        movel(posx(312.942, -0.617, 2.048, 85.089, 170.425, -176.229), v=50, a=50) #PE4 Tool 2 em
U_Frame_3
        set_output_register_int(0, 200)
        time.sleep(0.2)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.5)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        time.sleep(0.5)
        movel(posx(313.405, 4.783, -30.097, 85.089, 170.425, -176.229), v=120, a=120) #PA4 Tool 2
em U_Frame_3
        goto(1116)
        continue

    if line == 1116: #Passagem do P4 para o P5
        movej(posj(-17.87, 20.29, 82.96, -2.26, -103.59, 117.36), v=50, a=50)
        set_output_register_int(0, 210)
        time.sleep(1)
        goto(1117)
        continue

    if line == 1117: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1118)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1118:
        free_zone=get_input_register_int(4)
        if(free_zone==222): #Verifica se a zona está livre
            time.sleep(0.1)
            set_output_register_int(4, 888) #Info para PLC Zona Ocupada
            goto(1119)
            continue
```

```
        else:
            time.sleep(0.1)
            goto(1118)
            continue

    if line == 1119: #Aperto P5
        set_output_register_int(3, 0)
        set_output_register_int(4, 888) #Info para PLC Zona Ocupada
        set_tcp("U_Tool_3")
        movej(posj(-11.21, 46.73, 44.83, -79.67, -11.32, 183.7), v=220, a=150, r=20) #PA5
       movel(posx(-357.681, 272.845, -56.569, 87.244, -172.072, 176.963), v=120, a=120, r=5) #PA5
Tool 3 em U_Frame_3
        movel(posx(-357.495, 276.665, -29.037, 87.244, -172.073, 176.963), v=60, a=60) #PE5 Tool
3 em U_Frame_3
        set_output_register_int(0, 220)
        time.sleep(0.1)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.5)
       movel(posx(-357.681, 272.845, -56.569, 87.244, -172.072, 176.963), v=120, a=120, r=5) #PA5
Tool 3 em U_Frame_3
        goto(1120)
        continue

    if line == 1120: #Passagem do P5 para o P6
        set_output_register_int(0, 230)
        movej(posj(-9.1, 47.33, 52.49, -41.58, -13.67, 146.02), v=150, a=100, r=10) #PA6
       movel(posx(-357.515, 371.111, -71.685, 87.246, -172.072, 176.963), v=120, a=120) #PA6 Tool
3 em U_Frame_3
        goto(1121)
        continue

    if line == 1121: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1122)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)
                movej(posj(-20.41, 23.1, 64.06, 0, -87.17, 112.8), v=50, a=50, r=50)
                movej(posj(10.63, -25.59, 113.57, 3.03, -87.2, 101.83), v=50, a=50, r=50)
                movej(posj(98.73, -9.59, 111.82, 3.03, -102.59, 77.47), v=50, a=50, r=50)
                movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=50, a=50, r=50)
                movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=50, a=50, r=50)
                movej(carregamento, v=20, a=10)
                set_output_register_int(3, 0)
                set_tcp("U_Tool_2")
                set_output_register_int(4, 0) #Info para PLC Zona Livre
                set_output_register_int(0, 410)
                time.sleep(0.1)
                goto(1)
                continue

    if line == 1122: #Aperto P6
        set_output_register_int(3, 0)
        movel(posx(-357.331, 374.922, -44.335, 87.244, -172.073, 176.962), v=50, a=50) #PE6 Tool
3 em U_Frame_3
        set_output_register_int(0, 240)
        time.sleep(0.1)
        mwait(0)
        set_output_register_int(2, 55) #Sinal para iniciar o aperto da aparafusadora
        time.sleep(0.2)
        wait_torch() #Espera pelo torque pretendido da aparafusadora
        #time.sleep(0.5)
       movel(posx(-357.515, 371.111, -71.685, 87.246, -172.072, 176.963), v=120, a=120) #PA6 Tool
3 em U_Frame_3
```

```
        goto(1123)
        continue

    if line == 1123: #Verificação de possiveis erros
            step_ack=get_input_register_int(2)
            if(step_ack==5555):
                set_output_register_int(3, 777)
                time.sleep(0.1)
                goto(1124)
                continue

            error=get_input_register_int(2)
            if(error==9999):
                set_output_register_int(3, 777)

                time.sleep(0.1)
                goto(1124)
                continue

    if line == 1124:
        #Saida para a posição inicial depois de efetuar todos os apertos
        set_ref_coord(DR_BASE)
        set_output_register_int(0, 250)
        movej(posj(-16.84, 21.39, 68.64, 0.01, -90.03, 109.38), v=150, a=120, r=50)
        movej(posj(2.56, -26.6, 116.57, 2.88, -88.76, 128.26), v=150, a=120, r=50)
        movej(posj(103.74, -25.78, 122.5, 2.94, -97.34, 72.17), v=150, a=120, r=50)

        movej(posj(142.69, 7.76, 105.31, 73.37, -56.11, 118.17), v=200, a=150, r=80)
        movej(posj(142.01, 15.84, 101.65, 70.18, -56.9, 123.43), v=200, a=150, r=50)
        movej(carregamento, v=20, a=10)
        set_output_register_int(4, 0) #Info para PLC Zona Livre

        set_tcp("U_Tool_2")
        set_ref_coord(DR_BASE)
        set_output_register_int(3, 0)
        set_tcp("U_Tool_2")
        set_output_register_int(4, 0) #Info para PLC Zona Livre
        set_output_register_int(0, 410)
        time.sleep(0.1)
        goto(1)
        continue

    #Fim do código
```

Anexo 3 – Código Porteur do robô 1

```
from DRCF import*
import sys
import os
import time
import socket

#Entrada 1 + 2 = Cameras de segurança
#Entrada 3 + 4 = Interlock Reset
#Entrada 5 + 6 = Safety Zone Dynamic Enable (L)
#Entrada 8 = Feedback Home pos

#Saida 1 + 2 = Emergency Stop excl No Loopback Input (L)
#Saida 3 + 4 = Designated Zone = Home pos
#saida 5 = Zona Colisão
#Saida 7 = Task Operating (L)
#Saida 8 = Safe Operating Stop (L)
#Saida 13 = Safety Zone Dynamic Enable (L) = Entrada 5 + 6

#TCP da tool
#U_Tool_1 = -194.79, 1.113, 167.907, 0, 0, 0

global result
```

```
result = 0

class DoosanSick:

#*********************************

    #Função que inicializa e faz a conexão da câmara
    def __init__(self, ip, portIN, portOUT, state=0):

        self.ip = ip
        self.portIN = portIN
        self.portOUT = portOUT
        self.state = state
        client = socket.socket()
        client.settimeout(3)
        socket.setdefaulttimeout(3)

        try:
            client.connect((self.ip, self.portIN))
            self._socketIN = client_socket_open(self.ip, self.portIN)
            self.state=1

        except Exception as e:
            self.state=2
            tp_log("Socket INPUT falha de ligação. Erro: {0}".format(str(e)))

        time.sleep(0.1)

        try:
            client.connect((self.ip, self.portOUT))
            self._socketOUT = client_socket_open(self.ip, self.portOUT)
            self.state=1
        except Exception as e:
            self.state=2
            tp_log("Socket OUTPUT falha de ligação. Erro: {0}".format(str(e)))

#*********************************

    #Função usada para escrever no socket
    def write(self, cmd, socket=None):

        if socket == None:
            socket = self._socketIN

        cmd = bytes(cmd, encoding="ascii")

        STX = client_socket_write(socket, b"\x02")
        res = client_socket_write(socket, cmd)
        STX = client_socket_write(socket, b"\x03")
        time.sleep(0.1)
        res, rx_data = client_socket_read(socket, 10, 1)

        """
        rxd=None
        if res > 1:
            rx_msg = rx_data.decode()
            rxd =(rx_msg[1:3])
        if res == -1:
            tp_log("Erro durante a escrita comando no socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a escrita comando no socket: Erro de Socket")

        return rxd
        """

#*********************************

    #Função usada para ler a informação do command channel
    def readcmd(self, length, timeout, socket=None ):

        if socket == None:
           socket = self._socketIN
```

```
        res, rx_data = client_socket_read(socket, length, timeout)

        """
        rx_msg = rx_data.decode()
        rxd =(rx_msg[1:3])

        if res == -1:
            tp_log("Erro durante a leitura do socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a leitura do socket: Erro de Socket")
        elif res == -3:
            tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
        return rxd
        """

#*********************************
    #Função usada para ler do socket a informação recebida
    def readata(self, length, timeout, socket=None ):

        if socket == None:
           socket = self._socketOUT

        rx_msg = '0'
        res = 0
        rx_data = 0

        res, rx_data = client_socket_read(socket, length, timeout)

        if res == -1:
            tp_log("Erro durante a leitura dados do socket : Server não conectado")
            rx_msg = '0, 0, 0, 0, 0, 0, 0, 0, 0, 0'
            return rx_msg
        elif res == -2:
            tp_log("Erro durante a leitura dados do socket: Erro de Socket")
            rx_msg = '0, 0, 0, 0, 0, 0, 0, 0, 0, 0'
            return rx_msg
        elif res == -3:
            tp_log("Erro durante a leitura dados do socket: Tempo de espera excedido")
            rx_msg = '0, 0, 0, 0, 0, 0, 0, 0, 0, 0'
            return rx_msg
        else:
            rx_msg = rx_data.decode()

            return rx_msg

#*********************************

    #Função que irá fechar as conexões dos socket IN e OUT utilizados para efetuar a comunicação
com a câmara
    def closeIN(self, socket=None):

        if socket == None:
            socket = self._socketIN

        client_socket_close(socket)

    def closeOUT(self, socket=None):

        if socket == None:
            socket = self._socketOUT

        client_socket_close(socket)

#Função que vai aguardar que a aparafusadora esteja pronta novamente (que terminou o ciclo)
def wait_clear() :
        tt = True
        while tt:
            aperto=get_input_register_int(1)
            if(aperto==0):
                time.sleep(0.2)
                tt = False
```

```
            time.sleep(0.2)

#Função que irá esperar que o torque pretendido seja atingido para desativar a aparafusadora
def wait_torque() :
            yy = True
            zz = 0
            set_output_register_int(2, 55) #Manda sinal para o PLC para ativar a aparafusadora
            while yy:
                step_ack=get_input_register_int(1)  #Guarda  na  variavel  step_ack  o  valor  do
relatório
                if(step_ack==1111)  or  (step_ack==2222)  or  (step_ack==4444): #Só  entra  neste  if
quando recebe o relatório, seja OK ou NOK
                    time.sleep(0.1)
                    set_output_register_int(2, 0) #Output para desativar aparafusadora
                    result=get_input_register_int(1) #Guarda o relatório do aperto em result
                    yy = False
                    set_output_register_int(3, 777) #Info para PLC ACK (Informa o PLC que recebeu
a tomou a decisao relativa ao relatorio)
                    wait_clear() #Aguarda pelo termino do ciclo da aparafusadora
                    #Vai verificar o relatório enviado pela aparafusadora
                    if result==4444: #Valor caso o relatório seja OK
                        set_output_register_bit(10, 1)
                    if result!=4444: #Valor caso o relatório seja NOK
                        set_output_register_bit(10, 0)

                #TimeOut de 50 segundos para caso nao receber resposta da aparafusadora, o loop
termina e a ferramenta é desligada por segurança.
                if zz >= 50:
                    yy = False
                    set_output_register_int(2, 0)
                time.sleep(0.1)
                zz = zz + 0.1

#Função utilizada para confirmar que a aparafusadora está a fazer pressão no parafuso (20N)
def wait_touch() :
            xx = True
            while xx:
                xx = check_force_condition(axis=DR_AXIS_Z, max=20, ref=DR_TOOL)
                wait(0.1)


#*****************************************************************************

#Definição de algumas variáveis globais e outputs importantes
set_motion_end(DR_CHECK_ON) #Funçãoo para que o robo espere terminar o movimento atual antes de
executar o próximo comando
set_output_register_int(0, 0) # Info para PLC qual etapa está o Robô
set_output_register_int(1, 0) # Info para PLC resultado Visão
set_digital_output(13, OFF)
set_tool ("U_Tool_W1")  #Seleciona a tool que se pretende
set_tcp("U_Tool_0")     #Seleciona o TCP da respetiva tool
set_ref_coord(DR_BASE)  #Define o referencial de coordenadas para movimentar o robô
time.sleep(0.5)

#Definição de uma variável global que será usada posteriormente para indicar para onde irá a
execução do programa
def goto(linenum):
    global line
    line = linenum
line = 1

#Vai fazer conecção com a câmara de visão
camera_ip = "192.168.2.81"
trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114, state=0)
time.sleep(0.5)

