Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA

Integração e demonstração de plataformas de robótica móvel e colaborativa

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

Especialização em Automação e Comunicações em Sistemas Industriais

Autor

André Pessoa Gonçalves

Orientadores

Nuno Miguel Fonseca Ferreira

Fernanda de Madureira Coutinho

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, junho de 2025

Resumo

A indústria tem sido alvo de transformações significativas ao longo dos anos, particularmente na sequência das várias revoluções industriais, com o objetivo principal de aumentar a produtividade e a eficiência através da adoção de novas tecnologias. Neste contexto, destaca-se a área da robótica, uma área de aplicação ampla e diversificada no setor industrial. Um exemplo é a utilização crescente de robôs móveis e colaborativos em ambiente industrial. Os primeiros permitem o transporte automatizado e otimizado de materiais e equipamentos, enquanto os segundos possibilitam a realização de tarefas em colaboração direta com operadores humanos. Ambos contribuem para a melhoria da eficiência e rentabilidade dos processos industriais.

Motivado por um interesse nesta área, o aluno integrou, como bolseiro de investigação, o projeto de I&D intitulado "Agenda GreenAuto – Green Innovation for the Automotive Industry", do qual o ISEC é parceiro. A participação neste projeto ofereceu a oportunidade de o aluno colaborar com duas empresas de renome nos setores automóvel e da robótica, como a Stellantis e a Europneumaq, respetivamente.

Esta colaboração serviu de base para a definição dos objetivos desta tese de mestrado. Como resultado, o presente relatório descreve o desenvolvimento e validação experimental de duas soluções tecnológicas: i) uma plataforma Web para a monitorização de uma frota de robôs móveis, permitindo aos utilizadores acompanhar em tempo real o estado e localização destes dispositivos; ii) um protótipo de aparafusamento de componentes automóveis que combina a utilização de um robô colaborativo, um sistema de visão 3D e de uma aparafusadora elétrica.

O trabalho desenvolvido representa um contributo relevante para o aumento da segurança na indústria, a promoção da inovação tecnológica, o incremento da produtividade e a maior flexibilidade nos processos produtivos.

Parte do trabalho desenvolvido encontra-se publicado no Journal Automation [1], de Quartil 2, e encontra-se em processo de revisão um outro artigo submetido para uma revista de Quartil 1.

Palavras-chave: Robótica, Robô Móvel, Robô Colaborativo, Monitorização, Visão 3D

Abstract

Over the years, the industrial sector has undergone significant transformations, particularly as a result of successive industrial revolutions, with the primary goal of increasing productivity and efficiency through the adoption of new technologies. Within this context, robotics has emerged as a key area of broad and diverse application in industry. A notable example is the growing use of mobile and collaborative robots in industrial environments. Mobile robots enable the automated and optimized transport of materials and equipment, while collaborative robots are capable of performing tasks in direct cooperation with human operators. Both contribute significantly to enhancing the efficiency and profitability of industrial processes.

Driven by a strong interest in this field, the student joined the R&D project entitled "Agenda GreenAuto – Green Innovation for the Automotive Industry" as a research fellow, with ISEC as a project partner. This participation provided the opportunity to collaborate with two renowned companies in the automotive and robotics sectors: Stellantis and Europneumaq, respectively.

This collaboration served as the foundation for defining the objectives of the present master's thesis. Accordingly, this report describes the development and experimental validation of two technological solutions: i) a web-based platform for monitoring a fleet of mobile robots, enabling users to track the status and location of these devices in real time; ii) a prototype for automotive component screwing operations, combining the use of a collaborative robot, a 3D vision system, and an electric screwdriver.

The work carried out makes a relevant contribution to improving safety in industry, fostering technological innovation, increasing productivity, and providing greater flexibility in production processes.

Part of this work has been published in Journal Automation [1], a Quartile 2 journal, and another article is currently under review for a Quartile 1 journal.

Keywords: Robotics, Collaborative Robot, Mobile Robot, Monitoring, 3D Vision

Epígrafe

A robótica não existe para substituir o ser humano, mas para desempenhar as tarefas robotizadas dele.

Carlos Alberto Hang

Agradecimentos

Gostaria de expressar a minha profunda gratidão ao Professor Nuno Ferreira, meu orientador, pela oportunidade de trabalhar sob a sua orientação, pela partilha de conhecimentos e pela excelente oportunidade de fazer parte de um projeto tão enriquecedor. Foi uma experiência única, permitindo-me o contacto com empresas de referência no setor da robótica e da indústria automóvel, duas áreas que sempre me fascinaram.

À Professora Fernanda Coutinho, minha orientadora, agradeço igualmente pela possibilidade de colaborar consigo, pelo saber que partilhou e pelas sugestões e correções fundamentais que me ajudaram a aprimorar este trabalho. O seu acompanhamento constante e paciência foram cruciais para que este projeto pudesse ser concluído com sucesso.

Aos meus colegas e amigos, David Lopes e Francisco Cunha, deixo um especial agradecimento pela amizade e apoio contínuos ao longo destes anos. A nossa amizade ultrapassou em muito o contexto académico, e sem dúvida que este percurso teria sido muito mais difícil sem eles ao meu lado. Agradeço também ao Fernando Lopes, João Antunes e Tiago Pereira, meus colegas bolseiros no projeto, pelo companheirismo, dedicação e colaboração essencial ao longo do desenvolvimento do trabalho.

À minha família – pais, irmão, avós, tios, primos – e à minha namorada, sem os quais não teria sido possível chegar até aqui. Agradeço por me terem proporcionado o melhor ambiente familiar possível, pelo apoio incondicional, pela fé e por terem estado sempre disponíveis para me ajudar e facilitar os meus dias mais complicados. Se hoje estou aqui, devo-vos isso a vocês.

Por fim, a todos os que de alguma forma contribuíram para a realização deste trabalho, o meu sincero obrigado.

Índice

Índice de tabelas

Índice de figuras

Lista de abreviaturas

AGV	*Automated Guided Vehicle*
AMR	*Autonomous Mobile Robot*
API	*Application Programming Interface*
AP	*Access Point*
CSS	*Cascading Style Sheets*
DEE	Departamento de Engenharia Eletrotécnica
FMS	*Flexible Manufacturing System*
FTP	*File Transfer Protocol*
GPS	*Global Positioning System*
HMI	*Human-Machine Interface*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IA	Inteligência Artificial
ISEC	Instituto Superior de Engenharia de Coimbra
PME	Pequenas e Médias Empresas
QR	*Quick Response*
RFID	*Radio-Frequency Identification*
RMS	*Root Mean Square*
SLAM	*Simultaneous Localization and Mapping*
SMAP	*Simultaneous Mapping and Positioning*
SMTP	*Simple Mail Transfer Protocol*
TCP	*Tool Center Point*
TCP	*Transmission Control Protocol*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
ToF	*Time-of-Flight*
UDP	*User Datagram Protocol*

1 Introdução

O presente relatório descreve o trabalho de projeto elaborado no âmbito da Unidade Curricular de Projeto do Mestrado em Engenharia Eletrotécnica do Instituto Superior de Engenharia de Coimbra (ISEC), iniciado no Ano Letivo 2023/2024.

O trabalho desenvolvido enquadra-se na área da robótica colaborativa e da robótica móvel utilizada em contexto industrial. O plano de trabalhos foi definido de forma articulada com os objetivos do Projeto "Agenda GreenAuto: Inovação Verde para a Indústria Automóvel", financiado pelo Plano de Recuperação e Resiliência, no qual os orientadores são investigadores. O ISEC integra o respetivo consórcio, tendo o aluno sido Bolseiro de Investigação neste projeto.

Todo o trabalho desenvolvido envolveu uma forte componente de validação experimental em contexto real, em colaboração com dois dos parceiros do consórcio, nomeadamente as empresas Europneumaq e a Stellantis Mangualde.

1.1 Enquadramento

Estamos a viver o início de uma nova revolução industrial, onde os avanços tecnológicos transformam profundamente a interação entre as pessoas e o funcionamento das empresas. Neste contexto, é essencial que as empresas se mantenham atualizadas com as inovações tecnológicas, sob o risco de serem ultrapassadas pela concorrência.

O conceito de Indústria 4.0, que abrange temas como "Colaboração Homem-Robô", "Internet das Coisas", "*Big Data*" e "Simulação Industrial", procura melhorar a eficiência e produtividade dos processos. A União Europeia vê na Indústria 4.0 e na mais recente Indústria 5.0 uma oportunidade para revitalizar a indústria, promovendo e financiando projetos que incentivem a adoção de novas tecnologias com o envolvimento de instituições de ensino superior e de empresas. Tem-se assistido, nos últimos anos, a avanços significativos na área da robótica, especialmente no que diz respeito aos robôs móveis e robôs colaborativos em contexto industrial. Estas duas áreas têm ganho destaque em diversos setores industriais, proporcionando soluções inovadoras para melhorar a eficiência, produtividade e segurança nos ambientes de trabalho.

O trabalho desenvolvido teve dois grandes focos principais que envolveram robótica móvel (industrial) e robótica colaborativa:

1. Os ***Automated Guided Vehicle*** (**AGV**) são sistemas automatizados projetados

para transporte e movimentação de materiais em ambientes industriais. Estes robôs são equipados com sensores, sistemas de navegação e, por vezes, guias físicas, como fitas magnéticas ou marcadores no chão, que permitem a sua orientação e deslocação dentro de um espaço definido. Tipicamente, operam de forma autónoma e seguem rotas predefinidas.

Uma das principais características destes dispositivos é a sua capacidade de operar de forma autónoma, seguindo rotas predefinidas e evitar o choque contra obstáculos ao longo do caminho. Estes robôs são frequentemente utilizados em armazéns, fábricas e centros de distribuição para transporte de paletes, contentores e outros materiais pesados.

2. Os **Robôs Colaborativos** (**cobots**) são concebidos para interagir de forma segura e eficiente com os seres humanos no local de trabalho. Ao contrário dos robôs industriais tradicionais, que operam em ambientes isolados e requerem barreiras físicas de segurança, os cobots são projetados para trabalhar lado a lado com os trabalhadores, realizando tarefas de forma colaborativa.

   Uma das características distintivas dos cobots é a sua capacidade de detetar a presença humana e ajustar o seu comportamento em conformidade para evitar colisões e garantir a segurança do operador. Estes robôs são frequentemente utilizados em linhas de montagem e outras atividades que requerem uma interação próxima entre humanos e máquinas. Estes robôs são capazes de ajustar o seu comportamento em tempo real com base na presença humana e nas condições do ambiente de trabalho.

1.2 Motivação e metodologia

Tendo em vista a melhoria da eficiência do processo produtivo, foram identificados dois pontos críticos que se afiguram objeto de intervenção e resolução com este projeto. Em primeiro lugar, foi identificada a necessidade de melhorar a eficiência de gestão dos AGV existentes na fábrica da Stellantis-Mangualde, atendendo que não existe uma supervisão global centralizada destes equipamentos durante o seu funcionamento e carregamento nos locais estabelecidos. Por este motivo, quando existe uma anomalia num dos AGV que impedem o seu normal desempenho, esta situação é muitas vezes detetada apenas quando existem repercussões de maior dimensão, as quais por vezes despoletam paragens na cadeia de abastecimento, sendo possível minimizar a propagação do problema, se detetado precocemente, pelo que se propõe um sistema de supervisão geral do parque de AGV, por meio de um sistema conjunto com Sistema Flexível de Fabrico (FMS) / Interface Homem-Máquina (HMI). Para além disso, foi identificado que na linha de montagem do veículo, o processo produtivo de aperto do *Cloison* (placa divisória entre passageiros e mercadorias) e *Porteur* (elemento da roda

que contém o disco de travão e a manga de eixo) pode ser otimizado e melhorado através da substituição da execução manual da tarefa por uma abordagem automatizada, recorrendo à montagem dos componentes com o apoio de robôs colaborativos.

1.2.1 Gestão de frota de robôs móveis

Os AGV são dispositivos ambulatórios que estão sujeitos a perturbações que podem impedir o seu regular funcionamento. Estas perturbações podem incluir a necessidade de operações de manutenção (por exemplo, recarregar as baterias), o aparecimento de obstáculos no seu caminho (de natureza transitória ou permanente), ou então a perda de referência de posição (por exemplo, dificuldades de leitura de uma *tag* RFID) que compromete a sua capacidade de seguir uma rota predeterminada. Neste caso, o AGV desvia-se da rota pretendida e, em termos práticos, poderá efetivamente perder-se porque deixa de ter acesso aos necessários referenciais de localização.

A resolução deste tipo de problemas obriga, normalmente, a intervenção manual de um operador para repor o normal funcionamento (por exemplo, substituir a bateria ou reposicionar o AGV na rota correta). No entanto, para que tal seja possível, é necessário não só que o operador tenha conhecimento do evento adverso, como também que seja informado da real posição do AGV, para que o possa encontrar com facilidade. No caso de uma instalação fabril de grande dimensão com uma frota heterogénea de AGV, a identificação da necessidade de intervenção e determinação do real posicionamento não é trivial. A deteção de um problema pode ocorrer apenas tardiamente, por exemplo, através do alerta de um trabalhador quando as peças necessárias para uma determinada etapa da produção não chegam atempadamente. Nesta situação, a deteção do problema é tardia e irá implicar um atraso ou mesmo paragem da produção, com perdas de eficiência. Para além disso, caso a informação de localização não esteja disponível para consulta, o operador necessita de se deslocar para procurar o AGV em questão em todo o perímetro acessível, o que irá atrasar o período de resposta e aumentar o impacto negativo no processo de produção.

Um sistema de gestão de frota de AGV dá resposta a estas necessidades, fornecendo uma perspetiva centralizada de todos os parâmetros de funcionamento relevantes de todos os AGV em funcionamento. Esta informação inclui parâmetros operacionais de cada AGV, como por exemplo a carga da bateria (o que permite realizar as operações de recarregamento preventivamente sem disrupção da produção) ou a sua localização. O sistema de gestão inclui também, tipicamente, uma funcionalidade que permite o envio de notificações no caso de surgirem situações anómalas que obriguem à intervenção de um operador (por exemplo, uma obstrução na rota que precisa de ser removida).

Para ultrapassar estas dificuldades, podem ser investigadas soluções baseadas na instrumentação do ambiente, com capacidade para recolher informação de cada AGV independentemente do seu modelo, assim como a apresentação agregada, através de

Figura 1.1: ASTI BidiBot[1], TriBot[2] e EasyBot[3].

Tabela 1.1: Comparação das características de três AGV.

Características	ASTI BidBot	ASTI TriBot	ASTI EasyBot
Capacidade de Carga Máxima	2000 kg	5000 kg	6000 kg
Dimensões	2104 x 500 x 230 mm	1221 x 695 x 762 mm	1681 x 273 x 600 mm
Movimento	Bidirecional (AGV coloca-se por baixo da carga a rebocar)	Unidirecional (Sistema de reboque tradicional)	Unidirecional automático Bidirecional manual (AGV coloca-se por baixo da carga a rebocar)
Gama de Velocidade	0,01 até 1,5 m/s	0,035 até 2 m/s	0,1 até 0,83 m/s
Precisão de Posicionamento	$\pm$ 10 mm	$\pm$ 10 mm	-
Movimento Manual	Comando de controlo, Wireless	Comando de controlo, Wireless	Comando de controlo, Wireless
Sistemas de Segurança	-	-	-
• Laser Segurança	X2	X	X1
• PLC Segurança	X	X	X1
• Botão de emergência	X2	X4	X1
Navegação	Magnético / SLAM / RFID	Magnético / SLAM / RFID	Magnético / RFID
Sistema de Energia	-	Íões de Lítio	Chumbo-ácido
• Bateria	-	Íões de Lítio	Chumbo-ácido
• Tipo de Carregamento	Carregamento *online* ou carregador externo	Carregamento *online* ou carregador externo	Carregamento *online* ou carregador externo
Comunicação	WiFi, radio, M2M	WiFi, radio, M2M	WiFi, radio
Conectividade	2x HDMI, WiFi, USB & Ethernet, ERP, MES, WMS, tablets, PCL interface (OPC)	2x USB, Ethernet, IoT Through IT: ERP, MRP, MES, DDBB interface Industrial connectivity: OPC, wired connection	WiFi, Ethernet, PCL interface (OPC)

uma interface comum, de informação interna específica de cada modelo de AGV (p.ex., nível da bateria). Na Figura 1.1 são apresentados três robôs móveis diferentes da ASTI: BidiBot, TriBot e EasyBot.

Na Tabela 1.1 é apresentada uma comparação detalhada de três modelos de AGV diferentes, pertencentes todos à mesma marca, a ASTI. É possível verificar as diferenças ao nível das dimensões, carga máxima, entre outras características relevantes. O Asti EasyBot pertence a uma geração mais antiga comparativamente aos outros dois AGV alvos desta comparação.

Assim sendo, um dos principais objetivos do sistema de gestão de frotas é a criação de uma aplicação informática que permita visualizar em tempo real os AGV. Esta visu-

[1]Fonte: `https://www.leobotics.fr/fournisseur/asti-mobile-robotics/`
[2]Fonte: `https://www.europneumaq.com/pt/solucoes/robotica/robots-moveis`
[3]Fonte: `https://tinyurl.com/abf5c563`

alização irá conter a informação da localização de todos os AGV relativamente à rota escolhida. Nesta será possível verificar de forma intuitiva o estado de cada robô, bem como, a informação, em tempo real, do sentido que o mesmo tem. Para além da informação disponibilizada, se o AGV estiver sobre alguma anomalia irá mudar o seu estado e será gerado um alarme para informar ao utilizador, novamente em tempo real, dessa mesma anomalia ou paragem não programada.

Estas paragens não programadas dos AGV, normalmente devem-se às seguintes anomalias:

- Identificação do número do AGV que se encontra com anomalia ou em estado de alarme;
- Banda magnética danificada;
- Estado do nível de bateria baixo;
- Leitor de *tags* RFID danificado;
- Inexistência de *tag* no circuito;
- Erro no tratamento de dados;
- Necessidade de emparelhar o sistema de supervisão dos AGV à matrícula do AGV do circuito físico;
- Farol de comunicação entre os AGV;
- Sinótico das diferentes rotas.

1.2.2 Sistema de aparafusamento com manipulador robótico colaborativo

Foi identificado que na linha de montagem da Stellantis Mangualde o processo de produção e de montagem dos veículos poderia ser otimizado com a inclusão de robôs colaborativos para o aperto de componentes automóveis.

Os componentes automóveis alvos de intervenção são:

- *Cloison* (chapa divisória entre passageiros e mercadoria), presente na Figura 1.2;
- *Porteur* (conjunto do elemento da roda), que pode ser observado na Figura 1.3.

No caso do projeto *Cloison* é necessário efetuar 12 apertos de porcas (a vermelho na Figura 1.2), sendo que a estrutura onde são aplicadas estas porcas, possui outras 3 porcas previamente apertadas pelo operador (a verde na Figura 1.2), as quais chegam a esta fase do trabalho já apontadas no devido lugar, de modo a garantir que a estrutura esteja suficientemente segura e imóvel, quando o robô tiver de operar sobre a mesma.

Por outro lado no projeto *Porteur*, serão utilizadas duas aparafusadoras, as quais necessitam de ter capacidade de aparafusamento com torques distintos e regulados para

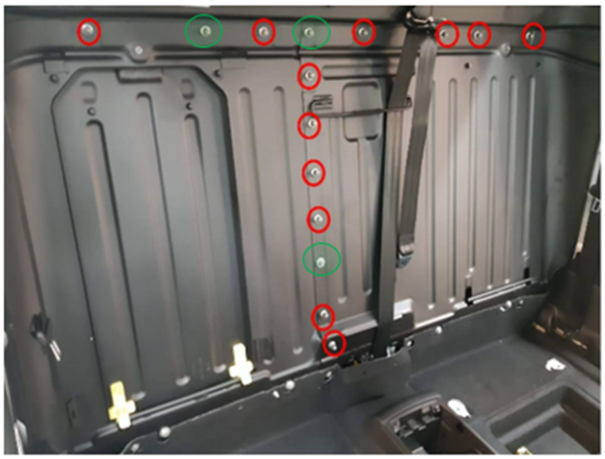
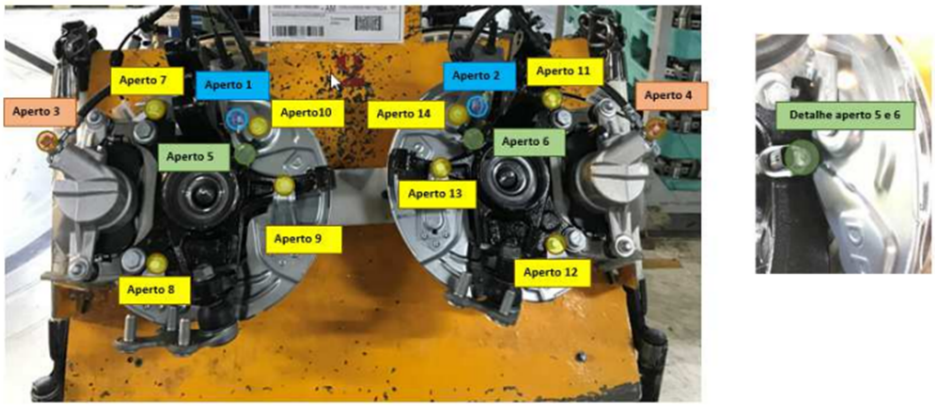


Figura 1.2: *Cloison*.



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

Figura 1.3: *Porteur*.

as especificações técnicas impostas para cada porca (Figura 1.3), bem como a possibilidade de troca para a chave adequada para aparafusar os elementos a que estão destinadas.

Assim, dada a necessidade de partilha do espaço de trabalho com os funcionários da empresa, foi estipulado inicialmente, que seriam utilizados os cobots da série M da Doosan, exibidos na Figura 1.4, tendo em consideração os requisitos de cada aplicação a desenvolver.

A Tabela 1.2 discrimina as principais características de cada um dos diferentes modelos da Série M da Doosan. Esta informação servirá de base à análise das várias soluções possíveis, para que seja tomada a melhor decisão tendo em conta os requisitos e as especificidades dos projetos a desenvolver.

[4]Fonte: `https://www.doosanrobotics.com/en/product-solutions/product/#mSeries`

Figura 1.4: Robôs da série M da Doosan Robotics[4].

Tabela 1.2: Principais características dos robôs da série M da Doosan [2].

Características	M0609	M0617	M1013	M1509
Capacidade de carga	6 kg	6 kg	10 kg	15 kg
Alcance	900 mm	1700 mm	1300 mm	900 mm
Precisão da repetibilidade	$\pm$0,03 mm	$\pm$0,1 mm	$\pm$0,05 mm	$\pm$0,03 mm

No sistema de aperto pertencente aos dois protótipos terá ainda de existir um sistema de visão atuando em conjunto com o robô.

Assim sendo, após uma avaliação cuidadosa das soluções para sensorização 3D disponíveis no mercado, e em linha com as tecnologias identificadas como mais promissoras para o presente caso de estudo, foram selecionados quatro sensores óticos 3D distintos, os quais são discriminados na Tabela 1.3.

Tabela 1.3: Lista dos sistemas de visão selecionados e suas principais características [3, 4, 5, 6].

Características	Sick TriSpector 1030	PhotoNeo S	Asus Xtion Pro	Zed Mini
Categoria	Triangulação Câmara Laser	Luz Estruturada Visível	Luz Estruturada Infravermelho	Visão Stereo
Resolução	2500 pontos por varrimento	0,8 – 3,2 M pontos	640 x 480	2208 x 1242
Campo de visão	65º	49º H, 36º V	58º H, 45º V, 70º D	90º H, 60º V, 100º D
Erro de profundidade	< 280 µm	< 0,05 mm	< 5 mm	< 1,5% até 3000 mm < 7% até 15000 mm
Nuvem de Pontos Cor/Escala de Cinza	Escala de Cinza	Escala de Cinza	Cor	Cor
Profundidade mínima	141 mm	384 mm	800 mm	100 mm
Profundidade máxima	541 mm	520 mm	3500 mm	15000 mm
Taxa de quadros (Frame rate)	5000 perfis 3D/s	5 – 6,6 Hz	30 Hz	15 Hz
Driver ROS	–	Sim	Sim	Sim
Sistemas Operativos Suportados	Linux, Windows	Linux, Windows	Linux, Windows	Linux, Windows
Conexão	Ethernet	USB 3.0	USB 3.0	USB 3.0
Dimensões	217 x 62 x 84 mm	77 x 68 x 296 mm	280 x 80 x 80 mm	124,5 x 30,5 x 26,5 mm
Peso	1,3 kg	900 g	$\pm$ 100 g	62,9 g
Preço	$\pm$ 5000 €	$\pm$ 10000 €	$\pm$ 250 €	$\pm$ 300 €

Através da análise da Tabela 1.3 é possível perceber que dois dos sensores identificados:

Tabela 1.4: Características da aparafusadora [7].

Modelo Aparafusadora	Desoutter ECS10-M20
Número da peça	6151654580
Drive de saída	1/4" Hex.
Velocidade máxima	1340 rpm
Intervalo de torque	2,5-10 Nm
Peso líquido	0,7 kg
Comprimento	285 mm

Sick TriSpector e PhotoNeo S são especificamente concebidos para serem utilizados na indústria, sendo de custo mais avultado e caracterizados por uma elevada precisão e fiabilidade. Por outro lado, da análise dos sensores utilizados apenas em investigação, tais como Asus Xtion Pro ou Zed Mini verifica-se que estes são significativamente menos dispendiosos, podendo ser suficientes para lidar com tarefas que requerem menor precisão. Ainda assim e devido ao facto do projeto ter que ser integrado em chão de fábrica, escolheu-se o sensor Sick TriSpector 1030 para os protótipos no ISEC.

Para completar o hardware necessário à realização destes projetos de aparafusamento, como a própria tarefa indica é necessária uma aparafusadora. Tendo em conta as necessidades de cada um dos projetos que envolve sistemas de aparafusamento, será necessária a utilização de ferramentas de aperto de parafusos/porcas específicas e adequadas às tarefas a realizar, sendo estas selecionadas tendo por base a experiência dos engenheiros e técnicos da empresa Europneumaq na aplicação de aparafusadoras em projetos anteriormente desenvolvidos. Assim sendo, a aparafusadora escolhida para os protótipos dos dois projetos foi a Desoutter ECS 10-M20, cujas características se encontram presentes na Tabela 1.4.

A aparafusadora tem que possuir o seu respetivo controlador compatível com a mesma, neste caso o controlador adquirido foi o Desoutter CVICII-H2 com as suas características presentes na Tabela 1.5.

Tabela 1.5: Características do controlador da aparafusadora [7].

Modelo do Controlador da Aparafusadora	Desoutter CVICII-H2
Número da peça	6159326770
Peso líquido	5 kg
Largura	130 mm
Profundidade	291 mm
Altura	278 mm
Consumo (corrente 115V)	4A 115V
Consumo (corrente 230V)	2A 230V
Energia 50/60Hz	Monofásico 100 a 250V

1.3 Projeto GreenAuto e colaboração com empresas parceiras

Nesta secção vai apresentar-se de forma breve as características globais do Projeto GreenAuto bem como sobre duas das empresas parceiras do consórcio deste projeto com as quais o aluno interagiu ativamente no decorrer deste trabalho, nomeadamente as empresas Europneumaq e a Stellantis.

1.3.1 Projeto Agenda GreenAuto

O Projeto Greenauto, cofinanciado pelo Plano de Recuperação e Resiliência Português, é uma iniciativa inovadora que pretende posicionar a indústria automóvel em Portugal como referência na produção de veículos com baixas emissões. Liderado pela Stellantis Mangualde, tem como objetivo preparar a fábrica para a produção de veículos comerciais ligeiros totalmente elétricos (BE-LCVs). Para alcançar, o projeto tem como objetivo o desenvolvimento de 18 produtos e processos inovadores, digitais e sustentáveis, associados ao "*Green Vehicle*" e "*Green Factory*". A Figura 1.5 traduz uma representação global do projeto *GreenAuto*, com os diversos produtos e processos inovadores envolvidos. O Projeto Agenda GreenAuto envolve um consórcio com 36 entidades do qual

Figura 1.5: Representação global do Projeto *GreenAuto* [5].

o ISEC faz parte. O ISEC está envolvido nas *work packages* responsáveis pelo desenvolvimento de soluções mais eficientes, inteligentes e autónomas baseadas em robótica colaborativa e robôs móveis industriais.

1.3.2 Empresa Europneumaq

A empresa Europneumaq, representada na Figura 1.6, é uma empresa fundada em 2001, pertencente ao ramo da robótica e automação industrial, que oferece diversas soluções, a diversos níveis, tais como:

- Aparafusamento e controlo;
- Automação industrial;
- *Grippers*;
- Máquinas e equipamentos;
- Robótica;
- Pneumática;
- Segurança industrial;
- Testes de qualidade.

Oferece também serviços ao nível do suporte técnico, desenvolvimento das soluções e ainda uma assistência pós venda. São representantes ainda de algumas marcas de relevo como por exemplo: ABB, Desoutter, Doosan Robotics, Kawasaki Robotics, OnRobot, entre outras [8].

1.3.3 Empresa Stellantis

A Stellantis Mangualde, localizada em Mangualde, é uma unidade de produção automóvel de grande relevância no contexto industrial nacional e internacional. Pertence ao grupo Stellantis que é um dos maiores fabricantes automóveis do mundo resultante da fusão entre a *PSA Group* e a *Fiat Chrysler Automobiles*. Esta fábrica desempenha um papel fundamental na produção de veículos comerciais ligeiros. É possível observar a extensão das suas instalações através da Figura 1.7. Ao longo dos anos esta fábrica tem-se vindo a destacar com vários marcos importantes desde a sua fundação, podendo-se destacar os seguintes:

- **1962**: Construção da fábrica;
- **1964**: Produção do primeiro Citroën AZL ("2 cavalos");
- **1977**: Início da exportação de veículos;

[5]Fonte: `https://www.agendagreenauto.pt/projeto/`
[6]Fonte: `https://www.facebook.com/Europneumaq/photos_by`
[7]Fonte: `https://tinyurl.com/mrx4jf9b`

Figura 1.6: Instalações da empresa Europneumaq e exemplos de alguns dos seus produtos [6].

Figura 1.7: Instalações da empresa Stellantis Mangualde [7].

- **1987**: Integração nas Unidades de Produção Citroën e comemoração dos 25 anos;
- **1990**: Produção do último Citroën 2CV e início do Citroën AX;
- **1998**: Lançamento da Citroën Berlingo e Peugeot Partner;
- **2009**: Segunda geração da Berlingo e Partner;
- **2012**: Produção do milionésimo veículo e comemoração dos 50 anos;
- **2016**: Integração no Comité Estratégico da iniciativa Indústria 4.0 em parceria com o Governo.

1.4 Objetivos e cronograma

Os principais objetivos deste trabalho de projeto, definidos em articulação com os objetivos do projeto de investigação apresentado na Secção 1.3.1 foram:

1. Desenvolver uma plataforma para a gestão de frota de robôs móveis industriais, capaz de fazer a monitorização do percurso e desempenho dos AGV;
2. Desenvolvimento de aplicações com sistemas robóticos colaborativos de forma a viabilizar a automação de processos produtivos na indústria automóvel.

Para a concretização destes objetivos, foram levadas a cabo as seguintes tarefas:

- **Tarefa 1** - Elaborar o estado da arte sobre a utilização da robótica em contexto

industrial e em particular sobre robótica móvel industrial e robótica colaborativa;

- **Tarefa 2** - Familiarização com as diferentes tecnologias de robôs móveis;
- **Tarefa 3** - Familiarização com as diferentes tecnologias que os robôs colaborativos permitem utilizar;
- **Tarefa 4** - Adquirir conhecimentos sobre segurança em robótica industrial, tanto para robótica móvel como para robótica colaborativa;
- **Tarefa 5** - Elaboração de um sistema de monitorização de robôs móveis numa página Web, utilizando a recolha de dados de um AMR;
- **Tarefa 6** - Elaboração de sistemas de aparafusamento com robôs colaborativos de modo a automatizar processos produtivos na indústria automóvel, com a elaboração do código para o robô colaborativo, algoritmo da câmara de visão 3D e programação de uma aparafusadora elétrica.

Com o intuito de ilustrar a distribuição temporal das tarefas ao longo do desenvolvimento do projeto, apresenta-se na Figura 1.8 um diagrama representativo da sua execução e do período de tempo alocado a cada.

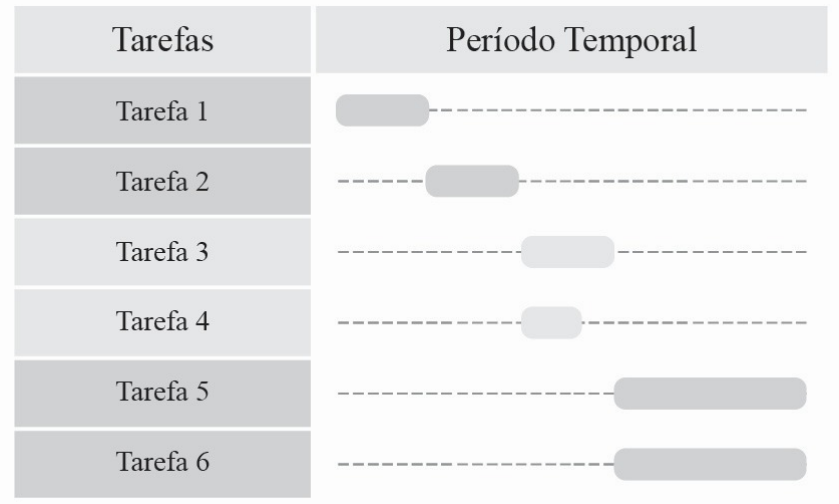


Figura 1.8: Diagrama temporal da execução das tarefas.

1.5 Estrutura do documento

Este documento está estruturado em seis capítulos e 3 anexos:

- No Capítulo 1 - Introdução - faz-se um enquadramento e apresenta-se a motivação subjacente a este trabalho. Antes de serem apresentados os objetivos e cronograma do trabalho desenvolvido, faz-se uma breve contextualização do projeto de investigação em que este trabalho esteve associado bem como duas das empresas com as quais houve colaboração no decorrer deste trabalho.

- No Capítulo 2 - Estado da arte - apresenta-se o enquadramento teórico dos principais temas do trabalho, incluindo uma visão histórica da robótica e a distinção entre robôs industriais, colaborativos e móveis. Explora-se ainda a evolução industrial até à Indústria 5.0 e abordam-se normas aplicáveis à robótica colaborativa e móvel.
- O Capítulo 3 - Gestão de frota de robôs móveis - detalha o desenvolvimento do gestor de robôs móveis.
- O Capítulo 4 - Sistema de aparafusamento com manipulador robótico colaborativo - descrevem-se os passos do desenvolvimento, incluindo detalhes de programação e preparação física dos demonstradores.
- O Capítulo 5 - Resultados experimentais e discussão - apresenta os resultados obtidos e analisa o grau de aplicabilidade dos demonstradores, conduzindo à discussão final. Este capítulo prepara o caminho para as propostas de trabalho futuro abordadas no capítulo seguinte.
- O Capítulo 6 - Conclusões e propostas de trabalho futuro - expõe as conclusões do trabalho e apresenta as propostas para trabalho futuro
- Por fim, nos anexos estão incluídos os códigos desenvolvidos para a plataforma de monitorização de robôs móveis (Anexo A e Anexo B) e para o protótipo de aparafusamento com o robô colaborativo (Anexo C).

2 Estado da arte

Este capítulo apresenta o estado da arte da robótica. É apresentado o enquadramento teórico dos temas principais do trabalho. Inicia-se com uma exposição teórica e enquadramento histórico sobre a robótica, fazendo a distinção entre robôs industriais, colaborativos e móveis. E, por fim, são abordadas as normas e dispositivos de segurança aplicados à robótica colaborativa e móvel.

2.1 Resenha histórica

Desde tempos antigos, os humanos têm procurado simplificar, acelerar e melhorar a eficácia na realização de diversas tarefas, utilizando inovações como os robôs. A palavra "robô" tem as suas raízes na obra de Karel Capek, originando-se da palavra checa "*robota*", que se traduz como "trabalhos forçados" [9]. Mais tarde, este termo foi adotado para se referir a um dispositivo automático capaz de executar tarefas humanas. Geralmente, um robô é ligado a outros dispositivos mecânicos e/ou eletrónicos, devendo ser programados para que possam controlar esses dispositivos de forma a desempenhar as tarefas pretendidas.

Leonardo Da Vinci é reconhecido como um dos primeiros investigadores no campo da robótica, tendo sido descoberto um esquema de um robô com forma humana nos seus manuscritos. Isaac Asimov, nos anos 50, estabeleceu os principais princípios que um robô deve respeitar. Estes princípios incluem a obrigação de não causar dano físico a um ser humano (ou impedir que tal aconteça por falha na execução de uma tarefa), a obediência e a responsabilidade de proteger a integridade física humana [10].

Com o passar dos anos, foi essencial melhorar os métodos de fabrico dos diversos objetos que utilizamos diariamente, dando origem ao campo da robótica no início do século XX - um período marcado pela Primeira e Segunda Guerras Mundiais, que impulsionaram a criação de robôs industriais. O primeiro robô industrial foi criado por George Devol e Joseph Engelber para ser utilizado numa fábrica da *General Motors*, conforme ilustrado na Figura 2.1 [11].

Desde aí, os avanços na robótica têm sido visíveis, abrangendo desde melhorias na qualidade de construção e na seleção de materiais até inovações nos atuadores, controladores e até mesmo na sofisticação das garras e *grippers* dos robôs. Esta evolução tem contribuído para melhoria da consistência das operações e acelerando os proces-

[8]Fonte: `https://tinyurl.com/futjbvj6`

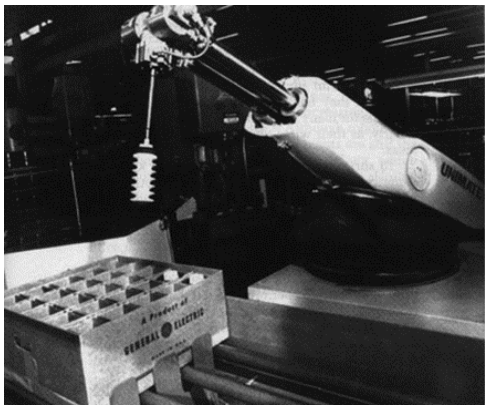


Figura 2.1: Robô *Unimate* - primeiro robô (industrial)[8].

sos de produção. No entanto, apesar dos progressos significativos, ainda há um grande potencial para desenvolver robôs mais leves e adaptáveis às diversas tarefas.

Em Portugal, o primeiro robô industrial, um ABB IRB6 (Figura 2.2) de cinco eixos para soldadura por arco, foi instalado nos anos 80 e permaneceu em funcionamento até ao ano de 2000. De acordo com a Federação Internacional de Robótica, os robôs mais comuns em Portugal são os robôs articulados, com 4 ou mais eixos, e os robôs cartesianos, com 3 ou mais eixos, sendo frequentemente utilizados, especialmente na indústria automóvel, como na *Stellantis*, para a montagem de veículos [12].

Figura 2.2: Robô ABB IRB6, primeiro robô (industrial) em Portugal[9].

[9]Fonte: `https://tinyurl.com/33zjm72e`

2.2 Implementação de robôs na indústria

A robótica tem uma vasta e variada gama de aplicações na indústria, abrangendo desde manipuladores especializados em soldadura a arco, pintura e paletização, entre outros [13]. Com o aumento da procura por manipuladores industriais, a produção destes equipamentos tem crescido de forma significativa. Atualmente, os principais fornecedores de robôs industriais são a *Fanuc*, a *Yaskawa* e a ABB (Figura 2.3).

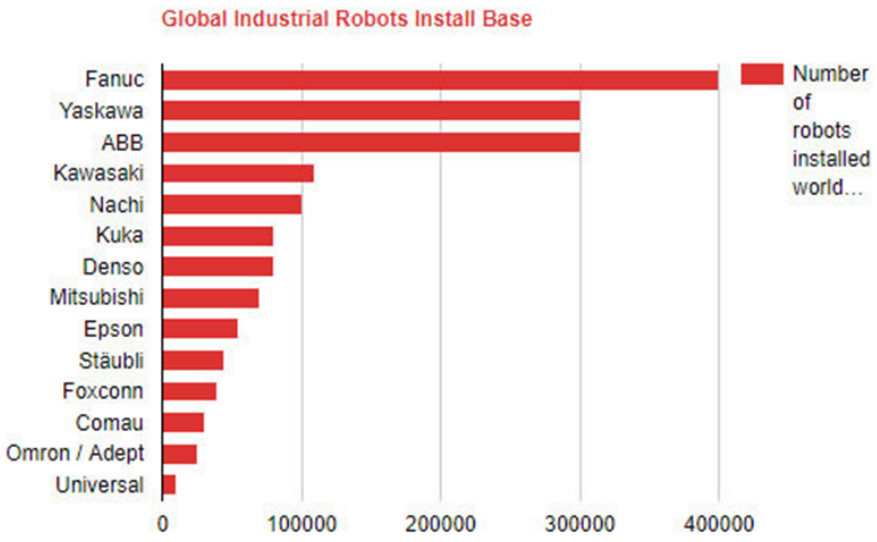


Figura 2.3: Gráfico com as catorze maiores fabricantes de robôs industriais[10].

2.3 Crescimento acentuado da venda de robôs destinados a outros fins

Segundo a Federação Internacional de Robótica, prevê-se um crescimento acentuado não apenas nas vendas de robôs industriais, mas também nos robôs de serviço profissionais. Ao contrário dos robôs industriais tradicionais, normalmente utilizados em linhas de montagem, os robôs de serviço profissionais são concebidos para desempenhar uma variedade de funções fora do ambiente de produção, sendo destinados à prestação de serviços diretos a humanos ou ao apoio em operações logísticas e técnicas. Em 2022, a venda destes robôs ultrapassou as 158 mil unidades, correspondendo a um crescimento aproximadamente de 48% face ao ano anterior [14].