#Verifica se a conecção foi estabelecida
if trispector.state != 1:
    set_output_register_int(1, 20) # Info para PLC resultado Visão (não estabelecida)
    set_output_register_int(0, 210) #Infor para PLC da etapa
    time.sleep(2)
```

```
    set_output_register_int(0, 401) #Indica que não foi efetuada corretamente a ligação com a
camara ao PLC
    exit() #Termina o programa

laser_off = trispector.write("set laser off") #Desliga o lazer

#Vai fechar os sockets uma vez que terminou a comunicação com a câmara
trispector.closeIN(None)
trispector.closeOUT(None)

#Dá set a 0 os seguintes outputs
set_output_register_int(0, 0) # Info para PLC etapa do Robô
set_output_register_int(1, 0) # Info para PLC resultado Visão
set_output_register_int(2, 0) # Pedido aperto
set_output_register_int(3, 0) # Limpa a informação anterior sobre ACK

time.sleep(0.5)

if get_digital_input(12)==1: #Segurança
 exit()
time.sleep(0.1)

zero = check_robot_mastering() #Verifica se o robô está num estado que requer masterização ou não
(calibração)

if zero == 1: #Se necessitar de recalibração
    move_home(DR_HOME_TARGET_USER) #Move-se para a posição Home definida pelo utilizador para fazer
recalibração


movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
mwait(0)
set_output_register_int(0, 402)
set_output_register_int(4, 0) # Info para PLC zona ocupada

set_digital_output(15, ON)
set_digital_output(16, ON)
time.sleep(10)
set_digital_output(15, OFF)
set_digital_output(16, OFF)

set_output_register_int(0, 402)
set_output_register_int(4, 0) # Info para PLC zona ocupada



#Vai definir os pontos de aproximacao e de aperto dos 4 parafusos de referência para posterior
correção do frame
#Foram definidos 2 conjuntos o 433 e o 753 que correspondem a discos diferentes devido ao diâmetro
e às pinças serem diferentes o que irá afetar no posicionamento dos parafusos

#**Pontos de aperto 433 (disco 1) chave 10

#Ponto1 direito 433

System_433d_pa1 = posx(-0.810, 4.800, 18.750, 81.53, -168.44, -69.0)

System_433d_p1 = posx(-1.050, 0.600, -0.560, 81.53, -168.44, -69.0)

#Ponto2 direito 433

System_433d_pa2 = posx(185.550, 4.910, 22.320, 80.36, -168.15, -34.44)

System_433d_p2 = posx(184.700, -0.090, -1.840, 80.36, -168.15, -34.44)

#Ponto3 direito 433

System_433d_pa3 = posx(117.710, 158.820, 19.530, 87.68, -168.34, -10.71)

System_433d_p3 = posx(117.570, 155.160, 1.760, 87.68, -168.34, -10.71)

#TXDireito 433
```

```
System_433d_pa4 = posx(67.770, 103.190, -5.130, 128.49, -80.13, 14.12)

System_433d_p4 = posx(77.860, 90.420, -2.310, 127.09, -80.48, 14.36)

#*****************************************************************************************
***
#*****************************************************************************************
***

#Ponto1 esquerdo 433

System_433e_pa1 = posx(-0.250, 2.350, -15.190, 88.88, -11.01, 128.45)

System_433e_p1 = posx(-0.300, -0.610, 0.020, 88.88, -11.01, 128.45)

#Ponto2 esquerdo 433

System_433e_pa2 = posx(182.980, 4.150, -18.980, 73.75, -11.44, -129.07)

System_433e_p2 = posx(181.850, 0.260, 1.050, 73.75, -11.44, -129.07)

#Ponto3 esquerdo 433

System_433e_pa3 = posx(112.410, 158.810, -22.640, 79.96, -12.37, -102.83)

System_433e_p3 = posx(111.550, 153.960, -0.160, 79.96, -12.37, -102.83)

#TXEsquerdo 433

System_433e_pa4 = posx(62.010, 108.130, 5.510, 128.49, -97.63, 135.08)

System_433e_p4 = posx(76.300, 90.160, 2.450, 128.49, -97.63, 135.08)


#*****************************************************************************************

#*****************************************************************************************

#*****************************************************************************************

#**Pontos de aperto 753 (disco 2) chave 10

#Ponto1 direito 753

System_753d_pa1 = posx(-2.510, 3.510, 12.180, 89.95, -168.71, -57.18)

System_753d_p1 = posx(-0.510, 0.630, -0.040, 89.95, -168.71, -57.18)

#Ponto2 direito 753

System_753d_pa2 = posx(184.260, 5.720, 26.220, 85.9, -168.03, -47.72)

System_753d_p2 = posx(183.870, 0.320, 0.630, 85.9, -168.03, -47.72)

#Ponto3 direito 511

System_753d_pa3 = posx(117.370, 160.010, 26.230, 87.24, -169.03, -31.32)

System_753d_p3 = posx(117.120, 154.930, 0.000, 87.24, -169.03, -31.32)

#TXDireito 753

System_753d_pa4 = posx(55.700, 117.840, -8.320, 130.78, -80.41, 10.23)

System_753d_p4 = posx(77.340, 92.280, -3.360, 130.4, -80.39, 10.31)

#*****************************************************************************************
***
#*****************************************************************************************
***
```

```
#Ponto1 esquerdo 753

System_753e_pa1 = posx(0.660, 4.780, -16.170, 91.5, -11.48, 123.66)

System_753e_p1 = posx(1.070, 1.600, 0.680, 91.5, -11.48, 123.66)

#Ponto2 esquerdo 753

System_753e_pa2 = posx(184.200, 2.520, -18.150, 90.68, -11.57, -121.91)

System_753e_p2 = posx(182.340, -0.140, 0.730, 61.37, -11.5, -147.72)

#Ponto3 esquerdo 753

System_753e_pa3 = posx(118.400, 158.970, -21.400, 74.06, -11.3, -168.32)

System_753e_p3 = posx(116.950, 155.550, 0.350, 74.06, -11.3, -168.32)

#TXEsquerdo 753

System_753e_pa4 = posx(56.840, 121.350, 8.590, 128.25, -97.81, 130.47)

System_753e_p4 = posx(80.290, 92.860, 3.620, 128.25, -97.81, 130.47)

#*********************************************************************************************
#*********************************************************************************************



#Inicio do programa de aperto
while True:

    time.sleep(0.1)

    #Vai ser feita a validação de qual tipo de disco é que chegou ao ponto de aperto para a
continuação do código de acordo com o dito disco
    while line == 1 and get_input_register_int(0) == 11 and get_digital_input(16)==0 :  # Disco
433
        set_output_register_int(0, 0) #Info para PLC etapa robo
        set_output_register_int(1, 0) #Info para PLC resultado visao
        set_output_register_int(2, 0) #Pedido aperto
        set_output_register_int(3, 777) # Info para PLC decisão recebida
        set_output_register_int(4, 0) # Info para PLC zona ocupada
        set_output_register_int(0, 111)
        traj=get_input_register_int(0)
        goto(1002) #Define a variável line a 1002 e será com esta variável que se irá controlar
qual parte do codigo correr a seguir
        break



    while line == 1 and get_input_register_int(0) == 12 and get_digital_input(16)==0 :  # Disco
753
        set_output_register_int(0, 0)   #Info para PLC etapa robo
        set_output_register_int(1, 0)   #Info para PLC resultado visao
        set_output_register_int(2, 0)   #Pedido aperto
        set_output_register_int(3, 777) #Info para PLC decisão recebida
        set_output_register_int(4, 0)   #Info para PLC zona ocupada
        set_output_register_int(0, 112)
        traj=get_input_register_int(0)
        goto(1102) #Define a variável line a 1102 e será com esta variável que se irá controlar
qual parte do codigo correr a seguir
        break



    #Vai verificar se a maquete chegou ao local, caso nao tenha chegado passa para o while em baixo
enquanto aguarda pelo disco
    while line == 1 and get_input_register_int(0) == 99 and get_digital_input(16)==0 :
        set_output_register_int(0, 0)
        set_output_register_int(1, 0)
        set_output_register_int(2, 0)
```

```
        set_output_register_int(3, 777)
        set_output_register_int(4, 0)
        time.sleep(0.5)
        traj=get_input_register_int(0)
        set_output_register_int(0, 999) # Fora serviço
        goto(99)
        continue

    #Irá permanecer aqui enquanto nao chega a maquete com os discos
    while line == 99:
        set_output_register_int(0, 999)
        outserv=get_input_register_int(0)
        if(outserv==0):
            time.sleep(0.2)
            set_output_register_int(0, 400)
            time.sleep(1)
            set_output_register_int(0, 0)
            goto(1)
            break
        time.sleep(2.1)
        set_output_register_int(3, 0)
        goto(99)
        break

##########################################################################################################
##########################################################
##########################################################################################################
##########################################################
#  Trajetória 11 433 (disco 1)

    while line == 1002:

        set_tcp("U_Tool_0")
        set_ref_coord(DR_BASE)

        #Faz a conecção com a câmara de visão
        camera_ip = "192.168.2.81"
        trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114, state=0)
        time.sleep(0.5)
        if trispector.state != 1: #Verifica se a conecção foi estabelecida corretamente
            set_output_register_int(1, 20)  #Info para PLC resultado Visão (não estabelecida)
            set_output_register_int(0, 210) #Infor para PLC da etapa
            time.sleep(3)
            set_output_register_int(0, 401) #Indica que não foi efetuada corretamente a ligação
com a camara ao PLC
            goto(1)
            continue

        job = trispector.write("set job \"433 55dir\"") #Seleciona o programa da câmara referente
ao disco 433 direito
        set_output_register_int(4, 888) # Info para PLC zona ocupada

        #Vai movimentar o robô para a posição para se dar ao inicio da aquisição da imagem pelo
sistema de visão
        movej(posj(53.85, 56.67, 59.84, -0.01, 63.49, 143.81), v=100, a=100, r=100)
        #Função usada para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a
velocidade "v", a aceleracao "a" e o radius "r"
        movel(posx(350.0, 552.8, 91.96, 0, 180, 90), v=100, a=100)

        set_output_register_int(3, 0)# Limpa a informação anterior sobre ACK
        set_output_register_int(0, 210) #Etapa do robô

        laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da
imagem
        time.sleep(0.3)
        amovel(posx(350.0, 164.03, 91.96, 0, -180, 90), v=50, a=200)
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio
à aquisição da imagem
        mwait(0)
        set_digital_output(15, ON) #Refere que se trata de valores do disco direito
        rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
```

```
        rxd =(rx_msg[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui irá dividir-se a informação pelas virgulas criando uma array
        data = rx_msg.split(', ')
        valid_locate  =(rx_msg[0:1])  #Vai  guardar  a  informação  se  a  imagem  foi  adquirida corretamente
        time.sleep(0.2)


        #Valores imagem referência (valores do frame de referência criado)
        refx1 = -74.745480
        refy1 = 134.420716
        refz1 = 213.746662
        refx2 = 110.374540
        refy2 = 147.874876
        refz2 = 214.483098
        refx3 = 28.898749
        refy3 = 292.389366
        refz3 = 249.882767

        if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência para saber quanto se moveu a chapa da posição de referência
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x3 =  float(data[7]) - refx3
            y3 =  float(data[8]) - refy3
            z3 =  float(data[9]) - refz3
            #Arredonda o valor da variável para 6 casas decimais.
            x1 = round(x1, 6)
            y1 = round(y1, 6)
            z1 = round(z1, 6)
            x2 = round(x2, 6)
            y2 = round(y2, 6)
            z2 = round(z2, 6)
            x3 = round(x3, 6)
            y3 = round(y3, 6)
            z3 = round(z3, 6)

            #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência em arrays
            d433visao1 = [x1, y1, -z1]
            d433visao2 = [x2, y2, -z2]
            d433visao3 = [x3, y3, -z3]

            #Define a posição exata dos 3 parafusos de referência
            d433pf1=posx(352.640, 332.070, -360.540, 29.17, -177.38, 52.84)
            d433pf2=posx(168.460, 317.250, -352.440, 66.04, -179.50, 126.72)
            d433pf3=posx(249.22, 172.71, -322.02, 15.22, -178.21, 91.68)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            d433visao1[0] = d433pf1[0] - d433visao1[0]
            d433visao1[1] = d433pf1[1] - d433visao1[1]
            d433visao1[2] = d433pf1[2] - d433visao1[2]
            d433visao2[0] = d433pf2[0] - d433visao2[0]
            d433visao2[1] = d433pf2[1] - d433visao2[1]
            d433visao2[2] = d433pf2[2] - d433visao2[2]
            d433visao3[0] = d433pf3[0] - d433visao3[0]
            d433visao3[1] = d433pf3[1] - d433visao3[1]
            d433visao3[2] = d433pf3[2] - d433visao3[2]

            #Acrescenta às novas posições o Rx, Ry e Rz das posições de referência pois vão ser iguais
            d433p1 = d433visao1[0:3]+d433pf1[3:]
            d433p2 = d433visao2[0:3]+d433pf2[3:]
            d433p3 = d433visao3[0:3]+d433pf3[3:]
```

```
            #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
            pose_user106 = calc_coord(d433p1, d433p2, d433p3, ref=DR_BASE, mod=0)

            #Vai alterar a pose (posição e orientação no espaço 3D) e o sistema de coordenadas de
referência para o novo sistema de coordenadas calculado
            overwrite_user_cart_coord(106, pose_user106, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(106)

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate!="1":
            set_output_register_int(1, 10)
            laser_off = trispector.write("set laser off") #Desliga o laser
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 401)
            time.sleep(1)
            set_output_register_int(1, 0)
            goto(1)
            continue

        #Sendo necessario saber a posição exata dos parafusos em ambos os discos, agora vai se
fazer o mesmo mas para o disco esquerdo
        #Isto é, vai se voltar a fazer a aquisição da imagem e o seu tratamento
        set_digital_output(15, ON)
        job = trispector.write("set job \"433 55esq\"") #Seleciona o programa relativo ao disco
esquerdo
        time.sleep(0.1)

        movel(posx(350.0, -8.24, 91.96, 174.09, 180, -95.91), v=100, a=100)
        time.sleep(0.3)
        amovel(posx(350.0, -386.7, 91.96, 128.95, -180, -141.05), v=50, a=200) #Posição para inicio
de aquisição da imagem
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio
à aquisição da imagem

        mwait(0)
        set_digital_output(16, ON)  #Refere que se trata de valores do disco esquerdo
        rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
        rxd =(rx_msg[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui
irá dividir-se a informação pelas virgulas criando uma array
        data = rx_msg.split(', ')
        valid_locate  =(rx_msg[0:1])  #Vai  guardar  a  informação  se  a  imagem  foi  adquirida
corretamente


        #Valores imagem referência do disco esquerdo (valores do frame de referência criado)
        refx1 = -79.326979
        refy1 = 228.698872
        refz1 = 213.101958
        refx2 = 105.795317
        refy2 = 221.585210
        refz2 = 214.578844
        refx3 = 29.718081
        refy3 = 74.771915
        refz3 = 250.553534


        if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
```

```
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x3 =  float(data[7]) - refx3
            y3 =  float(data[8]) - refy3
            z3 =  float(data[9]) - refz3
            x1 = round(x1, 6)
            y1 = round(y1, 6)
            z1 = round(z1, 6)
            x2 = round(x2, 6)
            y2 = round(y2, 6)
            z2 = round(z2, 6)
            x3 = round(x3, 6)
            y3 = round(y3, 6)
            z3 = round(z3, 6)

            #Guarda a diferença das posições dos pontos de aperto de referência
            e433visao1 = [x1, y1, -z1]
            e433visao2 = [x2, y2, -z2]

            #Define os pontos de referência das posições de aperto usadas para gerar o frame de
referência
            e433pf1=posx(356.63, -320.49, -360.85, 144.59, 176.19, 28.49)
            e433pf2=posx(176.21, -314.24, -352.04, 8.99, 176.59, -52.27)
            e433pf3=posx(252.12, -166.46, -322.35, 158.18, 179.3, 137.53)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir
da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            e433visao1[0] = e433pf1[0] - e433visao1[0]
            e433visao1[1] = e433pf1[1] - e433visao1[1]
            e433visao1[2] = e433pf1[2] - e433visao1[2]
            e433visao2[0] = e433pf2[0] - e433visao2[0]
            e433visao2[1] = e433pf2[1] - e433visao2[1]
            e433visao2[2] = e433pf2[2] - e433visao2[2]
            e433visao3[0] = e433pf3[0] - e433visao3[0]
            e433visao3[1] = e433pf3[1] - e433visao3[1]
            e433visao3[2] = e433pf3[2] - e433visao3[2]

            #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem os novos pontos
de referência
            e433p1 = e433visao1[0:3]+e433pf1[3:]
            e433p2 = e433visao2[0:3]+e433pf2[3:]
            e433p3 = e433visao3[0:3]+e433pf3[3:]

            #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
            pose_user108 = calc_coord(e433p1, e433p2, e433p3, ref=DR_BASE, mod=0)

            #Vai alterar a pose e o sistema de coordenadas de referência para o novo sistema de
coordenadas calculado
            overwrite_user_cart_coord(108, pose_user108, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(108)

            laser_off = trispector.write("set laser off") #Desliga o laser da câmara

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate!="1":
            set_digital_output(16, ON)
            set_output_register_int(1, 10)
            laser_off = trispector.write("set laser off") #Desliga o laser da câmara
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 401)
            time.sleep(1)
            set_output_register_int(1, 0)
            set_digital_output(16, OFF)
            goto(1)
            continue
```

```
        set_digital_output(16, ON)
        laser_off = trispector.write("set laser off") #Desliga o laser da câmara
        set_output_register_int(0, 220)
        trispector.closeIN(None)
        trispector.closeOUT(None)
        goto(1003)
        break

    while line == 1003:

        set_tcp("U_Tool_1")
        mwait(0)
        set_ref_coord(108) #Seleciona o novo frame criado com os valores do sistema de visão, que
irá corrigir os novos pontos de aperto das restantes porcas
        set_output_register_int(3, 0)# Limpa a informação anterior sobre ACK
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada

        #A partir deste ponto irá passar-se para a fase de aperto das porcas
        movej(posj(-41.6, 64.43, 55.36, 7.32, 60.42, -69.23), v=250, a=250, r=50)
        movej(posj(-43.71, 74.91, 47.92, -0.81, 56.03, -183.91), v=150, a=150, r=50)

        movel(System_433e_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
        mwait(0)
        set_output_register_int(0, 230) #Info PLC etapa do robô
        set_digital_output(13, ON)

        #Define o referêncial para o referêncial da tool pois para os apertos vai se usar 3 funções
para fazer o aperto
        #task_compliance_ctrl - O compliance permite que o robô possa sofrer pequenos desvios no
seu movimento para permitir a inserção no parafuso mais facilmente
        #set_stiffnessx - Com esta função vai se definir o valor da rigidez referente ao compliance
durante um determinado tempo
        #set_desired_force - Com a função force, aplica-se uma força no robo com a direção
pretendida
        set_ref_coord(DR_TOOL)
        #Aperto 1
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez do robô com os indicados durante 1 segundo
        fd = [0, 0, 50, 0, 0, 0] #Define o valor da força que se pretende aplicar no Z
        fctrl_dir= [0, 0, 1, 0, 0, 0] #Define o sentido da força, nesta caso no eixo Z positivo
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Aplica a força no robô

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP) #Paragem rápida e controlada do movimento do robô
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP) #Paragem rápida e controlada do movimento do robô
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -40.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        mwait(0)
        set_ref_coord(108) #Seleciona de novo o frame calculado

        #Começa a movimentar-se para as posicoes do parafuso 2
        movej(posj(-43.64, 68.97, 51.27, -0.77, 60.76, -183.91), v=100, a=100, r=50)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)# Limpa a informação anterior sobre ACK
        set_digital_output(13, OFF)

        movej(posj(-74.83, 49.69, 88.15, 0.58, 43.44, -153.88), v=100, a=100, r=50)
        set_output_register_int(0, 240)
```

```
        movel(System_433e_pa2, v=100, a=100, r=0) #Ponto de aproximação do segundo parafuso
        mwait(0)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP) #Paragem rápida e controlada do movimento do robô
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP) #Paragem rápida e controlada do movimento do robô
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -40.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_output_register_int(0, 245)

        goto(1034)
        break

    while line == 1034:

        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        set_ref_coord(108) #Seleciona de novo o frame calculado

        #Começa a movimentar-se para as posicoes do parafuso 3
        movej(posj(-87.51, 59.26, 100.14, 1.36, 22, -141.75), v=250, a=150, r=50)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)# Limpa a informação anterior sobre ACK
        set_output_register_int(0, 250)
        set_digital_output(13, OFF)

        movej(posj(-80.77, 58.58, 137.6, 2.13, -15.84, -103.47), v=250, a=150, r=50)

        movel(System_433e_pa3, v=100, a=100, r=0) #Ponto de aproximação do terceiro parafuso
        mwait(0)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
```

```
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(106)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0) # Limpa a informação anterior sobre ACK
        set_digital_output(13, OFF)
        set_output_register_int(0, 260)
        goto(1055)
        break

    #Vai verificar se pode passar ao aperto dos parafusos do disco direito
    while line == 1055:
        decision=get_input_register_int(2) #Lê o sinal enviado pelo PLC sobre a possibilidade de
avançar
        set_output_register_int(0, 270) #Info para PLC sobre etapa do robô
        set_output_register_int(3, 0)# Limpa a informação anterior sobre ACK

        #Vai verificar a resposta enviada pelo PLC
        if(decision==555): # 555 = Pode avançar
            set_output_register_int(3, 777) # Info para PLC decisão recebida
            time.sleep(0.1)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            goto(1005)
            break
        if(decision==999): # 999 = Não Pode Avançar
            set_output_register_int(3, 777) # Info para PLC decisão recebida
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            movej(posj(-39.09, 32.33, 116.07, 3.3, 39.01, -70.12), v=50, a=50, r=50)
            movej(posj(-23.48, 36.31, 113.85, 16.04, 15.59, -34.75), v=80, a=50, r=50)
            movej(posj(-12.82, 38.79, 113.27, 38.46, 12.65, 34.61), v=100, a=80, r=50)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 402) # Info para PLC etapa robô
            time.sleep(1)
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            set_output_register_int(1, 0) # Info para PLC resultado visao
            set_output_register_int(3, 0) # Limpa a informação anterior sobre ACK
            goto(1)
            break


    while line == 1005:

        #Código relativo ao aperto do disco direito
        mwait(0)
        #Seleciona o novo frame criado com os valores do sistema de visão relativo ao disco direito,
que irá corrigir os novos pontos de aperto das restantes porcas
        set_ref_coord(106)
        set_output_register_int(0, 300) # Info PLC sobre etapa robô
        set_output_register_int(3, 0)   # Limpa a informação anterior sobre ACK
        mwait(0)
        #Posições de aperto do primeiro parafuso do disco direito
        movej(posj(-39.09, 32.33, 116.07, 3.3, 39.01, -70.12), v=100, a=80, r=50)
        movej(posj(-23.48, 36.31, 113.85, 16.04, 15.59, -34.75), v=150, a=80, r=50)
        movej(posj(-12.82, 38.79, 113.27, 38.46, 12.65, 34.61), v=250, a=150, r=50)
        movej(posj(55.35, 47.13, 95.36, 6.41, 42.32, 100.11), v=150, a=150, r=50)
        movej(posj(62.2, 56.69, 105.32, 0.39, 19.34, 86.48), v=100, a=100)

        movel(System_433d_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
        mwait(0)
        set_output_register_int(0, 310)
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        set_digital_output(13, ON)
```

```
        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=80, a=50, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(106) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 320)
        set_digital_output(13, OFF)

        mwait(0)
        movej(posj(77.36, 60.03, 97.38, 0.71, 23.54, 137.05), v=150, a=150, r=10)

        movel(System_433d_pa2, v=100, a=100, r=0) #Ponto de aproximação do segundo parafuso
        mwait(0)
        set_digital_output(13, ON)
        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(106) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_digital_output(13, OFF)
        movej(posj(54.89, 56.44, 110.32, 3.93, 15.23, 128), v=150, a=150, r=10)
        movel(System_433d_pa3, v=100, a=100, r=0) #Ponto de aproximação do terceiro parafuso
```

```
        mwait(0)
        set_output_register_int(0, 330)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        set_output_register_int(0, 335)
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        release_force() #Desativa o force
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(106) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 340)
        set_digital_output(13, OFF)

        goto(1006)
        break
        exit()


    while line == 1006:
        free_zone=get_input_register_int(4) #Verifica se a zona está livre
        if(free_zone==3333):
            set_output_register_int(3, 777)
            time.sleep(0.5)
            goto(1077)
            break


    #Vai verificar se pode passar ao proximo aperto
    while line == 1077:
        decision=get_input_register_int(2)
        set_output_register_int(0, 350)
        set_output_register_int(3, 0)
        if(decision==555): #Caso possa avança
            set_output_register_int(3, 777)
            time.sleep(0.1)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            goto(1007)
            break
        if(decision==999): #Caso não possa volta ao inicio
            set_output_register_int(3, 777)
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 402)
            time.sleep(1)
            set_digital_output(15, OFF)
```