Destacam-se os seguintes setores de atividade:

- **Setor do transporte e logística**: robôs para o transporte de mercadorias ou de

[10]Fonte: `https://tinyurl.com/42dcjnp6`

carga (exemplo representado na Figura 2.4), como é o caso da entrega de alimentos e bebidas em restaurantes;

- **Setor da hotelaria e turismo**: teve um disparo considerável na ordem dos 125% nos robôs para orientação móvel, oferta de informação e telepresença;
- **Setor da agropecuária**: robôs para ordenha e limpeza de celeiros. A falta de mão de obra e a procura por práticas mais sustentáveis está na base da procura pela robotização neste setor.

Assim, é possível notar que a robótica continua em constante evolução e a estender-se por outros setores, tornando-os cada vez mais robotizados e automatizados.

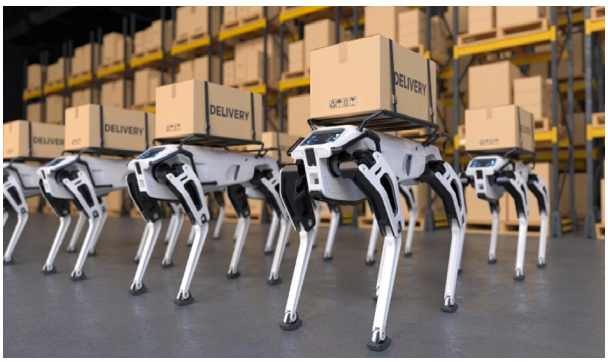


Figura 2.4: Robô destinado a entregas[11].

2.4 Revolução industrial e a Indústria 5.0

Na segunda metade do século XVIII, deu-se um marco significativo na História: a primeira Revolução Industrial, que teve início na Inglaterra. A invenção da máquina a vapor e a sua subsequente aplicação na produção de tecidos e fios, sobretudo na indústria têxtil, impulsionou a produção em série de componentes, conduzindo à industrialização do comércio. Esta primeira Revolução Industrial teve um impacto profundo na economia, na política e na sociedade, permitindo o aumento da produção de bens e serviços, promovendo o crescimento económico, criando oportunidades de emprego e melhorando, em termos gerais, o padrão de vida das populações [15].

Relativamente pouco tempo depois, entre o final do século XIX e o início do século XX, iniciou-se a segunda Revolução Industrial, que se manifestou principalmente nos países europeus e nos Estados Unidos. Esta revolução foi caracterizada pela introdução de novos paradigmas, como a eletricidade, a indústria química e a produção em massa. O surgimento de novas indústrias, como as do petróleo e dos automóveis, foi também uma característica importante deste período. A produção em massa, facilitada pela introdução de novas máquinas e processos de fabrico, permitiu a redução dos custos

[11] Fonte: `https://tinyurl.com/2bkhz3en`

de produção e o aumento da eficiência, o que resultou num incremento significativo das oportunidades de emprego [16].

A partir da década de 1970, iniciou-se uma nova fase da história industrial: a terceira Revolução Industrial, também conhecida como Revolução Digital. Esta foi impulsionada pelo desenvolvimento das tecnologias de informação e das telecomunicações, transformando profundamente a economia global e alterando a forma como as pessoas trabalhavam, se comunicavam e se relacionavam. A terceira Revolução Industrial foi marcada pelo surgimento de tecnologias como os computadores pessoais, a internet, os telemóveis e os e-mails. Estes avanços permitiram a criação de novos produtos e serviços, aumentando a eficiência e a produtividade das empresas, ao mesmo tempo que começaram a emergir discrepâncias nos salários e nas remunerações dos trabalhadores [17].

Pode ser observada na Figura 2.5 a cronologia das Revoluções Industriais ao longo dos anos desde a primeira até à evolução mais recente.

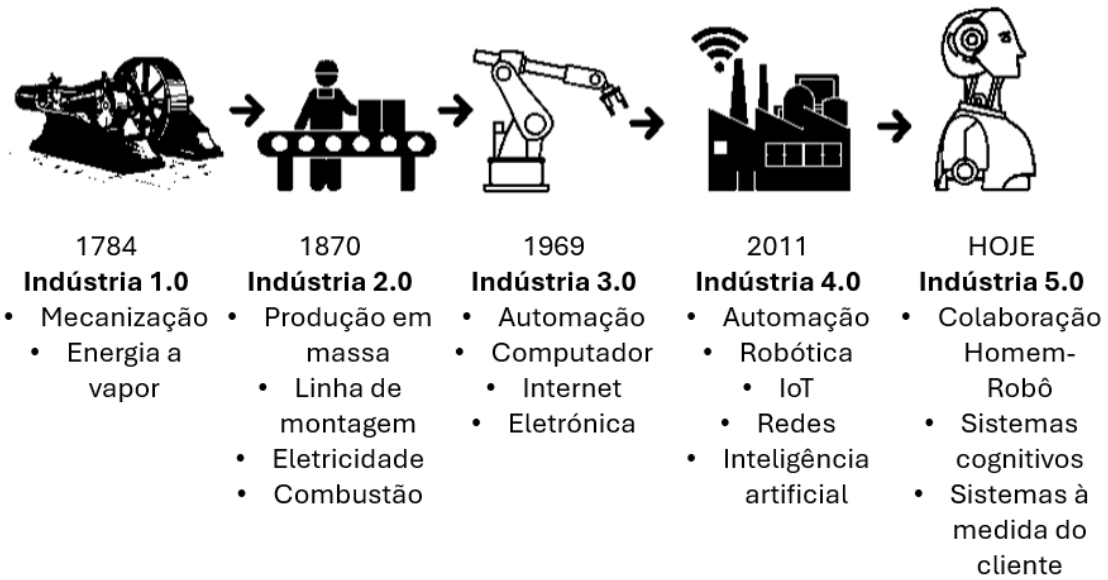


Figura 2.5: Cronologia das Revoluções Industriais (adaptado de [18]).

É relativamente simples estabelecer uma relação entre os acontecimentos mencionados anteriormente e compreender que esses momentos-chave partilham vários pontos em comum. Todos foram impulsionados pelo surgimento de novas tecnologias que transformaram os modos de produção, trabalho e comunicação, resultando em ganhos significativos de produtividade no setor industrial. É igualmente importante destacar a interdependência existente entre as várias revoluções industriais, uma vez que cada uma delas foi moldada pelos avanços tecnológicos precedentes, contribuindo para um progresso contínuo.

Adicionalmente, é inegável que a sociedade moderna se encontra cada vez mais dependente das tecnologias digitais, como o email e a Internet, entre outras. Estas inovações tornaram-se tão profundamente enraizadas nas nossas vidas quotidianas que é praticamente impensável conceber o mundo sem elas.

Mais recentemente surgiu a quarta Revolução Industrial, também conhecida como Indústria 4.0 [19]. Este movimento, iniciado na Alemanha, caracteriza-se pela combinação e interação de diversas tecnologias emergentes, como a inteligência artificial, a robótica, a Internet das Coisas (IoT), a impressão 3D, entre outras, tal como exemplificado na Figura 2.6. A quarta Revolução Industrial está a transformar profundamente a maneira como as empresas produzem, gerem e distribuem produtos. Melhorar a eficiência, reduzir os custos de produção e aumentar a produtividade são objetivos comuns a todas as revoluções industriais até ao momento.

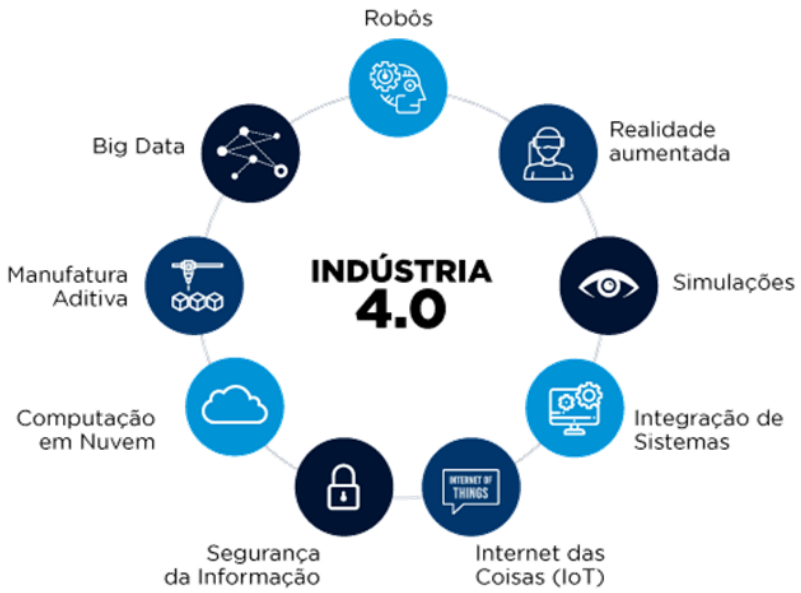


Figura 2.6: Representação da Indústria 4.0[12].

A Indústria 4.0 não só está a alterar os métodos de trabalho, mas também a exigir novas competências e habilidades por parte dos trabalhadores, acentuando a necessidade de formação e atualização contínua. Contudo, à medida que a Indústria 4.0 evolui, estamos também a presenciar o início de uma nova etapa: a quinta Revolução Industrial, ou Indústria 5.0.

A Indústria 5.0 traz uma nova perspetiva, focando-se na colaboração entre humanos e máquinas, onde a personalização em massa e a sustentabilidade desempenham um papel crucial. Ao invés de meramente automatizar processos, a Indústria 5.0 visa integrar a inteligência humana com a eficiência das máquinas, promovendo um equilíbrio

[12]Fonte: `https://site.ideia.cv/index.php/fr/blog-7/98-industria-4-0.html`

entre a tecnologia e o valor humano, conforme ilustrado na Figura 2.7. Além disto, há um crescente enfoque na responsabilidade ambiental e social, com o objetivo de criar soluções industriais mais sustentáveis e centradas nas necessidades individuais dos consumidores. Este novo paradigma procura não apenas aumentar a produtividade, mas também contribuir para o bem-estar da sociedade e do planeta.

Figura 2.7: Representação da Indústria 5.0[13].

2.5 Robótica móvel

Os AGV são veículos automatizados que são amplamente utilizados na indústria para realizar o transporte de materiais de forma independente num ambiente predefinido sem a necessidade de intervenção humana. Estes são equipados com diversos tipos de sensores que permitem a navegação pelas instalações fabris ao mesmo tempo que são capazes de evitar obstáculos. No geral, a escolha do tipo de AGV depende das necessidades específicas da aplicação, como o tipo e tamanho das cargas a serem transportadas, o *layout* da instalação e os requisitos operacionais [20]. Na Figura 2.8 é possível verificar a gama de AGV da marca ASTI (atual ABB).

Na indústria, os AGV são bastante importantes em aplicações que compreendem a movimentação, manipulação e carregamento de material. A utilização de AGV na indústria apresenta, entre outras, as seguintes vantagens:

- Redução da energia despendida;
- Deslocamento de carga não tripulado;

[13]Fonte: `https://blog-pt.checklistfacil.com/industria-5-0/`
[14]Fonte: `https://tinyurl.com/49rrww7f`

Figura 2.8: Gama de AGV da ASTI (atual ABB)[14].

- Fornecimento de material de acordo com o fluxo da produção, resultando num controlo mais eficiente;
- Segurança e redução dos danos provocados pelo transporte manual do material de produção.

No entanto identificam-se também algumas desvantagens, nomeadamente:

- Limitações quando utilizados em ambientes exteriores;
- Manutenção e suporte contínuo;
- São menos versáteis do que um operador, pois este é capaz de desempenhar diferentes tarefas e adaptar-se a eventuais alterações das necessidades de produção de uma forma mais inteligente e rápida;
- São mais indicados para executar tarefas repetitivas.

2.5.1 Tipos de AGV e tecnologias utilizadas

Alguns dos tipos mais comuns incluem:

- **AGV do tipo reboque** utilizados para transportar materiais ou mercadorias numa plataforma. Por exemplo, tipicamente são utilizados para mover cargas grandes e pesadas, como paletes de mercadorias ou máquinas pesadas;
- **AGV de carga unitária** desenvolvidos para transportar cargas individuais, como paletes e contentores. São utilizados em armazém e centros de distribuição;
- **AGV desenvolvidos para imitar a funcionalidade de uma empilhadora**, estes são equipados com mecanismo de elevação para pegar e colocar a carga. São tipicamente utilizados em operações de produção e armazenamento;
- **AGV especiais**, para aplicações específicas, como transporte de materiais perigo-

sos, transporte de cargas de formatos estranhos ou trabalho em condições ambientais extremas;

- **AGV autónomos (ou AMR, que são descritos mais à frente)** que não requerem assistência humana, podem mover e transportar mercadorias de forma autónoma. Estes robôs são equipados com sistema de navegação avançado que os permite movimentar livremente dentro da instalação.

A última geração de AGV (denominados AMR) incorpora tecnologias avançadas, como:

- Navegação autónoma usando sensores como LIDAR, radar e câmaras para navegar e evitar obstáculos;
- Munidos com algoritmos de Inteligência Artificial (IA) capazes de tomar decisões autónomas, ajustando a sua rota com base em dados em tempo real e adaptando-se dinamicamente às mudanças;
- Capacidade de comunicar com outros sistemas na instalação, como sistemas de gestão de armazéns, para otimizar as suas rotas e melhorar a eficiência;
- São flexíveis e adaptáveis, permitindo que sejam facilmente reconfigurados para lidar com diferentes tipos de materiais ou mercadorias;
- Apresentam recursos avançados de segurança, como deteção de obstáculos, prevenção de colisões e a presença de botões de paragem de emergência é abundante para garantir a segurança de pessoas e bens.

Ainda assim, os robôs móveis mais utilizados na indústria continuam a ser os modelos convencionais, utilizados sobretudo para o transporte de materiais entre as várias fases do processo produtivo. Estes robôs são frequentemente personalizados para responder às necessidades específicas de cada indústria e aplicação.

2.5.2 Sistemas de segurança

De modo a garantir a integridade do AGV, bem como a relação com o meio envolvente, como pessoas, outras máquinas e estruturas fixas, estes são construídos segundo a norma europeia para segurança dos robôs móveis e dispõem de dispositivos de segurança como:

- **Deteção de colisões**: possuem sensores de obstáculos que detetam objetos no seu campo de alcance e calculam a distância até estes. O AGV irá reduzir a velocidade ou até parar para prevenir colisões;
- **Sinal áudio / visual**: são equipados com um conjunto de dispositivos de sinalização, tais como sinalização luminosa intermitente ou alertas sonoros que indicam o estado do AGV ou avisam os trabalhadores da sua presença;
- **Controlo manual**: dispõem da possibilidade de controlo manual e de botões de

emergência para a paragem brusca do AGV [21].

2.5.3 Tipos de movimentação

Para diferentes tipos de aplicação há diferentes tipos de movimentação possíveis, tais como:

- **Unidirecional**: uma roda dianteira de tração/direção que roda sobre si mesma e um eixo traseiro fixo;
- **Bidirecional**: duas rodas de tração/direção e duas rodas livres que permitem ao AGV deslocar-se em ambos os sentidos;
- **Omnidirecional**: permite a movimentação em todas as direções, graças a um conjunto de rodas que giram de forma independente umas das outras [21].

2.5.4 Sistemas de localização

Cada AGV pode utilizar diferentes sistemas de localização, os quais, por sua vez, apresentam características distintas. Estes sistemas podem incluir tecnologias como *tags Radio-Frequency Identification* (RFID), *Quick Response* (QR) *Code*, laser, *Global Positioning System* (GPS) e *Simultaneous Localization and Mapping* (SLAM), descritas resumidamente como se segue:

- **Localização por *tags* RFID**: é uma tecnologia que utiliza *tags* RFID para determinar a posição do AGV; estas *tags* são colocadas em pontos estratégicos no chão ou nas paredes e emitem sinais de radiofrequência que são captados pelo AGV, permitindo que este determine a sua posição exata;
- **Localização por QR Code**: este método utiliza QR Code colocados em pontos estratégicos no chão ou nas paredes. O AGV utiliza uma câmara para detetar a presença destes códigos e assim determinar a sua posição no ambiente;
- **Localização por laser**: é caracterizada pelo uso de um *laser scanner* que, ao longo do trajeto, efetua um varrimento na procura de alvos refletores posicionados estrategicamente. Obtidas e processadas as distâncias e ângulos relativos aos alvos, o AGV consegue triangular a sua posição. Contudo, esta tecnologia não é caracterizada pela sua precisão, necessitando, tipicamente, de um sistema auxiliar para dupla validação;
- **Localização por GPS**: tem-se observado o recurso a estes sistemas de navegação, como alternativa viável e flexível em contexto de rotas dinâmicas. No entanto, no interior de edifícios, a orientação por GPS tende a apresentar constrangimentos de funcionamento, devido à limitação ou perda de sinal;
- **Localização por SLAM**: é uma das características que distingue um AGV tradici-

onal de um AMR. O SLAM é uma técnica que permite ao AMR criar um mapa do meio envolvente em simultâneo com a sua própria localização. Ou seja, enquanto o veículo se move pelo ambiente, recorre a sensores para fazer uma estimativa da sua própria posição e utiliza essa informação para construir um mapa do meio envolvente. Este processo é iterativo, e à medida que o veículo se move e recolhe mais informação, melhora a sua estimativa da posição e do mapa do ambiente. O SLAM é geralmente utilizado em ambientes desconhecidos, onde o veículo não tem um mapa pré-existente para se orientar.

2.5.5 Tipos de navegação

Existem dois principais de navegação nos robôs móveis, são eles:

- **Navegação natural**: este tipo de navegação é característico dos AMR, estes utilizam sensores para se orientar no ambiente, sem a necessidade de marcações no chão ou nas paredes. Os sensores podem ser câmaras, sensores de proximidade e sensores de posição, que permitem ao robô desviar-se de obstáculos e ajustar a própria rota em tempo real;
- **Navegação livre**: neste tipo de navegação, o AGV segue um mapa pré-existente do ambiente, que pode incluir marcações no chão ou nas paredes para ajudar na sua orientação. O AGV segue a rota definida no mapa, evitando a colisão com obstáculos através de sensores de proximidade. Este tipo de navegação é geralmente utilizado em ambientes com rotas bem definidas e estáveis, como em armazéns ou fábricas.

2.5.6 Programação dos AGV

Um sistema robótico móvel pode ter diferentes níveis de autonomia e inteligência, que dependem do tipo de sistema de controlo integrado no veículo. No caso de um AGV industrial, que normalmente segue uma marcação pré-estabelecida no chão, a sua capacidade para lidar com situações imprevistas é limitada. Por exemplo, se a marcação estiver danificada ou houver um obstáculo no caminho, o AGV irá parar até que o problema seja resolvido. Para superar estas limitações, o estudo relativo à robótica móvel autónoma procura desenvolver sistemas de controlo com comportamentos inteligentes que tornem os AGV mais confiáveis e robustos.

Ao programar um AGV, é importante considerar o tipo de robô e a sua capacidade de movimentação, bem como a linguagem de programação adequada a cada fabricante. É possível efetuar a programação usando uma interface HMI, ajustando assim o seu funcionamento de acordo com a aplicação pretendida [21].

A título de exemplo, para a programação dos robôs móveis da marca ABB pode ser utilizado a plataforma SIGAT que permite a programação de AGV da própria marca

através da definição de parâmetros associados às tarefas que o utilizador quer que o robô desempenhe. A Figura 2.9 representa um exemplo da definição de uma linha da tabela de trabalho.

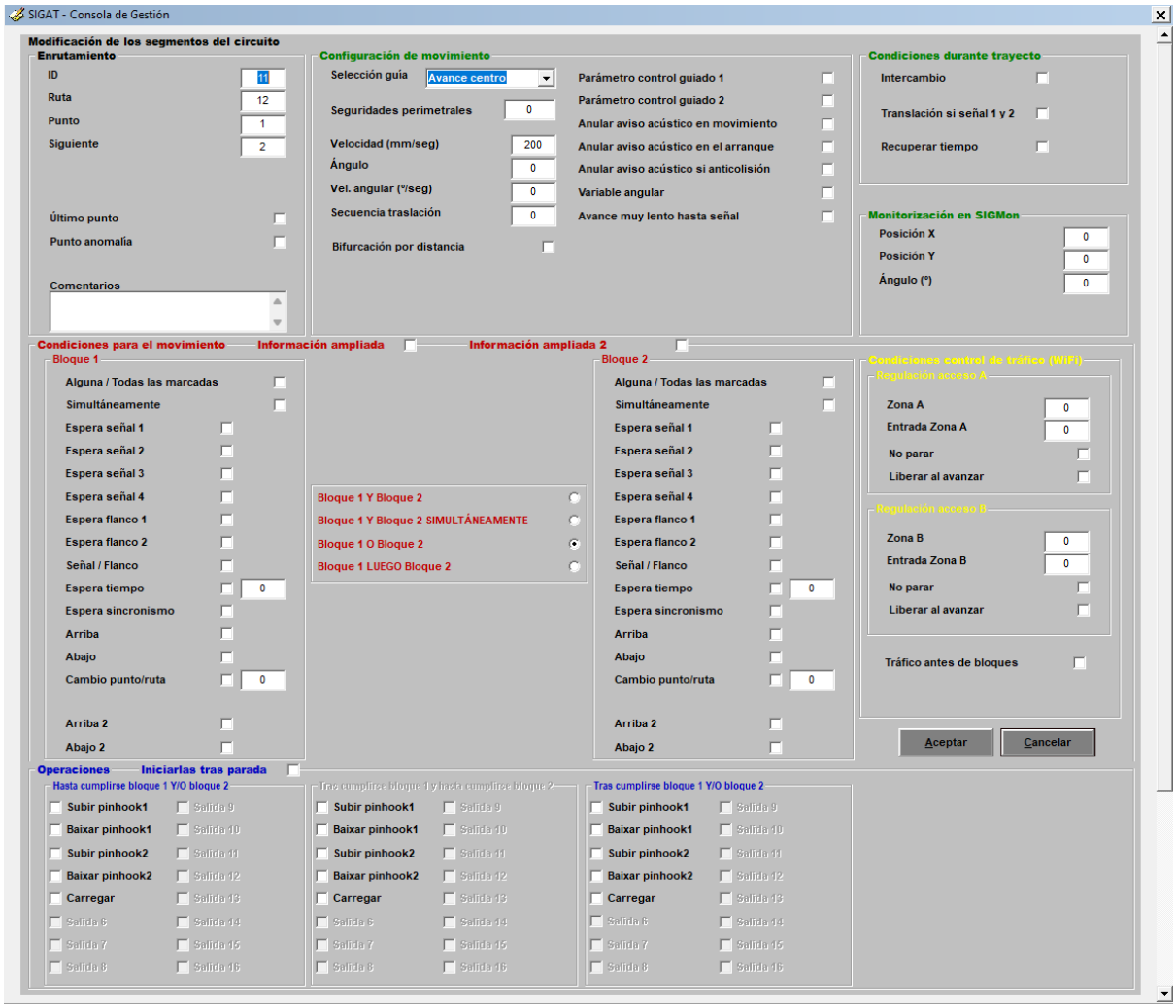


Figura 2.9: Definição de parâmetros do AGV na plataforma SIGAT.

2.5.7 Rotas dos AGV

Existem vários tipos de rotas que um AGV pode percorrer. Estas rotas são definidas consoante as operações a realizar, as condições das instalações e a localização. O tipo de rota irá influenciar a escolha do AGV mais adequado para realizar as tarefas.

As configurações a seguir apresentadas são as mais comuns. Contudo, sendo a flexibilidade uma das principais características do AGV, é possível adaptar as configurações e criar configurações que se adequem às situações pretendidas:

- **Linha única / Ida e volta**: é a configuração mais simples, pois o movimento é realizado apenas em linha reta permitindo que o AGV ande para a frente e para trás. Nesta configuração só pode haver um único AGV em cada linha;

- **Circuito duplo**: esta configuração envolve a colocação de duas linhas paralelas, permitindo que vários AGV circulem em ambos os sentidos;
- **Múltiplos loops**: constituído por vários loops, permitindo assim criar bifurcações e responder a sistemas com várias estações, ou seja, é uma configuração mais complexa, logo, exige controlo de tráfego mais sofisticado, mas oferece maior flexibilidade e otimização de rotas.

Se uma rota contiver apenas um único AGV, a gestão de tráfego é mais simples. Quando existem diversos AGV a circular no sistema, os problemas de interferência ou colisões entre veículos têm de ser evitados. Isto, normalmente, é implementado criando zonas de exclusividade para evitar colisões e ramais em locais apropriados do *layout* para resolver situações de conflito.

2.5.8 Tipos de carregamento

Existem diversos tipos de tecnologia de baterias e de sistemas de carregamento.

Relativamente aos sistemas de carregamento, podemos encontrar os seguintes tipos:

- **Carregamento manual ou troca manual**: envolve a remoção manual da bateria do AGV para carregá-la separadamente, ou a troca da bateria descarregada por outra já carregada;
- **Carregamento automático**: o AGV retorna automaticamente à estação de carregamento quando a bateria está fraca, e o processo de carregamento é iniciado sem a necessidade de intervenção manual;
- **Carregamento sem fios**: utiliza uma tecnologia de carga indutiva, em que o AGV é carregado simplesmente estacionando-o sobre uma superfície equipada com um carregador sem fios [21].

2.5.9 AMR versus AGV

Os AGV e os AMR revolucionaram a indústria, oferecendo maior eficiência, redução de erros e melhorias na segurança. Os AGV são robôs que seguem trajetos fixos (conforme já mencionado). Por outro lado, os AMR são mais avançados e autónomos, utilizando sensores e *software* para navegação. É importante perceber as suas principais diferenças para que se adeque a aquisição dos robôs consoante a necessidade específica para cada aplicação e empresa [22]. Pode ser observada na Figura 2.10 uma representação gráfica da capacidade de alteração de trajetória de um AGV comparativamente a um AMR, ou seja, o AGV como tem que obrigatoriamente ler sempre a banda magnética colada ao chão não se consegue desviar de um obstáculo, por outro lado o AMR como não precisa de um guia físico para se deslocar pode contornar esse mesmo obstáculo.

[15]Fonte: `https://idealworks.com/amr-vs-agv-2/`

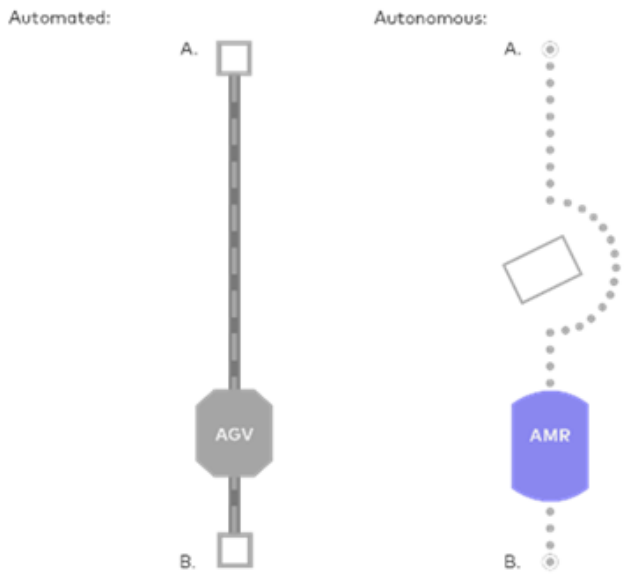


Figura 2.10: Possível trajeto de um AGV versus AMR[15].

Os AMR são dotados de vários sensores para que se orientem da melhor forma possível nos ambientes em que transitam:

- **Sensores de visão**: as câmaras são os "olhos" dos AMR, estas capturam imagens de alta resolução, permitindo que os robôs tenham a perceção do mundo à sua volta, estas imagens são processadas por algoritmos de visão, que identificam objetos, obstáculos e marcas no chão que podem servir como guias de navegação do robô;
- **LIDAR**: é uma tecnologia que utiliza feixes de laser para criar um mapa detalhado do ambiente envolvente, os AMR utilizam este sistema para medir distâncias com elevada precisão, detetar obstáculos e mapear o terreno.

Em suma, os AGV, utilizados desde a década de 1950, são robôs industriais que transportam materiais autonomamente dentro de fábricas e armazéns, reduzindo a necessidade de intervenção humana. Utilizam caminhos predeterminados definidos por fios, fitas magnéticas ou lasers no chão. Os sensores detetam estes guias, permitindo que os AGV naveguem com alta previsibilidade. Apresentam como vantagens:

- **Eficiência operacional**: automatizam tarefas repetitivas, aumentando a produtividade e diminuindo o esforço humano;
- **Melhoria na segurança**: operam de forma segura ao redor de pessoas e infraestruturas, reduzindo o risco de acidentes;
- **Ambientes perigosos**: funcionam em condições adversas para humanos, como temperaturas elevadas e/ou exposição a substâncias tóxicas;

- **Operação contínua**: podem operar 24/7, ideal para setores com operações ininterruptas.

Por outro lado, apresentam as seguintes limitações:

- **Dependência de trajetos fixos**: falta de flexibilidade, exigindo tempo e custo para modificar caminhos;
- **Tarefas limitadas**: projetados principalmente para transporte de materiais, não são adequados para tarefas complexas;
- **Sensibilidade a mudanças ambientais**: obstáculos imprevistos podem causar interrupções;
- **Investimento inicial elevado**: alto custo inicial, especialmente em instalações complexas.

Por outro lado, os AMR sendo robôs móveis mais avançados que navegam autonomamente, sem necessidade de guias físicos. Utilizam uma combinação de sensores, câmaras e *software* sofisticado para mapear o ambiente, planear rotas e evitar obstáculos, adaptando-se a diversas tarefas. Apresentam assim, as seguintes vantagens:

- **Flexibilidade na navegação**: adaptam-se a mudanças no ambiente sem modificações na infraestrutura;
- **Versatilidade nas tarefas**: realizam tarefas complexas, como a recolha e classificação de itens;
- **Eficiência aumentada**: automatizam tarefas que requerem trabalho humano, reduzindo tempo e esforço;
- **Operação em ambientes perigosos**: funcionam em condições adversas, promovendo a segurança;
- **Facilidade de integração**: integração simplificada com operações existentes, sem necessidade de guias físicos.

E as seguintes limitações:

- **Custo inicial elevado**: maior custo inicial devido às tecnologias avançadas;
- **Dependência de *software***: suscetíveis a falhas de *software*, exigindo alta confiabilidade;
- **Sensibilidade ambiental**: alterações no ambiente podem afetar o desempenho;
- **Integração complexa**: pode exigir adaptações significativas na infraestrutura e formação do pessoal.

Tendo em conta as vantagens e desvantagens destes dois diferentes tipos de robôs móveis é possível fazer uma comparação direta entre as características dos AGV e dos AMR:

- **Funcionalidade**:
  - **AGV**: transporta materiais em trajetos fixos, adequado para movimentações repetitivas em ambientes estáveis;
  - **AMR**: navega autonomamente e realiza tarefas complexas, ideal para ambientes dinâmicos.
- **Flexibilidade**:
  - **AGV**: menos adaptável a mudanças, requerendo modificações nos guias;
  - **AMR**: adapta-se facilmente a novas tarefas e mudanças no ambiente.
- **Custo**:
  - **AGV**: menor investimento inicial, mas custos adicionais para modificar trajetos;
  - **AMR**: maior investimento inicial, mas potencial para economia a longo prazo devido à adaptabilidade.
- **Implementação**:
  - **AGV**: requer instalação de guias físicos, processo demorado;
  - **AMR**: implementação simplificada, sem necessidade de guias físicos.

Assim sendo, para tomar uma decisão que decida entre a aquisição de uma frota de AGV em detrimento de AMR ou vice-versa, é necessária uma ponderada avaliação tendo em conta alguns aspetos, tais como:

- **Ambiente operacional**: se o ambiente é estável e as tarefas são repetitivas, os AGV podem ser suficientes. Para ambientes dinâmicos, os AMR são mais adequados;
- **Funcionalidade necessária**: avaliar as tarefas específicas a serem automatizadas. Tarefas complexas podem requerer AMR;
- **Processo de implementação**: considerar a facilidade de implementação e integração de cada tecnologia.
- **Custos**: comparar os custos iniciais e a longo prazo, considerando a escalabilidade;
- **Planeamento para crescimento**: os AMR oferecem maior flexibilidade para crescer e adaptar-se a novas necessidades.

Com uma análise cuidadosa destes fatores, as empresas podem fazer uma escolha informada entre AGV e AMR, garantindo que a tecnologia escolhida atenda às suas necessidades operacionais.

2.6 Robótica colaborativa

Os robôs colaborativos são projetados para trabalhar em colaboração com humanos, compartilhando o mesmo espaço de trabalho. Ao contrário dos robôs industriais tradicionais, que são concebidos para operar de forma autónoma em ambientes isolados, os cobots podem colaborar diretamente com os trabalhadores, realizando tarefas em conjunto de maneira segura e eficiente, sem necessidade de isolar o dispositivo mecânico. Para garantir uma interação eficaz e segura entre os cobots e os seres humanos, é essencial que o robô esteja equipado com sensores e sistemas de segurança para prevenir lesões.

De acordo com *Muller et al.* [23] existem três tipos de interação entre humanos e cobots:

- **Colaboração**: ocorre quando humanos e cobots partilham o mesmo espaço de trabalho e interagem diretamente. Neste caso, ambos realizam tarefas simultâneas e complementares com o objetivo de completar o mesmo produto ou peça;
- **Cooperação**: refere-se à interação indireta. Embora humanos e cobots trabalhem no mesmo espaço e executem tarefas diferentes, estas ocorrem em momentos distintos, sem que haja uma interação simultânea;
- **Coexistência**: os humanos e os cobots operam em áreas adjacentes sem uma barreira de proteção entre eles. No entanto, não partilham o mesmo espaço de trabalho, sendo as suas tarefas independentes e destinadas a processos diferentes.

Este modelo de interação reflete o nível de proximidade e coordenação entre humanos e cobots em diferentes ambientes industriais.

Os robôs colaborativos, ainda que sejam muito distintos dos industriais, são muitas vezes comparados com estes devido à sua estrutura física. Para uma melhor compreensão das principais funções e diferenças entres estes dois tipos de robôs apresenta-se na Tabela 2.1 uma análise comparativa com as principais diferenças entre os robôs industriais tradicionais e os robôs colaborativos.

Os robôs colaborativos podem, na maioria das vezes, operar sem a necessidade de barreiras de proteção, uma característica que, tradicionalmente, limita a flexibilidade. Além disto, os robôs colaborativos são facilmente reprogramáveis, mesmo por trabalhadores sem um conhecimento aprofundado das linguagens de programação utilizadas em robótica [25, 26]. Por estas razões, e tendo em conta a crescente procura por produtos personalizados com prazos de entrega mais curtos, a robótica colaborativa está a ser alvo de um grande crescimento. Portanto, espera-se uma rápida expansão da utilização da robótica colaborativa em novas aplicações.

Em 2008, a empresa *Universal Robotics* desenvolveu o primeiro robô colaborativo, denominado UR5 (Figura 2.11), impulsionando o interesse industrial na aplicação de cobots nas linhas de produção de diversas indústrias [27].

Tabela 2.1: Análise comparativa entre os robôs industriais tradicionais e os cobots [24].

Características	Robô Industrial Tradicional	Cobots
Tipo de instalação	Instalação fixa	Instalação flexível, fácil de realocar após instalada
Tipo de tarefas	Tarefas repetitivas	Mudança de tarefas frequente e flexível
Tipo de programação	Programação necessária (*online* ou *offline*)	Possibilidade de comandos simples que substituem a programação formal
Tipos de tarefas	Projetados para tarefas específicas	Podem ser facilmente realocados em novas aplicações
Tipo de interação com humanos	Espaço de trabalho separado por barreiras de segurança	Espaço de trabalho partilhado com os humanos
Investimento	Alto investimento inicial, ideal para grandes indústrias	Menor investimento, ideal para pequenas e médias empresas
Dimensão e velocidade	Tamanho variável e operação rápida	Tamanho pequeno, operação mais lenta por partilharem espaço com humanos
Avaliação de risco	Avaliação de risco não necessária	Avaliação de risco obrigatória por estarem em contacto com humanos

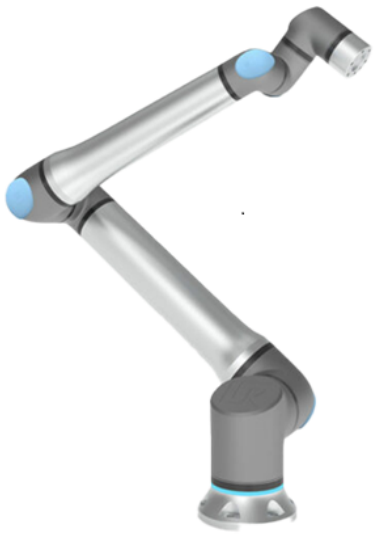


Figura 2.11: Primeiro robô colaborativo da Universal Robotics, UR5[16].

Este avanço levou empresas como a KUKA e a ABB a investirem no desenvolvimento de cobots. De facto, dado que a robótica colaborativa permite a operação conjunta entre humanos e robôs para a realização de tarefas em ambiente cooperativo, exigindo a proximidade entre humanos e máquinas, há um conjunto rigoroso de medidas de segurança que os cobots devem cumprir e que são abordadas mais à frente.

A robótica colaborativa permite, assim, um aumento na qualidade dos produtos, na eficiência da produção e na melhoria das condições de trabalho dos operadores [28, 29]. Por isso, empresas de diversos setores procuram integrar cobots nas suas instalações. Neste contexto, as Pequenas e Médias Empresas (PME) destacam-se pela dificuldade em automatizar os seus processos produtivos utilizando robôs industriais tradicionais, devido à falta de espaço e ao elevado investimento necessário [27], bem como empresas em setores caracterizados pela produção em massa, como o setor automóvel [30].

2.6.1 Sistemas de visão

Nos últimos anos, os sistemas de visão para reconhecimento de objetos na indústria têm registado uma procura exponencial por sensores óticos, tanto para imagens bi-dimensionais (2D) como tridimensionais (3D), impulsionando o desenvolvimento de dispositivos cada vez mais sofisticados e avançados.

No caso dos sensores 3D, estes podem operar de duas formas: de modo ativo, através da projeção de luz ou padrões, ou de modo passivo, captando a energia transmitida ou refletida por um objeto. Na área da robótica, diversas abordagens integram sensores 3D com a aquisição de informação tridimensional, permitindo dotar as máquinas de perceção espacial do ambiente envolvente. Esta capacidade possibilita a interação com o cenário, incluindo a manipulação de objetos e a comunicação com outras máquinas.

Os sistemas de visão oferecem algumas tecnologias que são bastante utilizadas nas aplicações dos mesmos:

- **Fotogrametria** [31]: é uma tecnologia que permite ultrapassar alguns desafios práticos relacionados com imprecisão, velocidade de medição, automação, integração de processos, desempenho de custo e integração sensorial. Estes sistemas de fotogrametria são extremamente precisos e é possível encontrá-los na indústria automóvel numa panóplia de aplicações, tais como, medições de deformação das carroçarias, controlo de peças, ajuste de ferramentas, entre outras aplicações. Para além disto, os sistemas de fotogrametria podem ser integrados numa rede, fornecendo informações 3D que possibilitam a ligação e controlo de processos. É importante ainda evidenciar que estes sistemas de fotogrametria podem ser acoplados nos robôs industriais, permitindo orientar a execução de tarefas de perfuração e montagem [31, 32];

[16] Fonte: `https://www.wiredworkers.io/cobot/brands/universal-robots/ur5/`

- **Tecnologia Time-of-Flight (ToF)**: esta tecnologia funciona emitindo um feixe de luz para uma superfície e medindo o tempo que a luz demora a ser refletida de volta, a fim de determinar a distância até ao objeto. Os sensores ToF são muito compactos e leves, possibilitando a captura de imagens 3D em tempo real com alta resolução, sendo utilizados nos braços móveis de sistemas robóticos;
- **Tecnologia de luz estruturada**: utiliza padrões de luz para capturar imagens 3D de um objeto ou superfície, sendo pouco influenciada pela variabilidade da iluminação, o que facilita a deteção precisa dos objetos. No entanto, os sensores e o seu tamanho considerável tornam difícil a sua fixação na extremidade de um sistema robótico;
- **Técnica de triangulação a laser**: permite a identificação de peças, podendo ser utilizado em diferentes etapas no processo de aparafusamento e parafusos ou porcas.

Por outro lado, as câmaras 2D permitem a deteção e identificação de objetos; no entanto, são bastante sensíveis às condições de luminosidade e à presença de poeiras ou sujidade no ambiente industrial [33]. Estas limitações podem conduzir a erros na identificação de componentes ou no posicionamento incorreto de elementos, como os pontos de aparafusamento. Nesse sentido, a integração de sistemas de visão 3D em robôs colaborativos revela-se uma solução mais viável, superando as limitações associadas às câmaras 2D.

Os sistemas de visão são essenciais em diversas aplicações robóticas. Estes são cruciais tanto para operações industriais automatizadas, como controlo de qualidade, quanto para tarefas mais complexas, como a seleção de objetos e as operações de montagem [34].

No entanto, em contextos industriais, surgem diversos desafios adicionais, como a presença de superfícies sem textura e variações nos ângulos e dimensões dos objetos. Estas variações fotométricas incluem alterações no brilho e contraste, deformações de perspetiva causadas pela mudança da posição do observador, objetos em movimento, entre outros fatores que representam desafios técnicos de resolução não trivial [31]. Para além disso, os sistemas de visão 3D possibilitam a identificação e quantificação da distância entre os operadores e a máquina, ajustam dinamicamente a trajetória do cobot à medida que este se aproxima de obstáculos e permitem ainda a interpretação de gestos humanos [35].

As soluções de robótica colaborativa visam o desenvolvimento de sistemas de captura tridimensional ou de características tridimensionais de objetos, utilizando técnicas de aquisição de imagens tridimensionais. Estes sistemas permitem a integração de diferentes tecnologias para complementar, por um lado, o conhecimento do ambiente envolvente do sistema robótico e, por outro, orientar o sistema robótico nas suas tarefas, como a identificação de objetos e as funções de aperto de componentes automóveis.