```
            set_digital_output(16, OFF)
            set_output_register_int(1, 0)
            set_output_register_int(3, 0)
            goto(1)
            break

    #Aperto Torx
    while line == 1007:

        set_ref_coord(106) #Seleciona de novo o frame calculado
        set_output_register_int(0, 360)
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        movej(posj(18.12, 45.17, 110.36, 43.43, 57.15, 53.41), v=150, a=150, r=50)
        movej(posj(-4.56, 86.98, 66.73, 51.98, 103.32, 58.84), v=150, a=150, r=50)
        set_output_register_int(3, 0)
        movel(System_433d_pa4, v=100, a=100, r=0) #Ponto de aproximação do parafuso torx
        mwait(0)
        set_output_register_int(0, 365)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 40, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force
        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa o force
        wait(0.2)
        delta_retract = [0.0, 0.0, -50.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=80, a=50, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(108) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 370)
        set_digital_output(13, OFF)
        mwait(0)
        goto(1099)
        break

    #Vai verificar se pode passar ao proximo aperto
    while line == 1099:
        decision=get_input_register_int(2)
        set_output_register_int(0, 380)
        set_output_register_int(3, 0)
        time.sleep(0.5)
        if(decision==555): #Caso possa avança
            set_output_register_int(3, 777)
            time.sleep(0.1)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            set_output_register_int(3, 0)
            goto(1009)
            break
        if(decision==999): #Caso não possa volta ao inicio
            set_output_register_int(3, 777)
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            movej(posj(-5.8, 80.2, 70.54, 37, 106.44, 38.35), v=100, a=100, r=50)
```

```
            movej(posj(-5.8, 68.71, 70.54, 25.89, 106.44, 71.79), v=100, a=100, r=100)
            movej(posj(-5.8, 51.79, 70.54, 2.81, 85.71, 86.06), v=100, a=100, r=100)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 402)
            time.sleep(1)
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            set_output_register_int(1, 0)
            set_output_register_int(3, 0)
            goto(1)
            break

    #Vai fazer o aperto do parafuso TORX do disco esquerdo
    while line == 1009:
        set_ref_coord(108) #Seleciona de novo o frame calculado
        set_output_register_int(0, 390)
        movej(posj(3.69, 58.14, 84.48, 35.86, 86.51, 50.07), v=150, a=150, r=50)
        movej(posj(5.36, 57.43, 74.46, -3.47, 71.84, -4.99), v=150, a=150, r=100)
        movej(posj(-0.52, 57.43, 74.46, -40.81, 101.77, -56.27), v=150, a=150, r=100)
        movej(posj(-8.27, 97.01, 49.45, -63.74, 107.5, -84.55), v=150, a=150, r=100)
        set_output_register_int(3, 0)
        set_output_register_int(0, 395)
        movel(System_433e_pa4, v=100, a=100, r=0) #Ponto de aproximação do parafuso torx
        mwait(0)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([10000, 10000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force
        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa o force
        wait(0.2)
        delta_retract = [0.0, 0.0, -50.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=80, a=50, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(108) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_digital_output(13, OFF)
        #Posições para retornar à Home Position
        movej(posj(-5.48, 95.65, 46.78, -61.75, 106.99, -93.03), v=150, a=150, r=50)
        movej(posj(-5.48, 65.7, 68.19, -45.25, 96.72, -37.01), v=250, a=250, r=50)
        movej(posj(-5.48, 49.75, 68.19, -19.27, 96.72, 40.64), v=250, a=250, r=50)
        set_output_register_int(0, 399)
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        movej(posj(-5.48, 49.75, 81.38, -6.81, 96.72, 83.36), v=150, a=150, r=50)

        movej(posj(0, 20, 115, 0, 45, 90), v=100, a=100) #Home Position
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        res = get_output_register_bit(10)
        if res==1:
            set_output_register_int(0, 402)
        if res==0:
            set_output_register_int(0, 400)
```

```
        set_output_register_int(4, 0)
        set_digital_output(15, OFF)
        set_digital_output(16, OFF)
        set_tcp("U_Tool_0")
        set_ref_coord(DR_BASE)
        goto(1)
        break



##########################################################################################################
############################################################
##########################################################################################################
############################################################
#  Trajetória 11 753 (disco 2)

    while line == 1102:

        set_tcp("U_Tool_0")
        set_ref_coord(DR_BASE)
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada

        #Faz a conecção com a câmara de visão
        camera_ip = "192.168.2.81"
        trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114, state=0)
        time.sleep(0.5)
        if trispector.state != 1: #Verifica se a conecção foi estabelecida corretamente
            set_output_register_int(1, 20)  #Info para PLC resultado Visão (não estabelecida)
            set_output_register_int(0, 210) #Infor para PLC da etapa
            time.sleep(3)
            set_output_register_int(0, 401) #Indica que não foi efetuada corretamente a ligação
com a camara ao PLC
            goto(1)
            continue


        job = trispector.write("set job \"753 55dir\"")  #Seleciona o programa da câmara referente
ao disco 753 direito

        #Vai movimentar o robô para a posição para se dar ao inicio da aquisição da imagem pelo
sistema de visão
        movej(posj(53.85, 56.67, 59.84, -0.01, 63.49, 143.81), v=100, a=100, r=100)
        #Função usada para movimentar o robô para as coordenadas "X, Y, Z, Rx, Ry, Rz" com a
velocidade "v", a aceleracao "a" e o radius "r"
        movel(posx(357.99, 552.8, 91.84, 0, 180, 90), v=100, a=100)

        set_output_register_int(3, 0)
        set_output_register_int(0, 210)

        laser_on = trispector.write("set laser on") #Ativa o laser da câmara para aquisição da
imagem
        time.sleep(0.3)
        amovel(posx(358.12, 164.03, 91.96, 0, -180, 90), v=50, a=200)
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio
à aquisição da imagem
        mwait(0)
        set_digital_output(15, ON) #Refere que se trata de valores do disco direito
        rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
        rxd =(rx_msg[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui
irá dividir-se a informação pelas virgulas criando uma array
        data = rx_msg.split(', ')
        valid_locate  =(rx_msg[0:1])  #Vai  guardar  a  informação  se  a  imagem  foi  adquirida
corretamente
        time.sleep(0.2)

        #Valores imagem referência (valores do frame de referência criado)
        refx1 = -70.901699
        refy1 = 143.633681
        refz1 = 213.160433
        refx2 = 114.831651
```

```
        refy2 = 140.310109
        refz2 = 216.565214
        refx3 = 46.082287
        refy3 = 291.541525
        refz3 = 252.419949


        if valid_locate=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x3 =  float(data[7]) - refx3
            y3 =  float(data[8]) - refy3
            z3 =  float(data[9]) - refz3
            x1 = round(x1, 6)
            y1 = round(y1, 6)
            z1 = round(z1, 6)
            x2 = round(x2, 6)
            y2 = round(y2, 6)
            z2 = round(z2, 6)
            x3 = round(x3, 6)
            y3 = round(y3, 6)
            z3 = round(z3, 6)

            #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência em
arrays
            d753visao1 = [x1, y1, -z1]
            d753visao2 = [x2, y2, -z2]
            d753visao3 = [x3, y3, -z3]

            #Define a posição exata dos 3 parafusos de referência
            d753pf1=posx(354.920, 323.090, -361.440, 30.37, -175.95, 50.87)
            d753pf2=posx(171.740, 324.400, -349.860, 62.72, -177.96, 94.46)
            d753pf3=posx(239.14, 172.900, -319.520, 31.88, -176.34, 93.61)
            d753f1=posx(354.920, 323.090, -361.440, 74.195, 13.394, 105.804)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir
da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            d753visao1[0] = d753pf1[0] - d753visao1[0]
            d753visao1[1] = d753pf1[1] - d753visao1[1]
            d753visao1[2] = d753pf1[2] - d753visao1[2]
            d753visao2[0] = d753pf2[0] - d753visao2[0]
            d753visao2[1] = d753pf2[1] - d753visao2[1]
            d753visao2[2] = d753pf2[2] - d753visao2[2]
            d753visao3[0] = d753pf3[0] - d753visao3[0]
            d753visao3[1] = d753pf3[1] - d753visao3[1]
            d753visao3[2] = d753pf3[2] - d753visao3[2]

            #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem os novos pontos
de referência
            d753p1 = d753visao1[0:3]+d753pf1[3:]
            d753p2 = d753visao2[0:3]+d753pf2[3:]
            d753p3 = d753visao3[0:3]+d753pf3[3:]

             #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
            pose_user102 = calc_coord(d753p1, d753p2, d753p3, ref=DR_BASE, mod=0)

            #Vai alterar a pose e o sistema de coordenadas de referência para o novo sistema de
coordenadas calculado
            overwrite_user_cart_coord(102, pose_user102, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(102)

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate!="1":
            set_output_register_int(1, 10)
```

```
            laser_off = trispector.write("set laser off") #Desliga o laser
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 401)
            time.sleep(1)
            set_output_register_int(1, 0)
            goto(1)
            continue

        #Vai selecionar o programa relativo ao disco esquerdo
        set_digital_output(15, ON)
        job = trispector.write("set job \"753 55esq\"")
        time.sleep(0.1)

        movel(posx(358.12, -8.24, 91.96, 174.09, 180, -95.91), v=100, a=100)
        #time.sleep(0.5)
        time.sleep(0.3)
        amovel(posx(358.12, -386.7, 91.96, 128.95, -180, -141.05), v=50, a=200) #Posição para
inicio de aquisição da imagem
        trispector.write("trigger") #Envia-se o sinal para dar trigger na câmara para dar inicio
à aquisição da imagem

        mwait(0)
        set_digital_output(16, ON) #Refere que se trata de valores do disco esquerdo
        rx_msg = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
        rxd =(rx_msg[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui
irá dividir-se a informação pelas virgulas criando uma array
        data = rx_msg.split(', ')
        valid_locate  =(rx_msg[0:1])  #Vai  guardar  a  informação  se  a  imagem  foi  adquirida
corretamente

        #Valores imagem referência do disco esquerdo (valores do frame de referência criado)
        refx1 = -72.807635
        refy1 = 219.099548
        refz1 = 215.918209
        refx2 = 112.165183
        refy2 = 225.036077
        refz2 = 217.896962
        refx3 = 46.181223
        refy3 = 72.956898
        refz3 = 252.125790


        #Verifica se a imagem adquirida foi adquirida com sucesso
        if valid_locate=="1":

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            x1 =  float(data[1]) - refx1
            y1 =  float(data[2]) - refy1
            z1 =  float(data[3]) - refz1
            x2 =  float(data[4]) - refx2
            y2 =  float(data[5]) - refy2
            z2 =  float(data[6]) - refz2
            x3 =  float(data[7]) - refx3
            y3 =  float(data[8]) - refy3
            z3 =  float(data[9]) - refz3
            x1 = round(x1, 6)
            y1 = round(y1, 6)
            z1 = round(z1, 6)
            x2 = round(x2, 6)
            y2 = round(y2, 6)
            z2 = round(z2, 6)
            x3 = round(x3, 6)
            y3 = round(y3, 6)
            z3 = round(z3, 6)
```

```
            #Guarda a diferença das posições dos pontos de aperto dos 3 pontos de referência
            e753visao1 = [x1, y1, -z1]
            e753visao2 = [x2, y2, -z2]
            e753visao3 = [x3, y3, -z3]

            #Define os pontos de referência das três posições de aperto usadas para gerar o frame
de referência
            e753pf1=posx(358.71, -313.42, -358.4, 39.92, -176.57, 51.6)
            e753pf2=posx(177.920, -316.090, -348.920, 13.22, 176.68, -42.67)
            e753pf3=posx(241.31, -164.52, -319.49, 16.66, -177.56, 6.34)
            e753f1=posx(358.71, -313.42, -358.4, 104.355, 167.350, 103.832)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir
da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            e753visao1[0] = e753pf1[0] - e753visao1[0]
            e753visao1[1] = e753pf1[1] - e753visao1[1]
            e753visao1[2] = e753pf1[2] - e753visao1[2]
            e753visao2[0] = e753pf2[0] - e753visao2[0]
            e753visao2[1] = e753pf2[1] - e753visao2[1]
            e753visao2[2] = e753pf2[2] - e753visao2[2]
            e753visao3[0] = e753pf3[0] - e753visao3[0]
            e753visao3[1] = e753pf3[1] - e753visao3[1]
            e753visao3[2] = e753pf3[2] - e753visao3[2]

            #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem os novos pontos
de referência
            e753p1 = e753visao1[0:3]+e753pf1[3:]
            e753p2 = e753visao2[0:3]+e753pf2[3:]
            e753p3 = e753visao3[0:3]+e753pf3[3:]


            #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
            pose_user104 = calc_coord(e753p1, e753p2, e753p3, ref=DR_BASE, mod=0) #cria user frame
com base na nova posição da  visão

            #Vai alterar a pose e o sistema de coordenadas de referência para o novo sistema de
coordenadas calculado
            overwrite_user_cart_coord(104, pose_user104, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(104)

            laser_off = trispector.write("set laser off") #Desliga o laser da câmara

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate!="1":
            set_digital_output(16, ON)
            set_output_register_int(1, 10)
            laser_off = trispector.write("set laser off") #Desliga o laser da câmara
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 401)
            time.sleep(1)
            set_output_register_int(1, 0)
            set_digital_output(16, OFF)
            goto(1)
            continue


        set_digital_output(16, ON)
        laser_off = trispector.write("set laser off") #Desliga o laser da câmara
        set_output_register_int(0, 220)
        trispector.closeIN(None)
        trispector.closeOUT(None)
        goto(1103)
        break

    while line == 1103:
```

```
        set_tcp("U_Tool_1")
        set_ref_coord(DR_BASE)
        mwait(0)
        set_ref_coord(104) #Seleciona o novo frame criado com os valores do sistema de visão, que
irá corrigir os novos pontos de aperto das restantes porcas
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        set_output_register_int(3, 0)

        #A partir deste ponto irá passar-se para a fase de aperto das porcas
        movej(posj(-41.6, 66.43, 57.36, 7.32, 60.42, -69.23), v=250, a=150, r=60)
        movej(posj(-41.01, 73.76, 52.56, -1.91, 56.7, -185.66), v=150, a=150, r=60)

        movel(System_753e_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
        mwait(0)
        set_output_register_int(0, 230)
        set_digital_output(13, ON)

        #Inicio dos apertos
        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez do robô
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Aplica a força no robô

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -40.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(104) #Seleciona de novo o frame calculado

        #Começa a movimentar-se para as posicoes do parafuso 2
        movej(posj(-40.96, 71.38, 53.22, -1.87, 58.42, -185.67), v=100, a=100, r=50)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_digital_output(13, OFF)
        movej(posj(-86.95, 40.39, 109.33, -6.07, 31.26, -121.44), v=100, a=100, r=50)
        set_output_register_int(0, 240)
        movel(System_753e_pa2, v=100, a=100, r=0) #Ponto de aproximação do segundo parafuso
        mwait(0)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)
```

```
        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -40.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_output_register_int(0, 245)

        goto(1134)
        break



    while line == 1134:

        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        set_ref_coord(104) #Seleciona de novo o frame calculado

        #Começa a movimentar-se para as posicoes do parafuso 3
        movej(posj(-59.4, 50.48, 108.45, 2.08, 22.28, -156.09), v=250, a=150, r=50)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 250)
        set_digital_output(13, OFF)
        movej(posj(-59.32, 57.11, 106.59, 2.63, 17.51, -156.59), v=150, a=150, r=50)
        movel(System_753e_pa3, v=100, a=100, r=0) #Ponto de aproximação do terceiro parafuso
        mwait(0)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(104)
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_digital_output(13, OFF)
        set_output_register_int(0, 260)
        goto(1155)
        break
```

```
    #Vai verificar se pode passar ao aperto dos parafusos do disco direito
    while line == 1155:
        decision=get_input_register_int(2)
        set_output_register_int(0, 270)
        set_output_register_int(3, 0)

        if(decision==555): #Caso possa passar ao disco direito avança
            set_output_register_int(3, 777)
            time.sleep(0.1)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            goto(1105)
            break
        if(decision==999): #Caso não possa, aborta e volta ao inicio
            set_output_register_int(3, 777)
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            movej(posj(-39.09, 32.33, 116.07, 3.3, 39.01, -70.12), v=100, a=100, r=50)
            movej(posj(-23.48, 36.31, 113.85, 16.04, 15.59, -34.75), v=100, a=100, r=50)
            movej(posj(-12.82, 38.79, 113.27, 38.46, 12.65, 34.61), v=100, a=100, r=50)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 402)
            time.sleep(1)
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            set_output_register_int(1, 0)
            set_output_register_int(3, 0)
            goto(1)
            break


    while line == 1105:

        #Código relativo ao aperto do disco direito
        mwait(0)
        #Seleciona o novo frame criado com os valores do sistema de visão relativo ao disco direito,
que irá corrigir os novos pontos de aperto das restantes porcas
        set_ref_coord(102)
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        set_output_register_int(0, 300)
        set_output_register_int(3, 0)
        mwait(0)
        set_output_register_int(3, 0)
        #Posições de aperto do primeiro parafuso do disco direito
        movej(posj(-39.09, 32.33, 116.07, 3.3, 39.01, -70.12), v=250, a=150, r=60)
        movej(posj(-23.48, 36.31, 113.85, 16.04, 15.59, -34.75), v=250, a=250, r=60)
        movej(posj(-12.82, 38.79, 113.27, 38.46, 12.65, 34.61), v=250, a=250, r=60)
        movej(posj(55.35, 47.13, 95.36, 6.41, 42.32, 100.11), v=150, a=150, r=60)
        movej(posj(57.96, 58.2, 103, 9.29, 21.1, 79.85), v=100, a=100)

        movel(System_753d_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
        mwait(0)
        set_output_register_int(0, 310)
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
```

```
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=80, a=50, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(102) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 320)
        set_digital_output(13, OFF)

        mwait(0)
        movej(posj(79.18, 59.54, 101.37, 7.26, 20.69, 118.99), v=100, a=100)

        movel(System_753d_pa2, v=100, a=100, r=0) #Ponto de aproximação do segundo parafuso
        mwait(0)
        set_output_register_int(4, 0)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa a força exercida no robô
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        mwait(0)
        set_ref_coord(102) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_digital_output(13, OFF)
        movej(posj(60.07, 56.59, 115.88, 9.26, 10.88, 112.64), v=100, a=100)
        movel(System_753d_pa3, v=100, a=100, r=0) #Ponto de aproximação do terceiro parafuso
        mwait(0)
        set_output_register_int(0, 330)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force

        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
```

```
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        set_output_register_int(0, 335)
        release_force() #Desativa o force
        wait(0.2)
        delta_retract = [0.0, 0.0, -60.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=100, a=100, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(102) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 340)
        set_digital_output(13, OFF)

        goto(1106)
        break
        exit()


    while line == 1106:
        free_zone=get_input_register_int(4) #Verifica se a zona está livre
        if(free_zone==3333):
            set_output_register_int(3, 777)
            time.sleep(0.5)
            goto(1177)
            break


    #Vai verificar se pode passar ao proximo aperto
    while line == 1177:
        decision=get_input_register_int(2)
        set_output_register_int(0, 350)
        set_output_register_int(3, 0)
        if(decision==555): #Caso possa avança
            set_output_register_int(3, 777)
            time.sleep(0.1)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            goto(1107)
            break
        if(decision==999): #Caso não possa volta ao inicio
            set_output_register_int(3, 777)
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 402)
            time.sleep(1)
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            set_output_register_int(1, 0)
            set_output_register_int(3, 0)
            goto(1)
            break

    #Aperto Torx
    while line == 1107:

        set_ref_coord(102) #Seleciona de novo o frame calculado
        set_output_register_int(0, 360)
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        movej(posj(18.12, 45.17, 110.36, 43.43, 57.15, 53.41), v=150, a=150, r=50)
        movej(posj(-4.56, 86.98, 66.73, 51.98, 103.32, 58.84), v=150, a=150, r=50)
        set_output_register_int(3, 0)
        movel(System_753d_pa4, v=100, a=100, r=0) #Ponto de aproximação do parafuso torx
        mwait(0)
```

```
        set_output_register_int(0, 365)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([20000, 20000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 40, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force
        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa o force
        wait(0.2)
        delta_retract = [0.0, 0.0, -50.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=80, a=50, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(104) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 370)
        set_digital_output(13, OFF)
        mwait(0)
        goto(1199)
        break

    #Vai verificar se pode passar ao proximo aperto
    while line == 1199:
        decision=get_input_register_int(2)
        set_output_register_int(0, 380)
        set_output_register_int(3, 0)
        time.sleep(0.5)
        if(decision==555): #Caso possa avança
            set_output_register_int(3, 777)
            time.sleep(0.1)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            set_output_register_int(3, 0)
            goto(1109)
            break
        if(decision==999): #Caso não possa volta ao inicio
            set_output_register_int(3, 777)
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            movej(posj(-5.8, 80.2, 70.54, 37, 106.44, 38.35), v=100, a=100, r=50)
            movej(posj(-5.8, 68.71, 70.54, 25.89, 106.44, 71.79), v=100, a=100, r=100)
            movej(posj(-5.8, 51.79, 70.54, 2.81, 85.71, 86.06), v=100, a=100, r=100)
            movej(posj(0, 20, 115, 0, 45, 90), v=50, a=50) #Home Position
            set_output_register_int(4, 0) # Info para PLC Zona Ocupada
            set_output_register_int(0, 402)
            time.sleep(1)
            set_digital_output(15, OFF)
            set_digital_output(16, OFF)
            set_output_register_int(1, 0)
            set_output_register_int(3, 0)
            goto(1)
            break

    #Vai fazer o aperto do parafuso TORX do disco esquerdo
    while line == 1109:
        set_ref_coord(104) #Seleciona de novo o frame calculado
        set_output_register_int(0, 390)
```

```
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        movej(posj(3.69, 58.14, 84.48, 35.86, 86.51, 50.07), v=150, a=150, r=50)
        movej(posj(5.36, 57.43, 74.46, -3.47, 71.84, -4.99), v=150, a=150, r=100)
        movej(posj(-0.52, 57.43, 74.46, -40.81, 101.77, -56.27), v=150, a=150, r=100)
        movej(posj(-7.56, 93.57, 51.83, -59.83, 102.46, -78.45), v=150, a=150, r=100)
        set_output_register_int(3, 0)
        set_output_register_int(0, 395)
        movel(System_753e_pa4, v=100, a=100, r=0) #Ponto de aproximação do parafuso torx
        mwait(0)
        set_digital_output(13, ON)

        set_ref_coord(DR_TOOL)
        task_compliance_ctrl() #Ativa o compliance
        set_stiffnessx([10000, 10000, 20000, 1000, 1000, 1000], time=1.0) #Define os valores da
rigidez
        fd = [0, 0, 50, 0, 0, 0]
        fctrl_dir= [0, 0, 1, 0, 0, 0]
        set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL) #Ativa o force
        delta_motion = [0.0, 0.0, 30.0, 0.0, 0.0, 0.0]
        amovel(delta_motion, v=50, a=50,  mod=DR_MV_MOD_REL)
        wait_touch() #Vai esperar pela pressão do parafuso
        stop(DR_QSTOP)
        wait(0.1)

        amovel(delta_motion, v=50, a=150, mod=DR_MV_MOD_REL)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        stop(DR_QSTOP)
        wait(0.5)
        release_force() #Desativa o force
        wait(0.2)
        delta_retract = [0.0, 0.0, -50.0, 0.0, 0.0, 0.0]

        movel(delta_retract, v=80, a=50, mod=DR_MV_MOD_REL)
        release_compliance_ctrl() #Desativa o compliance
        set_digital_output(13, OFF)
        mwait(0)
        set_ref_coord(104) #Seleciona de novo o frame calculado
        set_output_register_int(2, 0)
        set_output_register_int(3, 0)
        set_digital_output(13, OFF)
        #Posições para retornar à Home Position
        movej(posj(-5.48, 95.65, 46.78, -61.75, 106.99, -93.03), v=150, a=150, r=50)
        movej(posj(-5.48, 65.7, 68.19, -45.25, 96.72, -37.01), v=250, a=250, r=50)
        movej(posj(-5.48, 49.75, 68.19, -19.27, 96.72, 40.64), v=250, a=250, r=50)
        set_output_register_int(0, 399)
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        movej(posj(-5.48, 49.75, 81.38, -6.81, 96.72, 83.36), v=100, a=100, r=50)
        movej(posj(0, 20, 115, 0, 45, 90), v=100, a=100) #Home Position
        set_output_register_int(4, 0) # Info para PLC Zona Ocupada
        res = get_output_register_bit(10)
        if res==1:
            set_output_register_int(0, 402)
        if res==0:
            set_output_register_int(0, 400)
        set_output_register_int(4, 0)
        set_digital_output(15, OFF)
        set_digital_output(16, OFF)
        set_tcp("U_Tool_0")
        set_ref_coord(DR_BASE)
        goto(1)
        break