Além disto, o desenvolvimento de um sistema de visão artificial rápido e eficiente, capaz de controlar e garantir o rápido tempo de resposta dos cobots, representa um grande desafio, dado que este setor industrial é caracterizado por um tempo de cadência reduzido para maximizar a produtividade. A incorporação destas inovações na linha de produção confere uma vantagem competitiva única sobre os concorrentes. Por exemplo, o estudo de *Li et al.* [36] resultou no desenvolvimento de um cobot capaz de desaparafusar parafusos, e [37] desenvolveu uma solução para remover parafusos de computadores portáteis, com recurso à visão artificial para localizar os parafusos.

2.7 Segurança na robótica

Nesta secção, será abordado um dos temas mais importantes da robótica: a segurança, essencial numa indústria onde o contacto com pessoas e objetos inesperados é constante. São analisados os sistemas de segurança ativos e passivos tanto na robótica industrial tradicional, como na robótica colaborativa e móvel.

Além disso, são referidas as normas europeias que regulam as diferentes áreas da robótica mencionadas.

2.7.1 A importância dos sistemas de segurança na robótica

A robótica tem evoluído significativamente, especialmente na indústria, onde abrange tanto sistemas robóticos fixos como móveis [38, 39]. Dentro da robótica fixa, podemos distinguir entre a robótica industrial e a colaborativa, enquanto na robótica móvel destacam-se os AGV e os AMR [40, 41].

Os robôs industriais são amplamente utilizados como manipuladores para realizar tarefas específicas de forma autónoma, geralmente dentro de células robotizadas. São valorizados em várias indústrias pela sua capacidade de executar tarefas repetitivas com rapidez e eficiência, oferecendo um elevado retorno para as empresas que os adquirem [42].

Por outro lado, os robôs colaborativos diferenciam-se dos robôs industriais convencionais. Projetados para trabalhar em conjunto com os operadores, estes robôs podem partilhar o mesmo espaço de trabalho, ao contrário dos robôs industriais que, por razões de segurança, estão confinados a áreas específicas, como células robotizadas. Esta colaboração permite que os robôs e trabalhadores realizem tarefas em conjunto, refletindo a evolução dos sistemas de segurança, que garantem tanto a proteção dos operadores quanto do equipamento.

Um exemplo de aplicação de robótica colaborativa é a tarefa de apertar uma peça que separa a área dos passageiros da área de carga numa viatura comercial. Nesta situação, o uso de robôs colaborativos automatiza um processo que antes era exclusivamente

manual, melhorando o controlo de qualidade da operação.

Na robótica móvel, os robôs são frequentemente utilizados para transportar materiais de forma eficiente. Um exemplo notável são os AGV, programados para seguir trajetórias predefinidas, movendo materiais de forma autónoma e segura entre diferentes pontos.

Na indústria automóvel, por exemplo, esses sistemas desempenham um papel crucial no transporte de componentes durante o processo de montagem. A utilização de AGV e AMR não só aumenta a eficiência e reduz o risco de erros, como também otimiza a produção.

2.7.2 Medidas de segurança nos sistemas robóticos

As medidas de segurança dos sistemas robóticos são detalhadas nas normas apresentadas no final desta secção.

Os robôs industriais são projetados para realizar tarefas específicas em ambientes controlados e isolados.

Devido à sua elevada velocidade de operação e à aplicação de forças consideráveis, estes robôs necessitam de barreiras físicas de segurança, como cercas, para manter os operadores afastados das suas áreas de trabalho. Esta proteção é ilustrada na Figura 2.12.

Figura 2.12: Segurança nos robôs industriais[17].

Os robôs móveis [43, 44] estão equipados com vários sistemas de segurança que garantem uma operação segura. Graças aos seus sensores de perceção, conseguem detetar obstáculos no caminho, ajustando a velocidade ou parando, se necessário, para assegurar um funcionamento suave e sem incidentes. Estes veículos seguem rigorosamente as normas de robótica móvel, cumprindo todos os regulamentos de segurança da indústria. Os robôs móveis foram projetados para atender a padrões de segurança, sendo definidas zonas de proteção, conforme ilustrado na Figura 2.13.

[17]Fonte: `https://www.vecsa.com.br/cerca-de-protecao-de-equipamentos.php`
[18]Fonte: `https://www.kivnon.com/pt/agv-provide-more-safety-requirements/`

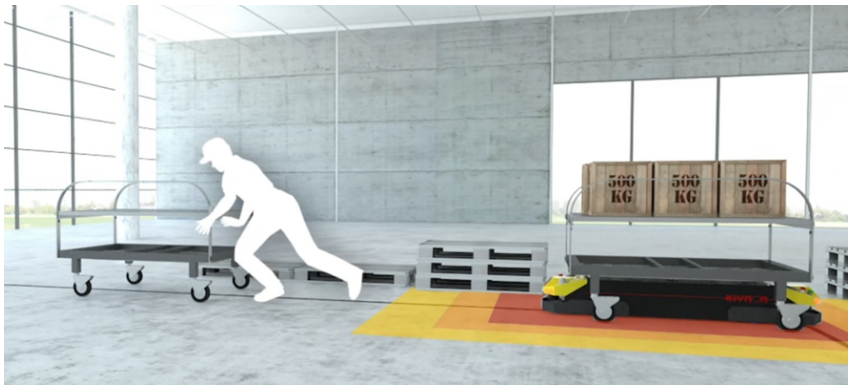


Figura 2.13: Zonas de segurança associadas aos robôs móveis[18].

Os cobots [45, 46] foram concebidos para interagir diretamente com humanos num ambiente de trabalho partilhado. Ao contrário dos robôs tradicionais, os cobots destacam-se pelos seus sistemas avançados de deteção e pela limitação de velocidade, garantindo uma colaboração segura. Estão equipados com sensores sofisticados, como câmaras, sensores de força e de proximidade, que lhes permitem detetar a presença de pessoas e reagir em tempo real a mudanças no ambiente. Estes robôs operam com forças e velocidades controladas para minimizar o risco de lesões em caso de contacto com humanos, sendo capazes de interromper as suas operações imediatamente em resposta a interações inesperadas. A Figura 2.14 ilustra as zonas de segurança de um cobot, distinguindo a zona de segurança, a zona de deteção e a zona de aviso.

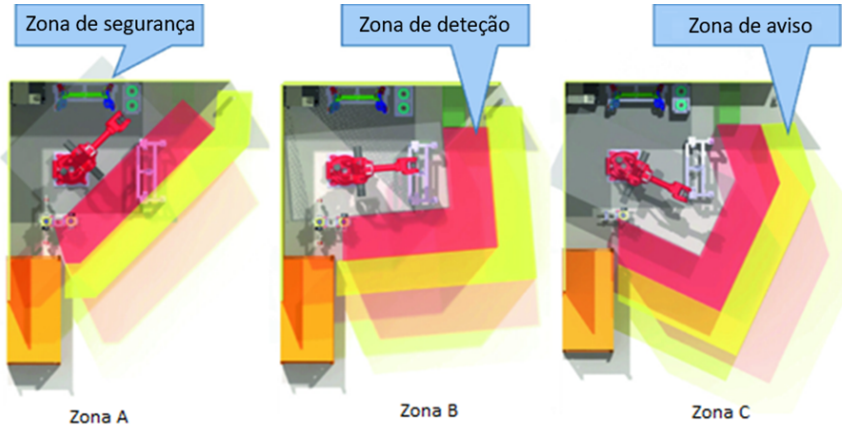


Figura 2.14: Zonas de segurança associadas aos robôs colaborativos[19].

No entanto, as suas características, objetivos e níveis de interação com os trabalhadores exigem abordagens diferentes no que à segurança diz respeito.

A comparação entre os sistemas de segurança de robôs colaborativos e robôs industriais tradicionais destaca a importância de adaptar as medidas de segurança às características e objetivos específicos de cada tipo de robô. Enquanto os cobots enfatizam a

[19]Fonte: `https://link.springer.com/chapter/10.1007/978-3-030-51591-1_14`

interação segura com operadores por meio de sensores avançados e limitações de força e velocidade, os robôs industriais tradicionais dependem mais de barreiras físicas para garantir a segurança [47].

2.7.3 Segurança ativa em cobots

Vamos abordar 5 soluções tipicamente adotadas no que diz respeito à segurança ativa de cobots, nomeadamente o *Safety Eye*, o tapete de segurança, os *Safety skin/pads*, o *laser scanner* e o *INXPECT*:

- ***Safety Eye***: este sistema é baseado em câmaras, oferece monitorização e controlo tridimensional ininterrupto de zonas de perigo, garantindo a deteção imediata de qualquer presença humana nas proximidades do robô (Figura 2.15);
- **Tapete de segurança**: este é utilizado para fornecer informações sobre a posição do operador dentro do local de trabalho do robô e geralmente são posicionados na zona de alcance do mesmo (Figura 2.16);
- ***Safety skin/pads***: são dispositivos de segurança importantes principalmente para operação de *hand-guiding*, estabelecendo uma colaboração homem-robô segura ao detetar qualquer contacto humano, interrompendo imediatamente o movimento do robô (Figura 2.17);
- ***Laser Scanner***: trata-se de um sensor de segurança 2D baseado na tecnologia laser que permite monitorizar a presença de objetos/operador humano na sua área de alcance, assegurando uma zona de segurança ao seu redor (Figura 2.18);
- ***INXPECT***: são sensores de segurança que têm por base de funcionamento a emissão de uma onda radio e respetiva reflexão no ambiente, garantindo uma deteção precisa de objetos ou pessoas nas proximidades do robô (Figura 2.19).

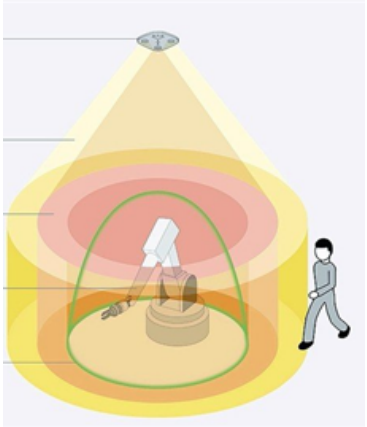


Figura 2.15: *Safety Eye*[20].

[20]Fonte: `https://tinyurl.com/4p29uaat`

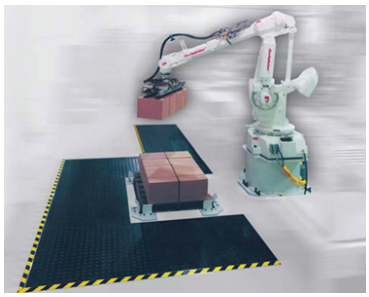


Figura 2.16: Tapete de Segurança[21].

Figura 2.17: *Safety Skin/Pads*[22].

Figura 2.18: *Laser Scanner*[23].

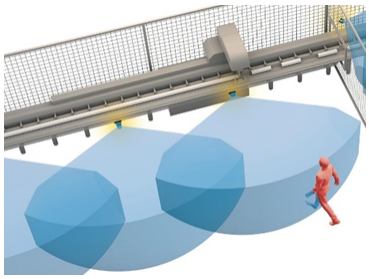


Figura 2.19: *INXPECT*[24].

[21]Fonte: `https://tinyurl.com/37pnmyp2`
[22]Fonte: `https://cordis.europa.eu/project/id/808534/reporting`
[23]Fonte: `https://tinyurl.com/yyfc8t8b`
[24]Fonte: `https://invision-news.de/invison-top-innovation-2021/radar-safety-2/`

2.7.4 Segurança ativa em robôs móveis

No caso dos robôs móveis podemos destacar dois elementos de segurança ativa, são eles o *laser scanner* (tal como temos nos cobots) e os para-choques:

- ***Laser scanner***: como apresentado anteriormente os *Laser Scanners* são dispositivos que permitem monitorizar a área de trabalho em torno de um robô móvel (Figura 2.20);
- **Para-choques**: equipados nos robôs móveis, estes para-choques criam pontos de segurança, interrompendo o movimento do robô ao detetar qualquer contacto físico, prevenindo acidentes (Figura 2.21).

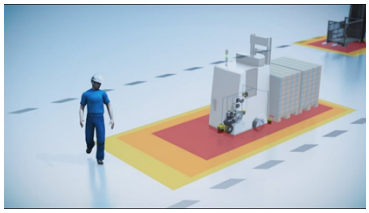


Figura 2.20: *Laser Scanner*[25].

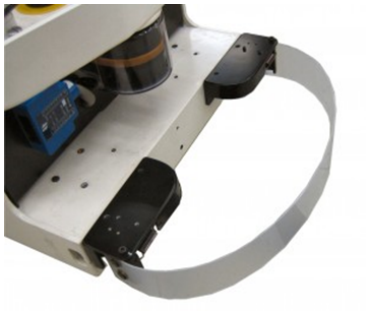


Figura 2.21: Para-choques[26].

2.7.5 Segurança passiva em robôs industriais

No caso dos robôs industriais tradicionais temos as barreiras físicas utilizadas como meio de segurança passiva, contrariamente aos elementos anteriores que são de segurança ativa.

- **Barreiras físicas**: utilizadas em robôs industriais tradicionais, as barreiras físicas servem para manter os operadores afastados do robô, prevenindo interação direta e minimizando os riscos (Figura 2.22).

[25]Fonte: `https://video.sick.com/media/t/0_dct8h6h7/`
[26]Fonte: `https://www.aiki-tcs.co.jp/english/products/another_function_english.html`
[27]Fonte: `https://x-cal.us/products/robot-accessories`

Figura 2.22: Barreiras físicas[27].

2.7.6 Informação normativa aplicável robótica móvel

Os sistemas de AGV estão sujeitos à Diretiva Máquinas 2006/42/CE, que foi incorporada na legislação portuguesa pelo Decreto-Lei n.º 103/2008 de 24 de junho de 2008. Esta diretiva estabelece requisitos obrigatórios para garantir a conformidade do produto. No entanto, a conformidade com uma norma harmonizada é voluntária. Quando um AGV é fabricado de acordo com uma norma harmonizada publicada no Jornal Oficial da União Europeia, presume-se que o produto cumpre os requisitos essenciais de saúde e segurança definidos pela diretiva.

Uma das normas mais importantes aplicáveis a AGV e AMR é a EN ISO 3691-4:2020, intitulada "*Industrial trucks — Safety requirements and verification — Part 4: Driverless industrial trucks and their systems*". Esta norma foi desenvolvida para acompanhar o rápido avanço das tecnologias de robôs móveis na indústria. Esta substitui a antiga EN 1525:1997, que vigorou por 23 anos, e atualmente é considerada a principal norma internacional para AGV na Europa. Destacam-se as seguintes normas para AGV:

- **EN ISO 3691-4:2020**: esta norma aborda de forma abrangente os requisitos de segurança para AGV, especificando as funcionalidades de segurança e os processos de validação necessários. Adota a metodologia de design de segurança da EN ISO 13849 e estabelece o nível de desempenho requerido para as funções de segurança, incluindo modos de operação e controlo de travagem. É importante destacar que esta norma não se aplica apenas aos fabricantes de AGV, mas também impõe requisitos aos utilizadores finais, como a definição de zonas de operação no chão de fábrica que garantam o funcionamento seguro dos AGV. Estas zonas devem ser definidas com base nos riscos específicos de cada aplicação;
- **EN ISO 1175:2020**: esta norma complementa a EN ISO 3691-4 e trata de aspetos elétricos específicos para AGV e AMR, estabelecendo requisitos técnicos para garantir a segurança no funcionamento elétrico destes veículos.

Para além destas normas específicas para os robôs móveis, podemos destacar outras normas complementares, tais como:

- **EN ISO 12100:2010**: "*Safety of machinery — Basic concepts, general principles for design — Risk assessment and risk reduction*". Esta norma estabelece os conceitos básicos e princípios gerais de design para garantir a segurança de máquinas, incluindo a avaliação de riscos e a sua mitigação;
- **EN IEC 60204-1:2016**: "*Safety of machinery — Electrical equipment of machines — Part 1: General requirements*". Esta norma define os requisitos gerais para a segurança dos sistemas elétricos de máquinas, com ênfase na avaliação dos testes elétricos;
- **EN IEC 61000-6-3:2020**: "*Electromagnetic compatibility (EMC) - Part 6-3: Generic standards - Emission standard for equipment in residential environments*". Estabelece os requisitos para a compatibilidade eletromagnética, especificamente para ambientes residenciais, comerciais e industriais leves;
- **EN IEC 61000-6-2:2019**: "*Electromagnetic compatibility (EMC) — Part 6-2: Generic standards — Immunity for industrial environments*". Especifica os requisitos de imunidade eletromagnética para ambientes industriais;
- **EN ISO 24134:2018**: "*Industrial trucks — Additional requirements for automated functions on trucks*". Esta norma trata de requisitos específicos para empilhadores automatizados, com foco nas funções de automação.

2.8 Segurança na robótica colaborativa

Duas das principais normas internacionais mais relevantes e que são aplicáveis sistemas robóticos colaborativos são a ISO 10218 e ISO/TS 15066:2016.

A norma EN ISO 10218:2011 foi desenvolvida para abordar os perigos específicos associados aos robôs industriais. Esta norma do tipo C (norma específica para certos tipos de máquinas) prevalece sobre normas do tipo A ou B, conforme definido na EN ISO 12100:2010. Esta estabelece orientações de segurança tanto no projeto quanto na operação de robôs industriais, levando em consideração as particularidades das diversas aplicações e dos perigos relacionados.

A estrutura desta norma é dividida em duas partes:

- **EN ISO 10218-1:2011**: "*Robots and robotic devices — Safety requirements for industrial robots*". Focando-se na segurança do projeto e construção dos robôs industriais;
- **EN ISO 10218-2:2011**: "*Robots and robotic devices — Safety requirements for industrial robots*". Fornecendo diretrizes para garantir a segurança dos operadores durante as fases de integração, instalação, programação, operação, manutenção e reparo dos robôs.

Por outro lado, a EN ISO/TS 15066:2016 foi lançada em 2016 para complementar a EN ISO 10218:2011, com foco específico em robôs colaborativos. Esta fornece diretrizes

sobre segurança em sistemas colaborativos, onde humanos e robôs compartilham o mesmo espaço de trabalho. Embora seja uma especificação técnica, o que significa as suas diretrizes ainda estão em processo de desenvolvimento, esta é amplamente utilizada como referência para implementar robôs colaborativos de forma segura. Esta norma apresenta os seguintes principais aspetos:

- Definição de espaço colaborativo, onde o robô e o operador humano executam tarefas de forma simultânea;
- Limites de força e pressão que um robô pode exercer sobre o operador, baseados em estudos sobre os limiares de dor humana;
- Detalhamento das operações colaborativas, como: paragem monitorizada, orientação manual, monitorização de velocidade e separação, e limitação de força.

A EN ISO/TS 15066:2016, embora ainda em desenvolvimento, oferece uma visão detalhada das melhores práticas e dos limites de segurança para robôs colaborativos. Esta continua a ser aprimorada com base nas evoluções tecnológicas e novas descobertas científicas, principalmente no que diz respeito à interação dos humanos com os robôs.

Com base nas normas EN ISO 10218-2:2011 e EN ISO/TS 15066:2016, mencionadas anteriormente, podemos classificar as soluções para tornar um sistema robótico colaborativo em quatro categorias diferentes, como ilustrado na Figura 2.23.

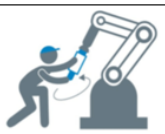
ISO 10218-1	Tipo de operação colaborativa	Velocidade	Principal Acão para a redução de risco	Pictograma (ISO 10218-2)
Tipo 1	Paragem de segurança monitorizada	A velocidade é nula quando o operador está na zona colaborativa	O robô para se o operador estiver na área de trabalho do robô	
Tipo 2	Movido manualmente	Velocidade controlada	O robô move-se de acordo com a ordem do operador	
Tipo 3	Monitorização da velocidade e da separação	Velocidade controlada	O robô apenas se move quando a distância de separação é superior à distância mínima de segurança	
Tipo 4	Limitação da forca por controlo inerente	Limita a força de impacto determinada pela avaliação de risco	Se houver contacto entre o operador e o robô, este só pode exercer forças limitadas	

Figura 2.23: Abordagens de colaboração em sistemas robóticos conforme a ISO 10218.

3 Gestão de frota de robôs móveis

Este capítulo descreve o desenvolvimento e validação da solução para gestão de frota aplicada a robôs móveis, designadamente AGV e AMR.

3.1 Desenvolvimento do *setup* para a gestão de frota com AGV

Para a elaboração do projeto da gestão de frotas de robôs móveis foram efetuadas algumas tarefas preliminares. A recolha da informação das rotas e *tags* presentes na Stellantis Mangualde foi uma das tarefas necessárias para o desenvolvimento da aplicação Web, assim sendo foi possível a recolha de informações relativas às posições (*tags*) presentes em cada rota na unidade fabril, bem como a conceção de mapas virtuais (conforme Figura 3.1) para o desenvolvimento das simulações futuras.

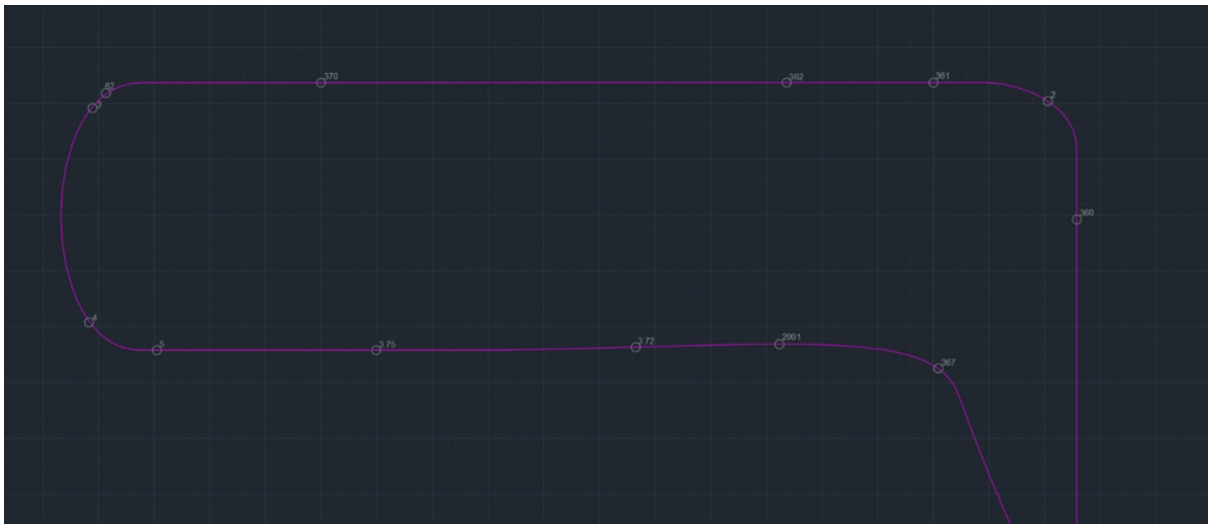


Figura 3.1: Excerto de uma rota retirada com as respetivas *tags*.

Além disto, foram efetuados testes de cobertura de uma rede própria, tendo sido necessário posicionar *Access Point* (AP) em diversos locais ao longo da unidade fabril, testando assim a cobertura com os módulos AP, permitindo que os dispositivos se conectem ao AP com o sinal mais forte. Na Figura 3.2 é possível visualizar o modelo de AP utilizado.

Esta rede de módulos AP é responsável por encaminhar toda a informação para um

Figura 3.2: *Access Point Grandstream GWN7660LR*.

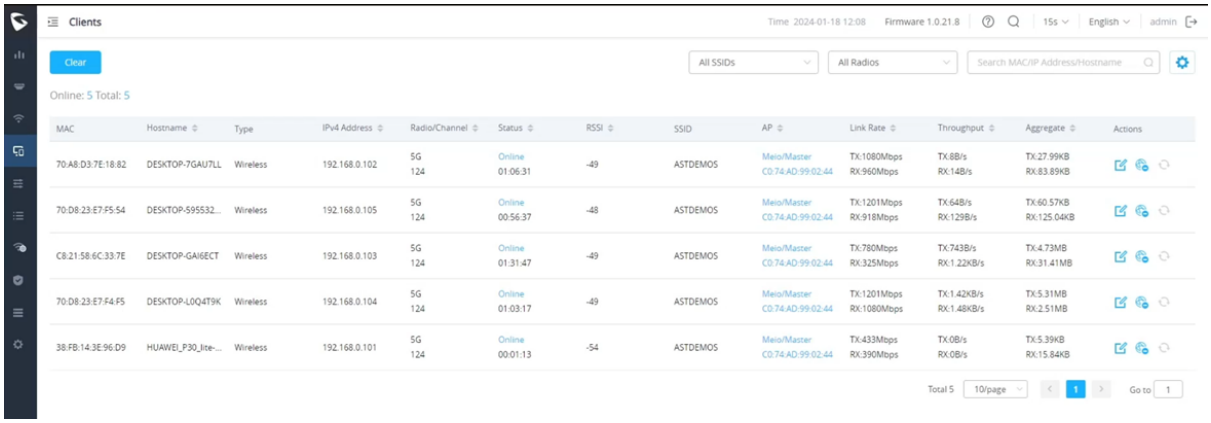


Figura 3.3: Janela de clientes do AP central.

ponto de acesso central, onde é possível consultar os detalhes de cada dispositivo conectado, conforme a Figura 3.3.

Numa fase inicial, o principal desafio para o início do desenvolvimento da aplicação Web para gestão de frota, é a recolha de dados em tempo real para que se tenha sempre uma informação completa e atualizada.

Assim sendo, foram-se testando alternativas ao longo do tempo.

Primeiramente testou-se a recolha de dados na versão dos AGV mais recentes. Nestes dispositivos, esta informação é obtida através do *Datalogger* existente no interior de cada AGV, onde todas as informações relativas à localização e possíveis erros existentes no AGV são armazenadas em base de dados. Os testes realizados nesta fase foram efetuados numa versão cujos *Datalogger* são de uma versão recente e nestas versões a informação está disponível em tempo real numa interface da própria marca, sendo assim possível recolhe-la conforme a Figura 3.4.

Ainda que a solução anterior fosse uma opção bastante válida para os AGV mais recentes, verificou-se nos testes realizados em chão de fábrica que a maioria dos AGV existentes na Stellantis, apresentam nos seus *Dataloggers* apenas informação encriptada

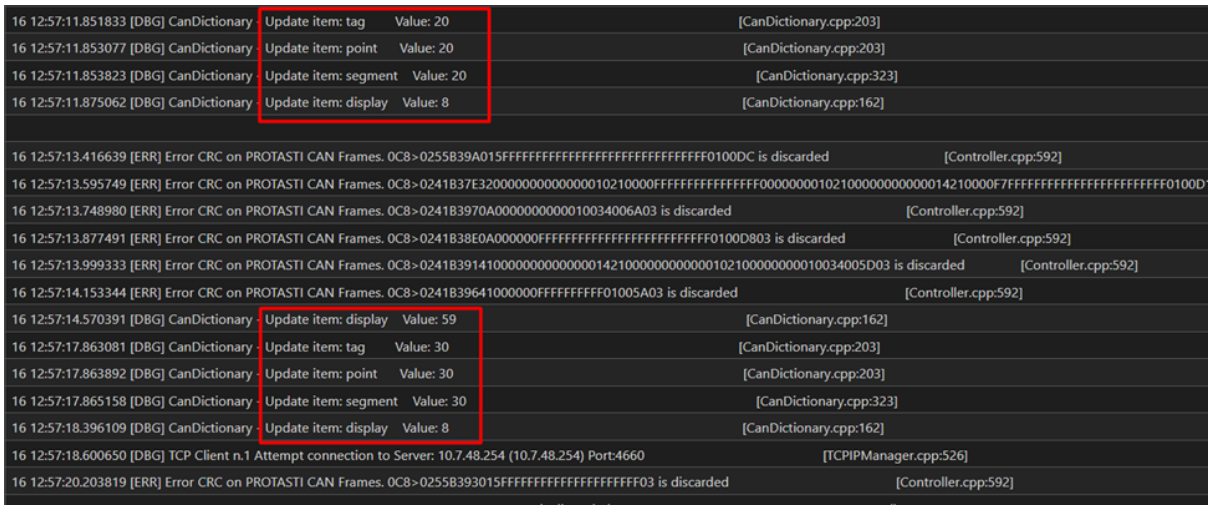


Figura 3.4: Informação retirada da plataforma da ASTI.

e, para além disso, os dados não são em tempo real.

Tornou-se imperativa a procura por outras alternativas para a recolha da informação, e após uma análise da aplicação SIGAT (aplicação da ASTI que permite a programação dos AGV da ASTI, representada na Figura 3.5) constatou-se que a informação necessária está disponível em tempo real também por esta via.

Figura 3.5: Aplicação SIGAT com a informação em tempo real.

Após a identificação desta nova solução e uma análise mais aprofundada, constatou-se que não existia um método eficaz para recolher os dados necessários ao posterior processamento. Verificou-se que, embora os dados fossem armazenados numa base de dados, não era possível extrair essa informação em tempo real a partir da aplicação SIGAT. Como alternativa, testou-se uma nova abordagem, que consistiu na instalação do *software* SIGAT MultiAGV. Com a instalação concluída, cuja interface se encontra representada na Figura 3.6, deram-se início aos procedimentos de teste e análise do

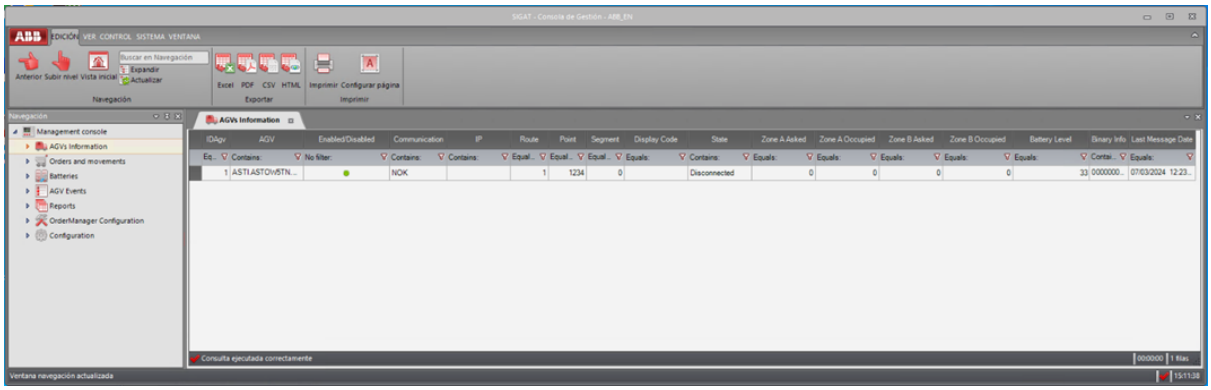


Figura 3.6: Interface do programa SIGAT MultiAGV.

referido *software*.

Para a realização dos testes, foi adquirido um AGV pelo ISEC. O modelo selecionado foi o ASTI Tribot, apresentado na Figura 3.7. Este equipamento permite explorar e compreender a abrangência e o tipo de informações que podem ser extraídas a partir do seu funcionamento.

Figura 3.7: AGV ASTI TriBot.

Posto isto, tornou-se necessária a elaboração de uma rota nas instalações do ISEC. Para a implementação da pista, foi inicialmente realizada uma análise detalhada do mapa das instalações, com o objetivo de identificar os trajetos mais adequados para a demonstração, conforme ilustrado na Figura 3.8. A seleção do itinerário baseou-se em vários critérios, tais como as dimensões do AGV, a sua capacidade de manobra, a localização da zona de carregamento (tendo em conta a proximidade de tomadas trifásicas),

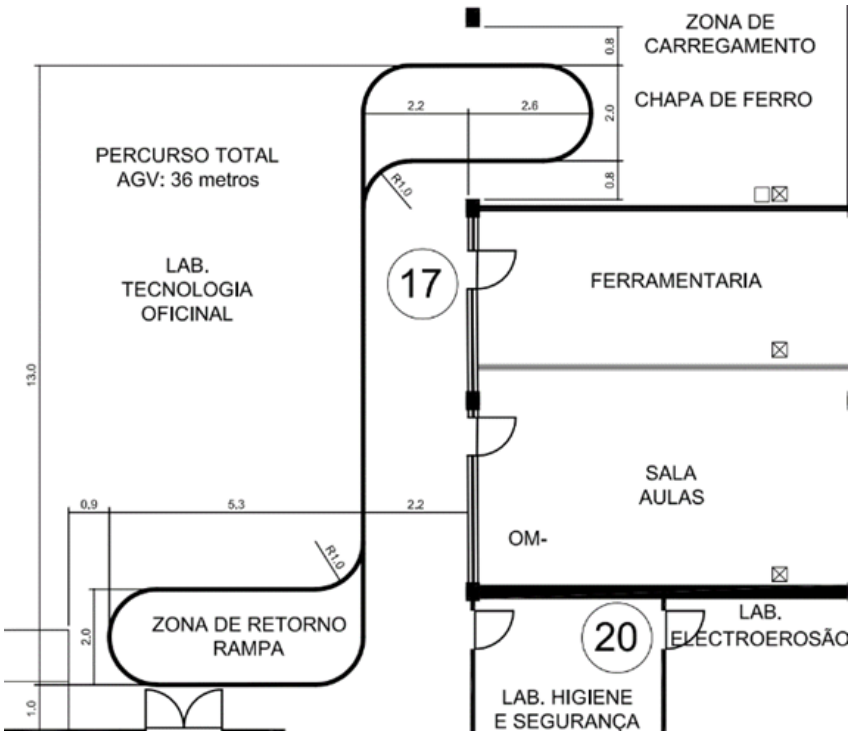


Figura 3.8: Layout da rota a ser construída no ISEC.

e a identificação de áreas com potencial para futuras expansões do projeto — como, por exemplo, a implementação de um sistema de reboque para o AGV. Esta avaliação teve em especial consideração a limitação do espaço disponível, procurando otimizar a utilização da área existente. Assim, avançou-se para a montagem da rota, a qual se encontra ilustrada na Figura 3.9.

Figura 3.9: Rota construída nas instalações do ISEC.

3.2 Integração de AMR na solução de gestão de frota

Nesta fase, utiliza-se o *setup* base descrito na secção anterior. Nele, a solução de gestão de frota assenta exclusivamente nos AGV da ASTI, sendo que todos os veículos equi-

Figura 3.10: AMR HIKROBOT MR-Q3-600LE-DI(CE).

pados com *Datalogger* (na versão mais recente) e com *router* conseguem conectar-se e comunicar com a base de dados. Contudo, foi necessário procurar melhorar a solução existente com o objetivo de evoluir para uma plataforma de gestão de frota universal. Para tal, estebeleceu-se como objetivo a criação de um sistema capaz de monitorizar não apenas AGV, mas também AMR de diferentes fabricantes, proporcionando assim uma gestão centralizada e abrangente de vários tipos de robôs móveis. O desenvolvimento desta tarefa teve o envolvimento direto e responsabilidade do aluno.

Por conseguinte, houve a necessidade de adquirir novas máquinas para testes de recolha de informação, designadamente os três AMR que, a seguir, se descrevem sumariamente:

- **AMR HIKROBOT MR-Q3-600LE-DI(CE)**: é um AMR com capacidade carga de até 600 Kg, utiliza uma tecnologia de navegação por Laser SLAM, ao contrário dos AGV apresentados anteriormente da ASTI. Possui um sistema de elevação elétrica, proporcionando assim uma maior versatilidade ao transportar diferentes tipos de cargas. A sua velocidade de locomoção é de até 2 m/s (Figura 3.10);
- **AMR Aiten EMP10**: comparativamente com o AMR anterior, possui uma maior capacidade de carga, com a possibilidade de até 1000 Kg; por outro lado, possui uma velocidade de 0,7 m/s (Figura 3.11);
- **AMR Seer AMB-1000JS**: apresenta, também, uma capacidade de carga de 1000 Kg e a sua velocidade máxima de locomoção é na ordem dos 1,5 m/s (Figura 3.12).

Portanto, ao contrário dos robôs mencionados inicialmente e do AGV adquirido para as instalações do ISEC, estes robôs são AMR em vez de AGV. Equipados com tecnologia Laser SLAM, estes robôs modernos oferecem uma navegação mais precisa, especialmente em ambientes industriais complexos. O "Seer AMB-1000JS" foi o modelo escolhido para o desenvolvimento do *software* de gestão de frota descrito neste documento,

Figura 3.11: AMR Aiten EMP10.

Figura 3.12: AMR Seer AMB-1000JS.

por ter sido o único disponível nas instalações do ISEC para a realização de testes.

3.3 Protocolos TCP/IP, HTTP e métodos GET e POST

Para a recolha de informação dos dispositivos robóticos, foram utilizados os métodos GET e POST. De seguida, apresenta-se uma breve explicação do funcionamento destes métodos e do seu enquadramento no contexto do sistema.

O *Transmission Control Protocol/Internet Protocol* (TCP/IP) é o conjunto de protocolos responsável por toda a comunicação via Internet. Define como os dados são divididos em pacotes, enviados pela rede e reagrupados no destino. Assim sendo, este protocolo é a base que permite que os dispositivos em redes diferentes comuniquem entre si, independentemente do tipo de *hardware* ou *software*.

O modelo TCP/IP está organizado em quatro camadas, cada uma com responsabilidades distintas no processo de comunicação:

- **Camada de acesso à rede** (*Link Layer*): esta é a camada mais baixa, trata da comunicação direta com o hardware (placas de rede, *switches*, entre outros). Esta camada é responsável por definir como é que os dados são transmitidos fisicamente através de diferentes tipos de redes (Ethernet, Wi-fi, entre outras);

- **Camada da internet** (*Internet Layer*): esta camada trata do encaminhamento dos pacotes de dados através da rede, o principal protocolo presente na camada é o IP (*Internet Protocol*) responsável pela definição de como os pacotes são enviados de um dispositivo para o outro, através de diferentes redes e *routers*;

- **Camada de transporte** (*Transport Layer*): a camada de transporte está associada à comunicação de ponta a ponta entre dispositivos, sendo que os principais protocolos associados a esta camada são o *Transmission Control Protocol* (TCP) que garante que todos os dados chegam sem erros e na ordem correta e o *User Datagram Protocol* (UDP) que se trata de um protocolo mais simples e rápido, no entanto sem garantias de fiabilidade, sendo por isso utilizado em situações onde a velocidade é mais importante que a precisão;

- **Camada de aplicação** (*Application Layer*): é nesta camada que residem protocolos de alto nível, fornecendo a interface entre o utilizador e a rede. Destacam-se os seguintes protocolos que atuam nesta camada: o *Hypertext Transfer Protocol* (HTTP) que é utilizado na navegação Web, o *File Transfer Protocol* (FTP) utilizado na transferência de ficheiros e o *Simple Mail Transfer Protocol* (SMTP) que é importante para o envio de emails.

Neste caso de estudo, para recolha de dados dos robôs é utilizado o HTTP presente na camada de aplicação, utilizado para a comunicação na web. O HTTP define como os pedidos (*requests*) e as respostas (*responses*) são trocados entre o cliente e o servidor. O HTTP trabalha diretamente com o TCP da camada de transporte, o que significa que, enquanto o HTTP define o conteúdo e o formato dos pedidos, o TCP garante a entrega segura e fiável desses pedidos e respostas entre o cliente e o servidor. O HTTP utiliza diferentes métodos para comunicar a intenção do cliente ao servidor. Os dois métodos mais comuns são o GET e o POST. Estes dois métodos irão ser brevemente explicados através da Tabela 3.1.

A Figura 3.13 ilustra o método de comunicação adotado neste caso de estudo. O AMR está ligado a uma rede à qual também está conectado o servidor que executa a plataforma Web. Desta forma, a comunicação entre o robô e a plataforma de visualização ocorre através dos métodos GET e POST, previamente descritos.

Tabela 3.1: Principais diferenças entre os métodos GET e POST [48].

Característica	GET	POST
Objetivo	Solicitar dados do servidor (leitura)	Enviar dados para o servidor (escrita/modificação)
Alteração no servidor	Não	Sim
Envio de dados	Através do URL	No corpo da mensagem (não visível no URL)
Segurança	Menos seguro (dados visíveis no URL)	Mais seguro (dados ocultos no corpo da mensagem)
Cacheável	Sim	Não
Tamanho de dados	Limitado (pelo comprimento do URL)	Ilimitado (pode enviar grandes quantidades de dados)
Utilização comum	Pedir páginas web, imagens, ou ficheiros estáticos	Submeter formulários, enviar ficheiros ou realizar operações no servidor
Visibilidade dos dados	Parâmetros visíveis no URL	Parâmetros ocultos no corpo do pedido
Repetibilidade	Seguro de repetir sem efeitos colaterais	Repetir pode causar duplicação de ações (ex: submissão de formulários)
Exemplo de utilização	http://exemplo.com/produto?id=123	Submissão de um formulário de registo ou login

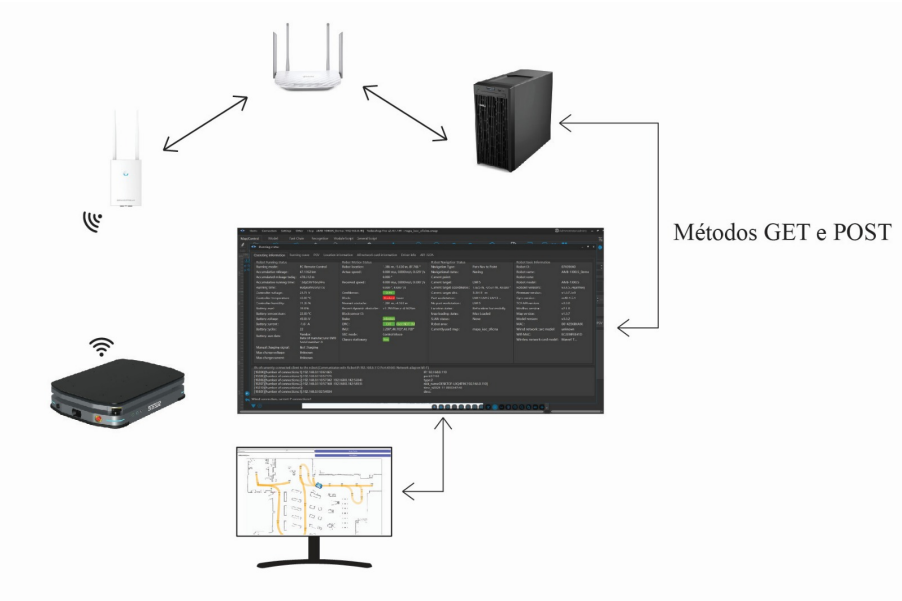


Figura 3.13: Método de comunicação com o AMR.