    #FIM DO CÓDIGO
```

Anexo 4 – Código Porteur do robô 2

```
from DRCF import*
import sys
import os
import time
```

```
import socket

#Entrada 1 + 2 = Cameras de segurança
#Entrada 3 + 4 = Interlock Reset
#Entrada 5 + 6 = Safety Zone Dynamic Enable (L)
#Entrada 8 = Feedback Home pos

#Saida 1 + 2 = Emergency Stop excl No Loopback Input (L)
#Saida 3 + 4 = Designated Zone = Home pos
#saida 5 = Zona Colisão
#Saida 7 = Task Operating (L)
#Saida 8 = Safe Operating Stop (L)
#Saida 13 = Safety Zone Dynamic Enable (L) = Entrada 5 + 6

class DoosanSick:

#*********************************

    #Função que inicializa e faz a conexão da câmara
    def __init__(self, ip="192.168.2.55", portIN=2115, portOUT=2114, state=0):

        self.ip = ip
        self.portIN = portIN
        self.portOUT = portOUT
        self.state = state
        client = socket.socket()
        client.settimeout(3)
        socket.setdefaulttimeout(3)

        try:
            client.connect((self.ip, self.portIN))
            self._socketIN = client_socket_open(self.ip, self.portIN)
            self.state=1

        except Exception as e:
            self.state=2
            tp_log("Socket INPUT falha de ligação. Error: {0}".format(str(e)))

        time.sleep(0.1)

        try:
            client.connect((self.ip, self.portOUT))
            self._socketOUT = client_socket_open(self.ip, self.portOUT)
            self.state=1
        except Exception as e:
            self.state=2
            tp_log("Socket OUTPUT falha de ligação. Error: {0}".format(str(e)))

#*********************************

    #Função usada para escrever no socket
    def write(self, cmd, socket=None):

        if socket == None:
            socket = self._socketIN

        cmd = bytes(cmd, encoding="ascii")

        STX = client_socket_write(socket, b"\x02")
        res = client_socket_write(socket, cmd)
        STX = client_socket_write(socket, b"\x03")
        time.sleep(0.1)
        res, rx_data = client_socket_read(socket, 10, 1)
        rxd=None
        if res > 1:
            rx_msg = rx_data.decode()
            rxd =(rx_msg[1:3])
        if res == -1:
            tp_log("Erro durante a escrita comando no socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a escrita comando no socket: Erro de Socket")
```

```
        return rxd

#*********************************

    #Função usada para ler a informação do command channel
    def readcmd(self, length, timeout, socket=None ):

        if socket == None:
           socket = self._socketIN

        res, rx_data = client_socket_read(socket, length, timeout)
        rx_msg = rx_data.decode()
        rxd =(rx_msg[1:3])

        if res == -1:
            tp_log("Erro durante a leitura do socket : Server não conectado")
        elif res == -2:
            tp_log("Erro durante a leitura do socket: Erro de Socket")
        elif res == -3:
            tp_log("Erro durante a leitura do socket: Tempo de espera excedido")
        return rxd

#*********************************

    #Função usada para ler do socket a informação recebida
    def readata(self, length, timeout, socket=None ):

        if socket == None:
           socket = self._socketOUT
        rx_msg = '0'
        res = 0
        rx_data = 0

        res, rx_data = client_socket_read(socket, length, timeout)

        if res == -1:
            tp_log("Erro durante a leitura dados do socket : Server não conectado")
            rx_msg = '0, 0, 0, 0, 0, 0, 0, 0, 0, 0'
            return rx_msg
        elif res == -2:
            tp_log("Erro durante a leitura dados do socket: Erro de Socket")
            rx_msg = '0, 0, 0, 0, 0, 0, 0, 0, 0, 0'
            return rx_msg
        elif res == -3:
            tp_log("Erro durante a leitura dados do socket: Tempo de espera excedido")
            rx_msg = '0, 0, 0, 0, 0, 0, 0, 0, 0, 0'
            return rx_msg
        else:
            rx_msg = rx_data.decode()

            return rx_msg

#*********************************

    #Função que irá fechar as conexões dos socket IN e OUT utilizados para efetuar a comunicação
com a câmara
    def closeIN(self, socket=None):

        if socket == None:
            socket = self._socketIN

        client_socket_close(socket)

    def closeOUT(self, socket=None):

        if socket == None:
            socket = self._socketOUT

        client_socket_close(socket)

#Função que irá esperar que o torque pretendido seja atingido para desativar a aparafusadora
def wait_torque() :
```

```
            yy = True
            zz = 0
            while yy:
                step_ack=get_input_register_int(1)
                if(step_ack==1111) or (step_ack==2222):
                    set_output_register_int(3, 777)
                    time.sleep(0.1)
                    set_output_register_int(2, 0) #Output para ativar e desativar aparafusadora
                    result=get_input_register_int(1)
                if zz >= 50:
                    yy = False
                    set_output_register_int(2, 0)
                time.sleep(0.1)
                zz = zz + 0.1

#*************************************************************************

#Definição de algumas variáveis globais e outputs importantes
set_motion_end(DR_CHECK_ON)
set_output_register_int(0, 0)
set_output_register_int(1, 0)
set_digital_output(13, OFF)
set_tool ("U_Tool_W1")  #Seleciona a tool que se pretende
set_tcp("U_Tool_1")     #Seleciona o TCP da respetiva tool
set_ref_coord(DR_BASE)  #Define o referêncial de coordenadas para movimentar o robô
time.sleep(0.5)

#Definição de uma variável global que será usada posteriormente para indicar para onde irá a
execução do programa
def goto(linenum):
    global line
    line = linenum
line = 1

#Dá set a 0 os seguintes outputs
set_output_register_int(0, 0) # Info para PLC etapa do Robô
set_output_register_int(1, 0) # Info para PLC resultado Visão
set_output_register_int(2, 0) # Pedido aperto
set_output_register_int(3, 0) # Info para PLC decisão recebida
#set_output_register_int(4, 0) # Info para PLC zona ocupada

time.sleep(0.5)

if get_digital_input(12)==1: #Segurança
 exit()
time.sleep(0.1)

zero = check_robot_mastering() #Verifica se o robô está num estado que requer masterização ou não
(calibração)

if zero == 1: #Se necessitar de recalibração
    move_home(DR_HOME_TARGET_USER) #Volta à posição Home


movej(posj(0, -80, 115, 0, 90, -90.0), v=50, a=50) #Home Position
set_output_register_int(4, 0) # Info para PLC zona ocupada
set_output_register_int(0, 402)

#Vai definir os pontos de aproximacao e de aperto dos 3 parafusos de referência para posterior
correção do frame
#Foram definidos 2 conjuntos o 433 e o 753 que correspondem a discos diferentes devido ao diâmetro
e às pinças serem diferentes o que irá afetar no posicionamento dos parafusos

#**Pontos de aperto 433 (disco 1)
System_433d_pa1 = posx(25.960, -110.490, 110.880, 44.75, 123.29, 33.02)
System_433d_p1 = posx(49.590, -87.070, 89.040, 44.75, 123.29, 33.02)

System_433e_pa1 = posx(17.020, -124.610, -120.000, 50.46, 58.82, 138.67)
System_433e_p1 = posx(44.700, -91.090, -93.690, 50.46, 58.82, 138.67)

#**Pontos de aperto 753 (disco 2)
```

```
System_753d_pa1 = posx(-17.870, -130.550, 129.090, 29.55, 120.41, 27.07)
System_753d_p1 = posx(43.590, -95.670, 87.660, 29.55, 120.41, 27.07)

System_753e_pa1 = posx(-27.130, -139.620, -142.200, 32.41, 58.22, 144.89)
System_753e_p1 = posx(40.170, -96.880, -92.810, 32.41, 58.22, 144.89)

#Inicio do programa de aperto
while True:

    time.sleep(0.1)

     #Vai ser feita a validação de qual tipo de disco é que chegou ao ponto de aperto para a
continuação do código de acordo com o dito disco
    while line == 1 and get_input_register_int(0) == 11 and get_digital_input(16)==0 :  # Disco
433
        set_output_register_int(0, 0)
        set_output_register_int(1, 0)
        set_output_register_int(2, 0)
        set_output_register_int(3, 777)
        set_output_register_int(4, 0)
        set_output_register_int(0, 111)
         goto(1002) #Define a variável line a 1002 e será com esta variável que se irá controlar
qual parte do codigo correr a seguir
        break



    while line == 1 and get_input_register_int(0) == 12 and get_digital_input(16)==0 :  # Disco
753
        set_output_register_int(0, 0)
        set_output_register_int(1, 0)
        set_output_register_int(2, 0)
        set_output_register_int(3, 777)
        set_output_register_int(4, 0)
        set_output_register_int(0, 112)
         goto(1102) #Define a variável line a 1102 e será com esta variável que se irá controlar
qual parte do codigo correr a seguir
        break

    #Função que fica à espera que a maquete com os discos cheguem ao ponto de aperto, enquanto a
mesma não chega o programa fica em loop aqui
    while line == 1 and get_input_register_int(0) == 99 and get_digital_input(12)==0 :
        set_output_register_int(0, 0)
        set_output_register_int(1, 0)
        set_output_register_int(2, 0)
        set_output_register_int(3, 777)
        set_output_register_int(4, 0)
        time.sleep(0.5)
        set_output_register_int(0, 999) # Fora serviço
        goto(99)
        continue


    while line == 99:
        set_output_register_int(0, 999)
        outserv=get_input_register_int(0)
        if(outserv==0):
            time.sleep(0.2)
            set_output_register_int(0, 400)
            time.sleep(1)
            set_output_register_int(0, 0)
            goto(1)
            break
        time.sleep(2.1)
        set_output_register_int(3, 0)
        goto(99)
        break


##########################################################################################################
#############################################################
```

```
#########################################################################################################
#########################################################
#  Trajetória 11 433 (disco 1)

    while line == 1002:

        time.sleep(0.1)
        set_tcp("U_Tool_1")
        set_ref_coord(DR_BASE)

        #Faz a conecção com a câmara de visão
        camera_ip = "192.168.2.81"
        trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114, state=0)
        time.sleep(0.5)
        if trispector.state != 1: #Verifica se a conecção foi estabelecida corretamente
            set_output_register_int(1, 20)
            set_output_register_int(0, 210)
            time.sleep(3)
            set_output_register_int(0, 401)
            goto(1)
            continue

        set_output_register_int(0, 210)
        wait_digital_input(15, ON)
        set_output_register_int(3, 0)
        set_output_register_int(0, 210)

         #Como apenas se usa uma câmara de visão, este segundo robô vai apenas comunicar com a
câmara para ir buscar os valores previamente recolhidos pelo robô 1
        rx_msg1 = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
        rxd1 =(rx_msg1[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui
irá dividir-se a informação pelas virgulas criando uma array
        data1 = rx_msg1.split(', ')
          valid_locate1 =(rx_msg1[0:1]) #Vai guardar a informação se a imagem foi adquirida
corretamente
        time.sleep(0.2)

        #Valores imagem referência (valores do frame de referência criado)
        refdx1 = -74.745480
        refdy1 = 134.420716
        refdz1 = 213.746662
        refdx2 = 110.374540
        refdy2 = 147.874876
        refdz2 = 214.483098
        refdx3 = 28.898749
        refdy3 = 292.389366
        refdz3 = 249.882767


        if valid_locate1=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

             #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            xd1 =  float(data1[1]) - refdx1
            yd1 =  float(data1[2]) - refdy1
            zd1 =  float(data1[3]) - refdz1
            xd2 =  float(data1[4]) - refdx2
            yd2 =  float(data1[5]) - refdy2
            zd2 =  float(data1[6]) - refdz2
            xd3 =  float(data1[7]) - refdx3
            yd3 =  float(data1[8]) - refdy3
            zd3 =  float(data1[9]) - refdz3
            xd1 = round(xd1, 6)
            yd1 = round(yd1, 6)
            zd1 = round(zd1, 6)
            xd2 = round(xd2, 6)
            yd2 = round(yd2, 6)
            zd2 = round(zd2, 6)
            xd3 = round(xd3, 6)
            yd3 = round(yd3, 6)
```

```
            zd3 = round(zd3, 6)

             #Guarda a diferença das posições dos pontos de aperto dos pontos de referência em
arrays
            d433visao1 = [-xd1, -yd1, -zd1]
            d433visao2 = [-xd2, -yd2, -zd2]
            d433visao3 = [-xd3, -yd3, -zd3]