3.4 Descrição do código da plataforma de visualização dos AMR

A plataforma de monitorização dos robôs foi elaborada essencialmente com recurso à linguagem de programação *JavaScrip* que é uma linguagem de programação interpretada estruturada que juntamente com *Hypertext Markup Language* (HTML) e *Cascading Style Sheets* (CSS) permite a execução de páginas Web interativas e, por isso, é essencial para a programação das mesmas. Os códigos completos, desenvolvidos pelo aluno, encontram-se nos anexos A (*JavaScript*) e B (HTML). Na Figura 3.14 é possível visua-

Figura 3.14: Interligação das diferentes tecnologias utilizadas (HTML, *JavaScript* e CSS).

lizar um esquema que demonstra a interligação das diferentes tecnologias utilizadas. Nas secções seguintes são explicadas, de forma sumária, algumas das principais funcionalidades implementadas.

3.4.1 Função de transformação de coordenadas

A função *transformCoords* adapta as coordenadas do mapa para o sistema de coordenadas do canvas HTML, aplicando uma escala proporcional. É uma função importante para desenhar os pontos e as formas corretamente no canvas, independentemente das dimensões. É chamada quer na função *drawCurve*, quer na *drawMap*. O código da função pode ser visualizado na Figura 3.15.

```
function transformCoords(x, y, canvasWidth, canvasHeight) {
    const scaleX = canvasWidth / (maxX - minX);
    const scaleY = canvasHeight / (maxY - minY);
    const newX = (x - minX) * scaleX;
    const newY = canvasHeight - (y - minY) * scaleY;
    return { newX, newY };
}
```

Figura 3.15: Função *transformCoords*.

3.4.2 Funções de desenho de pontos e formas

As seguinte funções são responsáveis pelo desenho de pontos e formas:

- Função *drawPoint*: desenha um ponto circular no canvas (Figura 3.16);
- Função *drawRect*: desenha um retângulo arredondado (Figura 3.17);
- Função *drawCurve*: cria trajetórias entre pontos com base em classes de curva (sejam elas *Bezier*, arco ou uma linha reta) (Figura 3.18).

```
function drawPoint(ctx, x, y, color, label) {
    ctx.beginPath();
    ctx.arc(x, y, 5, 0, 2 * Math.PI);
    ctx.fillStyle = color;
    ctx.fill();
    ctx.closePath();

    if (label) {
        ctx.font = "bold 12px Arial";
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.fillStyle = '#000';
        ctx.fillText(label, x, y - 22);
    }
}
```

Figura 3.16: Função *drawPoint*.

```
function drawRect(ctx, x, y, color, label) {
    ctx.fillStyle = color;
    ctx.strokeStyle = color;
    ctx.fillStyle = 'rgba(0, 0, 255, 0.6)';

    ctx.beginPath();
    ctx.roundRect(x - 7, y - 7, 14, 14, 5);
    ctx.fill();
    ctx.closePath();

    if (label) {
        ctx.font = "bold 12px Arial";
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.fillStyle = '#000';
        ctx.fillText(label, x, y + 20);
    }
}
```

Figura 3.17: Função *drawRect*.

```
function drawCurve(ctx, startPos, endPos, curve, canvasWidth, canvasHeight) {
    ctx.lineWidth = 25;
    ctx.lineJoin = 'round';
    ctx.lineCap = 'round';
    ctx.strokeStyle = 'rgba(255, 165, 0, 0.3)';

    if (curve.className === 'BezierPath') {
        const controlPos1 = transformCoords(curve.controlPos1.x, curve.controlPos1.y, canvasWidth, canvasHeight);
        const controlPos2 = transformCoords(curve.controlPos2.x, curve.controlPos2.y, canvasWidth, canvasHeight);
        ctx.beginPath();
        ctx.moveTo(startPos.newX, startPos.newY);
        ctx.bezierCurveTo(controlPos1.newX, controlPos1.newY, controlPos2.newX, controlPos2.newY, endPos.newX, endPos.newY);
        ctx.stroke();
    } else if (curve.className === 'DegenerateBezier') {
        const controlPos1 = transformCoords(curve.controlPos1.x, curve.controlPos1.y, canvasWidth, canvasHeight);
        const controlPos2 = transformCoords(curve.controlPos2.x, curve.controlPos2.y, canvasWidth, canvasHeight);
        ctx.beginPath();
        ctx.moveTo(startPos.newX, startPos.newY);
        ctx.bezierCurveTo(controlPos1.newX, controlPos1.newY, controlPos2.newX, controlPos2.newY, endPos.newX, endPos.newY);
        ctx.stroke();
    } else if (curve.className === 'ArcPath') {
        const controlPos = transformCoords(curve.controlPos1.x, curve.controlPos1.y, canvasWidth, canvasHeight);
        ctx.beginPath();
        ctx.moveTo(startPos.newX, startPos.newY);
        ctx.quadraticCurveTo(controlPos.newX, controlPos.newY, endPos.newX, endPos.newY);
        ctx.stroke();
    } else if (curve.className === 'StraightPath') {
        ctx.beginPath();
        ctx.moveTo(startPos.newX, startPos.newY);
        ctx.lineTo(endPos.newX, endPos.newY);
        ctx.stroke();
    }
}
```

Figura 3.18: Função *drawCurve*.

3.4.3 Funções de desenho dos elementos principais

A função *drawAGV* desenha uma imagem de um AGV, conforme se pode verificar na Figura 3.19. Esta função carrega uma imagem do AGV, identificada como "AGV.png" e define o seu posicionamento conforme a variável "*robotAngle*". Tem também um evento de clique que verifica se o clique ocorre dentro do raio do AGV, permitindo abrir uma interface de opções para o utilizador quando o AGV é selecionado.

```
function drawAGV(ctx, robotPos, robotAngle) {
    const image = new Image();
    image.src = 'AGV.png';
    const imageSizeX = 540;
    const imageSizeY = 400;
    const scale = 0.15;

    image.onload = function () {
        ctx.save();
        ctx.translate(robotPos.newX, robotPos.newY);
        ctx.rotate(robotAngle);
        ctx.drawImage(image, (-imageSizeX * scale) / 2, (-imageSizeY * scale) / 2, (imageSizeX * scale), (imageSizeY * scale));
        ctx.restore();

        canvas.addEventListener('click', (event) => {
            const rect = canvas.getBoundingClientRect();
            const x = event.clientX - rect.left;
            const y = event.clientY - rect.top;

            const agvRadius = (imageSizeX * scale) / 2;
            const distance = Math.sqrt(Math.pow(robotPos.newX - x, 2) + Math.pow(robotPos.newY - y, 2));

            // Se o clique estiver dentro do raio do AGV
            if (distance < agvRadius) {
                if (showinfo === 0) {
                    showinfo = 1;
                    showTooltipOptions(x, y);
                } else {
                    showinfo = 0;
                    hideTooltipAndTables();
                }
            }
        });
    };
}
```

Figura 3.19: Função *drawAGV*.

Por outro lado, a função *drawSLAM* é a função responsável pelo desenho dos pontos que constituem as paredes e obstáculos representados no mapa SLAM, é essencial para transmitir na página Web a perceção do espaço envolvente do mapa das posições do robô. A pequena função *drawSLAM* pode ser visualizada na Figura 3.20. A mesma vai ser chamada na função *drawMap* para aí sim fazer desempenhar o seu papel.

A função *drawReferencial* é responsável por desenhar o referencial utilizado no mapa de origem do robô — ou seja, aquele que pode ser configurado no programa *Roboshop Pro*. A representação dessa função pode ser vista na Figura 3.21. Esta função desenha dois eixos: uma linha horizontal vermelha com uma seta para a direita indicando o eixo do X e uma linha vertical verde com uma seta para cima indicando o eixo do Y. A função será também solicitada pela função *drawMap* para que o eixo de coordenadas do próprio robô seja mostrado na página Web de monitorização, permitindo uma melhor análise dos valores e das posições que estão a ser mostrados na página.

```
function drawSLAM(ctx, x, y) {
    color = '#606060';
    ctx.beginPath();
    ctx.arc(x, y, 1, 0, 2 * Math.PI);
    ctx.fillStyle = color;
    ctx.fill();
    ctx.closePath();
}
```

Figura 3.20: Função *drawSLAM*.

```
function drawReferencial(ctx, x, y) {
    const arrowLength = 10;
    const axisLength = 50;
    const labelOffset = 15;

    ctx.save();

    // Desenha o eixo X em vermelho
    ctx.lineWidth = 2;
    ctx.strokeStyle = 'red';
    ctx.beginPath();
    ctx.moveTo(x, y);
    ctx.lineTo(x + axisLength, y);
    ctx.stroke();

    // Desenha a seta vermelha para o eixo X
    ctx.beginPath();
    ctx.moveTo(x + axisLength, y);
    ctx.lineTo(x + axisLength - arrowLength, y - arrowLength / 2);
    ctx.moveTo(x + axisLength, y);
    ctx.lineTo(x + axisLength - arrowLength, y + arrowLength / 2);
    ctx.stroke();

    // Desenha o eixo Y em verde
    ctx.strokeStyle = 'green';
    ctx.beginPath();
    ctx.moveTo(x, y);
    ctx.lineTo(x, y - axisLength);
    ctx.stroke();

    // Desenha a seta verde para o eixo Y
    ctx.beginPath();
    ctx.moveTo(x, y - axisLength);
    ctx.lineTo(x - arrowLength / 2, y - axisLength + arrowLength);
    ctx.moveTo(x, y - axisLength);
    ctx.lineTo(x + arrowLength / 2, y - axisLength + arrowLength);
    ctx.stroke();

    // Rótulos dos eixos
    ctx.font = '12px Arial';
    ctx.fillStyle = 'black';
    ctx.fillText('X', x + axisLength + labelOffset, y);
    ctx.fillText('Y', x, y - axisLength - labelOffset);

    ctx.restore();
}
```

Figura 3.21: Função *drawReferencial*.

E finalmente, a função *drawMap* é responsável por desenhar o mapa completo, incluindo pontos, retângulos, as rotas e o próprio mapa SMAP. Está representada na Figura 3.22. Algumas das características da função são:

- As variáveis *advancedPointList* e *binLocationsList* são desenhadas no canvas como pontos e retângulos, respetivamente. Desenhando assim os pontos que constituem o mapa SLAM e os retângulos representativos dos pontos de destino do robô;
- A *advancedCurveList* desenha as rotas entre os pontos utilizando curvas;
- A *robotPosition* chama a *drawAGV* para renderizar o AGV na sua posição atual do mapa;
- É ainda chamada a função *drawReferencial* para desenhar então o referenical no mapa.

```
function drawMap(data) {
    const canvas = document.getElementById('canvas');
    const ctx = canvas.getContext('2d');
    canvas.width = 1675;
    canvas.height = 1100;
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    let instanceNames = [];

    if (data.advancedPointList) {
        data.advancedPointList.forEach(point => {
            const { newX, newY } = transformCoords(point.pos.x, point.pos.y, canvas.width, canvas.height);
            const color = '#00A8FF';
            drawPoint(ctx, newX, newY, color, point.instanceName);
            if (point.instanceName) {
                instanceNames.push(point.instanceName);
            }
        });
    }

    if (data.binLocationsList) {
        data.binLocationsList.forEach(point => {
            const { newX, newY } = transformCoords(point.binLocationList[0].pos.x, point.binLocationList[0].pos.y, canvas.width, canvas.height);
            const color = '#0000FF';
            drawRect(ctx, newX, newY, color, point.binLocationList[0].instanceName);
            if (point.binLocationList[0].instanceName) {
                instanceNames.push(point.binLocationList[0].instanceName);
            }
        });
    }

    if (data.advancedCurveList) {
        data.advancedCurveList.forEach(curve => {
            const startPos = transformCoords(curve.startPos.pos.x, curve.startPos.pos.y, canvas.width, canvas.height);
            const endPos = transformCoords(curve.endPos.pos.x, curve.endPos.pos.y, canvas.width, canvas.height);
            drawCurve(ctx, startPos, endPos, curve, canvas.width, canvas.height);
        });
    }

    if (data.robotPosition) {
        const robotPos = transformCoords(data.robotPosition.x, data.robotPosition.y, canvas.width, canvas.height);
        drawAGV(ctx, robotPos, data.robotPosition.angle);
    }
    if (data.normalPosList) {
        data.normalPosList.forEach(point => {
            const { newX, newY } = transformCoords(point.x, point.y, canvas.width, canvas.height);
            drawSLAM(ctx, newX, newY);
        });
    }
    const { newX: refX, newY: refY } = transformCoords('0', '0', canvas.width, canvas.height);
    drawReferencial(ctx, refX, refY);
}
```

Figura 3.22: Função *drawMap*.

3.4.4 Função de atualização das sugestões para o utilizador

A função *updateDatalist* tem o objetivo de atualizar a lista de sugestões no <*datalist*> para que o utilizador tenha a possibilidade de escolher locais disponíveis e atualizados, garantindo que só os nomes únicos e relevantes são apresentados pela seleção. O código desta função encontra-se ilustrado na Figura 3.23.

```
function updateDatalist(instanceNames) {
    const uniqueInstanceNames = [...new Set(instanceNames)];
    const dataList = document.getElementById('toLoc');
    dataList.innerHTML = '';

    uniqueInstanceNames.forEach(name => {
        const option = document.createElement('option');
        option.value = name;
        dataList.appendChild(option);
    });
}
```

Figura 3.23: Função *updateDatalist*.

3.4.5 Funções de manipulação de JSON e Formatação

A função *convertSingleToDoubleQuotes* ilustrada na Figura 3.24, converte aspas simples em duplas no JSON e ajusta valores boleanos para compatibilidade.

```
function convertSingleToDoubleQuotes(smapContent) {
    smapContent = smapContent.replace(/'/g, '"');
    smapContent = smapContent.replace(/\b(True|False)\b/g, match => match.toLowerCase());
    smapContent = smapContent.replace(/\b(None)\b/g, 'null');
    return smapContent;
}
```

Figura 3.24: Função *convertSingleToDoubleQuotes*.

Por outro lado, a função *inspectAndFixJson* tenta analisar o JSON para verificar se o mesmo está formatado corretamente. Na Figura 3.25 pode observar-se esta mesma função.

```
function inspectAndFixJson(smapContent) {
    try {
        return JSON.parse(smapContent);
    } catch (e) {
        console.error(`Erro ao decodificar JSON: ${e.message}`);
        return null;
    }
}
```

Figura 3.25: Função *inspectAndFixJson*.

3.4.6 Funções de comunicação com o servidor

A função *loadRobotIds* efetua uma requisição GET para */robotsStatus*, permitindo uma atualização da lista de IDs de robôs na interface. A função pode ser visualizada na Figura 3.26.

```
async function loadRobotIds() {
    const url2 = `http://${server_ip}:${server_port}/robotsStatus`;

    try {
        const response = await axios.get(url2);
        const robotData = response.data;

        const idRobotList = document.getElementById('IDRobot');

        idRobotList.innerHTML = '';

        robotData.report.forEach(robot => {
            const option = document.createElement('option');
            option.value = robot.vehicle_id;
            idRobotList.appendChild(option);
        });
    } catch (error) {
        console.error('Erro ao carregar os IDs dos robôs:', error);
    }
}
```

Figura 3.26: Função *loadRobotIds*.

Por outro lado, para fazer uma atualização dos dados do robô selecionado é utilizada a função *getData* que obtém dados da localização e estado do robô através do método POST para */robotSmap* e através do método GET para */robotsStatus*. Esta função pode ser observada através da Figura 3.27.

Para o envio de uma ordem para o robô é utilizada a função *sendOrder* (Figura 3.28), que permite inserir a localização do destino pretendido através de uma requisição do tipo POST. Assim, a comunicação efetuada nas funções supramencionadas são elaboradas através de requisições do tipo GET e, no caso da função *sendOrder* é elaborada uma requisição do tipo POST e com o recurso à *Application Programming Interface* (API) denominada de *setOrder*, é possível assim gerar um pedido de deslocamento do robô até ao ponto desejado.

3.4.7 Funções para a exibição das informações do robô

As funções *showTooltipOptions*, *showStatus*, *hideTooltipAndTables* e *updateTableData* controlam a visibilidade das tabelas de informação do robô (designadas por *updateRunningStatus* e *updateBasicInfo*), as mesmas são apresentadas na Secção 5.1.

3.4.8 Função para possibilitar o arrastamento de tabelas

A função *makeTableDraggable* permite que as tabelas de informações do robô possam ser movidas na interface da página Web, permitindo assim uma maior interatividade. Esta função encontra-se ilustrada na Figura 3.29.

```
// Função para buscar dados do robô
async function getData(selectedRobotId) {
    const url1 = `http://${server_ip}:${server_port}/robotSmap`;
    const url2 = `http://${server_ip}:${server_port}/robotsStatus`;

    const headers = {
        'Content-Type': 'application/json'
    };

    try {
        const response2 = await axios.get(url2);
        const robotData2 = response2.data;

        const selectedRobot = robotData2.report.find(robot => robot.vehicle_id === selectedRobotId);
        if (!selectedRobot) {
            console.error(`Robô com ID ${selectedRobotId} não encontrado.`);
            return;
        }

        // Se não existir uma instância do robô, cria uma nova
        if (!selectedRobotInstance) {
            selectedRobotInstance = new Robot(selectedRobotId);
        }

        // Atualiza os dados do robô
        selectedRobotInstance.updateData(selectedRobot);

        let data = {
            vehicle: selectedRobotId,
            map: selectedRobotInstance.data.currentMap + ".smap"
        };

        const response1 = await axios.post(url1, data, { headers });
        const robotData = response1.data;

        let smapContent = JSON.stringify(robotData);
        smapContent = convertSingleToDoubleQuotes(smapContent);
        const smapData = inspectAndFixJson(smapContent);

        if (smapData) {
            const smapdata_str = JSON.stringify(smapData);
            const positions = smapdata_str.match(/"pos":\s*{\s*"x":\s*([\d.-]+),\s*"y":\s*([\d.-]+)\s*}/g);

            if (positions) {
                const pos_list = positions.map(pos => {
                    const match = pos.match(/"pos":\s*{\s*"x":\s*([\d.-]+),\s*"y":\s*([\d.-]+)\s*}/);
                    return { x: parseFloat(match[1]), y: parseFloat(match[2]) };
                });

                const x_values = pos_list.map(pos => pos.x);
                const y_values = pos_list.map(pos => pos.y);

                if (x_values.length > 0 && y_values.length > 0) {
                    maxX = Math.max(...x_values);
                    minX = Math.min(...x_values);
                    maxY = Math.max(...y_values);
                    minY = Math.min(...y_values);

                    // Para dar uma margem de 2
                    maxX += 2;
                    minX -= 2;
                    maxY += 2;
                    minY -= 2;
                } else {
                    console.log("Não há valores válidos.");
                }
            } else {
                console.log("Nenhum valor de 'pos' encontrado.");
            }

            smapData.robotPosition = {
                x: selectedRobotInstance.data.positionX,
                y: selectedRobotInstance.data.positionY,
                angle: selectedRobotInstance.data.heading
            };
            drawMap(smapData);
            selectedRobotInstance.updateData(selectedRobot);
        } else {
            console.log('Não foi possível carregar os dados do mapa.');
        }
    } catch (error) {
        if (error.response) {
            const status = error.response.status;
            if (status === 400) {
                console.log('Os parâmetros solicitados estão incorretos ou faltando informações necessárias, consulte a documenta
            } else if (status === 404) {
                console.log('Os dados não existem ou não estão disponíveis');
            } else if (status === 406) {
                console.log('Solicitação indisponível');
            } else if (status === 500) {
                console.log('Erro interno do servidor');
            } else {
                console.log('Erro inesperado:', status);
            }
        } else {
            console.error('Erro:', error.message);
        }
    }
}
```

Figura 3.27: Função *getData*.

```
function sendOrder(serverIp, serverPort, taskId, toLoc) {
    const url = `http://${serverIp}:${serverPort}/setOrder`;
    const data = {
        id: taskId,
        blocks: [{
            blockId: "1",
            location: toLoc
        }],
        complete: true
    };

    console.log('Sending data:', data);

    const headers = {
        'Content-Type': 'application/json'
    };

    axios.post(url, data, { headers: headers })
        .then(response => {
            console.log('Response data:', response.data);
        })
        .catch(error => {
            if (error.response) {
                console.error('Error response data:', error.response.data);
                console.error('Error response status:', error.response.status);
                console.error('Error response headers:', error.response.headers);
                if (error.response.data.msg) {
                    alert(`Erro: ${error.response.data.msg}`);
                }
            } else if (error.request) {
                console.error('Error request:', error.request);
            } else {
                console.error('General error:', error.message);
            }
            console.error('Error config:', error.config);
        });
}
```

Figura 3.28: Função *sendOrder*.

```
function makeTableDraggable(tableId) {
    const table = document.getElementById(tableId);
    let offsetX = 0, offsetY = 0;
    let isDragging = false;

    function mouseDown(e) {
        e.preventDefault();

        offsetX = e.clientX ;
        offsetY = e.clientY ;

        isDragging = true;

        document.addEventListener('mousemove', mouseMove);
        document.addEventListener('mouseup', mouseUp);
    }

    function mouseMove(e) {
        if (isDragging) {
            table.style.position = 'absolute';
            table.style.left = `${e.clientX - offsetX}px`;
            table.style.top = `${e.clientY - offsetY}px`;
        }
    }

    function mouseUp() {
        isDragging = false;
        document.removeEventListener('mousemove', mouseMove);
        document.removeEventListener('mouseup', mouseUp);
    }

    table.addEventListener('mousedown', mouseDown);
}
function makeAllTablesDraggable() {
    const tables = document.querySelectorAll('.statusTable');
    tables.forEach(table => {
        makeTableDraggable(table.id);
    });
}
```

Figura 3.29: Função *makeTableDraggable*.

```
class Robot {
    constructor(vehicleId) {
        this.vehicleId = vehicleId;
        this.data = {};
    }

    updateData(selectedRobot) {
        // Atualiza os dados do robô a partir do objeto selectedRobot
        this.data.batteryLevel = selectedRobot.rbk_report.battery_level;
        this.data.batteryPercentage = (this.data.batteryLevel * 100).toFixed(1);
        this.data.positionX = selectedRobot.rbk_report.x;
        this.data.positionY = selectedRobot.rbk_report.y;
        this.data.heading = selectedRobot.rbk_report.angle;
        this.data.currentMap = selectedRobot.rbk_report.current_map;
        this.data.headingDegrees = Math.round(this.data.heading * (180 / Math.PI));

        this.data.vehicleId = this.vehicleId;

        // Armazena outros dados adicionais
        this.data.time = selectedRobot.rbk_report.time;
        this.data.todayOd = selectedRobot.rbk_report.today_od;
        this.data.totalTime = selectedRobot.rbk_report.total_time;
        this.data.odo = selectedRobot.rbk_report.odo;
        this.data.controllerVoltage = selectedRobot.rbk_report.controller_voltage;
        this.data.controllerTemp = selectedRobot.rbk_report.controller_temp;
        this.data.controllerHumi = selectedRobot.rbk_report.controller_humi;
        this.data.charging = selectedRobot.rbk_report.charging;
        this.data.batteryTemp = selectedRobot.rbk_report.battery_temp;
        this.data.voltage = selectedRobot.rbk_report.voltage;
        this.data.current = selectedRobot.rbk_report.current;
        this.data.batteryCycle = selectedRobot.rbk_report.battery_cycle;
        this.data.batteryUserData = selectedRobot.rbk_report.battery_user_data;
        this.data.manualCharge = selectedRobot.rbk_report.manual_charge;
        this.data.maxChargeCurrent = selectedRobot.rbk_report.max_charge_current;
        this.data.maxChargeVoltage = selectedRobot.rbk_report.max_charge_voltage;
    }
}
```

Figura 3.30: Classe *Robot*.

3.4.9 Classe *Robot* e função de atualização de dados

A classe *Robot* tem a função de manter os dados do robô, atualizando-os de acordo com os dados recebidos. O seu código está presente na Figura 3.30.

Por outro lado, para armazenar e converter os dados da classe *Robot* é utilizada a função *updateData* que permite a utilização direta da classe nas tabelas de informações do robô.

3.5 Criação do mapa SMAP utilizando o SLAM do robô Seer

Como já referido anteriormente, os AMR são dotados da tecnologia SLAM que permite mapear todo o ambiente envolvente ao robô, permitindo não apenas a criação de um mapa, mas também a localização precisa do robô no espaço. O SLAM permite que se construa um mapa do ambiente enquanto o robô se desloca, ajustando a sua localização em tempo real.

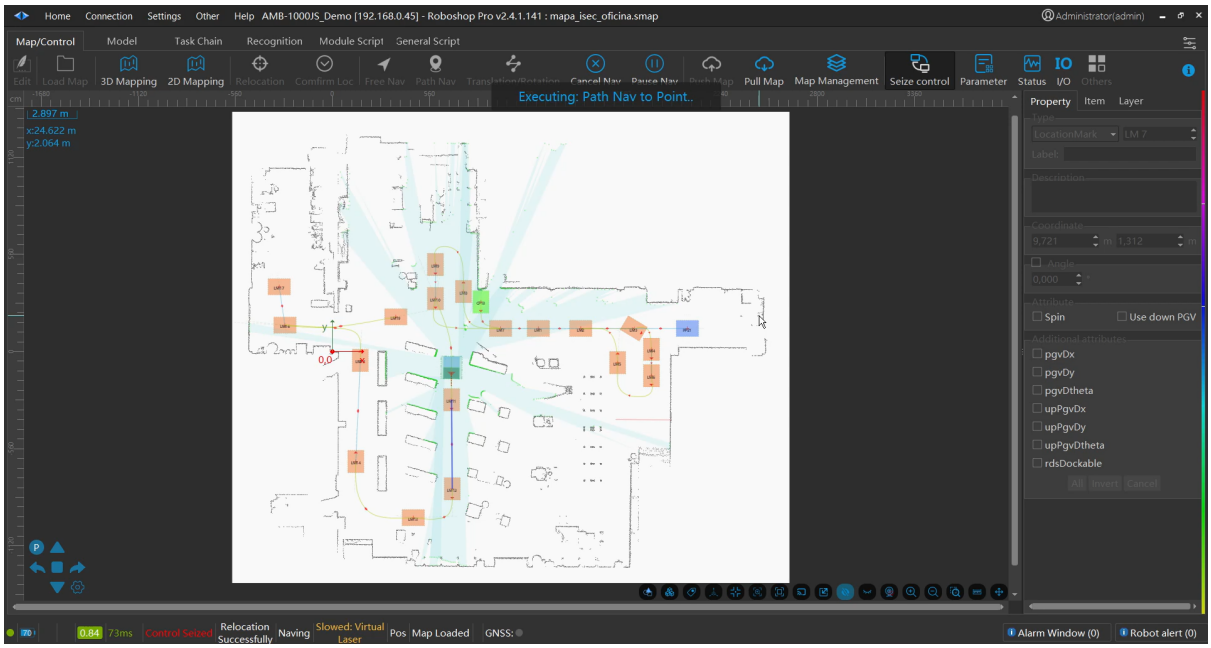


Figura 3.31: Mapa SLAM das intalações das oficinas do ISEC adquirido através do *software Roboshop Pro*.

A criação do mapa SMAP é um processo que envolve algumas etapas e é realizado com recurso ao programa *Roboshop Pro*. Este processo permite ao robô identificar obstáculos, corredores e outras características importantes do ambiente envolvente, permitindo uma navegação autónoma e segura.

Na Figura 3.31 pode ser visualizado o mapa SMAP adquirido nas instalações do ISEC que vai ser alvo de testes para a monitorização do robô em questão através da plataforma desenvolvida.

Ao observar o mapa, é possível identificar vários pontos de destino designados por "LM", os quais permitem a definição de tarefas que instruem o robô a dirigir-se a um destino específico. A Figura 3.32 apresenta um exemplo de tarefa com o *Target Point* configurado para o ponto de destino "LM10".

Por outro lado, um dos pontos mais relevantes do mapa é o que representa o posto de carregamento do robô. Em ambientes industriais, é essencial dispor de um local onde o robô possa recarregar a sua bateria de forma autónoma. A Figura 3.33 ilustra claramente a existência de um ponto de destino denominado "CP18", que corresponde precisamente a um posto de carregamento. Este ponto permite que o robô se desloque autonomamente até ele para realizar o carregamento da sua bateria. Por fim, o ponto de estacionamento que se refere a um local onde o robô possa estacionar e permanecer em repouso está identificado como "PP21" na Figura 3.34.

A Figura 3.35 fornece uma representação de alto nível da forma como é realizado o mapeamento e como os dados são transmitidos para a plataforma de visualização.

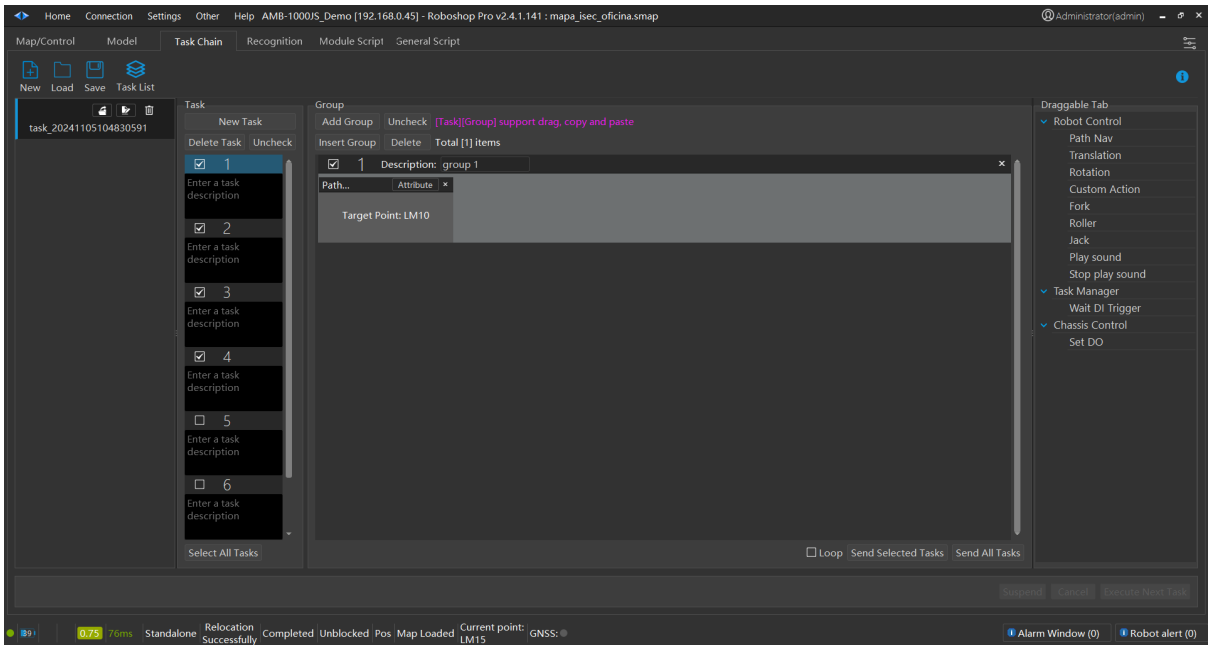


Figura 3.32: Tarefa efetuada no *Roboshop Pro*.

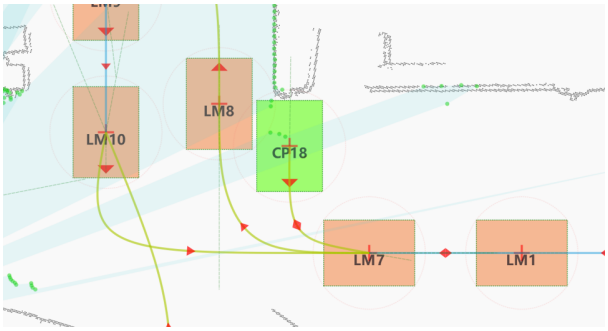


Figura 3.33: Ponto de carregamento do robô no *Roboshop Pro*.

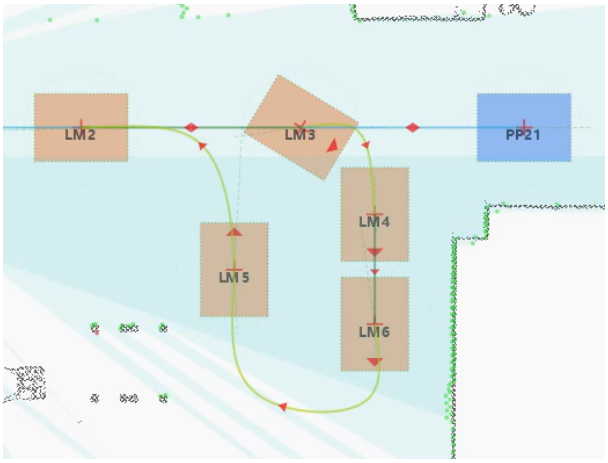


Figura 3.34: Ponto de estacionamento no *Roboshop Pro*.

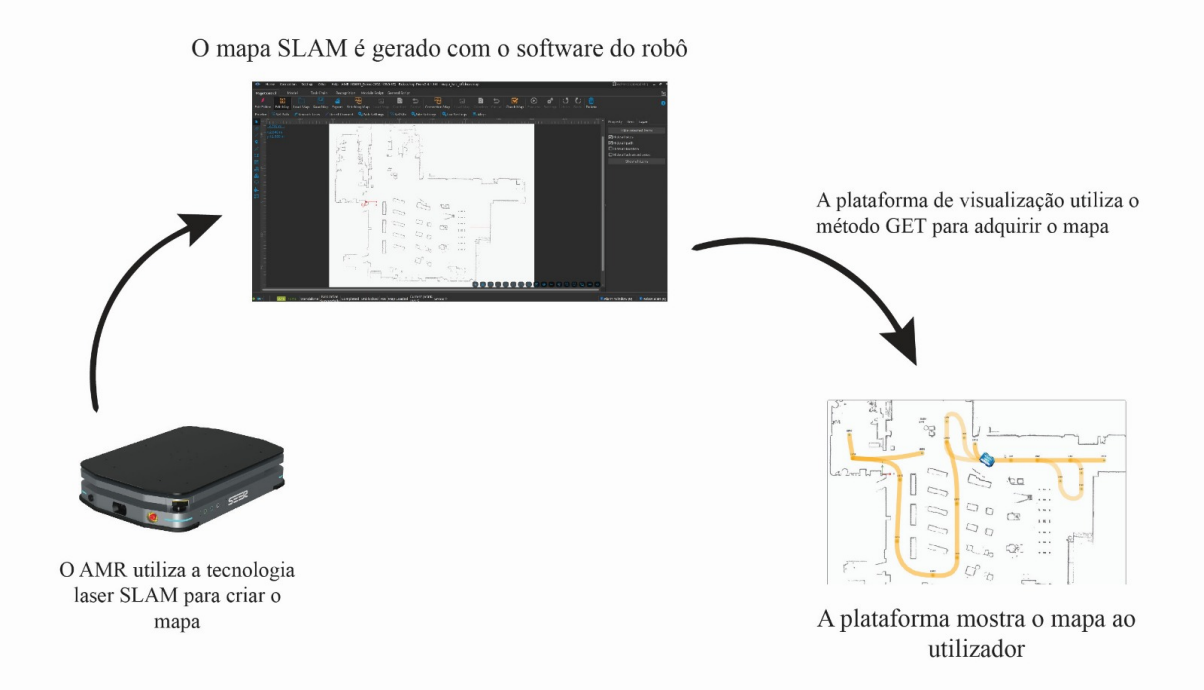


Figura 3.35: Representação de alto nível do mapeamento.

4 Sistema de aparafusamento com manipulador robótico colaborativo

Neste capítulo é apresentada a conceção e desenvolvimento das soluções de aparafusamento automatizado utilizando um manipulador robótico colaborativo. Tal como no capítulo anterior, é descrito um protótipo realizado nas instalações do ISEC.

4.1 Desenvolvimento do *setup* para o sistema de aparafusamento com cobot

Para a elaboração dos projetos 3D referentes às soluções de robótica colaborativa, tanto do aperto do *Cloison* como do *Porteur*, foi utilizado o *software* Fusion 360 que é um *software* de modulação 3D CAD/CAM/CAE desenvolvido pela Autodesk, sendo utilizado para design de produtos, engenharia mecânica e fabrico de diversos materiais. De entre as suas características, destacam-se as seguintes:

- Permite a criação de modelos e montagens 3D, utilizando uma vasta variedade de ferramentas intuitivas;
- Oferece recursos que permitem a simulação de movimento, tensões, temperatura, entre outros fatores, permitindo assim ajudar a entender os seus utilizadores a preverem o comportamento dos seus projetos antes do seu fabrico físico;
- Utiliza parâmetros e restrições para criar modelos que podem ser facilmente modificados e atualizados conforme pretendido;
- Tem a facilidade e comodidade dos utilizadores poderem compartilhar o seu trabalho, facilitando a colaboração em tempo real entre equipas, tornando assim os projetos mais eficientes.

Assim, face às necessidades inerentes às tarefas de aperto do *Cloison* e do *Porteur* foram projetados e desenhados os protótipos das demonstrações destas duas tarefas.

Depois de alguns estudos e ajustes sobre como seria a melhor solução mecânica para o desenvolvimento das demonstrações, ficou decidido que a solução mais prática e eficaz seria uma estrutura mecânica que albergasse as duas peças, o *Cloison* e o *Porteur*. Assim, a estrutura apresentada na Figura 4.1 satisfaz os critérios supramencionados.

A estrutura pode ser dividida em três partes distintas:

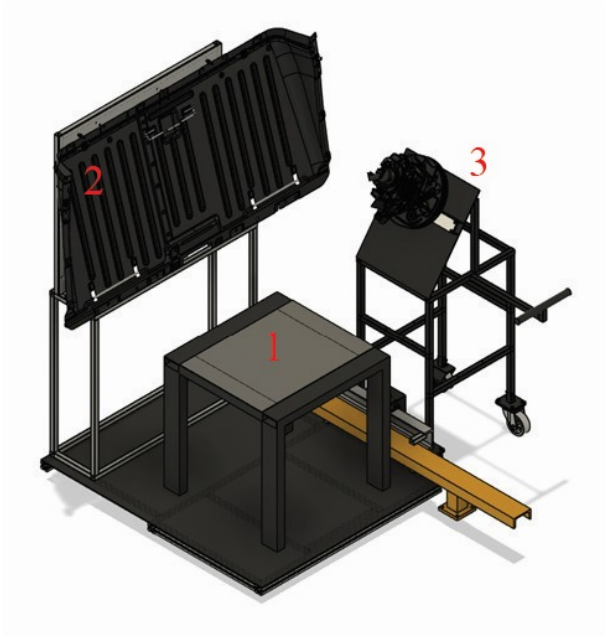


Figura 4.1: Estrutura desenvolvida para o protótipo do *Cloison* e *Porteur*.

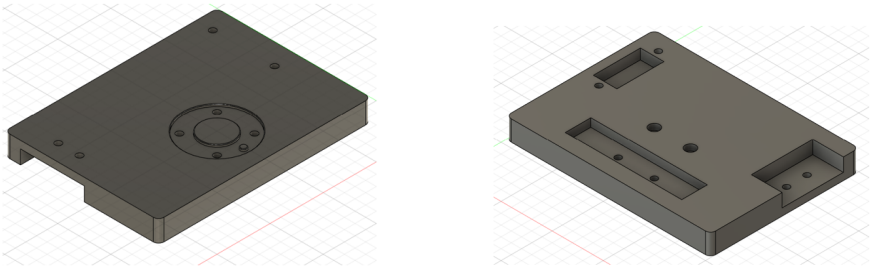
1. A Zona 1 contém o braço do robô Doosan M1013 que tem acoplado a aparafusadora Desoutter e a câmara de visão Sick TriSpector 1030, o respetivo controlador, bem como o controlador da aparafusadora;
2. A Zona 2 corresponde ao suporte para o *Cloison*;
3. A Zona 3 suporta o *Porteur*.

Assim sendo, já com o projeto da estrutura de suporte às peças alvo de intervenção concluído, foi elaborado um *gripper* para o robô.

Com as peças devidamente posicionadas na estrutura do protótipo e após a realização de vários testes, foi possível validar o melhor posicionamento do cobot e determinar o seu alcance relativamente às porcas e parafusos alvo do processo de aparafusamento. Com base nestes resultados, foi possível tomar uma decisão quanto ao tamanho do *gripper* a desenvolver, tendo em consideração a necessidade de incorporar tanto o sistema de visão (Sick Trispector 1030) como a aparafusadora. Ambos os componentes são essenciais para a localização precisa e a fixação eficaz das peças em questão. Desta forma, os principais objetivos definidos para o projeto do *gripper* foram:

- Fabricar o *gripper* recorrendo à impressão 3D, uma vez que esta solução é suficiente para utilização em ambientes não industriais e apresenta custos e exigências logísticas significativamente menores em comparação com outros materiais, como o metal;
- Desenhar uma estrutura simples, de fácil fabrico e integração;
- Adotar um conceito modular, permitindo uma impressão mais simples e alterações mais fáceis e rápidas.

Na Figura 4.2a é possível observar a vista isométrica dos encaixes no robô, enquanto que na Figura 4.2b observa-se a vista isométrica dos encaixes do sistema de visão e aparafusamento.



(a) (b)

Figura 4.2: Desenhos da base principal do *gripper*. (a) Vista isométrica do encaixe no robô; (b) Vista isométrica dos encaixes do sistema de visão e aparafusamento.

Perante os objetivos mencionados, o resultado foi um *gripper* impresso em 5 partes separadas.

Na base principal foi desenhado o espelho do encaixe do robô, o qual é posicionado por intermédio de um pino guia, sendo fixo ao robô por intermédio de dois parafusos que apertam a base diretamente ao robô. Os outros dois parafusos têm a função de apertar o apoio do sistema de visão à base principal. Para além disto, na base principal existem também duas caixas retangulares que têm como função fixar os apoios que servem para acoplar e posicionar a aparafusadora.

Na Figura 4.3 é possível observar a vista isométrica do apoio do sistema de visão presente no *gripper*. Este apoio contém na face superior o lugar dos parafusos que o atravessam e apertam na flange do robô. Existem ainda furos para colocar três parafusos, que apertam diretamente à câmara utilizada.

Na Figura 4.4 é possível visualizar o apoio à retaguarda que tem a função de suportar o peso da aparafusadora, contando com a flexão da abraçadeira aqui representada para manter a tensão sob a aparafusadora.

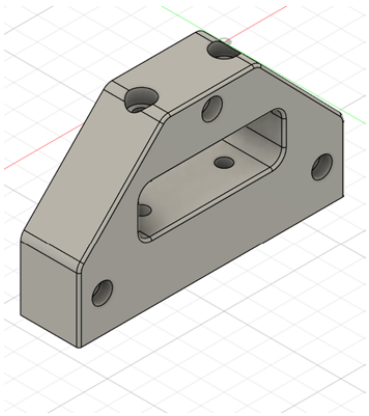


Figura 4.3: Vista isométrica do apoio do sistema de visão.

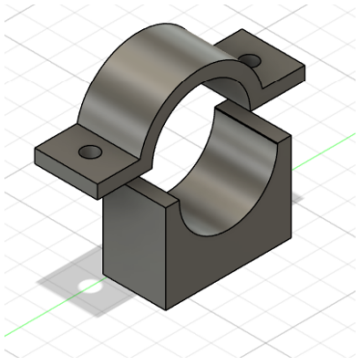


Figura 4.4: Vista isométrica do apoio à retaguarda.

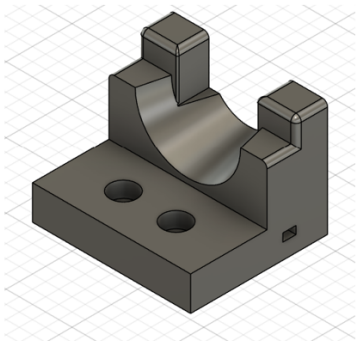


Figura 4.5: Vista isométrica do apoio frontal.

Por outro lado, o apoio frontal representando na Figura 4.5 apresenta uma geometria para encaixe da parte frontal da aparafusadora, com uma pequena passagem para uma abraçadeira de serrilha plástica, tendo o apoio frontal a função de absorver a torção criada durante o aperto dos parafusos.

Assim, conforme ilustrado na Figura 4.6, a solução idealizada foi desenvolvida com recurso ao *software* Fusion 360, consistindo no posicionamento da aparafusadora de um lado do *gripper* e da câmara do outro. Após várias iterações de design, concluiu-se que esta configuração seria a mais adequada, uma vez que permite minimizar o risco de colisão da ferramenta e/ou do robô com a estrutura durante as fases de aquisição de dados e aparafusamento. Simultaneamente, esta abordagem maximiza a operabilidade e o alcance do robô ao longo de todo o processo.

Depois de projetada a estrutura dos protótipos foi iniciado o desenvolvimento das simulações necessárias para o projeto, tendo sido utilizado o programa RoboDK que se trata de um *software* que simula sistemas robóticos por meio de programação *offline*, bastante flexível e generalista da área da robótica.

Trata-se de um programa bastante útil, na medida em que permite a programação *offline*, ou seja, a criação e simulação do projeto podem ser realizadas integralmente sem a necessidade de ligação direta ao robô físico. Adicionalmente, o software disponibiliza uma biblioteca extensa de robôs de diversas marcas — como ABB, Doosan Robotics,

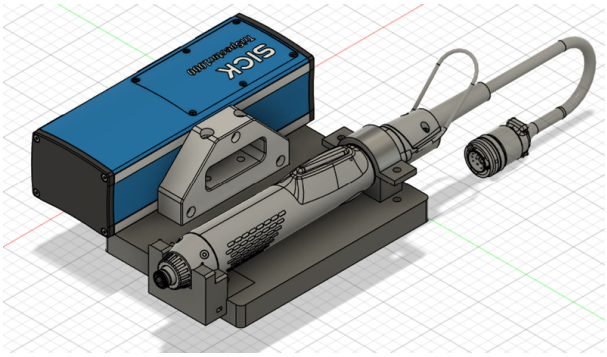


Figura 4.6: Desenho final do *gripper* completo com a aparafusadora e o sistema de visão.

Kawasaki, Yaskawa, Motoman, entre outras —, possibilitando a sua integração em simulações. É ainda possível utilizar estações pré-existentes em conjunto com o robô selecionado, bem como importar modelos CAD de estações e objetos desenvolvidos noutros programas. Neste caso, o modelo foi desenvolvido previamente no Fusion 360, conforme referido, e posteriormente importado para o RoboDK. Com a integração do robô escolhido (Doosan M1013), foi então possível elaborar o projeto pretendido.

Este *software* foi utilizado para analisar o comportamento dos cobots na linha de produção, permitindo realizar todos os testes necessários, nomeadamente a avaliação dos limites e alcances do braço robótico, a definição da posição ideal do robô no contexto do projeto, bem como a identificação de possíveis colisões e a otimização dos seus movimentos e trajetórias. O processo foi iniciado com a importação da estação previamente modelada. Em seguida, procedeu-se à seleção do robô a utilizar no projeto e à importação do *gripper* destinado à realização dos testes. Depois de posicionar o robô na estrutura pretendida e de acoplar o *gripper* desenhado ao robô, fazendo os ajustes necessários para o mesmo ficar na posição pretendida e com o TCP pretendido ficou-se com o projeto pronto para começar as simulações, conforme demonstra a Figura 4.7.

Observando a Figura 4.7 é possível verificar a Station Tree que é a lista de todos os componentes e objetos que estão presentes no projeto. Pode-se verificar também que o projeto 3D é constituído por toda a estrutura do projeto, bem como o robô utilizado e o respetivo referencial. Este projeto no RoboDK permitiu efetuar os testes preliminares do cobot, movendo o próprio robô para verificar se a sua posição era a mais adequada para alcançar todos os pontos necessários ao aparafusamento das porcas. Durante este processo, foram feitos os ajustes necessários na posição da sua base. Para a realização de uma simulação preliminar apenas utilizando o programa RoboDK, foram inseridas todas as posições de aproximação e de aperto, conforme se observa na Figura 4.8. De seguida, foi utilizada a ferramenta "Create Program" realizando um programa com todas as posições selecionadas. Ao executar este programa o robô executa os movimentos que foram inseridos. A partir daqui o programa será complementado com instruções de velocidade e aceleração desejadas.

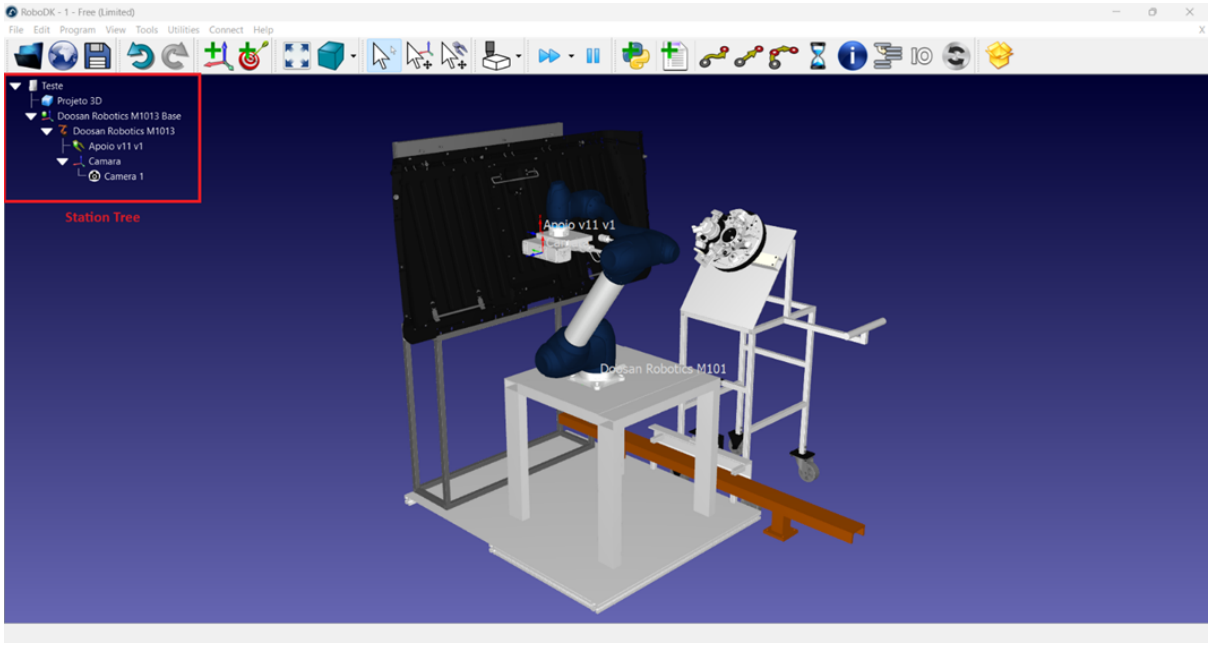


Figura 4.7: Protótipo no RoboDK.

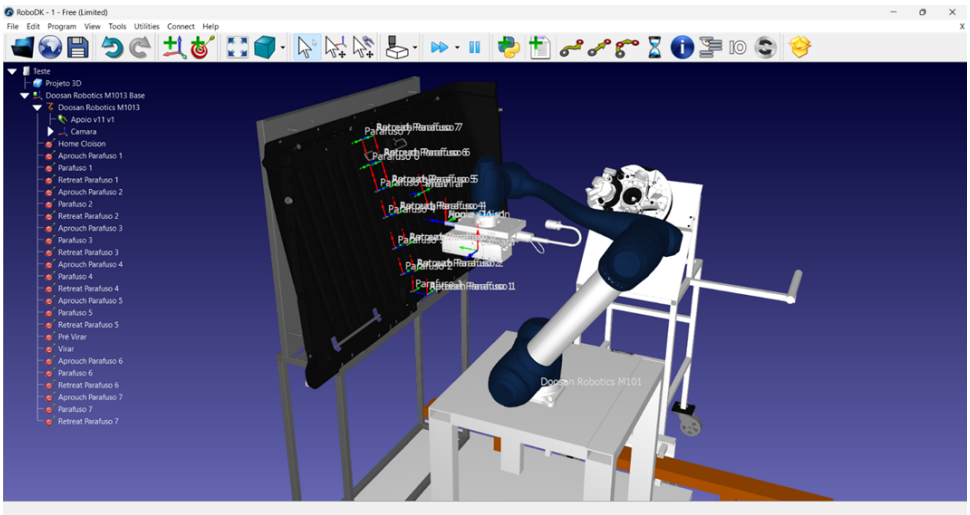


Figura 4.8: Posições de aperto no RoboDK.

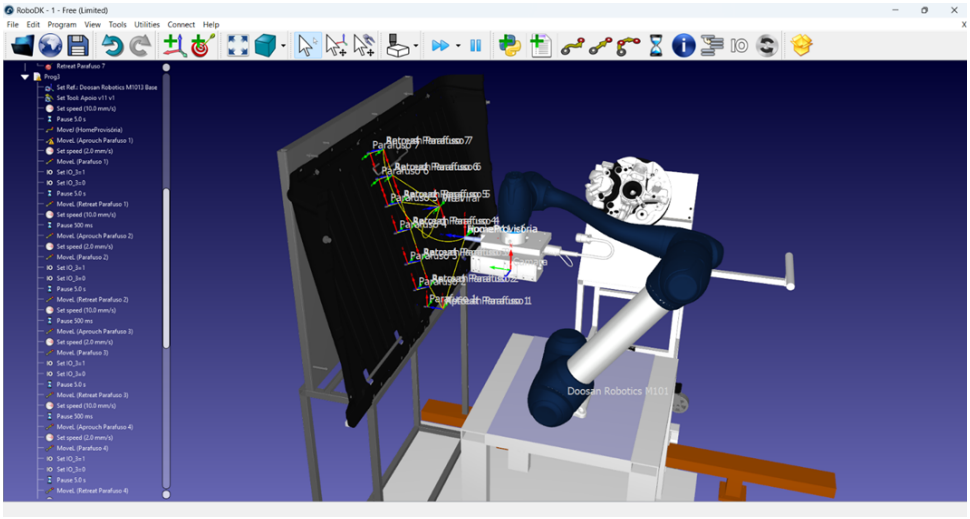


Figura 4.9: Simulação no RoboDK.

Figura 4.10: Estrutura mecânica do *Cloison*.

Na Figura 4.9 pode-se ver o programa desenvolvido, onde estão visíveis tanto os pontos utilizados quanto o trajeto do robô durante a simulação representado pelas linhas. Na simulação realizada nesta fase, a câmara não será incorporada; apenas o movimento do varrimento será simulado.

Após a realização de alguns testes na simulação e a determinação das posições ideais para a demonstração, iniciou-se a execução da parte mecânica do projeto do *Cloison*. Isto incluiu o corte dos perfis necessários e a subsequente montagem para completar a estrutura mecânica. Nesta fase, começou-se por montar o suporte para o robô e o seu controlador, bem como o suporte para o *Cloison* e a respetiva estrutura. As placas foram fixadas à estrutura, conforme indicado no projeto, permitindo um desvio de alguns centímetros para a esquerda ou para a direita. Após esta montagem, a estrutura mecânica ficou conforme ilustrado na Figura 4.10.

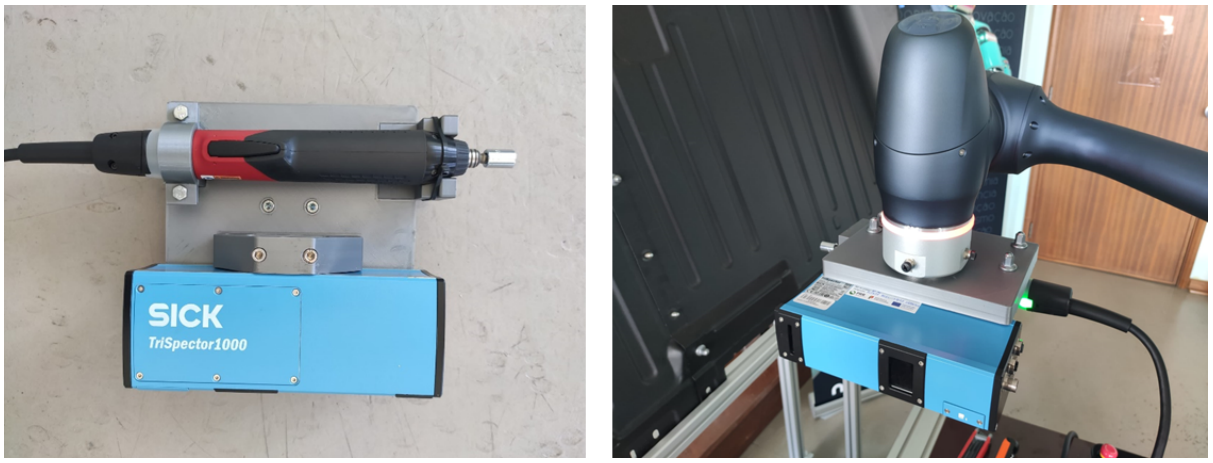


Figura 4.11: *Gripper* fora do robô e *gripper* acoplado ao robô.

De seguida, procedeu-se à impressão e montagem do *gripper*, que foi desenvolvido em ambiente 3D, conforme mencionado anteriormente. Este componente foi projetado para acoplar tanto a aparafusadora como a câmara 3D, com o intuito de fixá-los posteriormente no robô. Este conjunto será utilizado para a realização de testes em ambiente físico. Na Figura 4.11, é possível ver o *gripper* com a aparafusadora e a câmara já acopladas e instaladas no robô.

4.2 Desenvolvimentos preliminares da programação do aperto do *Cloison*

Uma vez concluída a estrutura mecânica destinada a acomodar todos os componentes — incluindo o robô, o respetivo controlador, e os suportes para acoplar a aparafusadora e a câmara —, o projeto encontrava-se em condições de avançar para a fase de desenvolvimento do *software*, bem como para a preparação de todas as condições necessárias a esse processo, conforme será descrito de seguida.

Inicialmente, procedeu-se ao estudo das técnicas de programação do robô. Após uma análise detalhada, identificaram-se duas alternativas viáveis para a programação deste robô, cada uma associada a diferentes programas. Um dos programas corresponde àquele que já vem integrado na interface do próprio *Teach Panel* do robô, mas que também permite a programação e configuração do robô via computador, proporcionando maior facilidade e agilidade no processo. Esta solução é especialmente recomendada para operadores inexperientes ou iniciantes na programação de cobots. O programa em questão é o *DART-Platform*, conforme ilustrado na Figura 4.12, e foi por meio dele que se iniciaram as configurações necessárias para o correto funcionamento do robô, permitindo, subsequentemente, a realização de testes com diferentes posições e movimentos.

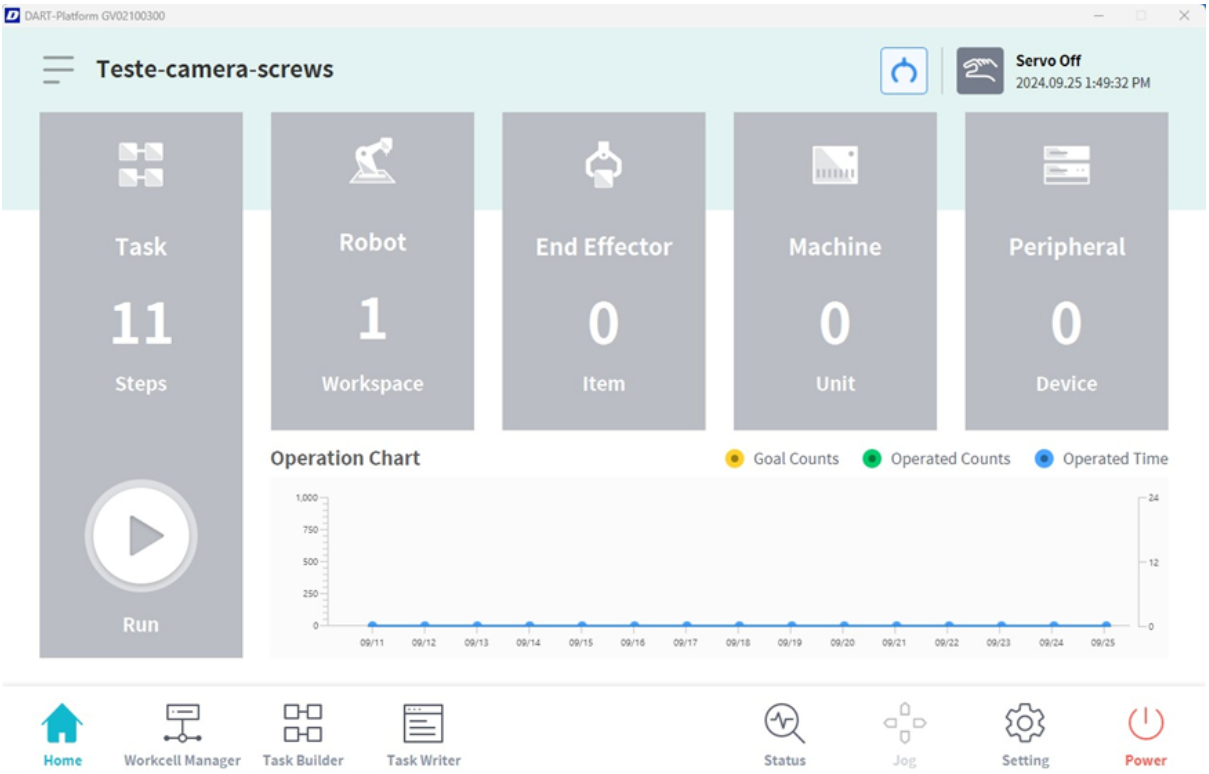


Figura 4.12: Página incial do programa *DART-Platform*.

Iniciou-se o processo com a verificação e configuração do robô, seguido pela criação de um programa simples de movimentos, em que o principal objetivo é a familiarização com as trajetórias e velocidades do equipamento. A partir deste programa, é possível ajustar diversos parâmetros do robô, tais como os endereços IP de comunicação, as zonas de segurança e colaborativas, além de verificar e definir os limites operacionais. Estes limites foram configurados de acordo com as necessidades específicas para a realização da demonstração. Em síntese, todas as configurações diretas do robô devem ser realizadas por meio do *DART-Platform*, como ilustrado na Figura 4.13.

Ainda que seja útil para testes iniciais e para uma programação simples do robô, esta aplicação apresenta limitações significativas em relação às necessidades do projeto, especialmente no que diz respeito à comunicação com sistemas externos, como a aparafusadora e a câmara, além do controlo de entradas e saídas (I/O), que são cruciais para a execução desta tarefa. Com base nestas limitações, optou-se pela utilização de um segundo programa, mais completo e flexível em termos de opções de programação do robô, o *DART-Studio*.

Ao contrário do *DART-Platform*, onde a programação é realizada por meio de blocos pré-configurados, nesta nova plataforma, a programação é efetuada diretamente através de código, proporcionando um controlo mais preciso tanto sobre a comunicação externa ao cobot quanto sobre os seus movimentos. Esta nova aplicação é o *DART-Studio* (Figura 4.14), um *software* pago que requer a aquisição de uma licença para a

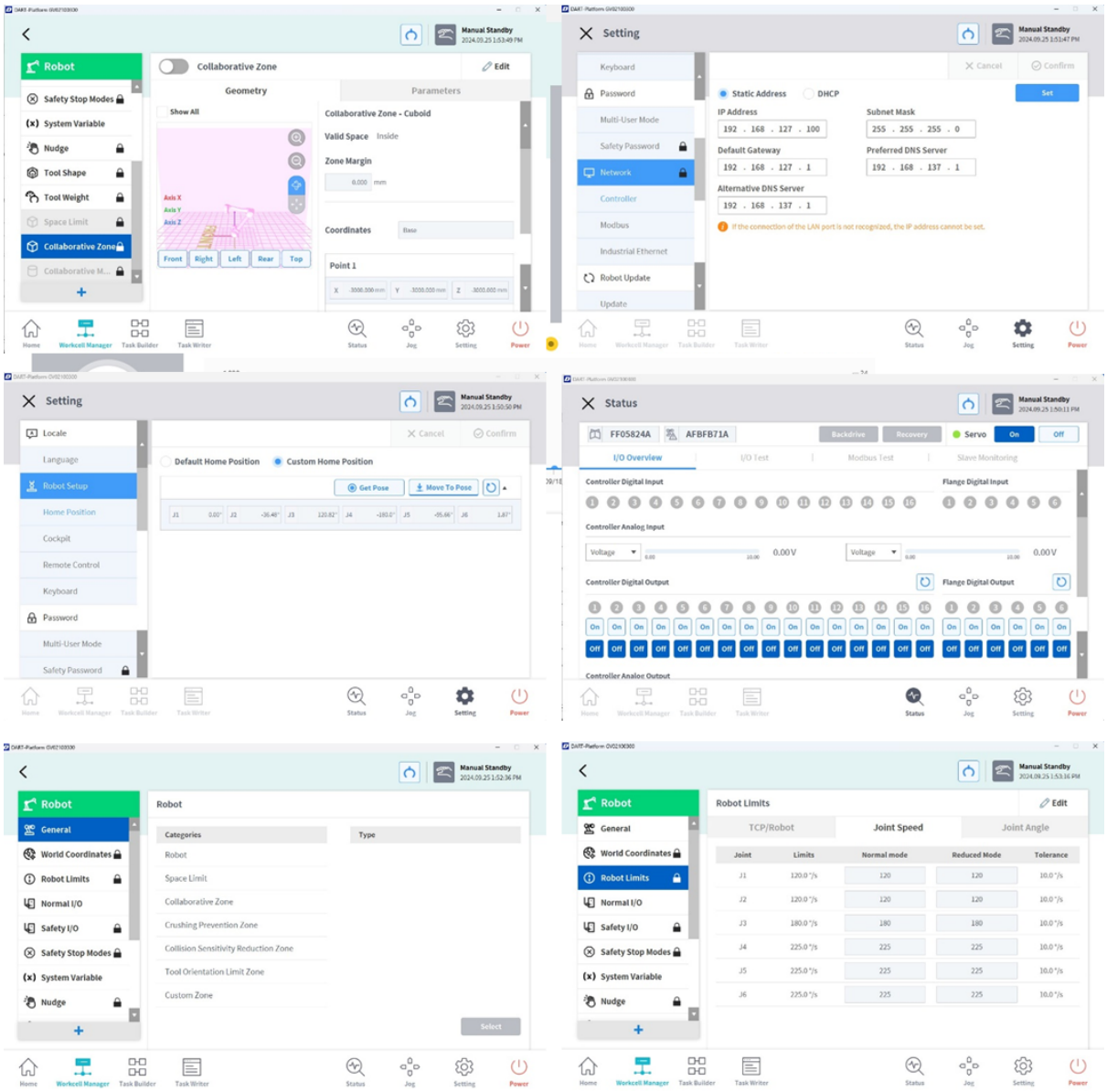


Figura 4.13: Configurações iniciais do robô no *DART-Platform*.

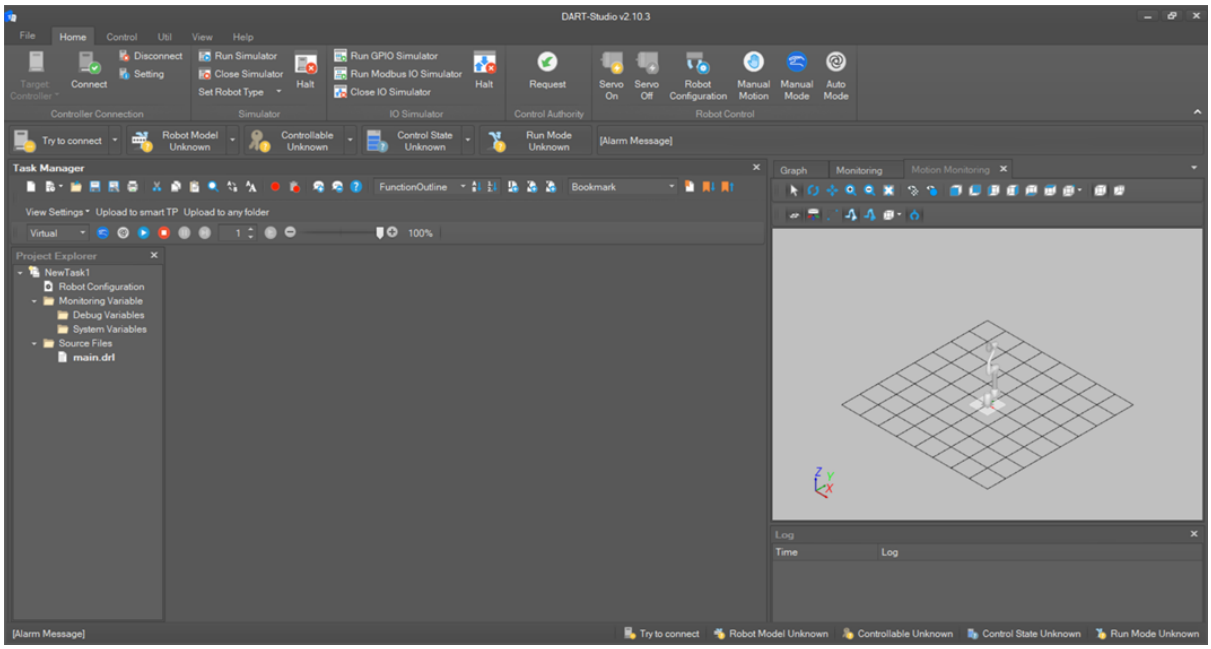


Figura 4.14: Página inicial *DART-Studio*.

sua utilização, ao contrário do *DART-Platform*, que é gratuito. O *DART-Studio* é amplamente utilizado em projetos deste tipo devido à sua capacidade de proporcionar controlo total sobre o robô e as suas comunicações externas.

Numa fase inicial, foram realizados testes preliminares com o robô, testes estes que se concentram exclusivamente na execução de movimentos. Foi desenvolvido um programa simples para simular o aperto da primeira porca (Figura 4.15). Trata-se de um código simples que simula o aperto da primeira porca, utilizando velocidades reduzidas devido à natureza inicial dos testes. As velocidades de movimentação do robô estão especificadas na área destacada com o número 1. Para a movimentação do robô, é necessário configurar seis parâmetros: X, Y, Z, RX, RY e RZ, que correspondem à posição desejada de destino do cobot. Para os movimentos articulados, utiliza-se o comando *'movej'*, enquanto o comando *'movel'* é utilizado para movimentos lineares, conforme indicado na área destacada com o número 2.

A simulação no *RoboDK* já apresentada, que gerou automaticamente um código na linguagem de programação do robô selecionado, permitindo a execução da tarefa simulada, permite também a elaboração de um código baseado na simulação contendo todas as posições necessárias para o aperto das porcas.

Esta simulação foi elaborada incluindo o processo de aperto da aparafusadora, simulando o mesmo com uma linha de código "$set\_digital\_output(3, 1)$". Esta linha instrui o robô a ativar a saída 3, que está conectada ao controlador da aparafusadora, enviando o sinal necessário para iniciar o programa de aperto da mesma.

Antes de se avançar para o processo de programação e integração do algoritmo da câmara de visão foi necessário configurar determinados parâmetros do robô, como a

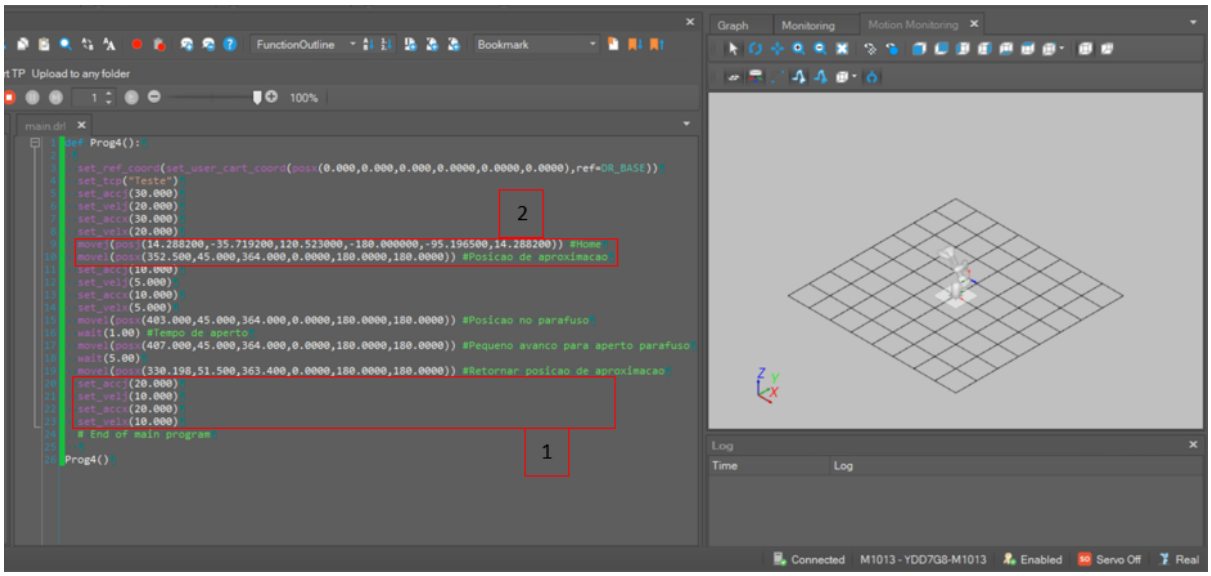


Figura 4.15: Programa simples para aperto de uma porca.

definição do *Tool Center Point* (TCP) e o peso do *gripper*, permitindo que o robô reconhecesse a presença do *gripper* na sua extremidade. Tanto o TCP quanto o peso do *gripper* influenciam diretamente os movimentos do robô. Além disso, foi essencial configurar o programa no controlador da aparafusadora para a mesma fazer o aperto dos parafusos quando dado o sinal para tal e para isso foi necessário também estabelecer as conexões das entradas e saídas do controlador do robô com o controlador da aparafusadora, de modo a garantir a comunicação eficaz entre os três sistemas.

Começou-se então por definir o TCP. Para isso o robô fornece 2 modos para a sua definição, sendo elas o modo manual e o modo automático, como é possível verificar na Figura 4.16.

Uma vez que o modo de cálculo automático apresenta um erro inferior, optou-se pela utilização deste método. Para este modo, é necessário registar 4 posições distintas no mesmo ponto, permitindo assim o cálculo do TCP do *gripper*, conforme ilustrado na Figura 4.17.

Para o registo destas 4 posições foi realizada a impressão 3D de um objeto em forma de cone, de modo a facilitar a obtenção das 4 posições distintas no mesmo ponto. Esta impressão pode ser observada na Figura 4.18.

Após a definição das quatro posições, o próprio sistema do robô faz o cálculo automático do TCP do *gripper* denominado de *Offset*, conforme ilustrado na Figura 4.19.

Relativamente à definição do peso do *gripper*, o robô possui um programa autónomo que calcula o peso exercido pelo *gripper*. Esta ferramenta foi utilizada para calcular o peso, conforme demonstrado na Figura 4.20.

Concluídos estes cálculos, procedeu-se à elaboração do programa para o processo de aparafusamento das porcas, o qual foi desenvolvido no controlador da aparafusadora,

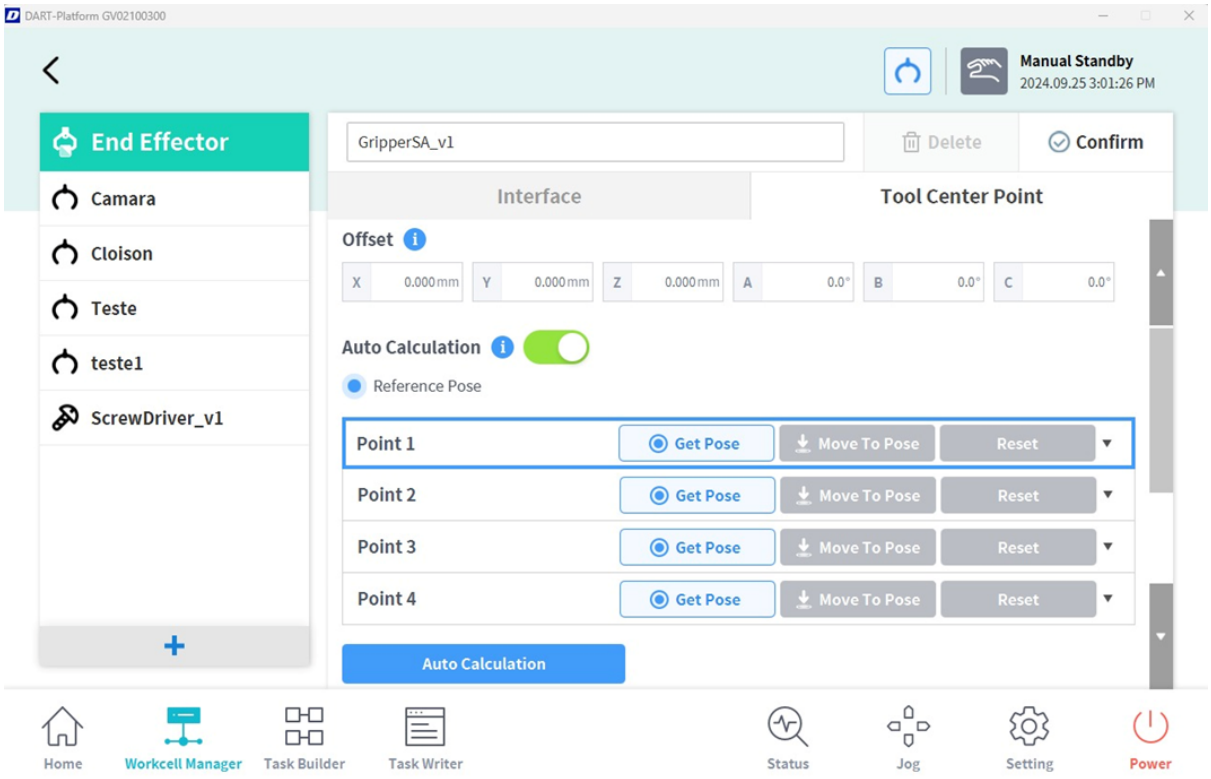


Figura 4.16: Definição do TCP.

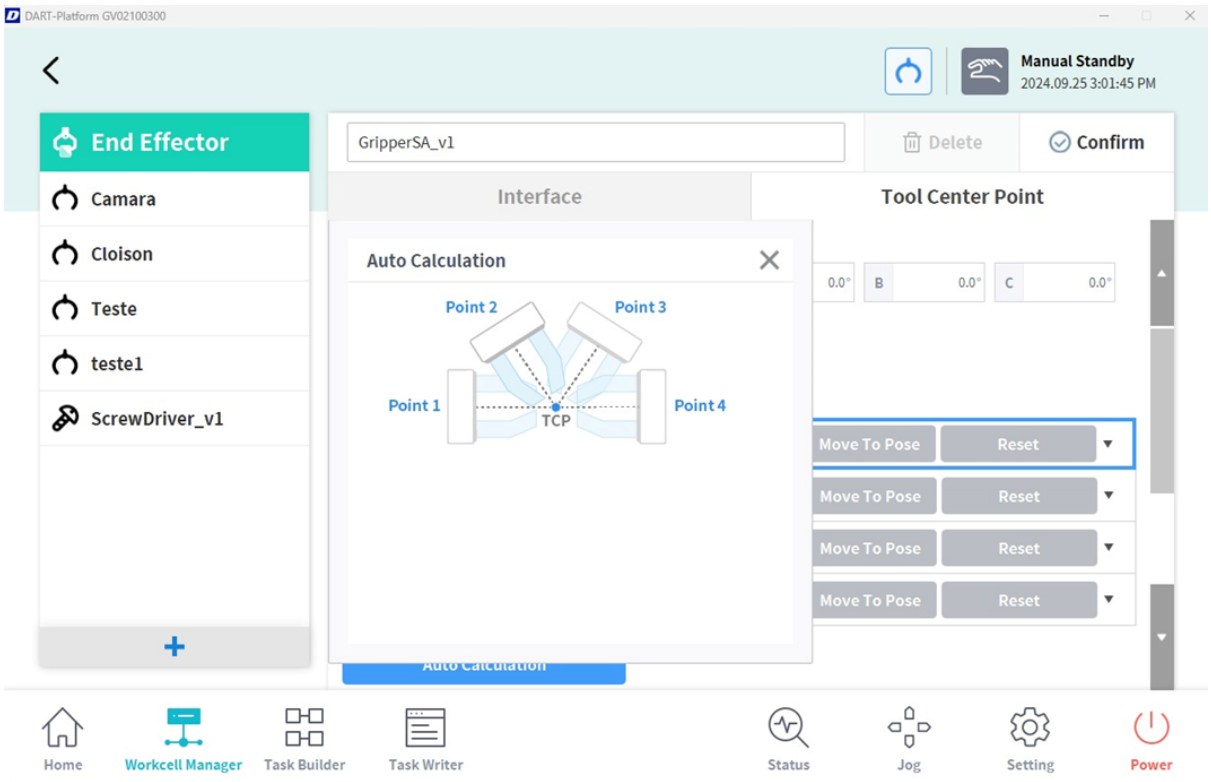


Figura 4.17: Registo de quatro posições distintas para cálculo do TCP.

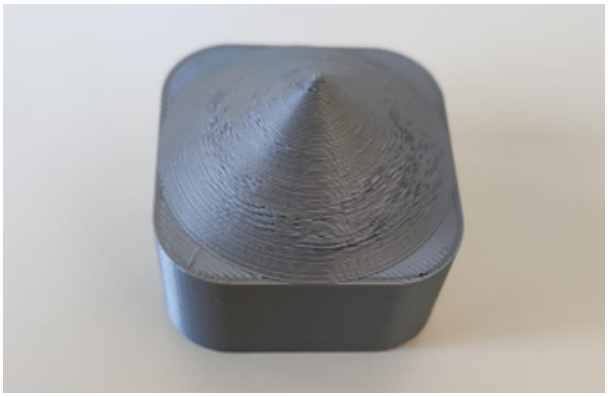


Figura 4.18: Cone impresso para obtenção das quatro posições.

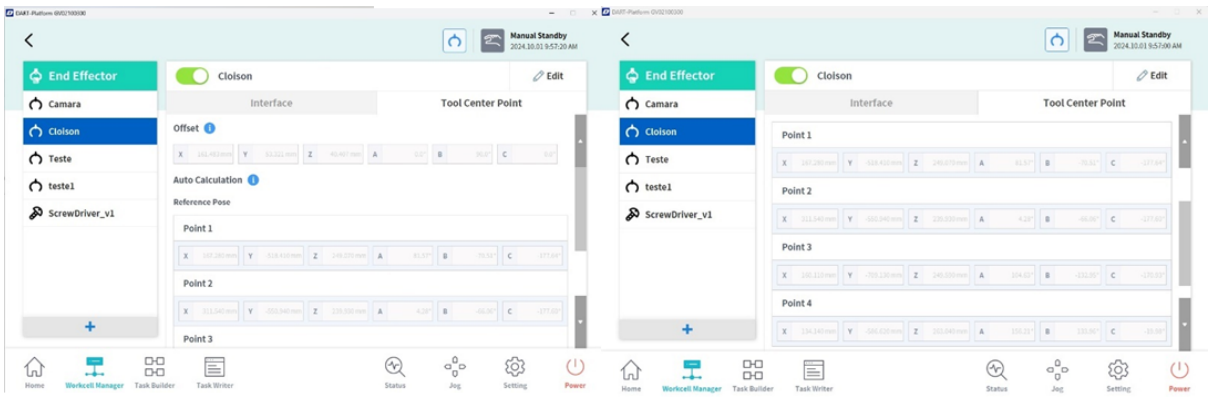


Figura 4.19: TCP do *gripper* calculado.