            #Define a posição exata dos parafusos de referência
            d433pf1=posx(564.61, -328.82, 370.52, 92.73, -177.64, 96.87)
            d433pf2=posx(749.52, -313.92, 377.44, 28.12, -178.56, 34.06)
            d433pf3=posx(667.45, -170.59, 408.23, 161.92, 179.41, 165.01)
            d433f1=posx(564.61, -328.83, 370.52, 85.134, -12.766, -80.288)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir
da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            d433visao1[0] = d433pf1[0] - d433visao1[0]
            d433visao1[1] = d433pf1[1] - d433visao1[1]
            d433visao1[2] = d433pf1[2] - d433visao1[2]
            d433visao2[0] = d433pf2[0] - d433visao2[0]
            d433visao2[1] = d433pf2[1] - d433visao2[1]
            d433visao2[2] = d433pf2[2] - d433visao2[2]
            d433visao3[0] = d433pf3[0] - d433visao3[0]
            d433visao3[1] = d433pf3[1] - d433visao3[1]
            d433visao3[2] = d433pf3[2] - d433visao3[2]

            #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem os novos pontos
de referência
            d433p1 = d433visao1[0:3]+d433pf1[3:]
            d433p2 = d433visao2[0:3]+d433pf2[3:]
            d433p3 = d433visao3[0:3]+d433pf3[3:]

            #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
            pose_user106 = calc_coord(d433p1, d433p2, d433p3, ref=DR_BASE, mod=0)

             #Vai alterar a pose e o sistema de coordenadas de referência para o novo sistema de
coordenadas calculado
            overwrite_user_cart_coord(106, pose_user106, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(106)

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate1 != "1":
            set_output_register_int(1, 10)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, -80, 115, 0, 90, -90.0), v=100, a=100, r=50) #Home Position
            set_output_register_int(1, 0)
            time.sleep(2)
            set_output_register_int(0, 401)
            goto(1)
            continue

        goto(1003)
        break

    while line == 1003:

        #Vai buscar os valores referentes ao segundo esquerdo
        wait_digital_input(16, ON)
        rx_msg2 = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
        rxd2 =(rx_msg2[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui
irá dividir-se a informação pelas virgulas criando uma array
        data2 = rx_msg2.split(', ')
          valid_locate2 =(rx_msg2[0:1]) #Vai guardar a informação se a imagem foi adquirida
corretamente
        tp_log("locate2: {0}".format(valid_locate2))
        tp_log("result2: {0}".format(data2))

        #Valores imagem referência do disco esquerdo (valores do frame de referência criado)
```

```
        refxe1 = -79.326979
        refye1 = 228.698872
        refze1 = 213.101958
        refxe2 = 105.795317
        refye2 = 221.585210
        refze2 = 214.578844
        refxe3 = 29.718081
        refye3 = 74.771915
        refze3 = 250.553534


        if valid_locate2=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            trispector.closeIN(None)
            trispector.closeOUT(None)

             #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            xe1 =  float(data2[1]) - refxe1
            ye1 =  float(data2[2]) - refye1
            ze1 =  float(data2[3]) - refze1
            xe2 =  float(data2[4]) - refxe2
            ye2 =  float(data2[5]) - refye2
            ze2 =  float(data2[6]) - refze2
            xe3 =  float(data2[7]) - refxe3
            ye3 =  float(data2[8]) - refye3
            ze3 =  float(data2[9]) - refze3
            xe1 = round(xe1, 6)
            ye1 = round(ye1, 6)
            ze1 = round(ze1, 6)
            xe2 = round(xe2, 6)
            ye2 = round(ye2, 6)
            ze2 = round(ze2, 6)
            xe3 = round(xe3, 6)
            ye3 = round(ye3, 6)
            ze3 = round(ze3, 6)

            #Guarda a diferença das posições dos pontos de aperto de referência
            e433visao1 = [-xe1, -ye1, -ze1]
            e433visao2 = [-xe2, -ye2, -ze2]
            e433visao3 = [-xe3, -ye3, -ze3]

             #Define os pontos de referência das posições de aperto usadas para gerar o frame de
referência
            e433pf1=posx(558.58, 327.27, 370.87, 88.3, -178.43, 91)
            e433pf2=posx(743.45, 320.38, 379.15, 157.06, -176.93, 160.62)
            e433pf3=posx(665.34, 173.33, 409.16, 34.75, 178.36, 51.46)
            e433f1=posx(558.58, 327.27, 370.87, 99.255, -167.242, -78.333)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir
da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            e433visao1[0] = e433pf1[0] - e433visao1[0]
            e433visao1[1] = e433pf1[1] - e433visao1[1]
            e433visao1[2] = e433pf1[2] - e433visao1[2]
            e433visao2[0] = e433pf2[0] - e433visao2[0]
            e433visao2[1] = e433pf2[1] - e433visao2[1]
            e433visao2[2] = e433pf2[2] - e433visao2[2]
            e433visao3[0] = e433pf3[0] - e433visao3[0]
            e433visao3[1] = e433pf3[1] - e433visao3[1]
            e433visao3[2] = e433pf3[2] - e433visao3[2]

            #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem os novos pontos
de referência
            e433p1 = e433visao1[0:3]+e433pf1[3:]
            e433p2 = e433visao2[0:3]+e433pf2[3:]
            e433p3 = e433visao3[0:3]+e433pf3[3:]

            #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
            pose_user108 = calc_coord(e433p1, e433p2, e433p3, ref=DR_BASE, mod=0)

             #Vai alterar a pose e o sistema de coordenadas de referência para o novo sistema de
coordenadas calculado
```

```
            overwrite_user_cart_coord(108, pose_user108, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(108)

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate2 != "1":
            set_output_register_int(1, 10)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, -80, 115, 0, 90, -90.0), v=100, a=100, r=50)
            set_output_register_int(1, 0)
            time.sleep(2)
            set_output_register_int(0, 401)
            goto(1)
            continue


        set_output_register_int(0, 220)
        set_tcp("U_Tool_1")

        goto(1004)
        break


    while line == 1004:

        free_zone=get_input_register_int(4) #Verifica se a zona está livre
        if(free_zone==3333):
            set_output_register_int(3, 777)
            set_output_register_int(0, 225)
            #time.sleep(0.1)
            mwait(0)

            set_ref_coord(108) #Seleciona o novo frame criado com os valores do sistema de visão,
que irá corrigir os novos pontos de aperto das restantes porcas
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            movej(posj(0, -55, 120, 0, 60, -90), v=100, a=100, r=50) #Home Position

            #A partir deste ponto irá passar-se para a fase de aperto das porcas
            movej(posj(65.1, 1.44, 88, 53.78, -52.74, -146.2), v=100, a=100, r=50)
            movej(posj(63.03, 11.21, 88, 95.37, -71.01, -104.08), v=100, a=100, r=50)
            movej(posj(80.77, 53.67, 36.76, 117.81, -132.73, -117.56), v=100, a=100, r=50)
            set_output_register_int(0, 229)
            set_output_register_int(3, 0)

            movel(System_433e_pa1, v=100, a=100, r=10) #Ponto de aproximação do primeiro parafuso
esquerdo
            set_output_register_int(0, 230)
            set_digital_output(13, ON)
            movel(System_433e_p1, v=50, a=50)  #Ponto de aperto do primeiro parafuso esquerdo

            #Aperto P1
            set_output_register_int(2, 55)
            time.sleep(0.2)
             wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o
aperto e espera pelo relatório de final de aperto
            time.sleep(0.1)
            set_output_register_int(2, 0)

             movel(System_433e_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
esquerdo
            set_digital_output(13, OFF)
            set_output_register_int(3, 0)
            set_output_register_int(0, 235)
            goto(1005)
            break


    while line == 1005:

        set_output_register_int(0, 239)
        set_output_register_int(3, 777)
```

```
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        time.sleep(0.2)
        set_output_register_int(3, 0)

        movej(posj(80.77, 53.67, 36.76, 117.81, -132.73, -117.56), v=100, a=100, r=50)
        movej(posj(63.03, 11.21, 88, 95.37, -71.01, -104.08), v=100, a=100, r=50)
        movej(posj(65.1, 1.44, 88, 53.78, -52.74, -146.2), v=100, a=100, r=50)

        #Movimentos para ir fazer o aperto no disco direito
        movej(posj(-53.04, -27.02, 83.45, 10.99, 80.35, -88.35), v=100, a=100, r=50)
        movej(posj(-85.59, 8.52, 73.82, 43.14, 92.85, -87.35), v=100, a=100, r=50)
        movej(posj(-99.43, 44.09, 49.3, 60.04, 128.19, -74.85), v=100, a=100, r=0)
        set_output_register_int(3, 0)##
        mwait(0)##
        set_ref_coord(106) #Seleciona de novo o frame calculado
        set_output_register_int(4, 0)
        set_output_register_int(0, 300)
        goto(1008)
        break


    while line == 1008:

        free_zone=get_input_register_int(4) #Verifica se a zona está livre
        if(free_zone==3333):
            set_output_register_int(3, 777)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            time.sleep(0.2)
            goto(1009)
            break


    while line == 1009:

        #Vai proceder ao aperto do parafuso do disco direito
        set_output_register_int(3, 0)
        movej(posj(-83.57, 54.81, 33.98, 59.47, 133.01, -69.99), v=100, a=100, r=50)
        set_output_register_int(0, 310)

         movel(System_433d_pa1, v=100, a=100, r=2)  #Ponto de aproximação do primeiro parafuso
direito
        set_digital_output(13, ON)
        movel(System_433d_p1, v=50, a=50) #Ponto de aperto do primeiro parafuso direito

        #Aperto P1
        set_output_register_int(2, 55)
        time.sleep(0.2)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        time.sleep(0.1)
        set_output_register_int(2, 0)

         movel(System_433d_pa1, v=100, a=100, r=20) #Ponto de aproximação do primeiro parafuso
direito
        movej(posj(-99.43, 44.09, 49.3, 60.04, 128.19, -74.85), v=100, a=100, r=0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 320)
        set_digital_output(13, OFF)
        set_output_register_int(0, 330)

        #Movimentos para voltar à Home Position
        movej(posj(-99.43, 44.09, 49.3, 60.04, 128.19, -74.85), v=150, a=150, r=50)
        movej(posj(-85.59, 8.52, 73.82, 43.14, 92.85, -87.35), v=150, a=150, r=50)
        movej(posj(-53.04, -27.02, 83.45, 10.99, 80.35, -88.35), v=150, a=150, r=50)
        movej(posj(-43.22, -49.01, 112.69, 28.16, 64.67, -136.33), v=150, a=150, r=50)
        movej(posj(0, -80, 115, 0, 90, -89.99), v=100, a=100, r=0)

        set_output_register_int(3, 0)
        set_output_register_int(4, 0)
        set_output_register_int(0, 400)
        time.sleep(1)
        set_ref_coord(DR_BASE)
```

```
        goto(1)
        break



########################################################################################################
########################################################
########################################################################################################
########################################################
#  Trajetória 12 753 (Disco 2)

    while line == 1102:

        time.sleep(0.1)
        set_tcp("U_Tool_1")
        set_ref_coord(DR_BASE)

        #Faz a conecção com a câmara de visão
        camera_ip = "192.168.2.81"
        trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT=2114, state=0)
        time.sleep(0.5)
        if trispector.state != 1: #Verifica se a conecção foi estabelecida corretamente
            set_output_register_int(1, 20)
            set_output_register_int(0, 210)
            time.sleep(3)
            set_output_register_int(0, 401)
            goto(1)
            continue

        set_output_register_int(0, 210)
        wait_digital_input(15, ON)
        set_output_register_int(3, 0)
        set_output_register_int(0, 210)

        rx_msg1 = trispector.readata(128, 6) #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
        rxd1 =(rx_msg1[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui
irá dividir-se a informação pelas virgulas criando uma array
        data1 = rx_msg1.split(', ')
          valid_locate1 =(rx_msg1[0:1]) #Vai guardar a informação se a imagem foi adquirida
corretamente
        time.sleep(0.2)

        #Valores imagem referência (valores do frame de referência criado)
        refdx1 = -70.901699
        refdy1 = 143.633681
        refdz1 = 213.160433
        refdx2 = 114.831651
        refdy2 = 140.310109
        refdz2 = 216.565214
        refdx3 = 46.082287
        refdy3 = 291.541525
        refdz3 = 252.419949


        if valid_locate1=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            xd1 =  float(data1[1]) - refdx1
            yd1 =  float(data1[2]) - refdy1
            zd1 =  float(data1[3]) - refdz1
            xd2 =  float(data1[4]) - refdx2
            yd2 =  float(data1[5]) - refdy2
            zd2 =  float(data1[6]) - refdz2
            xd3 =  float(data1[7]) - refdx3
            yd3 =  float(data1[8]) - refdy3
            zd3 =  float(data1[9]) - refdz3
            xd1 = round(xd1, 6)
            yd1 = round(yd1, 6)
            zd1 = round(zd1, 6)
```