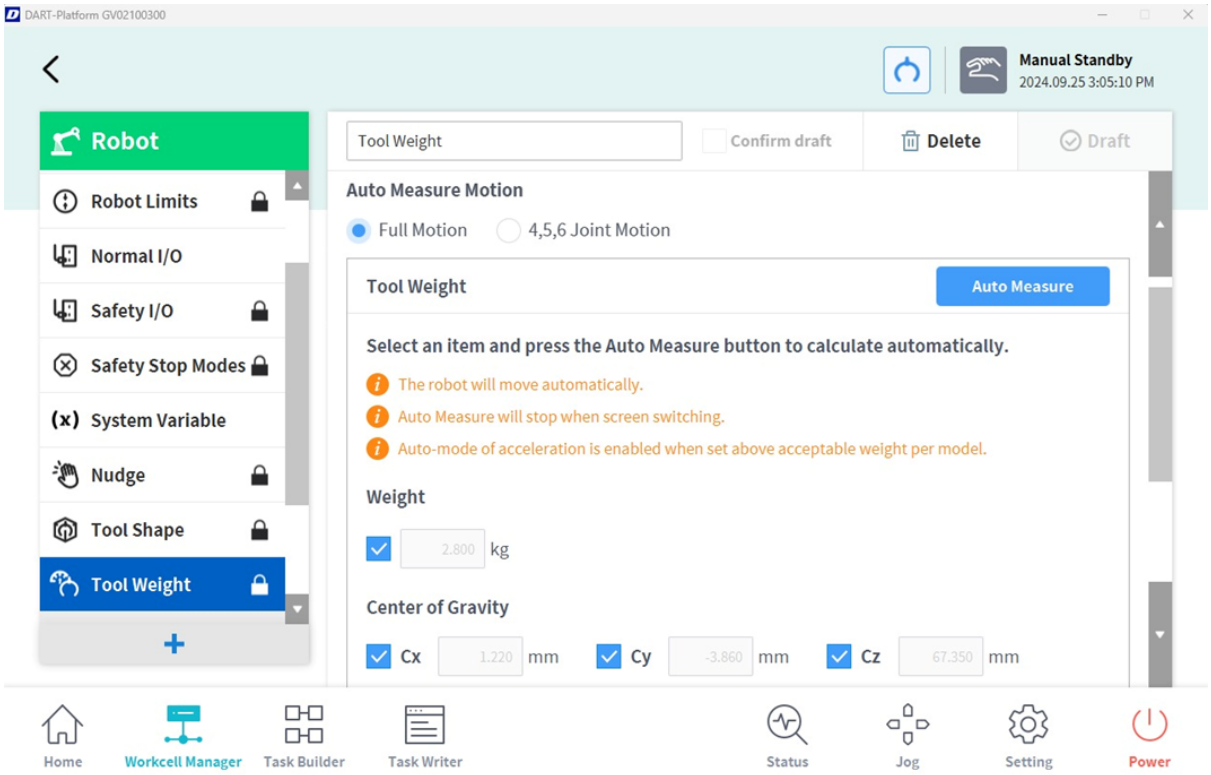


Figura 4.20: Programa para cálculo automático do peso do *gripper*.

Figura 4.21: Ecrã inicial do controlador da aparafusadora.

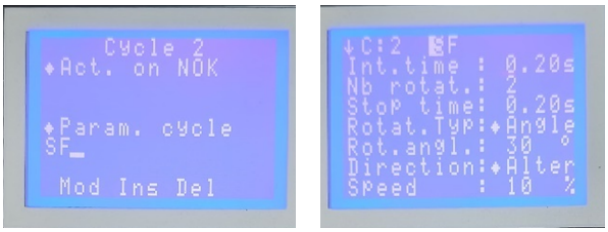


Figura 4.22: Fase denominada "*Search sequence*".

cujo ecrã de visualização inicial se encontra ilustrado na Figura 4.21.

Para a criação do ciclo de aparafusamento, começou-se pela fase denominada "*Search sequence*" ilustrada na Figura 4.22. Esta fase consiste na rotação lenta do bit da aparafusadora em ambos os sentidos, com o objetivo de encontrar a posição da porca, permitindo o encaixe correto para posterior aperto.

De seguida, a fase "*Final speed*" é responsável pelo aperto final da porca. Nesta etapa foi necessário definir os parâmetros de operação, tais como o tempo de aparafusamento e o toque pretendido. Todas estas métricas podem ser observadas na Figura 4.23.

Depois da programação dos ciclos da aparafusadora estarem concluídos, estabeleceram-se as conexões necessárias para permitir a comunicação entre o controlador do robô e a aparafusadora. No que diz respeito aos *inputs* e *outputs* do controlador, identificou-se a necessidade de um *input* (*Input* 6) que ativasse a aparafusadora. Durante os testes, concluiu-se também que seria necessário um *input* adicional (*Input* 8) para fazer o *reset* ao relatório enviado pela aparafusadora, uma vez que era necessário limpar o relatório entre cada aperto para evitar que os dados do parafuso anterior interferissem com o seguinte. Relativamente aos *Outputs*, era necessário obter relatórios sobre o sucesso de cada aperto. Para tal, utilizaram-se dois outputs pré-configurados: o *Output* 6, que

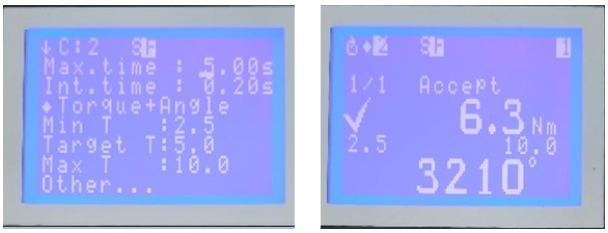


Figura 4.23: Fase denominada "*Final Speed*".

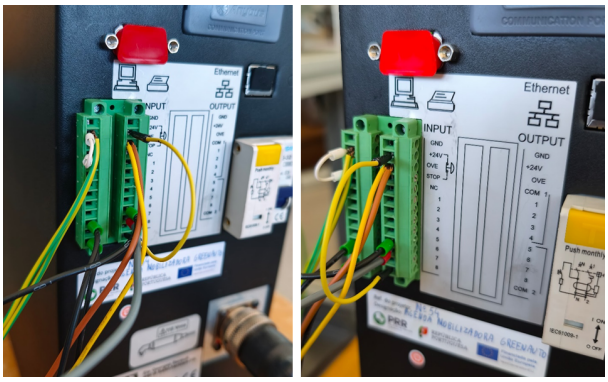


Figura 4.24: Ligações do controlador da aparafusadora.

Figura 4.25: Ligações do controlador do robô.

fica *'ON'* quando o relatório é positivo (OK), e o *Output* 7, que sinaliza uma falha no processo de aperto (NOK). Estas ligações estão ilustradas na Figura 4.24.

Relativamente ao controlador do robô, este irá receber dois *inputs* (*Input* 9 e 10), que correspondem aos relatórios sobre o processo de aperto da aparafusadora mencionados anteriormente. O controlador terá também três *outputs*: dois deles (*Output* 3 e 4) estão relacionados com a aparafusadora, sendo responsáveis pela sua ativação e pelo *reset*. O terceiro *output* (*Output* 13) terá a função de enviar o sinal para ativar a câmara 3D. As ligações do controlador do robô podem ser visualizadas na Figura 4.25.

Após a realização de alguns testes, verificou-se que o aperto das porcas não estava a ser efetuado da forma mais eficiente e adequada. Assim sendo, seguiu-se para a inclusão de dois recursos disponíveis no robô: as funções "*compliance*" e "*force*".

A função "*compliance*", quando ativada, permite desvios em qualquer um dos eixos X, Y, Z, RX, RY e RZ durante o movimento do robô, oferecendo um nível de rigidez semelhante ao comportamento de uma mola. Por outro lado, a função "*force*" permite especificar uma força num eixo ou eixos específicos, fazendo com que o robô se mova nesse eixo até atingir a força desejada.

A combinação destas duas funções permitiu alterar o modo de movimento do robô durante a fase de aparafusamento. O movimento, anteriormente baseado em trajetórias definidas entre as posições A e B, foi modificado para aplicar uma força no eixo Z, ali-

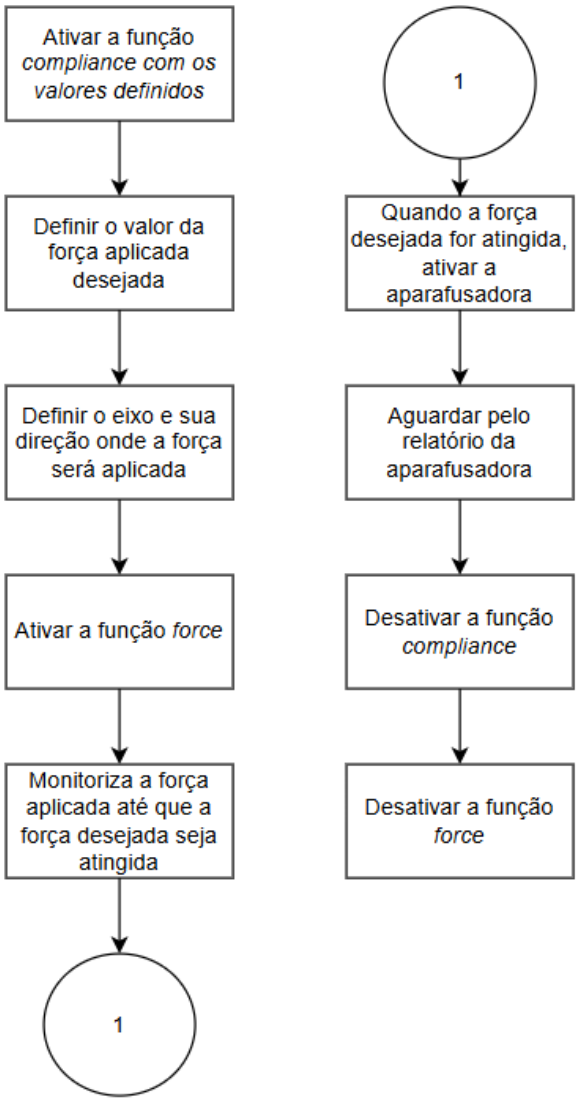


Figura 4.26: Fluxograma das funções "*compliance*" e "*force*".

nhado com o *gripper*. O "*compliance*" facilita o ajuste da posição da porca, assegurando a sua correta inserção no parafuso ao permitir pequenos ajustes automáticos, conforme necessário. Na Figura 4.26 pode ser observado um fluxograma exemplificativo da utilização destas duas funções.

Por fim, e antes da integração do algoritmo de visão 3D, explicitado na secção seguinte, é necessário implementar a leitura e a tomada de ações com base no relatório de aperto enviado pela aparafusadora, uma vez que o comportamento do robô terá que se ajustar dependendo se o relatório for positivo ou negativo. Para realizar esta leitura e permitir que o robô responda de forma adequada, foi desenvolvida uma função que aguarda a receção do relatório antes de acionar a ação desejada. O algoritmo de funcionamento desta função pode ser verificada na Figura 4.27.

O código completo e comentado, elaborado para o aperto do *Cloison*, encontra-se no Anexo C.

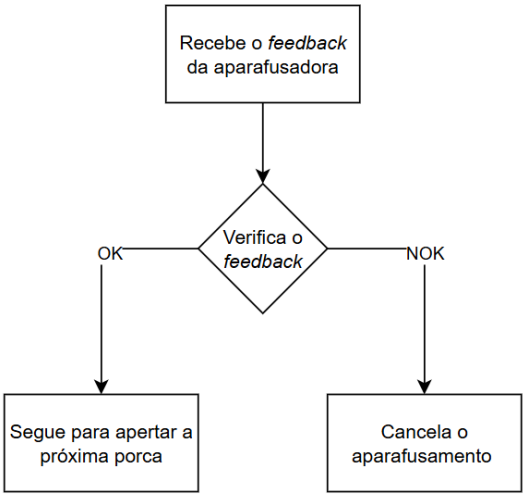


Figura 4.27: Algoritmo de funcionamento da função de espera do relatório da aparafusadora.

4.3 Descrição do sistema de aparafusamento da placa *Cloison*

Depois de toda a estrutura desenhada, preparada e montada, já estava tudo em condições de inicializar a programação de todos os dispositivos do sistema de aparafusamento: o controlador da aparafusadora, a câmara de visão 3D e, por último mas não menos importante, a programação do robô integrando as imagens capturadas, bem como a aparafusadora.

A Figura 4.10 mostra a disposição de todos os elementos, formando assim o sistema completo para a montagem dos parafusos na placa *Cloison*. O processo de montagem está dividido em 4 etapas, organizadas da seguinte forma:

1. **Calibração de uma referência entre o sistema de visão e a placa *Cloison*.**
   Nesta etapa é realizado um *scan* utilizando o sistema de visão através de um movimento linear do cobot. De seguida é definido manualmente a posição de cada parafuso de acordo com a referência dada pela câmara. Desta forma a calibração fica completa e o sistema fica apto para efetuar as futuras correções quando houver desvios da placa *Cloison*. É de salientar que este processo é realizado apenas uma vez.

2. **Deteção de uma nova referência**.
   Nesta etapa é inicializado o ciclo de trabalho do sistema onde primeiramente é realizado um *scan* com o sistema de visão. Após o *scan* realizado são retirado os pontos da nova referência.

3. **Cálculo e ajuste da posição do robô**.
   Nesta etapa é realizado um conjunto de cálculos de forma a ajustar a posição do

Figura 4.28: Estrutura mecânica *Cloison*.

robô para realizar o aperto dos parafusos. Desta maneira é comparada a referência do ponto 1 com a referência do ponto 2.

4. **Aperto das porcas**.
   Após o cálculo das posições de todos os parafusos com base nas referências é realizado o aperto das porcas nas posições corretas.

O *setup* de testes para o aparafusamento da placa divisória *Cloison* incorpora a seguinte arquitetura de elementos (que por sua vez, nas secções seguintes é feita uma abordagem mais aprofundada de cada elemento):

1. Estrutura mecânica que suporta o robô e a placa *Cloison*;
2. Robô colaborativo Doosan;
3. Sistema de visão Sick TriSpector;
4. Sistema de aparafusamento.

4.3.1 Estrutura mecânica que suporta o robô e a placa *Cloison*

A estrutura montada e representada na Figura 4.28 serve para simular a posição da chapa quando presente dentro do automóvel. Fixada a esta estrutura encontra-se a base do robô executada em perfil de alumínio. Como a base do robô encontra-se fixa à estrutura da chapa, para possibilitar os testes do sistema de visão, fixou-se a chapa de maneira que fosse possível alterar a sua posição lateralmente. Assim permite-nos ter uma estrutura de ensaios que reflete o processo da linha de montagem e habilita os testes para tornar todo o sistema robusto e confiável no seu resultado.

Tabela 4.1: Características do robô colaborativo [49].

Modelo do Robô Colaborativo	Doosan M1013
Peso	33 kg
Payload	10 kg
Alcance	1300 mm
Número de eixos	6
Repetibilidade	$\pm$ 0,1 mm ($\pm$ 0,05 mm)
Proteção (IP)	IP54
Velocidade Máxima	1 m/s
Tensão de alimentação	48V DC
Interfaces de comunicação	Ethernet, Modbus TCP/IP
Certificações	ISO 10218-1, ISO/TS 15066

4.3.2 Robô colaborativo Doosan

O robô utilizado neste caso de estudo é o Doosan M1013, um cobot desenvolvido pela Doosan. As suas especificações estão listadas na Tabela 4.1. Este robô colaborativo foi projetado para operar com segurança e eficiência ao lado de humanos em ambientes industriais. O robô possui 6 eixos e é conhecido por sua flexibilidade e facilidade de programação, permitindo que seja utilizado em diversas aplicações, como montagem, embalagem, inspeção de qualidade e manuseio de materiais.

4.3.3 Sistema de visão Sick TriSpector

O SICK Trispector 1030 é um sistema de visão 3D compacto e robusto projetado para aplicações industriais que requerem inspeção precisa e automatizada. Este sensor de visão é capaz de capturar e processar imagens tridimensionais de objetos em movimento, o que o torna ideal para uma variedade de tarefas, como controlo de qualidade, medição de volume, deteção de presença e posicionamento de peças. Na Tabela 4.2 encontram-se as especificações do equipamento.

4.3.4 Sistema de aparafusamento

O sistema de aparafusamento utilizado é composto por dois elementos principais: o controlador e a aparafusadora. Para o aperto das porcas, foi utilizada a aparafusadora linear Desoutter ECS10-M20, em conjunto com o controlador Desoutter CVICII-H2.

O controlador Desoutter CVICII-H2 é um dispositivo de alto desempenho, amplamente utilizado em linhas de produção industrial, nomeadamente no setor automóvel. Este equipamento permite a monitorização em tempo real do processo de aparafusamento, assegurando elevados níveis de qualidade e consistência, além de facilitar a integração com outros sistemas de automação. A sua interface intuitiva permite realizar ajustes de forma rápida e eficiente, adaptando-se facilmente às exigências da

Tabela 4.2: Características do sistema de visão [3].

Modelo do sensor de visão	Sick TriSpector 1030
Tipo de Sensor	Sensor de Visão 3D
Método de Medição	Triangulação laser 3D
Distância de trabalho	141 a 541 mm
Interface de trabalho	SOPAS
Interfaces de comunicação	Gigabit Ethernet (TCP/IP), serial (RS-232), I/O digitais configuráveis
Proteção (IP)	IP67
Peso	1300 g
Dimensões	217 mm x 62 mm x 84 mm
Erro (Perto/Longe)	40/280 µm
Velocidade máxima	2000 3D perfis/s
Número máximo de perfis	2500 por imagem
Resolução perfis 3D	0,215 mm/px

produção. As principais especificações deste controlador encontram-se apresentadas na Tabela 1.5.

Por sua vez, a aparafusadora Desoutter ECS10-M20, de funcionamento linear, é utilizada em conjunto com o referido controlador. As suas características técnicas estão detalhadas na Tabela 1.4.

4.4 Método proposto para o aparafusamento da placa *Cloison*

Esta secção descreve em detalhe o método de calibração de referência, a deteção de uma nova referência, o cálculo da posição do robô e o aperto das porcas.

4.4.1 Calibração de referência

O processo de calibração é realizado apenas uma vez, ou seja, só é feito no momento da instalação do robô no seu local de trabalho assim como os seus respetivos componentes. A calibração de uma nova referência inclui principalmente um sistema de coordenadas da base do robô B, um sistema de coordenadas do sistema de visão C e um sistema de coordenadas da placa *Cloison* P. Cada sistema de coordenadas é composto por um eixo X, um eixo Y e um eixo Z, conforme se pode observar na Figura 4.29.

Após a definição dos sistemas de coordenadas, a obtenção da imagem de referência é realizada com o auxílio do *software* SOPAS *Engineering Tool*, da SICK. Este software, desenvolvido pela própria SICK, destina-se à configuração, parametrização e diagnóstico de sensores e sistemas automatizados na área da sensorização. Amplamente utilizado

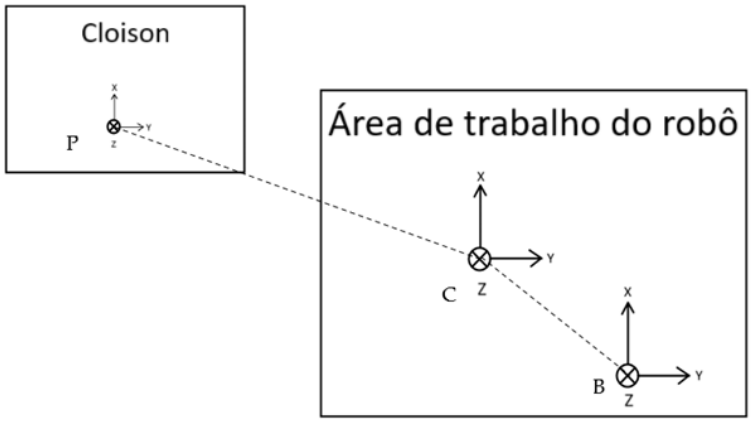


Figura 4.29: Sistemas de coordenadas do robô B, sistema de visão C e placa *Cloison* P.

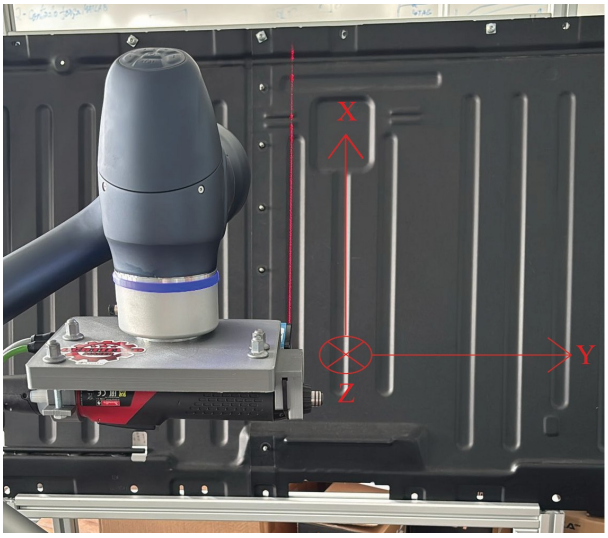


Figura 4.30: *Scan* realizado pelo sistema de visão ao longo do eixo Y.

em ambientes industriais, o SOPAS oferece uma interface gráfica intuitiva que facilita a ligação e configuração de diversos dispositivos da marca, como sensores de visão, sensores LiDAR e sistemas de segurança, mesmo sem a necessidade de ligação física direta.

Para a aquisição da referência, o robô executa um movimento ao longo do eixo Y — isto é, um deslocamento da esquerda para a direita — enquanto a câmara realiza a aquisição de dados, como ilustrado na Figura 4.30. Esta aquisição é desencadeada por um sinal de *trigger* enviado pelo robô e recebido pela câmara. Apenas após a captura da imagem de referência, esta etapa é considerada concluída.

Após a imagem de referência ser adquirida, e para saber as eventuais deslocações da placa *Cloison*, foram definidos como referência a posição dos parafusos nº 3 e nº 4, tal como se pode ver na Figura 4.31.

De seguida efetua-se o tratamento dos dados obtidos na imagem de referência tendo-

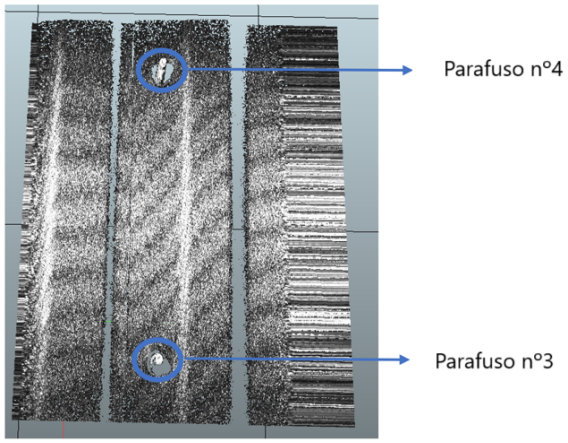


Figura 4.31: Parafusos que servem de referência de posição para o sistema de visão.

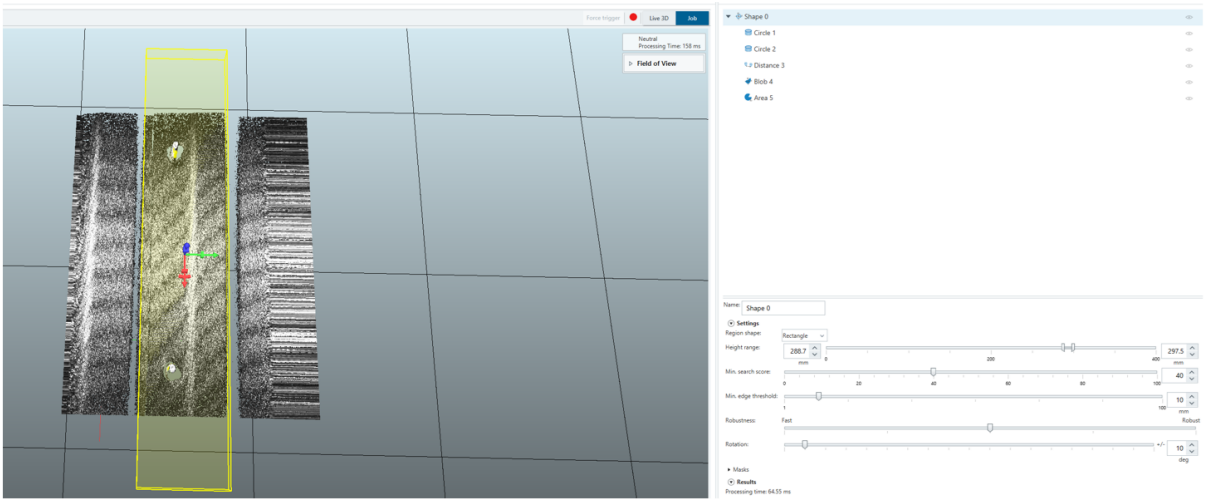


Figura 4.32: Ferramenta *Shape* utilizada.

se utilizado uma função para obter o *Shape* dentro de uma região de interesse (conforme mostra a Figura 4.32). Esta zona de interesse foi previamente definida com dimensões adequadas para recolher todas as informações necessárias à aplicação, garantindo simultaneamente que o sistema opera com rapidez suficiente para executar a tarefa de forma eficiente. A função *Shape* localiza uma forma de referência na imagem e reposiciona-a com outras funções de acordo com a sua orientação. Normalmente utiliza-se esta função quando a posição ou a rotação do objeto pode variar, o que é o caso deste cenário de teste.

Após a obtenção da forma do objeto, é possível associar ao *Shape* uma função adicional, designada por *Area* (conforme ilustrado na Figura 4.33). A função *Area* calcula a área da forma, correspondendo à superfície do *Cloison*, através da contagem dos pontos presentes numa região 3D definida ou dentro de um intervalo de intensidade específico. Esta funcionalidade permite verificar se a área de pontos adquirida corresponde, de facto, à região desejada, garantindo a precisão e fiabilidade da medição.

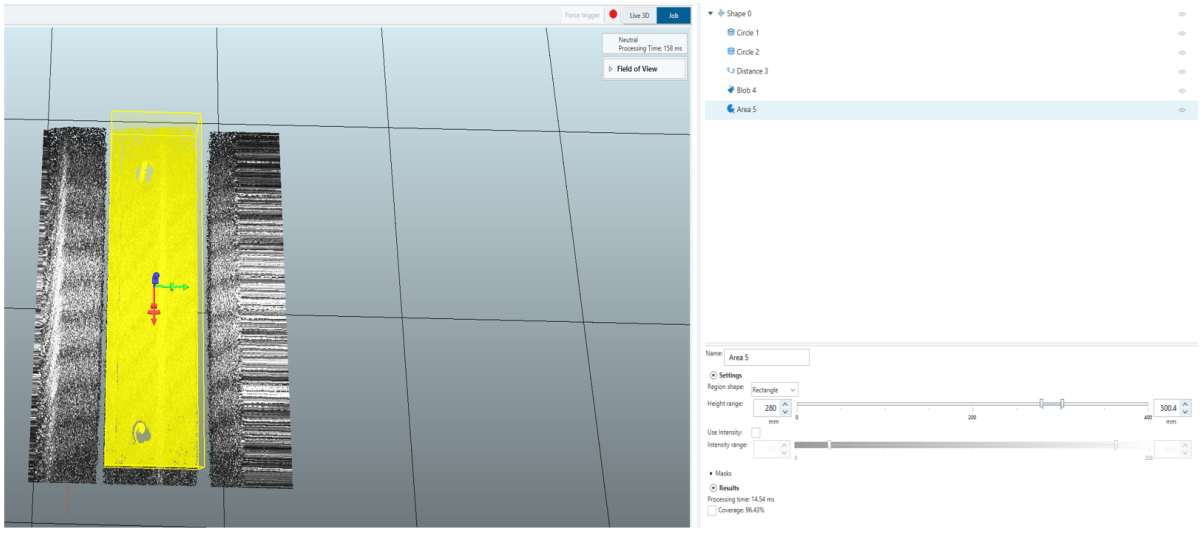


Figura 4.33: Ferramenta *Area* utilizada.

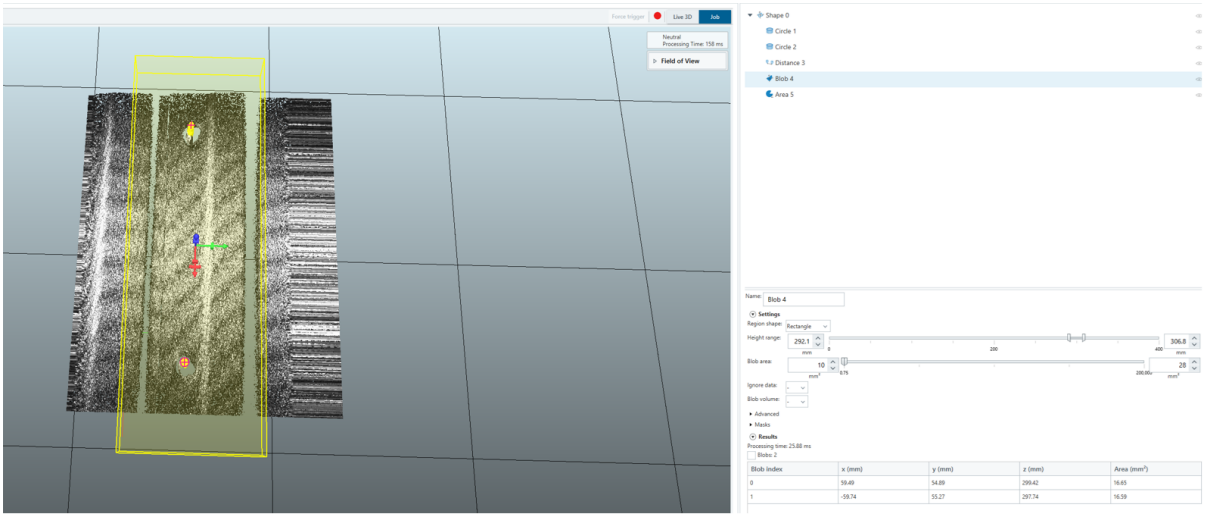


Figura 4.34: Ferramenta *Blob* utilizada.

A última etapa é a utilização de uma função designada de *Blob* (como se pode observar na Figura 4.34). Esta função *Blob* localiza aglomerados de pontos dentro de um intervalo de altura definido e calcula o tamanho de *cluster* específico. Assim, permite medir o volume, a área, o ângulo e a caixa delimitadora da imagem conforme se mostra na Figura 4.34. Desta forma esta função foi utilizada para detetar a posição dos parafusos em relação ao referencial da câmara, assim como o número de *blobs* detetados na região de interesse. A função *Blob* foi configurada de forma a criar *clusters* em volta de áreas compreendidas entre 10 mm^2 e 28 mm^2. Os resultados da função são as posições de cada *cluster* encontrado (sendo que cada *cluster* é a posição de um parafuso) nos eixos X, Y e Z assim como a área de cada *cluster*, ou seja, a dimensão de cada parafuso.

Por fim, o último passo da calibração, consiste em definir manualmente as posições de todos os parafusos no referencial da base do robô em relação à aparafusadora, para realizar o aperto. Desta forma, as posições ficam referenciadas conforme se mostra na

Tabela 4.3: Posições dos parafusos no referencial da base do robô em relação à aparafusadora.

	Eixo X	Eixo Y	Eixo Z	RX	RY	RZ
Parafuso 1	572,521	0,000	340,500	0,000	90,000	180,000
Parafuso 2	607,594	6,000	413,500	0,000	106,000	180,000
Parafuso 3	633,000	6,000	515,000	0,000	106,000	180,000
Parafuso 4	670,500	6,000	627,500	0,000	106,000	180,000
Parafuso 5	699,500	6,000	724,000	0,000	106,000	180,000
Parafuso 6	727,500	1,500	825,000	180,000	-107,000	-180,000
Parafuso 7	758,000	1,500	912,500	180,000	-107,000	-180,000

Tabela 4.4: Posições de referência dadas pela ferramenta *Blob*.

Blob index	X (mm)	Y (mm)	Z (mm)
0	59,49	54,89	299,42
1	-59,74	55,27	297,74

Tabela 4.3, e são definidas para posteriormente aplicar o método de correção desenvolvido, que é abordado nas secções seguintes.

As posições do robô para o aperto dos parafusos foram definidas com base nas posições de referência fornecidas pela câmara, através da ferramenta *Blob*, conforme apresentado na Tabela 4.4. Estas posições servem como base para a correção da posição dos parafusos em veículos subsequentes, uma vez que cada veículo possui um *Cloison* distinto que necessita de ser aparafusado. A relação entre a posição do *Cloison* e o robô, durante cada ciclo de trabalho, será descrita com mais detalhe nas secções seguintes.

4.4.2 Deteção de uma nova referência

A etapa principal é designada como o ciclo de trabalho. Aqui o robô vai realizar um novo movimento linear ao longo do eixo Y de forma a que o sistema de visão possa realizar uma aquisição da superfície do *Cloison* para obter uma referência dos parafusos e posteriormente compará-la com a primeira referência. Após a aquisição obtêm-se uma nova nuvem de pontos, conforme se mostra na Figura 4.35.

Após ser realizada aquisição do *Cloison* são aplicadas as funções descritas na secção anterior. Por exemplo, para a figura acima obtiveram-se os seguintes parâmetros conforme se mostra na Tabela 4.5. Após a aquisição dos dados obtidos de cada função é realizado um processo que calcula a posição correta do robô em relação aos parafusos de referência.

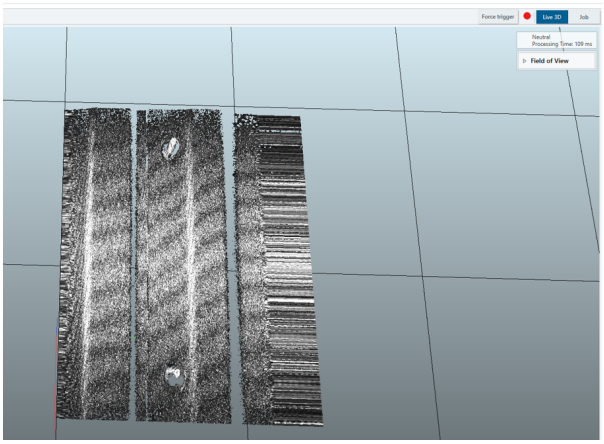


Figura 4.35: Nova nuvem de pontos dada pelo *scan*.

Tabela 4.5: Parâmetros obtidos para a Figura 4.35.

	Informação
Área	94,63 %
Quantidade	2 unidades
Posição	Blob 1 (mm) : X: 59,66, Y: 56,23, Z: 300,2 Blob 2 (mm): X: -59,41, Y: 56,54, Z: 298,72

4.4.3 Cálculo e ajuste da posição do robô

O próximo passo consiste no cálculo da posição dos parafusos, com base nos dados fornecidos pelo sistema de visão. Os dados são enviados do sistema de visão através do *software SOPAS Engineering Tool* para o *software Dart Studio*, via *sockets*, conforme ilustra a Figura 4.36.

Os dados enviados via *socket* consistem numa trama (conforme mostra a Figura 4.37) com informações sobre os dados adquiridos pelo sistema de visão. A trama contém dados sobre a área de aquisição, o número de blobs detetados e a posição de cada blob.

Após a receção dos dados é realizado o processo de ajuste de posição. O ajuste da

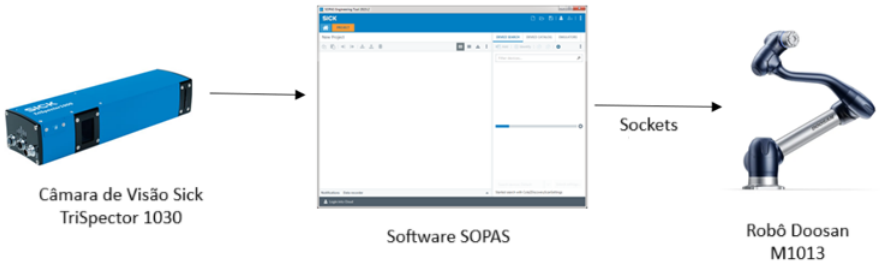


Figura 4.36: Representação do fluxo de dados.

(%)	(Unidades)	(x, y, z)	(x, y, z)
Área	Número de Blobs	Posição do Blob 1	Posição do Blob 2

Figura 4.37: Estrutura da trama de dados.

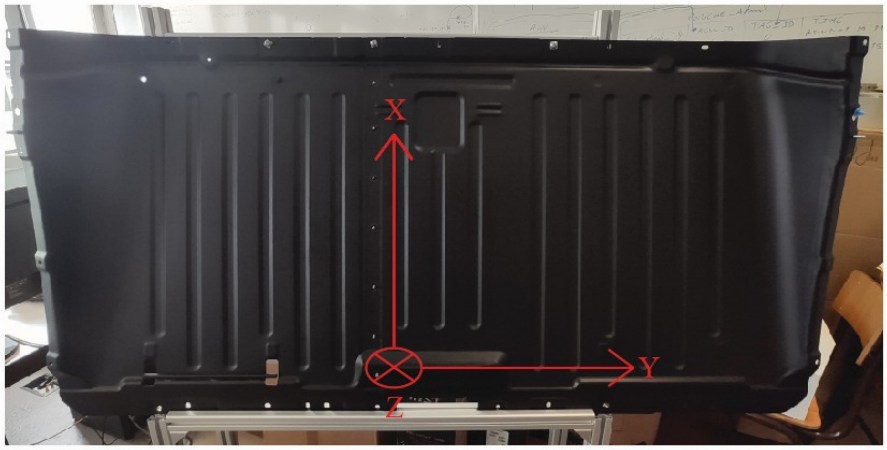


Figura 4.38: Sistema de eixos na placa *Cloison*.

posição é baseado na verificação do desvio da posição dos parafusos em cada eixo. Neste caso de estudo os desvios ocorrem principalmente no eixo Y, conforme ilustra a Figura 4.38.

O cálculo da cinemática inversa só é efetuado após a obtenção do deslocamento que é a diferença entre os dados obtidos pela câmara (proveniente da trama de dados) e os dados definidos na calibração de referência:

$$Dif_y = Data_y - DataCalibrationReference_y \tag{4.1}$$

Porém a variável Dif_y representa apenas a variação entre os dados atuais fornecidos pelo sistema de visão e os dados de referência, que correspondem à posição previamente definida da câmara no sistema de referência global. No entanto, essa variação não reflete com precisão a variação real no referencial da base do robô. Desta forma, é possível aferir se os eixos do sistema estão ou não alinhados, de seguida adicionou-se um *offset* no respetivo eixo. Esse *offset* foi estimado através do cálculo de uma interpolação linear inversa. A interpolação linear inversa é usada para obter o valor de *offset* com base na diferença Dif_y, e a fórmula segue o princípio de interpolação linear entre os pontos que foram medidos. Um deles foi medido diretamente na chapa *Cloison* e essa variação corresponde ao ponto inicial de calibração. O segundo ponto foi obtido a partir de medições adicionais durante a operação, permitindo um mapeamento preciso da variação do eixo Y.

A interpolação linear inversa foi implementada da seguinte forma: os dois pontos de

referência (x_1,y_1) e (x_2,y_2) foram utilizados para criar uma relação entre a variação observada nos dados da câmara e a variação real da posição real do robô. Estes pontos foram medidos com precisão, sendo o primeiro ponto definido durante a calibração inicial e o segundo adquirido após uma série de medições durante o processo de ajuste. A equação usada para calcular o *offset* y_{offset} foi:

$$y_{offset} = y_1 + \frac{(Dif_y - x_1)}{(x_2 - x_1)} * (y_2 - y_1) \tag{4.2}$$

Nesta fórmula, y_1 e y_2 representam as posições de referência conhecidas, enquanto x_1 e x_2 representam as variações observadas nesses mesmos pontos. A variável Dif_y, é a diferença entre a posição medida pela câmara e a posição definida durante a calibração de referência. Para garantir a estabilidade numérica, esta fórmula pressupõe que x_1 é diferente de x_2.

A interpolação linear inversa permite, assim, ajustar com precisão a posição real do parafuso no eixo Y, compensando quaisquer desvios detetados na medição. Ao aplicar este método, foi possível calcular o deslocamento exato do robô para que ele possa realizar a tarefa de fixação das porcas nos parafusos, independentemente de pequenas variações no posicionamento detetadas pela chapa do *Cloison*.

Finalmente apenas é adicionado o valor do *offset* na posição do robô para compensar os desvios que possam ocorrer no sistema.

Deve-se notar que o mesmo processo pode ser aplicado aos eixos X e Z, caso existam discrepâncias entre a posição esperada e a posição medida pela câmara. Assim, com base no algoritmo proposto, o sistema fornece as posições X, Y e Z, juntamente com os respetivos *offsets* a aplicar em cada eixo (se aplicável). Desta forma, é possível calcular a posição desejada para o aperto Px_{offset}, Py_{offset} e Pz_{offset}.

A cinemática inversa é utilizada para calcular os ângulos das juntas necessários para que o robô alcance a posição ajustada ao espaço cartesiano, garantindo que o *gripper* se alinhe corretamente com o ponto desejado. Esta abordagem baseia-se na decomposição da transformação homogénea T_{06}, definida como:

$$T_{06} = \begin{bmatrix} R_{3\times3} & P_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3}$$

Nesta matriz, R representa a rotação e P a posição final do *gripper*. Ao expandir os componente de R e P, a matriz T_{06} assume a seguinte forma:

$$T_{06} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_{X_{\text{offset}}} \\ r_{21} & r_{22} & r_{23} & P_{Y_{\text{offset}}} \\ r_{31} & r_{32} & r_{33} & P_{Z_{\text{offset}}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Com base nesta matriz transformada, o robô executa a trajetória correspondente, posicionando a aparafusadora no local correto para aparafusar as porcas com precisão.

4.4.4 Aparafusamento das porcas

Primeiramente é realizada uma calibração de uma referência, em que esta referência corresponde a dois parafusos na chapa *Cloison*, denominados por *screw* nº 3 e *screw* nº 4. Para a referência calibrada são determinadas as posições pelas quais o robô se deve de deslocar sendo essas posições também usadas como referência. De seguida é realizado um *scan* com o sistema de visão onde o *trigger* é ativado pelo robô para a obtenção da nuvem de pontos. Posteriormente é realizado o processamento e comparado com os dados de referência. Depois é feita a conversão entre os dados da câmara para os dados do robô para calcular as novas posições e o robô realizar o aparafusamento das porcas na chapa *Cloison*.

Por fim, a Figura 4.39 ilustra o processo completo do sistema robótico e seus componentes durante o ciclo de trabalho.

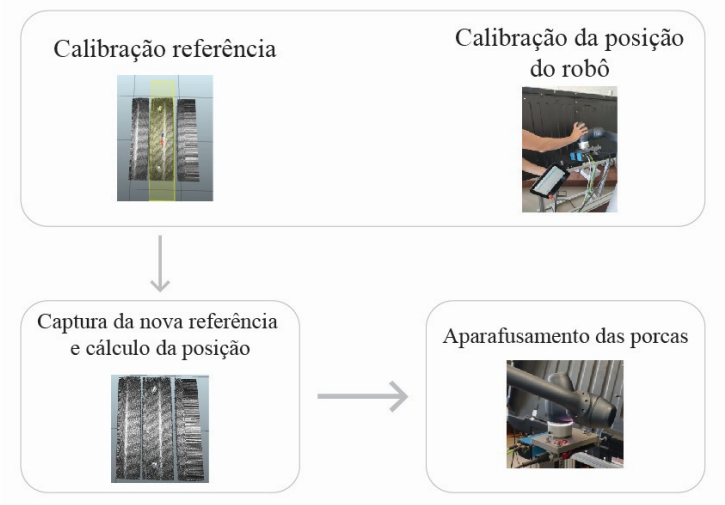


Figura 4.39: Fluxograma descritivo de todo o processo do ciclo de trabalho.

5 Resultados experimentais e discussão

Este capítulo apresenta os resultados experimentais e discussão das plataformas desenvolvidas e, por isso, encontra-se dividido em duas secções: uma para os testes do sistema de gestão de frota de robôs móveis e outra para os testes do sistema de aparafusamento com o manipulador robótico colaborativo.

5.1 Testes da gestão de frota de robôs móveis

Nesta secção são apresentados os testes realizados no sistema de gestão de frota de robôs móveis. Para verificar a fiabilidade, precisão e robustez do sistema, foram realizados alguns testes.

A execução dos testes teve como objetivo validar o desempenho do *software* de gestão de frota, assim como identificar possíveis melhorias que possam ser integradas para uma maior eficiência operacional. Seguidamente, são apresentados os procedimentos dos testes realizados, acompanhados de figuras ilustrativas, que representam as várias etapas do processo de avaliação e os respetivos resultados.

Os testes efetuados desta parte relativa ao *software* de gestão de frota consistiram em testar várias vezes a conexão da plataforma ao robô, quer através do envio de uma instrução para o robô se deslocar até determinado ponto, quer através da análise em tempo real da sua localização no mapa.

Foi implementado um formulário para que seja possível trocar o robô e, consequentemente, o mapa que se pretende visualizar na gestão de frota, uma vez que o *software* irá desenhar o mapa que está naquele momento associado ao robô, ou seja o mapa em que o robô está a "navegar". Este formulário pode ser visualizado na Figura 5.1.

Após o utilizador clicar no botão "Trocar Mapa" irá ser apresentado o mapa escolhido e o robô nele contido (neste caso, é o mapa visualizado através do *Roboshop* no capítulo anterior), conforme demonstra a Figura 5.2.

Antes da visualização do movimento e das informações do robô em tempo real, é possível interagir com um formulário especificamente desenvolvido para comandar o robô a deslocar-se até um ponto de destino. Este formulário, localizado na parte superior

Figura 5.1: Formulário da troca de mapa.

Figura 5.2: Visualização do mapa escolhido do robô na página Web.

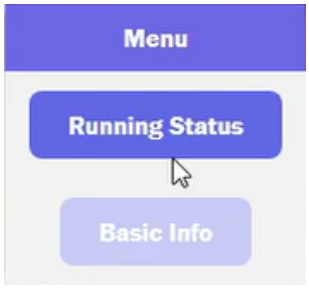


Figura 5.3: Opções das tabelas informativas disponíveis.

da página Web (conforme ilustrado na Figura 5.2), permite ao utilizador selecionar o destino desejado. No exemplo apresentado, foi enviada ao robô a instrução para se deslocar até ao ponto "LM17". Após clicar no botão "Enviar Tarefa", o robô inicia o movimento em direção ao destino solicitado.

Para além da visualização gráfica em tempo real do movimento do robô, a plataforma disponibiliza ao utilizador a visualização de informação complementar através de duas tabelas que podem ser escolhidas quando se clica em cima da imagem do robô em movimento, conforme de pode visualizar na Figura 5.3.

Ao clicar na tabela "*Running Status*", o utilizador terá acesso a informações tais como, tempo que o robô percorreu, informação da distância percorrida pelo robô, se está a carregar ou não, entre outras informações. Pode visualizar-se a informação presente nesta tabela através da Figura 5.4.

Por fim, a outra tabela que o *software* permite visualizar é a tabela "*Basic Info*". Como o próprio nome indica, contém informações básicas do robô, tais como a identificação

Running Status	
Time	44.01 min
Total Time	14.05 dias
Odo	67.35 km
Charging	No
Battery Voltage	49.65 V
Battery Current	-1.69 A
Manual Charge	No

Figura 5.4: Visualização da informação da tabela "*Running Status*".

Basic Information	
IDAGV	AMB-1000JS_Demo
Current Map	mapa_isec_oficina
Battery Level	69.0%
Pose X	6.0037
Pose Y	4.3988
Heading	91°

Figura 5.5: Visualização da informação da tabela "*Basic Info*".

do robô que está a ser visualizado, o nome do mapa que está ser visualizado, o nível de bateria, a posição X e Y em relação ao referencial do robô e também a sua orientação. Esta tabela está ilustrada na Figura 5.5.

Concluindo, os testes realizados no sistema de gestão de frota demonstraram a sua capacidade de conectar de forma eficiente com o robô móvel, permitindo o envio de instruções precisas e a monitorização em tempo real da sua posição. Os resultados obtidos evidenciam a fiabilidade do sistema, bem como a sua robustez em contextos operacionais variados. No entanto, os testes também destacaram áreas de potencial melhoria, as quais podem ser exploradas para otimizar ainda mais o desempenho e a eficiência do *software*. As observações extraídas reforçam a importância de um sistema de gestão de frota bem estruturado para maximizar a eficácia do mesmo.

5.2 Testes do sistema de aparafusamento com manipulador robótico colaborativo

O teste experimental foi realizado com o objetivo de validar a robustez do algoritmo desenvolvido. O protótipo desenvolvido é o *setup* de testes e encontra-se ilustrado na Figura 5.6. O primeiro teste consistiu em deslocar a placa *Cloison* entre várias distân-

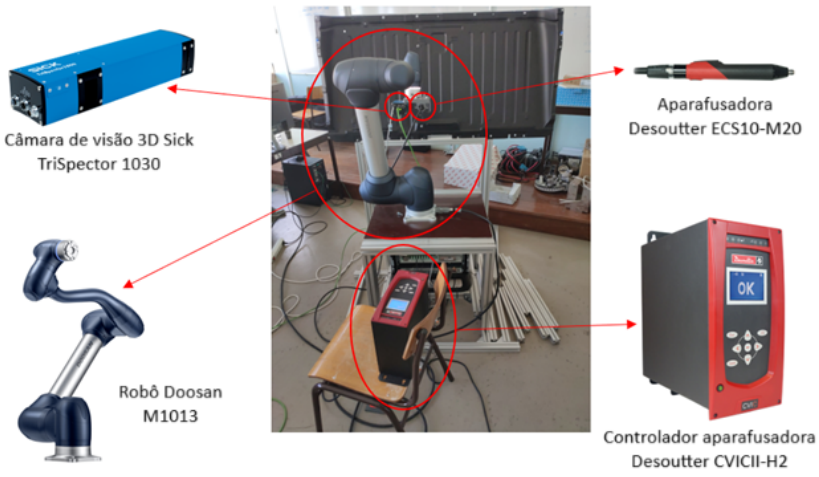


Figura 5.6: *Setup* de testes para o aparafusamento da placa *Cloison*.

cias, executando o ciclo de trabalho para cada uma delas. Para cada distância, foram realizadas 5 iterações. Com este procedimento, pretende-se verificar as diferenças entre os valores das posições calculadas pela câmara e os valores reais da posição de cada parafuso.

Como métricas de avaliação, foram utilizadas a distância máxima (Max) entre o valor de referência e o valor calculado, assim como a métrica *Root Mean Square* (RMS). A métrica de distância máxima avalia o maior erro entre o valor de referência e o valor calculado nas 5 iterações realizadas para cada distância em cada parafuso, enquanto o RMS calcula o erro quadrático médio, proporcionando uma visão global da precisão do sistema ao longo de várias iterações.

As Tabelas 5.1, 5.2 e 5.3 apresentam os resultados dos testes experimentais realizados no aperto das porcas da placa *Cloison*. Estes testes foram efetuados com desvios de 0, 3, 6, 9, 12 e 15 mm ao longo do eixo Y, simulando pequenas variações na posição da placa. Os valores escolhidos refletem desvios plausíveis em contexto real, onde se estima que não ultrapassem os 15 mm. Nas tabelas apresentadas a legenda das mesmas é a seguinte:

- **Distância**: corresponde à distância em milímetros do deslocamento a que a placa *Cloison* foi sujeita para cada teste;
- **Y real**: corresponde à posição real da localização dos apertos em relação ao eixo do Y definido previamente;
- **Eixo Y calc.**: corresponde à posição calculada pela câmara da localização dos apertos em relação ao eixo do Y;
- **Erro**: corresponde ao erro associado à aquisição da posição por parte da câmara de visão, é calculado através do valor absoluto da diferença entre o "Y calc." e o "Y real".

Tabela 5.1: Resultados experimentais para desvios de 0 mm e 3 mm da placa *Cloison*.

Iteração	Porca	Distância = 0 mm			Distância = 3 mm		
		Y real	Y calc.	Erro	Y real	Y calc.	Erro
1ª	Porca 1	0	0,202	0,202	-3	-3,305	0,305
	Porca 2	6	7,702	1,702	3	4,089	1,089
	Porca 3	6	7,702	1,702	3	4,089	1,089
	Porca 4	6	7,702	1,702	3	4,089	1,089
	Porca 5	6	7,702	1,702	-1,5	-0,411	1,089
	Porca 6	1,5	3,202	1,702	-1,5	-0,411	1,089
	Porca 7	1,5	3,202	1,702	-1,5	-0,411	1,089
2ª	Porca 1	0	0,181	0,181	-3	-3,305	0,305
	Porca 2	6	7,681	1,681	3	4,195	1,195
	Porca 3	6	7,681	1,681	3	4,195	1,195
	Porca 4	6	7,681	1,681	3	4,195	1,195
	Porca 5	6	7,681	1,681	-1,5	-0,305	1,195
	Porca 6	1,5	3,181	1,681	-1,5	-0,305	1,195
	Porca 7	1,5	3,181	1,681	-1,5	-0,305	1,195
3ª	Porca 1	0	0,765	0,765	-3	-4,419	1,419
	Porca 2	6	8,265	2,265	3	3,081	0,081
	Porca 3	6	8,265	2,265	3	3,081	0,081
	Porca 4	6	8,265	2,265	3	3,081	0,081
	Porca 5	6	8,265	2,265	-1,5	-1,419	0,081
	Porca 6	1,5	3,765	2,265	-1,5	-1,419	0,081
	Porca 7	1,5	3,765	2,265	-1,5	-1,419	0,081
4ª	Porca 1	0	0,965	0,965	-3	-4,064	1,064
	Porca 2	6	8,465	2,465	3	3,436	0,436
	Porca 3	6	8,465	2,465	3	3,436	0,436
	Porca 4	6	8,465	2,465	3	3,436	0,436
	Porca 5	6	8,465	2,465	-1,5	-1,064	0,436
	Porca 6	1,5	3,965	2,465	-1,5	-1,064	0,436
	Porca 7	1,5	3,965	2,465	-1,5	-1,064	0,436
5ª	Porca 1	0	0,291	0,291	-3	-3,638	0,638
	Porca 2	6	7,791	1,791	3	3,862	0,862
	Porca 3	6	7,791	1,791	3	3,862	0,862
	Porca 4	6	7,791	1,791	3	3,862	0,862
	Porca 5	6	7,791	1,791	-1,5	-0,638	0,862
	Porca 6	1,5	3,291	1,791	-1,5	-0,638	0,862
	Porca 7	1,5	3,291	1,791	-1,5	-0,638	0,862

Tabela 5.2: Resultados experimentais para desvios de 6 mm e 9 mm da placa *Cloison*.

Iteração	**Porca**	**Distância = 6 mm**			**Distância = 9 mm**		
		Y real	**Y calc.**	**Erro**	**Y real**	**Y calc.**	**Erro**
1ª	Porca 1	-6	-6,7932	0,7932	-9	-10,204	1,204
	Porca 2	0	0,7068	0,7068	-3	-2,704	0,296
	Porca 3	0	0,7068	0,7068	-3	-2,704	0,296
	Porca 4	0	0,7068	0,7068	-3	-2,704	0,296
	Porca 5	-4,5	-3,7932	0,7068	-7,5	-7,204	0,296
	Porca 6	-4,5	-3,7932	0,7068	-7,5	-7,204	0,296
	Porca 7	-4,5	-3,7932	0,7068	-7,5	-7,204	0,296
2ª	Porca 1	-6	-6,3930	0,3930	-9	-9,353	0,353
	Porca 2	0	1,1070	1,1070	-3	-1,853	1,147
	Porca 3	0	1,1070	1,1070	-3	-1,853	1,147
	Porca 4	0	1,1070	1,1070	-3	-1,853	1,147
	Porca 5	-4,5	-3,3930	1,1070	-7,5	-6,353	1,147
	Porca 6	-4,5	-3,3930	1,1070	-7,5	-6,353	1,147
	Porca 7	-4,5	-3,3930	1,1070	-7,5	-6,353	1,147
3ª	Porca 1	-6	-5,5255	0,4745	-9	-8,535	0,465
	Porca 2	0	1,9745	1,9745	-3	-1,035	1,965
	Porca 3	0	1,9745	1,9745	-3	-1,035	1,965
	Porca 4	0	1,9745	1,9745	-3	-1,035	1,965
	Porca 5	-4,5	-2,5255	1,9745	-7,5	-5,535	1,965
	Porca 6	-4,5	-2,5255	1,9745	-7,5	-5,535	1,965
	Porca 7	-4,5	-2,5255	1,9745	-7,5	-5,535	1,965
4ª	Porca 1	-6	-5,5831	0,4169	-9	-8,327	0,673
	Porca 2	0	1,9169	1,9169	-3	-0,827	2,173
	Porca 3	0	1,9169	1,9169	-3	-0,827	2,173
	Porca 4	0	1,9169	1,9169	-3	-0,827	2,173
	Porca 5	-4,5	-2,5831	1,9169	-7,5	-5,327	2,173
	Porca 6	-4,5	-2,5831	1,9169	-7,5	-5,327	2,173
	Porca 7	-4,5	-2,5831	1,9169	-7,5	-5,327	2,173
5ª	Porca 1	-6	-5,3213	0,6787	-9	-8,696	0,304
	Porca 2	0	2,1787	2,1787	-3	-1,196	1,804
	Porca 3	0	2,1787	2,1787	-3	-1,196	1,804
	Porca 4	0	2,1787	2,1787	-3	-1,196	1,804
	Porca 5	-4,5	-2,3213	2,1787	-7,5	-5,696	1,804
	Porca 6	-4,5	-2,3213	2,1787	-7,5	-5,696	1,804
	Porca 7	-4,5	-2,3213	2,1787	-7,5	-5,696	1,804

Tabela 5.3: Resultados experimentais para desvios de 12 mm e 15 mm da placa *Cloison*.

Iteração	Porca	Distância = 12 mm			Distância = 15 mm		
		Y real	Y calc.	Erro	Y real	Y calc.	Erro
1ª	Porca 1	-12	-11,410	0,590	-15	-15,435	0,435
	Porca 2	-6	-3,910	2,090	-9	-7,935	1,065
	Porca 3	-6	-3,910	2,090	-9	-7,935	1,065
	Porca 4	-6	-3,910	2,090	-9	-7,935	1,065
	Porca 5	-6	-3,910	2,090	-9	-7,935	1,065
	Porca 6	-10,5	-8,410	2,090	-13,5	-12,435	1,065
	Porca 7	-10,5	-8,410	2,090	-13,5	-12,435	1,065
2ª	Porca 1	-12	-11,527	0,473	-15	-15,536	0,464
	Porca 2	-6	-4,027	1,973	-9	-7,913	1,087
	Porca 3	-6	-4,027	1,973	-9	-7,913	1,087
	Porca 4	-6	-4,027	1,973	-9	-7,913	1,087
	Porca 5	-6	-4,027	1,973	-9	-7,913	1,087
	Porca 6	-10,5	-8,527	1,973	-13,5	-12,413	1,087
	Porca 7	-10,5	-8,527	1,973	-13,5	-12,413	1,087
3ª	Porca 1	-12	-11,472	0,528	-15	-15,489	0,511
	Porca 2	-6	-4,527	1,473	-9	-7,989	1,011
	Porca 3	-6	-4,527	1,473	-9	-7,989	1,011
	Porca 4	-6	-4,527	1,473	-9	-7,989	1,011
	Porca 5	-6	-4,527	1,473	-9	-7,989	1,011
	Porca 6	-10,5	-9,027	1,473	-13,5	-12,489	1,011
	Porca 7	-10,5	-9,027	1,473	-13,5	-12,489	1,011
4ª	Porca 1	-12	-11,642	0,358	-15	-15,580	0,420
	Porca 2	-6	-4,578	1,422	-9	-8,080	0,920
	Porca 3	-6	-4,578	1,422	-9	-8,080	0,920
	Porca 4	-6	-4,578	1,422	-9	-8,080	0,920
	Porca 5	-6	-4,578	1,422	-9	-8,080	0,920
	Porca 6	-10,5	-9,078	1,422	-13,5	-12,580	0,920
	Porca 7	-10,5	-9,078	1,422	-13,5	-12,580	0,920
5ª	Porca 1	-12	-11,642	0,358	-15	-15,580	0,420
	Porca 2	-6	-4,142	1,858	-9	-8,080	0,920
	Porca 3	-6	-4,142	1,858	-9	-8,080	0,920
	Porca 4	-6	-4,142	1,858	-9	-8,080	0,920
	Porca 5	-6	-4,142	1,858	-9	-8,080	0,920
	Porca 6	-10,5	-8,642	1,858	-13,5	-12,580	0,920
	Porca 7	-10,5	-8,642	1,858	-13,5	-12,580	0,920

Por último, na Tabela 5.4 são apresentados os valores de MAX e RMS para as sete porcas em diferentes distâncias, variando de 0 mm a 15 mm. Observa-se uma variação significativa nas medições em função da distância e da porca analisada:

- **Porca 1**: Para a distância de 0 mm, o valor de MAX foi de 0,96 mm e o de RMS foi de 0,57 mm, indicando uma pequena discrepância entre o valor calculado e o valor de referência. No entanto, para distâncias maiores, como 12 mm e 15 mm, os valores de MAX e RMS diminuíram para 0,58 mm e 0,37 mm a 12 mm, e para 0,57 mm e 0,49 mm a 15 mm, sugerindo que a precisão do algoritmo melhorou com o aumento da distância para esta porca;
- **Porcas 2 a 7**: As restantes porcas apresentaram um padrão semelhante no comportamento das métricas MAX e RMS. Para a distância de 0 mm, o valor de MAX foi consistentemente mais elevado, atingindo 2,46 mm, enquanto o RMS foi de 2,00 mm. Com o aumento das distâncias, a MAX variou entre 1,19 mm e 2,17 mm, e o RMS entre 0,84 mm e 1,78 mm, evidenciando uma variação mais acentuada nas maiores distâncias.

Analisando os dados recolhidos pode-se verificar que de um modo geral, o algoritmo apresentou erros menores nas distâncias maiores para a Porca 1, enquanto nas Porcas 2 a 7, os erros permaneceram mais elevados nas distâncias intermédias (3 mm e 6 mm), antes de uma ligeira estabilização nas distâncias maiores (12 mm e 15 mm). Este comportamento pode ser explicado pela maior complexidade na leitura de posições em distâncias mais curtas, onde as interferências tendem a ser mais significativas.

Tabela 5.4: Valores de MAX e RMS para as sete porcas nas diferentes distâncias testadas.

Porca	**Distância (mm)**	**MAX**	**RMS**
Porca 1	0	0,9647	0,5785
	3	1,419	0,8735
	6	0,7932	0,5733
	9	1,2035	0,6833
	12	0,5898	0,3758
	15	0,5797	0,4945
Porca 2	0	2,465	2,0066
	3	1,1946	0,8428
	6	2,1787	1,676
	9	2,1727	1,6274
	12	2,0898	1,7834
	15	1,0865	1,0113
Porca 3	0	2,465	2,0066
	3	1,1946	0,8428
	6	2,1787	1,676
	9	2,1727	1,6274
	12	2,0898	1,7834
	15	1,0865	1,0113
Porca 4	0	2,465	2,0066
	3	1,1946	0,8428
	6	2,1787	1,676
	9	2,1727	1,6274
	12	2,0898	1,7834
	15	1,0865	1,0113
Porca 5	0	2,465	2,0066
	3	1,1946	0,8428
	6	2,1787	1,676
	9	2,1727	1,6274
	12	2,0898	1,7834
	15	1,0865	1,0113
Porca 6	0	2,465	2,0066
	3	1,1946	0,8428
	6	2,1787	1,676
	9	2,1727	1,6274
	12	2,0898	1,7834
	15	1,0865	1,0113
Porca 7	0	2,465	2,0066
	3	1,1946	0,8428
	6	2,1787	1,676
	9	2,1727	1,6274
	12	2,0898	1,7834
	15	1,0865	1,0113

6 Conclusões e propostas de trabalho futuro

Com base no trabalho apresentado é possível concluir que os objetivos iniciais foram, de um modo geral, atingidos com sucesso. A organização do projeto em várias etapas e vertentes permitiu um aprofundamento tanto no conhecimento teórico quanto na implementação prática, culminando em soluções aplicáveis na monitorização de robôs móveis e automação do aparafusamento de componentes no setor automóvel.

Inicialmente, foi essencial adquirir um conhecimento geral sobre o projeto e da atividade das empresas envolvidas, o que permitiu alinhar os objetivos com as necessidades reais do setor em questão, garantido a relevância e a aplicabilidade dos protótipos desenvolvidos. A familiarização com as tecnologias associadas aos robôs móveis e colaborativos constituiu um passo fundamental, promovendo a capacitação técnica necessária ao sucesso do projeto.

A pesquisa sobre o contexto histórico da robótica industrial, com foco tanto na robótica móvel quanto na colaborativa, foi relevante para entender a evolução dos sistemas e tecnologias robóticas aplicadas à produção industrial. A realização da resenha histórica demonstrou a importância crescente da robótica colaborativa e móvel e o modo como estas tecnologias se adaptam aos requisitos de produção moderna. Além disto, o estudo das normas e práticas de segurança na robótica industrial trouxe à tona as melhores práticas e as especificidades de segurança tanto para a robótica móvel quanto para a colaborativa, um ponto central para garantir a implementação responsável de sistemas robóticos.

A criação do visualizador de frotas de robôs móveis visa colmatar uma das principais lacunas existentes no ambiente de chão de fábrica no que diz respeito à gestão desses dispositivos: a ausência de uma visualização remota. Esta ferramenta permite ao utilizador monitorizar à distância a posição de cada robô, detetar avarias, consultar o nível de bateria, entre outras funcionalidades essenciais para uma gestão eficiente da frota.

Na vertente da robótica colaborativa, a implementação de sistemas de aparafusamento com robôs colaborativos trouxe uma solução prática e eficiente para a automação deste processo. Este sistema de aparafusamento contribuiu para a padronização e agilização dos processos produtivos, diminuindo as tarefas repetitivas e todos os problemas que as mesmas acarretam para os operadores humanos.

Como desenvolvimentos futuros podem-se mencionar:

- Na parte do gestor de frotas propõe-se a integração do protocolo VDA 5050, com

o intuito de tornar a plataforma transversal a diferentes marcas de robôs móveis, permitindo a comunicação e recolha de dados de forma padronizada. Para além desta integração, poderão ser consideradas funcionalidades adicionais, tais como a implementação de alertas de emergência, alertas de deteção de obstáculos, bem como a visualização simultânea de vários robôs no mesmo mapa, recorrendo aos restantes robôs já adquiridos para essa integração;

- Na parte da automação do aparafusamento de componentes de automóveis, sugere-se o desenvolvimento de todo o processo destinado ao aperto do *Cloison*, aplicado agora ao elemento da roda *Porteur*, dando continuidade ao trabalho anteriormente realizado e aprofundando a automação desta etapa específica da montagem automóvel.

Por fim, sublinha-se que parte do trabalho desenvolvido pelo aluno e aqui descrito foi já alvo de uma publicação científica internacional num periódico de Quartil 2 e à data atual encontra-se em processo de revisão um outro artigo submetido para um periódico de Quartil 1.

Referências bibliográficas

[1] A. Gonçalves, T. Pereira, D. Lopes, F. Cunha, F. Lopes, F. Coutinho, J. Barreiros, J. Durães, P. Santos, F. Simões, P. Ferreira, E. D. C. Freitas, J. P. F. Trovão, V. Santos, J. P. Ferreira, and N. M. F. Ferreira, "Enhancing nut-tightening processes in the automotive industry: Integration of 3d vision systems with collaborative robots," *Automation*, vol. 6, no. 1, 2025. [Online]. Available: https://www.mdpi.com/2673-4052/6/1/8

[2] Doosan, "M-series," https://www.doosanrobotics.com/en/product-solutions/product/#mSeries, 2024.

[3] Sick, "Trispector1000," https://www.sick.com/media/docs/2/72/272/product_information_trispector1000_intiutive_3d_inspection_en_im0062272.pdf.

[4] Photoneo, "Phoxi 3d scanner xs," https://www.photoneo.com/products/phoxi-scan-xs/.

[5] Asus, "Xtion pro," https://www.asus.com/supportonly/xtion%20pro/helpdesk_manual/.

[6] Stereolabs, "Zed mini stereo camera," https://www.stereolabs.com/en-pt/store/products/zed-mini.

[7] Ets Georges Renault, *CVIC II Controllers V 5.1.X Operator's Manual*, Nantes, France, 2019.

[8] Europneumaq, "Sobre a europneumaq," https://www.europneumaq.com/pt/empresa/sobre-a-europneumaq, 2024.

[9] R. Silva, "A automação industrial é a tecnologia de aplicação," *Robótica - Revista Técnico-Científica*, no. 87, 2012.

[10] A. Grau, M. Indri, L. L. Bello, and T. Sauter, "Industrial robotics in factory automation: From the early stage to the internet of things," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, pp. 6159–6164.

[11] J. N. Pires, *Automação Industrial*. Portugal: ETEP - Edições Técnicas e Profissionais, 2007.

[12] N. Mineiro, "Estado da arte da robótica industrial em portugal," *Robótica - Revista Técnico-Científica*, no. 87, 2012.

[13] P. Abreu, "Aplicações industriais de robôs," Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2002.

[14] Thinkwork, "Venda de robôs de serviço profissionais cresce 48% em 2022," https://thinkworklab.com/transformacao-digital/venda-de-robos-de-servico-profissionais-cresce-48-em-2022/, 2023.

[15] R. C. Allen, *The Industrial Revolution: a very short introduction*. United Kingdom: OXFORD, 2017.

[16] E. Brynjolfsson and A. McAfee, *The second age machine*. United States: Norton & Company, 2014.

[17] P. Marsh, *The New Industrial Revolution: Consumers, Globalization and the End of Mass Production*. Yale University Press, 2012.

[18] Justyna Matuszak, "Is your business ready for industry 5.0?" https://knowhow.distrelec.com/manufacturing/is-your-business-ready-for-industry-5-0/, 2022.

[19] K. Schwab, *The Fourth Industrial Revolution*. Crown Currency, 2016.

[20] A Voz da Indústria, "Descubra o que são os agvs (automated guided vehicle)," https://avozdaindustria.com.br/inovacao/descubra-o-que-sao-os-agvs-automated-guided-vehicle, 2023.

[21] L. A. F. da Rocha, "Logística flexível baseada em agvss," Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2010.

[22] 4am Robotics, "Autonomous mobile robots: Trends and challenges," https://www.scio-automation.com/update/autonomous-mobile-robots-trends-and-challenges, 2022.

[23] R. Müller, M. Vette, and A. Geenen, "Skill-based dynamic task allocation in human-robot-cooperation with the example of welding application," *Procedia Manufacturing*, vol. 11, pp. 13–21, 2017, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978917303177

[24] Universal Robots Brasil, "Como escolher entre robôs colaborativos e robôs industriais tradicionais?" https://www.universal-robots.com/br/blog/como-escolher-entre-rob%C3%B4s-colaborativos-e-rob%C3%B4s-industriais-tradicionais/, 2022.

[25] S. El Zaatari, M. Marei, W. Li, and Z. Usman, "Cobot programming for collaborative industrial tasks: An overview," *Robotics and Autonomous Systems*, vol. 116, pp. 162–180, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188901830602X

[26] T. Pentikäinen and S. Richard, "How to make collaborative robot programming easier: Workflow visualization on a tablet device," Master's thesis, UPPSALA UNIVERSITET, Sweden, Jun. 2016.

[27] Universal Robotics, "Our history," https://www.universal-robots.com/about-universal-robots/our-history/, 2022.

[28] R. Müller, M. Vette, and O. Mailahn, "Process-oriented task assignment for assembly processes with human-robot interaction," *Procedia CIRP*, vol. 44, pp. 210–215, 12 2016.

[29] L. Peternel, W. Kim, J. Babič, and A. Ajoudani, "Towards ergonomic control of human-robot co-manipulation and handover," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 55–60.

[30] E. Commission, "Periodic reporting for period 1 - colrobotics (collaborative robotics for assembly and kitting in smart manufacturing)," European Commission, Tech. Rep., 2018.

[31] L. Pérez, I. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. F. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors*, vol. 16, no. 3, 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/3/335

[32] T. Clarke and X. Wang, "The control of a robot end-effector using photogrammetry," *Int. Arch. Photogramm. Remote Sens.*, vol. 33, 01 2000.

[33] R.-J. Halme, M. Lanz, J. Kämäräinen, R. Pieters, J. Latokartano, and A. Hietanen, "Review of vision-based safety systems for human-robot collaboration," *Procedia CIRP*, vol. 72, pp. 111–116, 2018, 51st CIRP Conference on Manufacturing Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827118301434

[34] A. Tellaeche, I. Maurtua, and A. Ibarguren, "Use of machine vision in collaborative robotics: An industrial case," in *21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2016, Berlin, Germany, September 6-9, 2016*. IEEE, 2016, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/ETFA.2016.7733689

[35] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio, and G. Sansoni, "Deep learning based machine vision: First steps towards a hand gesture recognition set up for collaborative robots," 04 2018.

[36] R. Li, D. T. Pham, J. Huang, Y. Tan, M. Qu, Y. Wang, M. Kerin, K. Jiang, S. Su, C. Ji, Q. Liu, and Z. Zhou, "Unfastening of hexagonal headed screws by a collaborative robot," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1455–1468, 2020.

[37] N. M. DiFilippo and M. K. Jouaneh, "A system combining force and vision sensing for automated screw removal on laptops," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 887–895, 2018.

[38] L. Sanneman, C. Fourie, and J. A. Shah, "The state of industrial robotics: Emerging technologies, challenges, and key research directions," *Foundations and Trends® in Robotics*, vol. 8, no. 3, pp. 225–306, 2021. [Online]. Available: http://dx.doi.org/10.1561/2300000065

[39] S. Cebollada, L. Payá, M. Flores, A. Peidró, and O. Reinoso, "A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data," *Expert Syst. Appl.*, vol. 167, no. C, apr 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2020.114195

[40] A. J. Moshayedi, J. Li, and L. Liao, "Agv (automated guided vehicle) robot: Mission and obstacles in design and performance," vol. 12, pp. 5–0018, 11 2019.

[41] G. Fragapane, R. De Koster, F. Sgarbossa, and J. O. Strandhagen, "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," *European Journal of Operational Research*, vol. 294, 01 2021.

[42] L. Perez, E. Diez, R. Usamentiaga, and F. D. Garcia, "Industrial robot control and operator training using virtual reality interfaces," *Computers in Industry*, vol. 109, pp. 114–120, 08 2019.

[43] H. Sobreira, A. Moreira, P. Costa, and J. Lima, "Robust mobile robot localization based on security laser scanner," *Proceedings - 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2015*, pp. 162–167, 05 2015.

[44] J. Diaz Bayona, J. Gonzalez Pilonieta, R. Molina Lache, and S. Roa Prada, "Development of a control system for path following applications in an agv using computer vision," in *2020 9th International Congress of Mechatronics Engineering and Automation, CIIMA 2020 - Conference Proceedings*, ser. 2020 9th International Congress of Mechatronics Engineering and Automation, CIIMA 2020 - Conference Proceedings, L. Colorado, Ed. Institute of Electrical and Electronics Engineers Inc., Nov. 2020, publisher Copyright: © 2020 IEEE.; 9th International Congress of Mechatronics Engineering and Automation, CIIMA 2020 ; Conference date: 04-11-2020 Through 06-11-2020.

[45] A. M. Hosseini, C. Fischer, M. Bhole, W. Kastner, T. Sauter, and S. Schlund, "A safety and security requirements management methodology in reconfigurable collaborative human-robot application," in *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*. IEEE, Apr. 2023.

[46] B. A. Abishek, T. Kavyashree, R. Jayalakshmi, S. Tharunkumar, and R. Raffik, "Collaborative robots and cyber security in industry 5.0," in *2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, 2023, pp. 1–6.

[47] A. M.Hosseini, T. Sauter, and W. Kastner, "Formal verification of safety and secu-

rity properties in industry 4.0 applications," 09 2023, pp. 1–8.

[48] w3schools, "Http request methods," https://www.w3schools.com/TAgs/ref_httpmethods.asp.

[49] Doosan Robotics, *Doosan Robotics Installation Manual v2.1*, Suwon, Republic of Korea, 2021.