```
            xd2 = round(xd2, 6)
            yd2 = round(yd2, 6)
            zd2 = round(zd2, 6)
            xd3 = round(xd3, 6)
            yd3 = round(yd3, 6)
            zd3 = round(zd3, 6)

             #Guarda a diferença das posições dos pontos de aperto dos pontos de referência em
arrays
            d753visao1 = [-xd1, -yd1, -zd1]
            d753visao2 = [-xd2, -yd2, -zd2]
            d753visao3 = [-xd3, -yd3, -zd3]

            #Define a posição exata dos parafusos de referência
            d753pf1=posx(561.11, -318.9, 368.59, 164.22, -175.95, 164.98)
            d753pf2=posx(747.34, -320.96, 379.03, 1.43, 176.12, 2.24)
            d753pf3=posx(677.14, -170.13, 409.81, 132.84, -178.8, 136.38)
            d753f1=posx(561.11, -318.9, 368.59, 75.778, -13.42, -76.045)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir
da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            d753visao1[0] = d753pf1[0] - d753visao1[0]
            d753visao1[1] = d753pf1[1] - d753visao1[1]
            d753visao1[2] = d753pf1[2] - d753visao1[2]
            d753visao2[0] = d753pf2[0] - d753visao2[0]
            d753visao2[1] = d753pf2[1] - d753visao2[1]
            d753visao2[2] = d753pf2[2] - d753visao2[2]
            d753visao3[0] = d753pf3[0] - d753visao3[0]
            d753visao3[1] = d753pf3[1] - d753visao3[1]
            d753visao3[2] = d753pf3[2] - d753visao3[2]

            #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem os novos pontos
de referência
            d753p1 = d753visao1[0:3]+d753pf1[3:]
            d753p2 = d753visao2[0:3]+d753pf2[3:]
            d753p3 = d753visao3[0:3]+d753pf3[3:]

           #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
            pose_user102 = calc_coord(d753p1, d753p2, d753p3, ref=DR_BASE, mod=0)

             #Vai alterar a pose e o sistema de coordenadas de referência para o novo sistema de
coordenadas calculado
            overwrite_user_cart_coord(102, pose_user102, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(102)

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate1 != "1":
            set_output_register_int(1, 10)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, -80, 115, 0, 90, -90.0), v=100, a=100, r=50) #Home Position
            set_output_register_int(1, 0)
            time.sleep(2)
            set_output_register_int(0, 401)
            goto(1)
            continue

        goto(1103)
        break

    while line == 1103:

        #Vai buscar os valores referentes ao disco esquerdo
        wait_digital_input(16, ON)
        rx_msg2 = trispector.readata(128, 6)  #Guarda na variável rx_msg a informação enviada pela
câmara relativa à imagem adquirida
        rxd2 =(rx_msg2[0:128])
        #Uma vez que a informação enviada pela câmara vem toda seguida separada por virgulas, aqui
irá dividir-se a informação pelas virgulas criando uma array
        data2 = rx_msg2.split(', ')
```

```
          valid_locate2 =(rx_msg2[0:1]) #Vai guardar a informação se a imagem foi adquirida
corretamente

        #Valores imagem referência do disco esquerdo (valores do frame de referência criado)
        refxe1 = -72.807635
        refye1 = 219.099548
        refze1 = 215.918209
        refxe2 = 112.165183
        refye2 = 225.036077
        refze2 = 217.896962
        refxe3 = 46.181223
        refye3 = 72.956898
        refze3 = 252.125790


        if valid_locate2=="1": #Verifica se a imagem adquirida foi adquirida com sucesso

            trispector.closeIN(None)
            trispector.closeOUT(None)

             #Diferença entre o frame adquirido pelo sistema de visão com o frame de referência
para saber quanto se moveu a chapa da posição de referência
            xe1 =  float(data2[1]) - refxe1
            ye1 =  float(data2[2]) - refye1
            ze1 =  float(data2[3]) - refze1
            xe2 =  float(data2[4]) - refxe2
            ye2 =  float(data2[5]) - refye2
            ze2 =  float(data2[6]) - refze2
            xe3 =  float(data2[7]) - refxe3
            ye3 =  float(data2[8]) - refye3
            ze3 =  float(data2[9]) - refze3
            xe1 = round(xe1, 6)
            ye1 = round(ye1, 6)
            ze1 = round(ze1, 6)
            xe2 = round(xe2, 6)
            ye2 = round(ye2, 6)
            ze2 = round(ze2, 6)
            xe3 = round(xe3, 6)
            ye3 = round(ye3, 6)
            ze3 = round(ze3, 6)

            #Guarda a diferença das posições dos pontos de aperto de referência
            e753visao1 = [-xe1, -ye1, -ze1]
            e753visao2 = [-xe2, -ye2, -ze2]
            e753visao3 = [-xe3, -ye3, -ze3]

             #Define os pontos de referência das posições de aperto usadas para gerar o frame de
referência
            e753pf1=posx(556.87, 320.99, 372.7, 45.87, 175.04, 52.39)
            e753pf2=posx(742.6, 325.49, 381.97, 49.99, 176.05, 57.43)
            e753pf3=posx(676.53, 172.99, 411.64, 86.42, 178.9, 100.51)
            e753f1=posx(556.87, 320.99, 372.7, 104.152, -167.301, -76.926)

            #Vai calcular os novos pontos de referência subtraindo a diferença calculada a partir
da imagem adquirida pela câmara aos pontos exatos de cada parafuso de referência
            e753visao1[0] = e753pf1[0] - e753visao1[0]
            e753visao1[1] = e753pf1[1] - e753visao1[1]
            e753visao1[2] = e753pf1[2] - e753visao1[2]
            e753visao2[0] = e753pf2[0] - e753visao2[0]
            e753visao2[1] = e753pf2[1] - e753visao2[1]
            e753visao2[2] = e753pf2[2] - e753visao2[2]
            e753visao3[0] = e753pf3[0] - e753visao3[0]
            e753visao3[1] = e753pf3[1] - e753visao3[1]
            e753visao3[2] = e753pf3[2] - e753visao3[2]

            #Acrescenta às novas posições de referência o Rx, Ry e Rz de onde vem os novos pontos
de referência
            e753p1 = e753visao1[0:3]+e753pf1[3:]
            e753p2 = e753visao2[0:3]+e753pf2[3:]
            e753p3 = e753visao3[0:3]+e753pf3[3:]

           #Cria o novo user frame com base na nova posição da chapa obtida pelo sistema de visão
```

```
            pose_user104 = calc_coord(e753p1, e753p2, e753p3, ref=DR_BASE, mod=0)

            #Vai alterar a pose e o sistema de coordenadas de referência para o novo sistema de
coordenadas calculado
            overwrite_user_cart_coord(104, pose_user104, ref=DR_BASE, apply_mod=DR_TEMPORARY)

            pose, eref = get_user_cart_coord(104)

        #Caso a imagem adquirida não tenha sido adquirida com sucesso aborta e volta ao inicio
        if valid_locate2 != "1":
            set_output_register_int(1, 10)
            trispector.closeIN(None)
            trispector.closeOUT(None)
            movej(posj(0, -80, 115, 0, 90, -90.0), v=100, a=100, r=50)
            set_output_register_int(1, 0)
            time.sleep(2)
            set_output_register_int(0, 401)
            goto(1)
            continue

        set_output_register_int(0, 220)
        set_tcp("U_Tool_1")

        goto(1104)
        break


    while line == 1104:

        free_zone=get_input_register_int(4) #Verifica se a zona está livre
        if(free_zone==3333):
            set_output_register_int(3, 777)
            set_output_register_int(0, 225)
            mwait(0)

            set_ref_coord(104) #Seleciona o novo frame criado com os valores do sistema de visão,
que irá corrigir os novos pontos de aperto das restantes porcas
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            movej(posj(0, -55, 120, 0, 60, -90), v=100, a=100, r=50) #Home Position

            #A partir deste ponto irá passar-se para a fase de aperto das porcas
            movej(posj(65.1, 1.44, 88, 53.78, -52.74, -146.2), v=100, a=100, r=50)
            movej(posj(63.03, 11.21, 88, 95.37, -71.01, -104.08), v=100, a=100, r=50)
            movej(posj(83.74, 30.96, 72.2, 120.47, -110.89, -109.79), v=100, a=100, r=50)
            set_output_register_int(0, 229)
            set_output_register_int(3, 0)

            movel(System_753e_pa1, v=100, a=100, r=10) #Ponto de aproximação do primeiro parafuso
esquerdo
            set_output_register_int(0, 230)####
            set_digital_output(13, ON)
            movel(System_753e_p1, v=50, a=50) #Ponto de aperto do primeiro parafuso esquerdo

            #Aperto P1
            set_output_register_int(2, 55)
            time.sleep(0.2)
            wait_torque()
            time.sleep(0.1)
            set_output_register_int(2, 0)

            movel(System_753e_pa1, v=100, a=100, r=0) #Ponto de aproximação do primeiro parafuso
esquerdo
            set_digital_output(13, OFF)
            set_output_register_int(3, 0)
            set_output_register_int(0, 235)
            goto(1105)
            break


    while line == 1105:

        set_output_register_int(0, 239)
```

```
        set_output_register_int(3, 777)
        set_output_register_int(4, 888) # Info para PLC Zona Ocupada
        time.sleep(0.2)
        set_output_register_int(3, 0)

        movej(posj(83.74, 30.96, 72.2, 120.47, -110.89, -109.79), v=100, a=100, r=50)
        movej(posj(63.03, 11.21, 88, 95.37, -71.01, -104.08), v=100, a=100, r=50)
        movej(posj(65.1, 1.44, 88, 53.78, -52.74, -146.2), v=100, a=100, r=50)

        #Movimentos para ir fazer o aperto no disco direito
        movej(posj(-53.04, -27.02, 83.45, 10.99, 80.35, -88.35), v=100, a=100, r=50)
        movej(posj(-85.59, 8.52, 73.82, 43.14, 92.85, -87.35), v=100, a=100, r=50)
        movej(posj(-99.43, 44.09, 49.3, 60.04, 128.19, -74.85), v=100, a=100, r=0)
        set_output_register_int(3, 0)##
        mwait(0)##
        set_ref_coord(102) #Seleciona de novo o frame calculado
        set_output_register_int(4, 0)
        set_output_register_int(0, 300)
        goto(1108)
        break


    while line == 1108:

        free_zone=get_input_register_int(4) #Verifica se a zona está livre
        if(free_zone==3333):
            set_output_register_int(3, 777)
            set_output_register_int(4, 888) # Info para PLC Zona Ocupada
            time.sleep(0.2)
            goto(1109)
            break


    while line == 1109:

        #Vai proceder ao aperto do parafuso do disco direito
        set_output_register_int(3, 0)
        movej(posj(-88.8, 33.24, 76.59, 57.34, 110.69, -84.66), v=100, a=100, r=50)
        set_output_register_int(0, 310)

         movel(System_753d_pa1, v=100, a=100, r=2) #Ponto de aproximação do primeiro parafuso
direito
        set_digital_output(13, ON)
        movel(System_753d_p1, v=50, a=50) #Ponto de aperto do primeiro parafuso direito

        #Aperto P1
        set_output_register_int(2, 55)
        time.sleep(0.2)
        wait_torque() #Dá inicio ao aperto, mandando sinal para a aparafusadora começar o aperto
e espera pelo relatório de final de aperto
        time.sleep(0.1)
        set_output_register_int(2, 0)

         movel(System_753d_pa1, v=100, a=100, r=20) #Ponto de aproximação do primeiro parafuso
direito
        movej(posj(-99.43, 44.09, 49.3, 60.04, 128.19, -74.85), v=100, a=100, r=0)
        set_output_register_int(3, 0)
        set_output_register_int(0, 320)
        set_digital_output(13, OFF)
        set_output_register_int(0, 330)

        #Movimentos para voltar à Home Position
        movej(posj(-99.43, 44.09, 49.3, 60.04, 128.19, -74.85), v=150, a=150, r=50)
        movej(posj(-85.59, 8.52, 73.82, 43.14, 92.85, -87.35), v=150, a=150, r=50)
        movej(posj(-53.04, -27.02, 83.45, 10.99, 80.35, -88.35), v=150, a=150, r=50)
        movej(posj(-43.22, -49.01, 112.69, 28.16, 64.67, -136.33), v=150, a=150, r=50)
        movej(posj(0, -80, 115, 0, 90, -89.99), v=100, a=100, r=0)

        set_output_register_int(3, 0)
        set_output_register_int(4, 0)
        set_output_register_int(0, 400)
        time.sleep(1)
```

```
        set_ref_coord(DR_BASE)
        goto(1)
        break

    #FIM DO CÓDIGO
```