Anexos

Anexo A - Código JavaScript da plataforma de monitorização

```
const currentPage = window.location.pathname;

if(currentPage.endsWith('enviaordersmap.html')){

    // Esta função adapta as coordenadas do mapa para o sistema de
        coordenadas do canvas HTML, aplicando uma escala
       proporcional
    function transformCoords(x, y, canvasWidth, canvasHeight) {
        const scaleX = canvasWidth / (maxX - minX);
        const scaleY = canvasHeight / (maxY - minY);
        const newX = (x - minX) * scaleX;
        const newY = canvasHeight - (y - minY) * scaleY;
        return { newX, newY };
    }

    // Desenha um ponto circular
    function drawPoint(ctx, x, y, color, label) {
        ctx.beginPath();
        ctx.arc(x, y, 5, 0, 2 * Math.PI);
        ctx.fillStyle = color;
        ctx.fill();
        ctx.closePath();

        if (label) {
            ctx.font = "bold 12px Arial";
            ctx.textAlign = "center";
            ctx.textBaseline = "middle";
            ctx.fillStyle = '#000';
            ctx.fillText(label, x, y - 22);
        }
    }

    // Desenha um retângulo arredondado
    function drawRect(ctx, x, y, color, label) {
```

```
    ctx.fillStyle = color;
    ctx.strokeStyle = color;
    ctx.fillStyle = 'rgba(0, 0, 255, 0.6)';

    ctx.beginPath();
    ctx.roundRect(x - 7, y - 7, 14, 14, 5);
    ctx.fill();
    ctx.closePath();

    if (label) {
        ctx.font = "bold 12px Arial";
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.fillStyle = '#000';
        ctx.fillText(label, x, y + 20);
    }
}

// Cria trajetórias entre pontos com base em classes de curva
   (sejam elas Bezier, arco ou uma linha reta)
function drawCurve(ctx, startPos, endPos, curve, canvasWidth,
   canvasHeight) {
    ctx.lineWidth = 25;
    ctx.lineJoin = 'round';
    ctx.lineCap = 'round';
    ctx.strokeStyle = 'rgba(255, 165, 0, 0.3)';

    if (curve.className === 'BezierPath') {
        const controlPos1 = transformCoords(curve.controlPos1.
           x, curve.controlPos1.y, canvasWidth, canvasHeight);
        const controlPos2 = transformCoords(curve.controlPos2.
           x, curve.controlPos2.y, canvasWidth, canvasHeight);
        ctx.beginPath();
        ctx.moveTo(startPos.newX, startPos.newY);
        ctx.bezierCurveTo(controlPos1.newX, controlPos1.newY,
           controlPos2.newX, controlPos2.newY, endPos.newX,
           endPos.newY);
        ctx.stroke();
    } else if (curve.className === 'DegenerateBezier') {
        const controlPos1 = transformCoords(curve.controlPos1.
           x, curve.controlPos1.y, canvasWidth, canvasHeight);
        const controlPos2 = transformCoords(curve.controlPos2.
           x, curve.controlPos2.y, canvasWidth, canvasHeight);
```

```
            ctx.beginPath();
            ctx.moveTo(startPos.newX, startPos.newY);
            ctx.bezierCurveTo(controlPos1.newX, controlPos1.newY,
               controlPos2.newX, controlPos2.newY, endPos.newX,
               endPos.newY);
            ctx.stroke();
        } else if (curve.className === 'ArcPath') {
            const controlPos = transformCoords(curve.controlPos1.x
               , curve.controlPos1.y, canvasWidth, canvasHeight);
            ctx.beginPath();
            ctx.moveTo(startPos.newX, startPos.newY);
            ctx.quadraticCurveTo(controlPos.newX, controlPos.newY,
                endPos.newX, endPos.newY);
            ctx.stroke();
        } else if (curve.className === 'StraightPath') {
            ctx.beginPath();
            ctx.moveTo(startPos.newX, startPos.newY);
            ctx.lineTo(endPos.newX, endPos.newY);
            ctx.stroke();
        }
    }

    // A função carrega uma imagem do AGV, identificada como "AGV.
       png" e define o seu posicionamento conforme a variável "
       robotAngle"
    function drawAGV(ctx, robotPos, robotAngle) {
        const image = new Image();
        image.src = 'AGV.png';
        const imageSizeX = 540;
        const imageSizeY = 400;
        const scale = 0.15;

        image.onload = function () {
            ctx.save();
            ctx.translate(robotPos.newX, robotPos.newY);
            ctx.rotate(robotAngle);
            ctx.drawImage(image, (-imageSizeX * scale) / 2, (-
               imageSizeY * scale) / 2, (imageSizeX * scale), (
               imageSizeY * scale));
            ctx.restore();

            canvas.addEventListener('click', (event) => {
                const rect = canvas.getBoundingClientRect();
```

```
            const x = event.clientX - rect.left;
            const y = event.clientY - rect.top;

            const agvRadius = (imageSizeX * scale) / 2;
            const distance = Math.sqrt(Math.pow(robotPos.newX
               - x, 2) + Math.pow(robotPos.newY - y, 2));

            // Se o clique estiver dentro do raio do AGV
            if (distance < agvRadius) {
                if (showinfo === 0) {
                    showinfo = 1;
                    showTooltipOptions(x, y);
                } else {
                    showinfo = 0;
                    hideTooltipAndTables();
                }
            }
        });
    };
}

// Esta é a função responsável pelo desenho dos pontos que
   constituem as paredes e obstáculos representados no mapa
   SLAM
function drawSLAM(ctx, x, y) {
    color = '#606060';
    ctx.beginPath();
    ctx.arc(x, y, 1, 0, 2 * Math.PI);
    ctx.fillStyle = color;
    ctx.fill();
    ctx.closePath();
}

// A função que permite o desenho do referencial utilizado no
   mapa de origem do robô
function drawReferencial(ctx, x, y) {
    const arrowLength = 10;
    const axisLength = 50;
    const labelOffset = 15;

    ctx.save();

    // Desenha o eixo X em vermelho
```

```
    ctx.lineWidth = 2;
    ctx.strokeStyle = 'red';
    ctx.beginPath();
    ctx.moveTo(x, y);
    ctx.lineTo(x + axisLength, y);
    ctx.stroke();

    // Desenha a seta vermelha para o eixo X
    ctx.beginPath();
    ctx.moveTo(x + axisLength, y);
    ctx.lineTo(x + axisLength - arrowLength, y - arrowLength /
        2);
    ctx.moveTo(x + axisLength, y);
    ctx.lineTo(x + axisLength - arrowLength, y + arrowLength /
        2);
    ctx.stroke();

    // Desenha o eixo Y em verde
    ctx.strokeStyle = 'green';
    ctx.beginPath();
    ctx.moveTo(x, y);
    ctx.lineTo(x, y - axisLength);
    ctx.stroke();

    // Desenha a seta verde para o eixo Y
    ctx.beginPath();
    ctx.moveTo(x, y - axisLength);
    ctx.lineTo(x - arrowLength / 2, y - axisLength +
       arrowLength);
    ctx.moveTo(x, y - axisLength);
    ctx.lineTo(x + arrowLength / 2, y - axisLength +
       arrowLength);
    ctx.stroke();

    // Rótulos dos eixos
    ctx.font = '12px Arial';
    ctx.fillStyle = 'black';
    ctx.fillText('X', x + axisLength + labelOffset, y);
    ctx.fillText('Y', x, y - axisLength - labelOffset);

    ctx.restore();
}
```

```
// A função drawMap é responsável por desenhar o mapa completo
   , incluindo pontos, retângulos, as rotas, o referencial e o
    próprio mapa SMAP
function drawMap(data) {
    const canvas = document.getElementById('canvas');
    const ctx = canvas.getContext('2d');
    canvas.width = 1675;
    canvas.height = 1100;
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    let instanceNames = [];

    if (data.advancedPointList) {
        data.advancedPointList.forEach(point => {
            const { newX, newY } = transformCoords(point.pos.x
               , point.pos.y, canvas.width, canvas.height);
            const color = '#00A8FF';
            drawPoint(ctx, newX, newY, color, point.
               instanceName);
            if (point.instanceName) {
                instanceNames.push(point.instanceName);
            }
        });
    }

    if (data.binLocationsList) {
        data.binLocationsList.forEach(point => {
            const { newX, newY } = transformCoords(point.
               binLocationList[0].pos.x, point.binLocationList
               [0].pos.y, canvas.width, canvas.height);
            const color = '#0000FF';
            drawRect(ctx, newX, newY, color, point.
               binLocationList[0].instanceName);
            if (point.binLocationList[0].instanceName) {
                instanceNames.push(point.binLocationList[0].
                   instanceName);
            }
        });
    }

    if (data.advancedCurveList) {
        data.advancedCurveList.forEach(curve => {
```

```
            const startPos = transformCoords(curve.startPos.
               pos.x, curve.startPos.pos.y, canvas.width,
               canvas.height);
            const endPos = transformCoords(curve.endPos.pos.x,
                curve.endPos.pos.y, canvas.width, canvas.
               height);
            drawCurve(ctx, startPos, endPos, curve, canvas.
               width, canvas.height);
        });
    }

    if (data.robotPosition) {
        const robotPos = transformCoords(data.robotPosition.x,
            data.robotPosition.y, canvas.width, canvas.height)
           ;
        drawAGV(ctx, robotPos, data.robotPosition.angle);
    }
    if (data.normalPosList) {
        data.normalPosList.forEach(point => {
            const { newX, newY } = transformCoords(point.x,
               point.y, canvas.width, canvas.height);
            drawSLAM(ctx, newX, newY);
        });
    }
    const { newX: refX, newY: refY } = transformCoords('0',
       '0', canvas.width, canvas.height);
    drawReferencial(ctx, refX, refY);
}

// Converte aspas simples em duplas no JSON e ajusta valores
   boleanos para compatibilidade
function convertSingleToDoubleQuotes(smapContent) {
    smapContent = smapContent.replace(/'/g, '"');
    smapContent = smapContent.replace(/\b(True|False)\b/g,
       match => match.toLowerCase());
    smapContent = smapContent.replace(/\b(None)\b/g, 'null');
    return smapContent;
}

// Analisa o JSON para verificar se o mesmo está bem formatado
function inspectAndFixJson(smapContent) {
    try {
        return JSON.parse(smapContent);
```

```
    } catch (e) {
        console.error(`Erro ao decodificar JSON: ${e.message
            }`);
        return null;
    }
}

function generateTaskId() {
    return 'task-' + Date.now();
}

// Permite inserir a localização do destino pretendido através
    de uma requisição do tipo POST
function sendOrder(serverIp, serverPort, taskId, toLoc) {
    const url = `http://${serverIp}:${serverPort}/setOrder`;
    const data = {
        id: taskId,
        blocks: [{
            blockId: "1",
            location: toLoc
        }],
        complete: true
    };

    console.log('Sending data:', data);

    const headers = {
        'Content-Type': 'application/json'
    };

    axios.post(url, data, { headers: headers })
        .then(response => {
            console.log('Response data:', response.data);
        })
        .catch(error => {
            if (error.response) {
                console.error('Error response data:', error.
                    response.data);
                console.error('Error response status:', error.
                    response.status);
                console.error('Error response headers:', error
                    .response.headers);
                if (error.response.data.msg) {
```

```
                    alert(`Erro: ${error.response.data.msg}`);
                }
            } else if (error.request) {
                console.error('Error request:', error.request)
                    ;
            } else {
                console.error('General error:', error.message)
                    ;
            }
            console.error('Error config:', error.config);
        });
}

// Tabela de info
function showTooltipOptions(mouseX, mouseY) {
    const tooltip = document.getElementById('tooltip');
    const optionTable = document.getElementById('optionTable')
        ;

    optionTable.style.display = 'table';

    tooltip.style.left = `${mouseX + 5}px`;
    tooltip.style.top = `${mouseY + 5}px`;
    tooltip.style.display = 'block';
}

function showStatus(tableId) {
    const tables = document.querySelectorAll('.statusTable');

    tables.forEach(table => {
        table.style.display = 'none';
    });

    const optionTable = document.getElementById('optionTable')
        ;
    const selectedTable = document.getElementById(tableId);

    if (selectedTable) {
        optionTable.style.display = 'none';
        selectedTable.style.display = 'table';
        activeTable = tableId;
        updateTableData();
    } else {
```

```
        console.log(`Tabela com ID ${tableId} não encontrada
            !`);
    }
}

function hideTooltipAndTables() {
    const tooltip = document.getElementById('tooltip');
    tooltip.style.display = 'none';

    const tables = document.querySelectorAll('.statusTable');
    tables.forEach(table => {
        table.style.display = 'none';
    });
}

function updateTableData() {
    if (selectedRobotId != null) {
        switch (activeTable) {
            case 'running':
                updateRunningStatus(selectedRobotInstance.data
                    );
                break;
            case 'basic':
                updateBasicInfo(selectedRobotInstance.data);
                break;
            default:
                console.log("Nenhuma tabela ativa.");
        }
    } else {
        console.log("Nenhum robô selecionado");
    }
}

// Atualizar as tabelas
function updateRunningStatus(robotData) {
    const runningBody = document.getElementById('runningBody')
        ;
    runningBody.innerHTML = `
        <tr><td>Time</td><td>${(robotData.time/60000).toFixed
            (2)} min</td></tr>
        <tr><td>Total Time</td><td>${(robotData.totalTime
            /86400000).toFixed(2)} dias</td></tr>
```

```
            <tr><td>Odo</td><td>${(robotData.odo/1000).toFixed(2)}
                km</td></tr>
            <tr><td>Charging</td><td>${robotData.charging ? 'Yes'
               : 'No'}</td></tr>
            <tr><td>Battery Voltage</td><td>${(robotData.voltage).
               toFixed(2)} V</td></tr>
            <tr><td>Battery Current</td><td>${(robotData.current).
               toFixed(2)} A</td></tr>
            <tr><td>Manual Charge</td><td>${robotData.manualCharge
                ? 'Yes' : 'No'}</td></tr>
        `;
    }

    function updateBasicInfo(robotData) {
        const basicBody = document.getElementById('basicBody');
        basicBody.innerHTML = `
             <tr><td>IDAGV</td><td>${robotData.vehicleId}</td></tr
                >
             <tr><td>Current Map</td><td>${robotData.currentMap}</
                td></tr>
             <tr><td>Battery Level</td><td>${robotData.
                batteryPercentage}%</td></tr>
             <tr><td>Pose X</td><td>${robotData.positionX}</td></
                tr>
             <tr><td>Pose Y</td><td>${robotData.positionY}</td></
                tr>
             <tr><td>Heading</td><td>${robotData.headingDegrees}º
                </td></tr>
        `;
    }

    // Função para tornar as tabelas arrastáveis
    function makeTableDraggable(tableId) {
        const table = document.getElementById(tableId);
        let offsetX = 0, offsetY = 0;
        let isDragging = false;

        if (!table.style.left) table.style.left = "0px";
        if (!table.style.top) table.style.top = "0px";

        function mouseDown(e) {
            e.preventDefault();
```

```
        offsetX = e.clientX - parseInt(table.style.left, 10);
        offsetY = e.clientY - parseInt(table.style.top, 10);

        isDragging = true;

        document.addEventListener('mousemove', mouseMove);
        document.addEventListener('mouseup', mouseUp);
    }

    function mouseMove(e) {
        if (isDragging) {
            table.style.position = 'absolute';
            table.style.left = `${e.clientX - offsetX}px`;
            table.style.top = `${e.clientY - offsetY}px`;
        }
    }

    function mouseUp() {
        isDragging = false;
        document.removeEventListener('mousemove', mouseMove);
        document.removeEventListener('mouseup', mouseUp);
    }

    table.addEventListener('mousedown', mouseDown);
}

function makeAllTablesDraggable() {
    const tables = document.querySelectorAll('.statusTable');
    tables.forEach(table => {
        makeTableDraggable(table.id);
    });
}

document.addEventListener('DOMContentLoaded', () => {
    makeAllTablesDraggable();
});

class Robot {
    constructor(vehicleId) {
        this.vehicleId = vehicleId;
        this.data = {};
    }
```

```
updateData(selectedRobot) {

    // Atualiza os dados do robô a partir do objeto
       selectedRobot
    this.data.batteryLevel = selectedRobot.rbk_report.
       battery_level;
    this.data.batteryPercentage = (this.data.batteryLevel
       * 100).toFixed(1);
    this.data.positionX = selectedRobot.rbk_report.x;
    this.data.positionY = selectedRobot.rbk_report.y;
    this.data.heading = selectedRobot.rbk_report.angle *
       -1;
    this.data.currentMap = selectedRobot.rbk_report.
       current_map;
    this.data.headingDegrees = Math.round(this.data.
       heading * (180 / Math.PI));

    this.data.vehicleId = this.vehicleId;

    // Armazena outros dados adicionais
    this.data.time = selectedRobot.rbk_report.time;
    this.data.todayOd = selectedRobot.rbk_report.today_od;
    this.data.totalTime = selectedRobot.rbk_report.
       total_time;
    this.data.odo = selectedRobot.rbk_report.odo;
    this.data.controllerVoltage = selectedRobot.rbk_report
       .controller_voltage;
    this.data.controllerTemp = selectedRobot.rbk_report.
       controller_temp;
    this.data.controllerHumi = selectedRobot.rbk_report.
       controller_humi;
    this.data.charging = selectedRobot.rbk_report.charging
       ;
    this.data.batteryTemp = selectedRobot.rbk_report.
       battery_temp;
    this.data.voltage = selectedRobot.rbk_report.voltage;
    this.data.current = selectedRobot.rbk_report.current;
    this.data.batteryCycle = selectedRobot.rbk_report.
       battery_cycle;
    this.data.batteryUserData = selectedRobot.rbk_report.
       battery_user_data;
    this.data.manualCharge = selectedRobot.rbk_report.
       manual_charge;
```

```
        this.data.maxChargeCurrent = selectedRobot.rbk_report.
           max_charge_current;
        this.data.maxChargeVoltage = selectedRobot.rbk_report.
           max_charge_voltage;
    }
}

let selectedRobotInstance = null;
let showinfo = 0;
let maxX=5;
let minX=5;
let maxY=5;
let minY=5;
let activeTable = '';
const server_ip = '10.220.128.53';
const server_port = 8088;

async function loadRobotIds() {
    const url2 = `http://${server_ip}:${server_port}/
       robotsStatus`;

    try {
        const response = await axios.get(url2);
        const robotData = response.data;

        const idRobotList = document.getElementById('IDRobot')
           ;

        idRobotList.innerHTML = '';

        robotData.report.forEach(robot => {
            const option = document.createElement('option');
            option.value = robot.vehicle_id;
            idRobotList.appendChild(option);
        });
    } catch (error) {
        console.error('Erro ao carregar os IDs dos robôs:',
           error);
    }
}

document.getElementById('taskIDRobot').addEventListener("
   submit", function(event) {
```

```
    event.preventDefault();
    selectedRobotId = document.getElementById('IDRobotInput').
       value;
    getData(selectedRobotId);
});

// Função para buscar dados do robô
async function getData(selectedRobotId) {
    const url1 = `http://${server_ip}:${server_port}/robotSmap
       `;
    const url2 = `http://${server_ip}:${server_port}/
       robotsStatus`;

    const headers = {
        'Content-Type': 'application/json'
    };

    try {
        const response2 = await axios.get(url2);
        const robotData2 = response2.data;

        const selectedRobot = robotData2.report.find(robot =>
           robot.vehicle_id === selectedRobotId);
        if (!selectedRobot) {
            console.error(`Robô com ID ${selectedRobotId} não
               encontrado.`);
            return;
        }

        // Se não existir uma instância do robô, cria uma nova
        if (!selectedRobotInstance) {
            selectedRobotInstance = new Robot(selectedRobotId)
               ;
        }

        // Atualiza os dados do robô
        selectedRobotInstance.updateData(selectedRobot);

        let data = {
            vehicle: selectedRobotId,
            map: selectedRobotInstance.data.currentMap + ".
               smap"
        };
```

```
const response1 = await axios.post(url1, data, {
   headers });
const robotData = response1.data;

let smapContent = JSON.stringify(robotData);
smapContent = convertSingleToDoubleQuotes(smapContent)
   ;
const smapData = inspectAndFixJson(smapContent);

if (smapData) {
    const smapdata_str = JSON.stringify(smapData);
    const positions = smapdata_str.match(/"pos":\s*{\s
       *"x":\s*([\d.-]+),\s*"y":\s*([\d.-]+)\s*}/g);

    if (positions) {
        const pos_list = positions.map(pos => {
            const match = pos.match(/"pos":\s*{\s*"x
               ":\s*([\d.-]+),\s*"y":\s*([\d.-]+)\s
               *}/);
            return { x: parseFloat(match[1]), y:
               parseFloat(match[2]) };
        });

        const x_values = pos_list.map(pos => pos.x);
        const y_values = pos_list.map(pos => pos.y);

        if (x_values.length > 0 && y_values.length >
           0) {
            maxX = Math.max(...x_values);
            minX = Math.min(...x_values);
            maxY = Math.max(...y_values);
            minY = Math.min(...y_values);

            // Para dar uma margem de 2
            maxX += 2;
            minX -= 2;
            maxY += 2;
            minY -= 2;
        } else {
            console.log("Não há valores máximos.");
        }
    } else {
```

```
                console.log("Nenhum valor de 'pos' encontrado
                    .");
            }

            smapData.robotPosition = {
                x: selectedRobotInstance.data.positionX,
                y: selectedRobotInstance.data.positionY,
                angle: selectedRobotInstance.data.heading
            };
            drawMap(smapData);
            selectedRobotInstance.updateData(selectedRobot);
        } else {
            console.log('Não foi possível carregar os dados do
                mapa.');
        }
    } catch (error) {
        if (error.response) {
            const status = error.response.status;
            if (status === 400) {
                console.log('Os parâmetros solicitados estão
                    incorretos ou faltando informações necessá
                    rias, consulte a documentação');
            } else if (status === 404) {
                console.log('Os dados não existem ou não estão
                     disponíveis');
            } else if (status === 406) {
                console.log('Solicitação indisponível');
            } else if (status === 500) {
                console.log('Erro interno do servidor');
            } else {
                console.log('Erro inesperado:', status);
            }
        } else {
            console.error('Erro:', error.message);
        }
    }
}

loadRobotIds();

setInterval(() => {
    if(selectedRobotInstance != null){
        getData(selectedRobotInstance.vehicleId);
```

```
            updateTableData();
        }
        else{
            console.log("Nenhum robô selecionado");
        }
    }, 1000);

    document.getElementById('taskForm').addEventListener("submit",
        function(event) {
        event.preventDefault();
        const taskId = generateTaskId();
        const toLoc = document.getElementById('toLocInput').value;

        if (!toLoc) {
            alert('Por favor, selecione as localizações de origem
               e destino.');
            return;
        }

        sendOrder(server_ip, server_port, taskId, toLoc);
    });
}
```

Anexo B - Código HTML da página WEB

```
<!DOCTYPE html>
<html lang="pt-pt">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
       scale=1.0">
    <title>Testes SEER</title>

    <!-- Estilo CSS com controlo de versão (para evitar cache) -->
    <link rel="stylesheet" href="css/style.css?v=<?php echo
       $version; ?>">

    <!-- Bibliotecas JavaScript externas -->
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min
       .js"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"
       defer></script>

    <!-- Script principal da aplicação -->
    <script src="js/script.js" defer></script>
</head>
<body>
    <!-- Formulário para envio de tarefa ao robô -->
    <form id="taskForm" style="padding-top: 20px;">
        <datalist id="toLoc"></datalist>
        <input type="text" list="toLoc" id="toLocInput" name="
           toLoc" placeholder="To Location">
        <input type="submit" value="Enviar Tarefa" style="color:
           rgb(255, 255, 255);">
    </form>

    <!-- Formulário para selecionar robô e trocar o mapa -->
    <form id="taskIDRobot">
        <datalist id="IDRobot"></datalist>
```

```
        <input type="text" list="IDRobot" id="IDRobotInput" name="
            IDRobot" placeholder="Robot">
        <input type="submit" id="changeMapButton" value="Trocar
            Mapa" style="color: rgb(255, 255, 255);">
    </form>

    <!-- Tooltip com opções e tabelas de estado do robô -->
    <div id="tooltip" class="tableinfor">
        <!-- Menu de opções -->
        <table id="optionTable">
            <tr><th>Menu</th></tr>
            <tr><td><button onclick="showStatus('running')">
                Running Status</button></td></tr>
            <tr><td><button onclick="showStatus('basic')">Basic
                Info</button></td></tr>
        </table>

        <!-- Tabelas com dados detalhados do robô -->
        <div id="tables">
            <table id="running" class="statusTable" style="display
                :none;">
                <thead><tr><th colspan="2">Running Status</th></tr
                    ></thead>
                <tbody id="runningBody"></tbody>
            </table>

            <table id="basic" class="statusTable" style="display:
                none;">
                <thead><tr><th colspan="2">Basic Information</th
                    ></tr></thead>
                <tbody id="basicBody"></tbody>
            </table>
        </div>
    </div>

    <!-- Canvas onde o mapa e o robô são desenhados -->
    <div class="canvas-container">
        <canvas id="canvas"></canvas>
    </div>
</body>
</html>
```

Anexo C - Código completo do robô colaborativo

```
from DRCF import*
import numpy as np
import sys
import os
import time
import socket

# In 9 - Accept Aparafusadora (Aperto correto)
# In 10 - Reject Aparafusadora (Aperto incorreto)
# Out 3 - Ativa Aparafusadora
# Out 4 - RESET Relatório Aparafusadora

#Função para a espera do relatório da aparafusadora
def espera_relatorio() :
            yy = True
            zz = 0
            while yy:
                repA=get_digital_input(9) # Relatório positivo
                repR=get_digital_input(10) # Relatório negativo
                if(repA==1) or (repR==1):
                    set_digital_output(3, 0) # Desativa
                       aparafusadora
                    wait(0.1)
                    release_compliance_ctrl() # Desativa o
                       compliance
                    release_force() # Desativa o force
                    wait(0.1)
                    yy = False
                if zz >= 35:
                    yy = False
                wait(0.1)
                zz = zz + 0.1

# Coordenadas das câmaras, posição fixa no sistema de referência
   global
```

```
CAMERA_COORDS = np.array([
    [-59.7, 55.7, 297.58],
    [59.87, 55.2, 297.58]
])

# Posições dos parafusos 3 e 4 no referencial do robô
ROBOT_COORDS = np.array([
    [635.000,7.500,515.000],#parafuso 3
    [670.500,7.500,630.000] #parafuso 4
])

# Posições dos outros parafusos no referencial do robô
POSE_OTHER_SCREW= np.array([
    [572.521,0.000,339.500], #Parafuso 1
    [607.594,7.500,417.000], #parafuso 2
    [699.500,7.500,728.500], #Parafuso 5
    [732.500,3.000,825.000], #Parafuso 6
    [758.000,3.000,915.000]  #Parafuso 7
])

# Ip do controlador do robô 192.168.127.100
# Ip da câmara 192.168.127.101

class DoosanSick:
# Classe para a conexão com a câmara do sistema de visão.

    def __init__(self, ip="192.168.127.101", portIN=2115, portOUT
       =2114): # Inicializa a conexão com a câmara de visão.
    # Parâmetros:
    # ip: Endereço IP da câmara (192.168.127.101)
    # portIN: Porta de entrada para comandos (2115)
    # portOUT: Porta de saída para dados de resposta (2114)
        # Inicialização das variáveis de conexão
        self.ip = ip
        self.portIN = portIN
        self.portOUT = portOUT
        client = socket.socket()
        client.settimeout(3) # Timeout para a conexão
        socket.setdefaulttimeout(3) # Timeout padrão para o socket
        client.connect((self.ip, self.portIN)) # Conexão inicial

        # Conectar à câmara para enviar comandos e receber dados
        tp_log("Conexão com a câmara de visão...")
```

```
        try:
            self._socketIN = client_socket_open(self.ip, self.
               portIN)
            tp_log("Ligação com visão INPUT OK!")
        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)
            tp_log("Ligação com visão OUTPUT OK!")
        except Exception as e:
            tp_popup("Socket OUTPUT falha de ligação. Error: {0}".
               format(
                str(e)), DR_PM_ALARM)
            raise e

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

    def write(self, cmd, socket=None):

        # Envia um comando ASCII para o socket especificado
        # Parâmetros:
        # cmd: O comando a ser enviado (string ASCII)
        # socket: Socket para enviar o comando (padrão: _socketIN)

        if socket == None:
            socket = self._socketIN

        cmd = bytes(cmd, encoding="ascii") # Converte o comando
           para bytes ASCII

        # Envia o comando com os delimitadores STX e ETX
        STX = client_socket_write(socket, b"\x02") # Início do
           comando
        res = client_socket_write(socket, cmd) # Comando
        STX = client_socket_write(socket, b"\x03") # Fim do
           comando
```

```
        time.sleep(0.1)

        # Lê a resposta do comando
        res, rx_data = client_socket_read(socket, 10, 1)
        print (res)
        print(rx_data)

        rxd=None
        if res > 1:
            rx_msg = rx_data.decode()
            rxd =(rx_msg[1:3]) # Extrai resposta
        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

    def read_data(self, socket=None):


        # Lê dados do socket especificado

        if socket is None:
            socket = self._socketOUT

        try:
            res, rx_data = client_socket_read(socket, 1024, 1)  #
               Buffer de 1024 bytes
            if res > 0:
                return rx_data.decode('ascii') # Descodifica
                   resposta
            else:
                return None
        except Exception as e:
            tp_log("Erro durante a leitura do socket: {0}".format(
               str(e)))
            return None


def Prog_Screws (robot_coords_screws):
```

```
# Programa para mover o robô até os parafusos e aplicar a forç
   a desejada
# para aparafusar. Repete para todos os parafusos
   especificados.
# Possui como parâmetro as coordenadas dos parafusos no
   referencial do robô

# Configurações iniciais e posicionamento home
set_ref_coord(set_user_cart_coord(posx
   (0.000,0.000,0.000,0.0000,0.0000,0.0000),ref=DR_BASE))
set_tcp("Cloison")
set_ref_coord(DR_WORLD)
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)
wait(1.000)

# Posição de início (Home Position)
movel(posx(437.16,14.09,610.33,3.56,85.21,179.11))

# ------------------------ Parafuso 1
   ------------------------------------------------
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)
wait(0.500)

# Movimenta o robô para o ponto de aproximação do Parafuso 1
movel(posx(robot_coords_screws[0][0]-100,robot_coords_screws
   [0][1],robot_coords_screws[0][2],0.0000,90.0000,180.0000))

# Reduz a velocidade para a aproximação precisa
set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

# Aproximação do parafuso 1
```

```
movel(posx(robot_coords_screws[0][0],robot_coords_screws
   [0][1],robot_coords_screws[0][2],0.0000,90.0000,180.0000))
wait(0.5)

# Configurações de force e compliance
set_ref_coord(DR_TOOL)
task_compliance_ctrl([250,1000,50,200,200,200]) # Ativa o
   compliance com os valores específicos para a aplicação
force_desired = 20.0 # Força aplicada desejada
f_d = [0.0, 0.0, force_desired, 0.0, 0.0, 0.0] # Força que vai
    ser aplicada
f_dir = [0, 0, 1, 0, 0, 0] # Direção que vai ser efetuada a
   força
set_desired_force(f_d, f_dir) # Ativa o Force Control

force_ext = get_tool_force(DR_TOOL) # Vai buscar a força que
   está a ser exercida
while(force_ext[2] >= -18): # Enquanto a força não for quase
   os -20N pertendidos vai esperar
    force_ext = get_tool_force(DR_TOOL)

set_digital_output(3, 1) # Ativa a aparafusadora
wait(0.5)

espera_relatorio()

repR=get_digital_input(10)
wait(0.2)

if(repR==1): # Se o relatório for NotOK volta à home position
    set_ref_coord(DR_WORLD)
    movel(posx(545.421,0.000,339.500,0.0000,90.0000,180.0000))
               movej(posj
       (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
       ) # Home Position
    exit()

# Muda a referência para DR_WORLD
set_ref_coord(DR_WORLD)


# Recuar apoés a operação no Parafuso 1
```

```
movel(posx(robot_coords_screws[0][0]-100,robot_coords_screws
   [0][1],robot_coords_screws[0][2],0.0000,90.0000,180.0000))

# ------------------------------------------------------------

# ------------------------------- Parafuso 2
   --------------------------------
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)
wait(0.500)

# Movimenta o robô para o ponto de aproximação do Parafuso 2
movel(posx(robot_coords_screws[1][0]-100,robot_coords_screws
   [1][1],robot_coords_screws[1][2],0.0000,106.0000,180.0000))

# Reduz a velocidade para a aproximação precisa
set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

# Aproximação do parafuso 2
movel(posx(robot_coords_screws[1][0],robot_coords_screws
   [1][1],robot_coords_screws[1][2],0.0000,106.0000,180.0000))
wait(0.5)

# Configurações de force e compliance
set_ref_coord(DR_TOOL)
task_compliance_ctrl([250,1000,50,200,200,200]) # Ativa o
   compliance com os valores específicos para a aplicação
force_desired = 20.0 # Força aplicada desejada
f_d = [0.0, 0.0, force_desired, 0.0, 0.0, 0.0] # Força que vai
    ser aplicada
f_dir = [0, 0, 1, 0, 0, 0] # Direção que vai ser efetuada a
   força
set_desired_force(f_d, f_dir) # Ativa o Force Control

force_ext = get_tool_force(DR_TOOL) # Vai buscar a força que
   está a ser exercida
```

```
while(force_ext[2] >= -18): # Enquanto a força não for quase
   os -20N pertendidos vai esperar
    force_ext = get_tool_force(DR_TOOL)

set_digital_output(3, 1) # Ativa a aparafusadora
wait(0.5)

espera_relatorio()

repR=get_digital_input(10)
wait(0.2)

if(repR==1): # Se o relatório for NotOK volta à home position
    set_ref_coord(DR_WORLD)
    movel(posx(545.421,0.000,339.500,0.0000,90.0000,180.0000))
    movej(posj
       (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
       ) # Home Position
    exit()

# Muda a referência para DR_WORLD
set_ref_coord(DR_WORLD)


# Recuar apoés a operação no Parafuso 2
movel(posx(robot_coords_screws[1][0]-100,robot_coords_screws
   [1][1],robot_coords_screws[1][2],0.0000,106.0000,180.0000))

#-----------------------------------------------------------------------

# ------------------------- Parafuso 3
   ---------------------------
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)
wait(0.500)

# Movimenta o robô para o ponto de aproximação do Parafuso 3
movel(posx(robot_coords_screws[2][0]-100,robot_coords_screws
   [2][1],robot_coords_screws[2][2],0.0000,106.0000,180.0000))
```

```
# Reduz a velocidade para a aproximação precisa
set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

#ponto de aproximação
movel(posx(robot_coords_screws[2][0],robot_coords_screws
   [2][1],robot_coords_screws[2][2],0.0000,106.0000,180.0000))
wait(0.5)

# Configurações de force e compliance
set_ref_coord(DR_TOOL)
task_compliance_ctrl([250,1000,50,200,200,200]) # Ativa o
   compliance com os valores inseridos
force_desired = 20.0 # Força aplicada desejada
f_d = [0.0, 0.0, force_desired, 0.0, 0.0, 0.0] # Força que vai
    ser aplicada
f_dir = [0, 0, 1, 0, 0, 0] # Direção que vai ser efetuada a
   força
set_desired_force(f_d, f_dir) # Ativa o Force

force_ext = get_tool_force(DR_TOOL) # Vai buscar a força que
   está a ser exercida
while(force_ext[2] >= -18): # Enquanto a força não for quase
   os -20N pertendidos vai esperar
    force_ext = get_tool_force(DR_TOOL)

set_digital_output(3, 1) # Ativa a aparafusadora
wait(0.5)

espera_relatorio()

repR=get_digital_input(10)
wait(0.2)

if(repR==1): # Se o relatório for NotOK volta à home position
    set_ref_coord(DR_WORLD)
    movel(posx(545.421,0.000,339.500,0.0000,90.0000,180.0000))
    movej(posj
       (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
       ) # Home Position
    exit()
```

```
set_ref_coord(DR_WORLD)


# Recuar apoés a operação no Parafuso 3
movel(posx(robot_coords_screws[2][0]-100,robot_coords_screws
    [2][1],robot_coords_screws[2][2],0.0000,106.0000,180.0000))
#
    ------------------------------------------------------------------


# ---------------------------  Parafuso 4
    ---------------------------------------------
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)
wait(0.500)

# Movimenta o robô para o ponto de aproximação do Parafuso 4
movel(posx(robot_coords_screws[3][0]-100,robot_coords_screws
    [3][1],robot_coords_screws[3][2],0.0000,106.0000,180.0000))

# Reduz a velocidade para a aproximação precisa
set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

# Aproximação do parafuso 4
movel(posx(robot_coords_screws[3][0],robot_coords_screws
    [3][1],robot_coords_screws[3][2],0.0000,106.0000,180.0000))
wait(0.5)

# Configurações de force e compliance
set_ref_coord(DR_TOOL)
task_compliance_ctrl([250,1000,50,200,200,200]) # Ativa o
    compliance com os valores inseridos
force_desired = 20.0 # Força aplicada desejada
f_d = [0.0, 0.0, force_desired, 0.0, 0.0, 0.0] # Força que vai
     ser aplicada
f_dir = [0, 0, 1, 0, 0, 0] # Direção que vai ser efetuada a
    força
```

```
set_desired_force(f_d, f_dir) # Ativa o Force Control

force_ext = get_tool_force(DR_TOOL) # Vai buscar a força que
   está a ser exercida
while(force_ext[2] >= -18): # Enquanto a força não for quase
   os -20N pertendidos vai esperar
    force_ext = get_tool_force(DR_TOOL)

set_digital_output(3, 1) # Ativa a aparafusadora
wait(0.5)

espera_relatorio()

repR=get_digital_input(10)
wait(0.2)

if(repR==1): # Se o relatório for NotOK volta à home position
    set_ref_coord(DR_WORLD)
    movel(posx(545.421,0.000,339.500,0.0000,90.0000,180.0000))
    movej(posj
       (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
       ) # Home Position
    exit()

# Muda a referência para DR_WORLD
set_ref_coord(DR_WORLD)

# Recuar apoés a operação no Parafuso 4
movel(posx(robot_coords_screws[3][0]-100,robot_coords_screws
   [3][1],robot_coords_screws[3][2],0.0000,106.0000,180.0000))

#
   ------------------------------------------------------------------------



# -------------------------------- Parafuso 5
   ---------------------------------------
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)
wait(0.500)
```

```
# Movimenta o robô para o ponto de aproximação do Parafuso 5
movel(posx(robot_coords_screws[4][0]-100,robot_coords_screws
   [4][1],robot_coords_screws[4][2],0.0000,106.0000,180.0000))

# Reduz a velocidade para a aproximação precisa
set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

# Aproximação do parafuso 5
movel(posx(robot_coords_screws[4][0],robot_coords_screws
   [4][1],robot_coords_screws[4][2],0.0000,106.0000,180.0000))
wait(0.5)

# Configurações de force e compliance
set_ref_coord(DR_TOOL)
task_compliance_ctrl([250,1000,50,200,200,200]) # Ativa o
   compliance com os valores inseridos
force_desired = 20.0 # Força aplicada desejada
f_d = [0.0, 0.0, force_desired, 0.0, 0.0, 0.0] # Força que vai
    ser aplicada
f_dir = [0, 0, 1, 0, 0, 0] # Direção que vai ser efetuada a
   força
set_desired_force(f_d, f_dir) # Ativa o Force Control

force_ext = get_tool_force(DR_TOOL) # Vai buscar a força que
   está a ser exercida
while(force_ext[2] >= -18): # Enquanto a força não for quase
   os -20N pertendidos vai esperar
    force_ext = get_tool_force(DR_TOOL)

set_digital_output(3, 1) # Ativa a aparafusadora
wait(0.5)

espera_relatorio()

repR=get_digital_input(10)
wait(0.2)

if(repR==1): # Se o relatório for NotOK volta à home position
    set_ref_coord(DR_WORLD)
```

```
    movel(posx(545.421,0.000,339.500,0.0000,90.0000,180.0000))
    movej(posj
       (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
       ) # Home Position
    exit()

# Muda a referência para DR_WORLD
set_ref_coord(DR_WORLD)

# Recuar apoés a operação no Parafuso 5
movel(posx(robot_coords_screws[4][0]-100,robot_coords_screws
   [4][1],robot_coords_screws[4][2],0.0000,106.0000,180.0000))

#
   ------------------------------------------------------------------------



# Virar robô para os parafusos 6 e 7
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)
wait(0.500)

movel(posx(545.520,6.000,744.500,0.0000,106.0000,180.0000))

set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

movel(posx(545.520,1.500,744.500,180.0000,-107.0000,-180.0000)
   )
# ------------------------------- Parafuso 6
   ---------------------------------------
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)

# Movimenta o robô para o ponto de aproximação do Parafuso 6
```

```
movel(posx(robot_coords_screws[5][0]-100,robot_coords_screws
   [5][1],robot_coords_screws
   [5][2],180.0000,-107.0000,-180.0000))

# Reduz a velocidade para a aproximação precisa
set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

# Aproximação do parafuso 6
movel(posx(robot_coords_screws[5][0],robot_coords_screws
   [5][1],robot_coords_screws
   [5][2],180.0000,-107.0000,-180.0000))
wait(0.5)

# Configurações de force e compliance
set_ref_coord(DR_TOOL)
task_compliance_ctrl([250,1000,50,200,200,200]) # Ativa o
   compliance com os valores inseridos
force_desired = 20.0 # Força aplicada desejada
f_d = [0.0, 0.0, force_desired, 0.0, 0.0, 0.0] # Força que vai
    ser aplicada
f_dir = [0, 0, 1, 0, 0, 0] # Direção que vai ser efetuada a
   força
set_desired_force(f_d, f_dir) # Ativa o Force

force_ext = get_tool_force(DR_TOOL) # Vai buscar a força que
   está a ser exercida
while(force_ext[2] >= -18): # Enquanto a força não for quase
   os -20N pertendidos vai esperar
    force_ext = get_tool_force(DR_TOOL)

set_digital_output(3, 1) # Ativa a aparafusadora
wait(0.5)

espera_relatorio()

repR=get_digital_input(10)
wait(0.2)

if(repR==1): # Se o relatório for NotOK volta à home position
    set_ref_coord(DR_WORLD)
```

```
    movel(posx(545.421,0.000,339.500,0.0000,90.0000,180.0000))
    movej(posj
       (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
       ) # Home Position
    exit()

# Muda a referência para DR_WORLD
set_ref_coord(DR_WORLD)

# Recuar apoés a operação no Parafuso 6
movel(posx(robot_coords_screws[5][0]-100,robot_coords_screws
   [5][1],robot_coords_screws
   [5][2],180.0000,-107.0000,-180.0000))

#
   ------------------------------------------------------------------------


# --------------------------------- Parafuso 7
   ---------------------------------------
# Configurações de velocidade e aceleração
set_accj(100.000)
set_velj(50.000)
set_accx(100.000)
set_velx(50.000)

# Movimenta o robô para o ponto de aproximação do Parafuso 7
movel(posx(robot_coords_screws[6][0]-100,robot_coords_screws
   [6][1],robot_coords_screws
   [6][2],180.0000,-107.0000,-180.0000))

# Reduz a velocidade para a aproximação precisa
set_accj(50.000)
set_velj(25.000)
set_accx(50.000)
set_velx(25.000)

# Aproximação do parafuso 7
movel(posx(robot_coords_screws[6][0],robot_coords_screws
   [6][1],robot_coords_screws
   [6][2],180.0000,-107.0000,-180.0000))
wait(0.5)
```

```
# Configurações de force e compliance
set_ref_coord(DR_TOOL)
task_compliance_ctrl([250,1000,50,200,200,200]) # Ativa o
   compliance com os valores inseridos
force_desired = 20.0 # Força aplicada desejada
f_d = [0.0, 0.0, force_desired, 0.0, 0.0, 0.0] # Força que vai
    ser aplicada
f_dir = [0, 0, 1, 0, 0, 0] # Direção que vai ser efetuada a
   força
set_desired_force(f_d, f_dir) # Ativa o Force Control

force_ext = get_tool_force(DR_TOOL) # Vai buscar a força que
   está a ser exercida
while(force_ext[2] >= -18): # Enquanto a força não for quase
   os -20N pertendidos vai esperar
    force_ext = get_tool_force(DR_TOOL)

set_digital_output(3, 1) # Ativa a aparafusadora
wait(0.5)

espera_relatorio()

repR=get_digital_input(10)
wait(0.2)

if(repR==1): # Se o relatório for NotOK volta à home position
    set_ref_coord(DR_WORLD)
    movel(posx(545.421,0.000,339.500,0.0000,90.0000,180.0000))
    movej(posj
       (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
       ) # Home Position
    exit()

# Muda a referência para DR_WORLD
set_ref_coord(DR_WORLD)

# Recuar apoés a operação no Parafuso 7
movel(posx(robot_coords_screws[6][0]-100,robot_coords_screws
   [6][1],robot_coords_screws
   [6][2],180.0000,-107.0000,-180.0000))
#
   ------------------------------------------------------------------
```

```
while True:

    # Parametros iniciais fundamentais do robô
    set_ref_coord(set_user_cart_coord(posx
       (0.000,0.000,0.000,0.0000,0.0000,0.0000),ref=DR_BASE))
    set_tcp("Cloison")
    set_ref_coord(DR_WORLD)

    # Configurações de velocidade e aceleraçã
    set_accj(100.000)
    set_velj(60.000)
    set_accx(35.000)
    set_velx(35.000)
    time.sleep(0.1)

    camera_ip = "192.168.127.101"
    trispector = DoosanSick(ip=camera_ip, portIN=2115, portOUT
       =2114)
    time.sleep(0.5)

    laser_on = trispector.write("set laser on") # Laser ativado
       para captura de imagem

    while (True):
        movej(posj
           (0.000000,-36.480000,120.820000,-180.000000,-95.660000,1.870000)
           ) # Home Position
        wait(5.000)

        # FAZER SCAN -> CRIAR ROTINA
        movel(posx
           (292.020,-4.00,680.570,90.0000,-90.0000,-16.0000))
        set_digital_output(13, 1) # Ativa a recolha pela câmara
        movel(posx
           (292.020,-124.00,680.570,90.0000,-90.0000,-16.0000))
        set_digital_output(13, 0) # Desativa a recolha pela câmara
        wait(1.000)

        data_from_camera= trispector.read_data()
        tp_log(str(data_from_camera))
```

```
    data_list = data_from_camera.split(',') # Armazena dados
       em data_list

    # Converter os elementos que são números para float
    for i in range(len(data_list)):
        try:
            data_list[i] = float(data_list[i])
            tp_log(str(data_list[i]))
        except ValueError:
            # Se a conversão falhar, mantém o valor original (
               por exemplo, para "true")
            pass
    if data_list[1]==2 and data_list[0]>=70:
        laser_off = trispector.write("set laser off")

        break


# PONTOS LIDOS DA CAMERA -> ALTERAR AQUI DEPOIS
camera_point_1 = np.array([data_list[2], data_list[3],
   data_list[4]])
camera_point_2 = np.array([data_list[5],data_list[6],
   data_list[7]])

# Calculo auxiliar de diferença entre eixos
dif_x_1= abs(data_list[5])- abs(CAMERA_COORDS[0][0])
dif_y_1= abs(data_list[6])- abs(CAMERA_COORDS[0][1])
dif_z_1= abs(data_list[7])- abs(CAMERA_COORDS[0][2])

dif_x_2= abs(data_list[2])- abs(CAMERA_COORDS[1][0])
dif_y_2= abs(data_list[3])- abs(CAMERA_COORDS[1][1])
dif_z_2= abs(data_list[4])- abs(CAMERA_COORDS[1][2])


# Cálculo do offsset em y por interpolação linear inversa
y1, x1 = 1, 1.7
y2, x2 = 4, 11.3

##Parafuso 3 e 4 ref
y_offset = y1 + (dif_y_1 - x1) * (y2 - y1) / (x2 - x1)
y_offset_final= y_offset + dif_y_1
```

```
robot_coords_1_final= [POSE_OTHER_SCREW[0][0],POSE_OTHER_SCREW
   [0][1]-y_offset_final,POSE_OTHER_SCREW[0][2]]
robot_coords_2_final= [POSE_OTHER_SCREW[1][0],POSE_OTHER_SCREW
   [1][1]-y_offset_final,POSE_OTHER_SCREW[1][2]]
robot_coords_3_final= [ROBOT_COORDS[0][0],ROBOT_COORDS[0][1]-
   y_offset_final,ROBOT_COORDS[0][2]]
robot_coords_4_final= [ROBOT_COORDS[1][0],ROBOT_COORDS[1][1]-
   y_offset_final,ROBOT_COORDS[1][2]]
robot_coords_5_final= [POSE_OTHER_SCREW[2][0],POSE_OTHER_SCREW
   [2][1]-y_offset_final,POSE_OTHER_SCREW[2][2]]
robot_coords_6_final= [POSE_OTHER_SCREW[3][0],POSE_OTHER_SCREW
   [3][1]-y_offset_final,POSE_OTHER_SCREW[3][2]]
robot_coords_7_final= [POSE_OTHER_SCREW[4][0],POSE_OTHER_SCREW
   [4][1]-y_offset_final,POSE_OTHER_SCREW[4][2]]

tp_log ("Pose Y P1 "+ str(robot_coords_1_final[1]))
tp_log ("Pose Y P2 "+ str(robot_coords_2_final[1]))
tp_log ("Pose Y P3 "+ str(robot_coords_3_final[1]))
tp_log ("Pose Y P4 "+ str(robot_coords_4_final[1]))
tp_log ("Pose Y P5 "+ str(robot_coords_5_final[1]))
tp_log ("Pose Y P6 "+ str(robot_coords_6_final[1]))
tp_log ("Pose Y P7 "+ str(robot_coords_7_final[1]))

robot_coords_all= [robot_coords_1_final,robot_coords_2_final,
   robot_coords_3_final,
                   robot_coords_4_final,robot_coords_5_final,
                      robot_coords_6_final,
                   robot_coords_7_final]

# Criar rotina de deslocação para os pontos calculados
Prog_Screws (robot_coords_all)
```

Instituto Superior de Engenharia
Politécnico de Coimbra